Diplomarbeit

Lower bounds and algorithms for covering sequences of subsets of a finite set

im Rahmen des Studiums
Technische Mathematik

unter der Anleitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther R. Raidl
Univ.Ass. Dipl.-Ing. Enrico Iurlano, BSc

eingereicht von
Alexander Weissenfels, BSc
Matrikelnummer 11713131

an der Fakultät für Mathematik und Geoinformation
der Technischen Universität Wien

ausgeführt am
Institut für Logic and Computation

Wien, August 2026

Alexander Weissenfels

Günther Raidl

Abstract

In this thesis we focus on the generalisation of universal cycles to covering sequences. Universal cycles are compact strings encoding a class of combinatorial objects. A P_n-covering sequence is a string such that every subset of $\{1, \ldots, n\}$ is contained as a contiguous substring. We provide new lower bounds on the minimal length of P_n-covering sequences and related problems. Additionally theorems are given to obtain a lower bound on such sequences containing a predefined substring which we in turn use in a computational approach to further improve upon the known lower bounds. We derive and exploit a novel reduction to the Generalised Traveling Salesperson Problem. For an alternative construction algorithm, we introduce the concept of pruning admissible equivalence relations and preorders. In a modified variant of beam search, an incomplete tree search algorithm, pruning admissible preorders along with heuristics are then used to substantially reduce the search space. Experiments on different heuristics are run and the best heuristic is chosen for the algorithm in a bigger set of experiments, which yield new shortest-known P_n-sequences.

Contents

1 Introduction

An infamous problem statement due to de Bruijn given in 1948 [6], given $n \in \mathbb{N}$ is concerned with the task of constructing a cyclic string u on the alphabet $\{0, 1\}$ such that every word in $\{0, 1\}^n$ can be found exactly once as a cyclical factor of u.

For example the cyclic string $u = 00010111$ contains all words on $\{0, 1\}$ of length 3. The cyclic nature is understood as allowing the string to wrap back around at the end to the start, so the word 110 is present in u as the cyclic substring $u_7u_8u_1$.

These and similar cycles have many applications in real life such as in digital fault testing, pseudo-random number generation, modern public- key cryptographic schemes, and many more [5]. Generally, the study of universal cycles is concerned with the existence of such compact strings, where every k-substring represents or encodes a unique combinatorial object of some class of combinatorial objects. These may be classes of strings (i.e. languages), permutations, subsets of a set, trees, or other combinatorial structures that can be effectively encoded as a string.

Note that unlike in the example above we may have several representations for the same combinatorial object. For example the string

$$\hat{u} = \texttt{12345674527635716327415624135217346}$$

is a universal cycle for the subsets of $\{1, \ldots, 7\}$ of size 3. Here the set $\{4, 6, 7\}$ is found as the substring 674 (starting at index 6) but it would have been perfectly legal for $\hat{u}$ to contain any one of the other five ways of encoding this subset. Such cyclic strings encoding subsets of a set are relevant to the problem of file organisation, in particular for consecutive retrieval problems [10, 20].

For the majority of this thesis we will then however drop the condition of cyclic strings and will not consider wrap-around at the end. Furthermore, where universal cycles and sequences are concerned with the existence of strings where every (cyclical) k-factor represents a unique object of some class we want to generalise to so-called covering strings that just require all objects of interest are encoded somewhere as a substring but possibly more than once and instead concern ourselves with the minimum length of such a string.

Definition. We call a string s a P_n^k-covering sequence if for every k-subset $A \subseteq \{1, \ldots, n\}$ there exists at least one k-substring t in s for which $\{t_1, \ldots, t_k\} = A$.

While this definition has a counterpart in the world of universal cycles we are also interested in P_n-covering sequences, where we require every subset of $\{1, \ldots, n\}$ be encoded as a substring in a P_n-covering sequence. Our main result is the following.

Theorem. If $n \nmid \binom{n}{k}$ then every P_n^k-covering sequence is at least of length

$$n\left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \left\lceil \frac{kr}{n} \right\rceil - 1$$

where $r = \binom{n}{k} \bmod n$.

We also prove that this bound is larger than existing bounds if $kr > n$. Note that since any P_n-covering is also a P_n^k-covering sequence for $k \in \{1, \ldots, n\}$ we can also apply this theorem to P_n-covering sequences. We then provide several generalisations of previous results bounding the length of P_n^k-covering sequences, which contain a certain predefined substring, from below — a result which we then use in iterative deepening depth-first search to further strengthen lower bounds on lengths of such sequences.

We then turn our interest on constructive approaches, which may of course also be thought of as finding upper bounds. To this end we propose the novel reduction of the problem of finding P_n^k-covering sequences to an instance of the Generalised Traveling Salesperson problem.

With the limitations of this approach in mind we then go on to present an adaptation of the well-known beam search algorithm. In particular we are able to exploit specific details of the problem allowing us to prune a vast amount of the search space and improve on the efficacy of the algorithm. The fruits of our labour are harvested in the form of new shortest P_n^k- and P_n-covering sequences (which can all be found in full in the Appendix of this thesis).

2 Universal objects for combinatorial structures

The next subsections strive to give a formal definition of what was just informally described and also provide several examples, but before we can dive in we need to recall some common definitions and also introduce some of our own notation.

2.1 Notation

We will usually denote an *alphabet* with the letter A. An *alphabet* is just a set of "symbols" or *letters*, for our purposes mostly finitely many. A finite sequence of letters of A is called an *A-word* or just *word* if the relevant alphabet A is clear from context, which is usually the case. We will also commonly refer to words as *strings*. The *empty word* is denoted by ε. Also denote by $\mathbb{N}$ the natural numbers starting with 1, i.e. $\mathbb{N} = \{1, 2, \dots\}$ and subsequently let $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Definition 2.1.

(i) *Concatenation*: For words $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_m$ we define the word $u \cdot v$ as $u_1 \cdots u_n v_1 \cdots v_m$. Oftentimes we may also just write uv.

(ii) *Potentiation*: For a word w and $k \in \mathbb{N}$ we define w^k recursively by $w^0 = \varepsilon$ and $w^{k+1} = ww^k$.

(iii) *Length*: For a word $s = s_1 \cdots s_n$ we define $|s| = n$.

(iv) *Words of length k on A*: For an alphabet A and $k \in \mathbb{N}_0$ we define A^k as the words of length k. More formally we define $A^0 = \{\varepsilon\}$, $A^1 = A$, $A^{k+1} = A \cdot A^k$ and $A \cdot A = \{x \cdot y : x, y \in A\}$.

(v) *Words on A (Kleene Star)*: We define the set of all words on A as

$$A^* = \bigcup_{k \geq 0} A^k.$$

(vi) *Reversion*: For a word $s = s_1 \cdots s_n$ we define its reverse word $\overleftarrow{s} = s_n \cdots s_1$.

While the preceding definitions are commonplace, the following notation is not but will be useful for us in this thesis.

Definition 2.2. Let $w = w_1 \cdots w_n$ be a word. We denote by $\mathrm{suff}(w)$ the set of all suffixes of w. More formally

$$\mathrm{suff}(w) = \{\varepsilon, w_n, w_{n-1}w_n, \dots\}.$$

Analogously we define the set of prefixes by

$$\mathrm{pref}(w) = \{\varepsilon, w_1, w_1w_2, \dots\}.$$

Also for $k \in \mathbb{N}_0$ define $\mathrm{suff}_k(w)$ as the suffix of w of length k if it exists (i.e. $|w| \geq k$). If on the other hand $|w| < k$ then we simply set $\mathrm{suff}(w) = w$. The same definitions apply analogously for $\mathrm{pref}_k(w)$.

For the sake of better legibility of some results that we will prove in Section 3.2 we introduce the following notation, which we state here for its generality.

Definition 2.3. Let $L, U, x \in \mathbb{Z}$. Define

$$\begin{aligned}\operatorname{clamp}_L(x) &= \max(x, L),\\ \operatorname{clamp}^U(x) &= \min(x, U),\\ \operatorname{clamp}_L^U(x) &= \operatorname{clamp}^U(\operatorname{clamp}_L(x)).\end{aligned}$$

2.2 Universal sequences and universal cycles

In the following we aim to define universal sequences and universal cycles in the most general setting inspired by and adapting the notation of [5, 9, 29, 31].

Definition 2.4. Let $\mathcal{F}$ be a family of combinatorial objects and A a finite alphabet. We call $\varphi_{\mathcal{F}}\colon A^* \to \mathcal{F}$ a *representation function* if φ is a partial surjective function. For a combinatorial object $F \in \mathcal{F}$ we call $\varphi_{\mathcal{F}}^{-1}(\{F\})$ the *representations* of F.

If we are only talking about a single representation function or it is clear from the context we may just write φ instead of $\varphi_{\mathcal{F}}$.

Remark 2.5.

(i) Some authors also allow countably infinite alphabets [9].

(ii) In addition to "representations", the term *encodings* is frequently used in the literature, often interchangeably with representations [9, 29, 33].

Definition 2.6. Let X be a set. Then we denote by $\binom{X}{k}$ the family of subsets of X having cardinality k. We also refer to these as *k-subsets*.

Example 2.7. Consider $A = \{1, 2, 3, 4, 5\}$ and let $\mathcal{F} = \binom{A}{3}$ denote the family of 3-subsets on A. A natural choice for a representation function $\varphi_{\mathcal{F}}$ could be

$$\begin{aligned}\varphi_{\mathcal{F}}\colon A^* &\to \mathcal{F},\\ w = v_1 \cdots v_n &\mapsto \begin{cases}\{v_1, v_2, v_3\} & \text{if } |w| = n = 3, v_1 \neq v_2, v_2 \neq v_3 \text{ and } v_3 \neq v_1,\\ \text{undefined} & \text{otherwise.}\end{cases}\end{aligned}$$

In this case the representations for $\{2, 3, 5\}$, i.e. $\varphi_{\mathcal{F}}^{-1}(\{2, 3, 5\})$ would be

$$235, 253, 325, 352, 523, 532.$$

This example should illustrate why we require a partial mapping from $A^* \to \mathcal{F}$.

Definition 2.8 (Universal sequences and universal cycles)**.** Let $\mathcal{F}_n$ be a family of m combinatorial objects of "rank n"[1], A a finite alphabet, and $\varphi_{\mathcal{F}_n}$ a representation

[1] e.g. permutations on n objects, sets of size n, etc. This is just stated rather informally here but can be thought of more formally as assigning weights to combinatorial objects akin to how it is done in the field of *Enumerative combinatorics*.

function for which all representations have the same length k. Then a string

$$s = s_1 \cdots s_{m+k-1} \in A^*$$

is a *universal sequence* if for all $F \in \mathcal{F}_n$ there exists exactly one $i \in \{1, \ldots, m\}$ such that $\varphi_{\mathcal{F}_n}(s_i \cdots s_{i+k-1}) = F$. We call a string $u = u_1 \cdots u_m$ a *universal cycle* if and only if

$$u_1 \cdots u_m u_1 \cdots u_{k-1}$$

is a universal sequence.

Remark 2.9. Equivalently, we may say that $u = u_1 \cdots u_m$ is a universal cycle, if for every $F \in \mathcal{F}_n$ there exists exactly one $i \in \{1, \ldots, m\}$ such that

$$\varphi_{\mathcal{F}_n}(s_{((i-1) \bmod m)+1} \cdots s_{((i+k-2) \bmod m)+1}) = F,$$

where $x \bmod y \in \{0, \ldots, y-1\}$ denotes the bivariate modulo function on $\mathbb{N} \times \mathbb{N}$ returning the remainder of the Euclidean division of x by y. The equation above is an alternative description of the allowed "wrap-around" at the end of the string.

Definition 2.10. In the setting of Definition 2.8 it is common to refer to factors (cyclic factors in the case of universal cycles) $u_i \cdots u_{i+k-1}$ as *k-blocks* and we say a block is *valid* if and only if it is the representation of a combinatorial object $F \in \mathcal{F}_n$ for which we may then say that $u_i \cdots u_{i+k-1}$ *covers* F. Additionally we may say that a word $u = u_1 \cdots u_m$ *covers* F if there exists a valid block of u that covers F. Some authors prefer to use the word *window* over block [29].

Remark 2.11. Notice that since representations always identify a single combinatorial object and a universal sequence or cycle u consists of exactly m k-blocks which is also the number of combinatorial objects that need to be covered, every k-block of u already has to be a valid block.

In the following and whenever necessary we will use the notation $A^{(\times,n)}$ to mean the n-ary Cartesian power of A as opposed to A^n, the set of all n-letter words on A to properly distinguish these objects, although the set-theoretic implementations of them may coincide.

Example 2.12. A well-known example is the following. Let $A = \{0, 1\}$, $n = 3$ and $F_3 = A^{(\times,3)}$, i.e. our combinatorial objects here are three-dimensional bitvectors which we represent in the canonical way by the representation function $\varphi_{\mathcal{F}_3} \colon A^3 \to A^{(\times,3)}$, $\varphi_{\mathcal{F}_3}(x_1x_2x_3) = (x_1, x_2, x_3)$. Then an example of a universal cycle in this setting is

$$u = \texttt{00010111}$$

as can be quickly verified. For instance the bitvector $(1, 1, 0)$ is covered by the block (i.e. represented by the cyclic factor) $u_7u_8u_1$ in this case.

Remark 2.13. As is apparent from Definition 2.8 any universal cycle $u = u_1 \cdots u_m$ can be viewed as a universal sequence s by appending the $k-1$ letter long prefix of u onto its end, i.e.

$$s = u_1 \cdots u_{k-1} u_k \cdots u_m u_1 \cdots u_{k-1},$$

where $|s| = m + k - 1$. The universal cycle shown in Example 2.12 becomes

$$s = \texttt{0001011100},$$

where the bitvector $(1, 1, 0)$ is covered by $u_7u_8u_9$.

In fact, the objects studied in Example 2.12, which have come to be known as de Bruijn sequences[2], can be shown to exist for all finite alphabets A, $n > 0$, $F_n = A^{(\times,n)}$ and the canonical representation function above. The proof is based on showing a certain graph has a Eulerian circuit. We recall the definition of Eulerian circuits and the well-known characterisation when graphs contain such circuits here for the reader's convenience.

Definition 2.14. Let $G = (V, E)$ be a directed graph. A closed walk in G is called a Eulerian circuit if it visits every edge $e \in E$ exactly once.

Theorem 2.15. *Let $G = (V, E)$ be a directed graph. Then G contains a* Eulerian circuit *if and only if G is strongly connected and for every vertex $v \in V$ their in- and out-degree are equal, i.e. $d^+(v) = d^-(v)$. In this case we may say G is* Eulerian*.*

Definition 2.16. Let A be an alphabet of size r. We call the directed graph $G_n^r = (V, E)$ the de Bruijn graph of order n, where $V = A^n$ the set of all strings of A of length n and $(u, v) = (u_1 \cdots u_n, v_1 \cdots v_n) \in E$ if and only if $u_2 \cdots u_n = v_1 \cdots v_{n-1}$.

It is common to set $A = \{0, \ldots, r - 1\}$.

Example 2.17. A visualisation of G_3^2 ($A = \{0, 1\}$ and $n = 3$) is given in Fig. 1, where every edge $(u_1 \cdots u_n, v_1 \cdots v_n)$ has been labelled as $u_1 \cdots u_n \cdot v_n$, which is possible due to the structure of the edge set of G_3^2 as laid out in Definition 2.16.

Theorem 2.18 (De Bruijn sequences)**.** *For a finite alphabet A of size r and $n > 0$ there exists a (r, n)-de Bruijn sequence of length r^n.*

Proof. Let $n > 0$ and A a finite alphabet of size r. Consider the de Bruijn graph G_{n-1}^r. Associate for every word $w = w_1 \cdots w_n$ over A of length n the unique edge in G_{n-1}^r connecting $w_1 \cdots w_{n-1}$ and $w_2 \cdots w_n$. In this way a Eulerian circuit $v^{(1)}, \ldots, v^{(r^n)}$ in G_{n-1}^r corresponds to a de Bruijn sequence by writing down the first letter of each v_i, $i = 1, \ldots, r^n$ consecutively. By the well-known characterisation of Eulerian graphs (see Theorem 2.15) we see that indeed G_{n-1}^r is Eulerian as for all vertices v of the graph the in-degree as well as the out-degree are equal to r and clearly the graph is connected, which is what we wanted to show. □

Importantly universal cycles need not exist for all choices of families of combinatorial objects and representation functions as the following examples show.

Example 2.19. Let $F_3 = S_3$ be the set of permutations on $\{1, 2, 3\}$.

[2] "cycles" would be a much more consistent naming in this case, but for historic reasons they are called sequences.

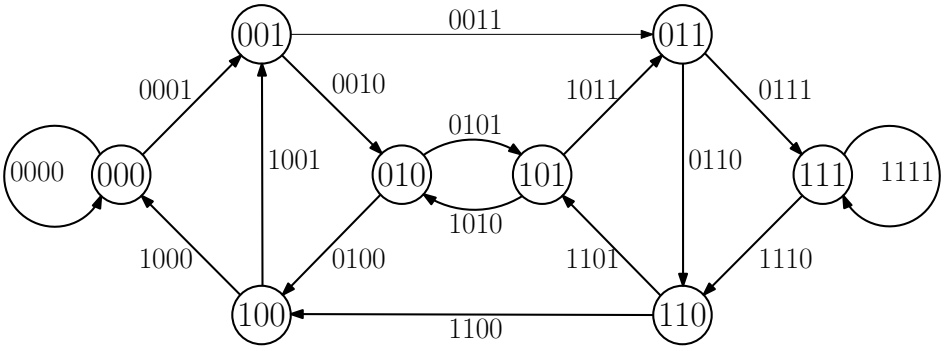

Figure 1: De Bruijn graph on $\{0,1\}$ of order 3.

(i) Consider the alphabet $A = \{1,2,3\}$ let $\varphi(x_1x_2x_3)$ be the permutation of S_3 that maps $1 \mapsto x_1$, $2 \mapsto x_2$ and $3 \mapsto x_3$ whenever x_1, x_2 and x_3 are pairwise distinct, and undefined otherwise. What was just formally defined for permutations of S_3 is just the common way of notating permutations by their image on $1,2,3$ in that order and will also be the way we notate permutations from here on out unless otherwise specified. Clearly the representation function φ is injective and the unique representation for the identity is 123. Due to the cyclic nature of universal cycles and the fact that there are 6 permutations on a 3-set let us make the Ansatz (a, b, c are variables)

`123abc`.

Since the factor `23a` has to be a representation for a permutation we must have that $a = 1$. Similarily it follows that $b = 2$ and $c = 3$. It is apparent though, that `123123` is not a universal cycle since there is no representation `321` for instance.

(ii) However, consider now a representation function that takes words of length two instead. Formally, let $\varphi(x_1x_2)$ be the permutation of S_3 that maps $1 \mapsto x_1$, $2 \mapsto x_2$ and $3 \mapsto z$ where z is the unique missing element in $\{1,2,3\} \setminus \{x_1, x_2\}$ whenever x_1, x_2 are pairwise distinct, and undefined otherwise. These are known as *shorthand permutations* and clearly no information is lost here, since the missing element can always be inferred. Indeed in this setting a universal cycle exists for the very same combinatorial objects. For example

`121323`

is one such universal cycle. The correctness is easily verified. For instance the permutation `321` from afore is covered by the block `32` at indices 4–5.

This example easily generalizes to any $n \geq 2$ and the existence of universal cycles for shorthand permutations will follow from Theorem 2.28 which is proved later in this subsection and can be found there as Corollary 2.30.

Here we have seen how different representation functions on the same alphabet can lead to very different conclusions. Clearly changing the alphabet also has an impact. Consider as an extreme example the alphabet $\{1, \ldots, 6\}$ and a representation function that simply associates each permutation in S_3 with a unique letter in the alphabet. Then `123456` is a universal cycle or any permutation thereof for that matter. This is of course not of much interest, but the general idea of expanding the alphabet slightly can be.

Definition 2.20 ([9, 28])**.** Let $A \subseteq \mathbb{N}$ and let $v, w \in A^*$ be words over the alphabet A, that have the same length $n = |v| = |w|$. Then v and w are called *order-isomorphic* if and only if for all $i, j \in \{1, \ldots, n\}$ we have that

$$v_i < v_j \iff w_i < w_j$$

holds.

Example 2.21 ([28])**.** Suppose we are still interested in finding a compact string containing the permutations of a 3-set S_3 as in Example 2.19. To that end consider instead the alphabet $A = \{1, 2, 3, 4\}$ with one more letter added. Define the representation function φ such that for all $\pi \in S_3$ we have that

$$\varphi^{-1}(\pi) = \{v \in A^* \mid v \text{ is order-isomorphic to } \pi(1)\pi(2)\pi(3)\}.$$

The following string works as a universal cycle in this context:

`123143`.

As an example the permutation 132 is covered by the block `143`.

As a matter of fact, Example 2.21 can be generalised for arbitrary n, i.e. it can be shown (see Theorem 2.22) that a universal cycle always exists for the permutations S_n over an alphabet with $n + 1$ symbols using the representation described above. The proof is involved, so it is not reproduced here.

Theorem 2.22 ([28])**.** *For all $n \geq 3$ there exists a word of length $n!$ over the alphabet $\{1, \ldots, n+1\}$ such that each element of S_n is order-isomorphic to exactly one of the $n!$ cyclic intervals of length n.*

Remark 2.23. Note that "cyclic intervals" can also be referred to as blocks in our notation and it should also be remarked that we could have equivalently chosen the set $\{0, \ldots, n\}$ as the alphabet in Theorem 2.22 or any other chain of length $n + 1$.

Definition 2.24. We call the number $n^{\underline{k}}$ the falling factorial defined by

$$n^{\underline{k}} = \overbrace{n(n-1)(n-2)\cdots(n-k+1)}^{k \text{ factors}}.$$

Example 2.25. There is an interesting spin on the topic of permutations. Consider an alphabet $A = \{1, \ldots, n\}$ but instead of permutations on A we consider all permutations on k-subsets of A where $1 \leq k < n$ (sometimes called k-subpermutations

or k-permutations). Equivalently let $\mathcal{F}$ be the set of injective strings on A of length $1 \leq k < n$. Clearly then

$$|\mathcal{F}| = k!\binom{n}{k} = \frac{n!}{(n-k)!} = n^{\underline{k}}.$$

With the latter interpretation we may just set $\varphi_{\mathcal{F}}\big|_{\mathcal{F}} = \mathrm{id}_{\mathcal{F}}$. Notice that if we were to allow $k = n$ then we have already seen in Example 2.19 (i) that even for k as little as 3 no universal cycles or sequences can exist. The story changes for $k < n$ however, in fact we can use a similar construction as we have used in the proof of the existence of de Bruijn sequences introduced in Definition 2.16. An example for a universal cycle with $(n, k) = (3, 2)$ is

131232.

Definition 2.26. Let A be an alphabet of size n. Denote by $D_n^k = (V, E)$, $1 \leq k < n$ the directed graph where V contains all injective strings of length k over A and $(u, v) = (u_1 \cdots u_k, v_1 \cdots v_k) \in E$ if and only if $u_2 \cdots u_k = v_1 \cdots v_{k-1}$ and $u_1 \neq v_k$.

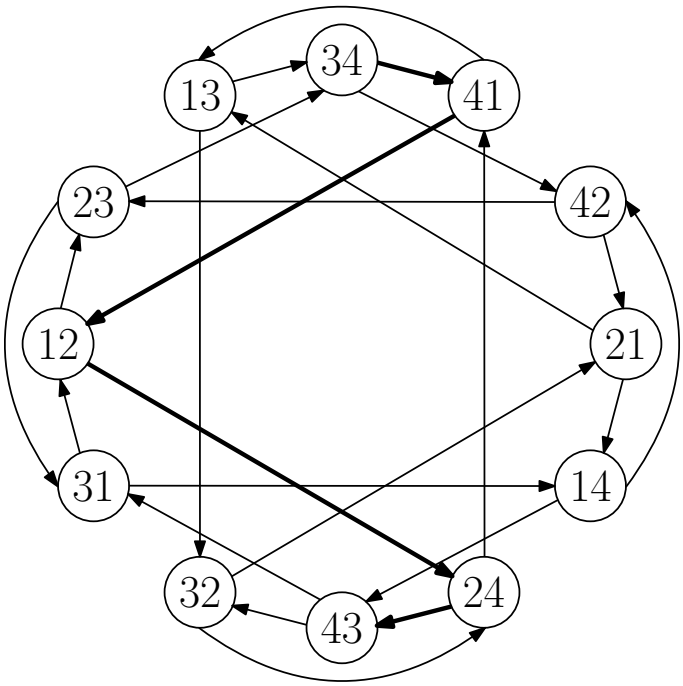

Figure 2: Transition graph D_4^2 as defined in Definition 2.26. A Eulerian circuit for the depicted graph is given by the list of vertices (14, 43, 31, 12, 24, 43, 32, 24, 41, 13, 32, 21, 14, 42, 21, 13, 34, 42, 23, 34, 41, 12, 23, 31).

Lemma 2.27. *Let $1 \leq k < n$. Then there is a directed path from $u_1 \cdots u_{k-1}$ to any of its cyclical rotations $u_{r+1} \cdots u_{k-1} u_1 \cdots u_r$ for $r \in \{1, \ldots, k-2\}$ in D_n^{k-1}.*

Proof. Let $u_1 \cdots u_{k-1}$ be a vertex in D_n^{k-1}. Notice, that it suffices to find a directed path from $u_1 \cdots u_{k-1}$ to $u_2 \cdots u_{k-1}u_1$ since any cyclical r-rotation can be achieved repeating this rotation the appropriate number of times. Since $k < n$ we have that $n-(k-1) \geq 2$ and hence there exists $x, y \in \{1, \ldots, n\} \setminus \{u_1, \ldots, u_{k-1}\}$ with $x < y$. Clearly then there is a path $u_1 \cdots u_{k-1} \to u_2 \cdots u_{k-1}x \to u_3 \cdots xy$. Critically now the first letter is u_3 ($\neq u_2$) so we may continue this like $u_3 \cdots xy \to u_4 \cdots yu_2 \to u_5 \cdots u_2u_3$ and so on to $xy \cdots u_{k-2} \to yu_2 \cdots u_{k-1} \to u_2 \cdots u_{k-1}u_1$. In this way we have cyclically rotated the string by one step, which is what we wanted to show. □

For a more visual illustration, we indicated by fat arrows in Fig. 2 the path as constructed in the proof of Lemma 2.27 to reach the 1-rotation 43 of 34 from 34.

The following theorem and its proof are taken from [17]. We found it useful to supplement their abridged proof with more details, including factoring out Lemma 2.27 which is used in their proof without justification. The proof of this lemma uses similar methods to the ones used in the main proof of Theorem 2.28. We also want to remark that an error has been made in [3, Lemma 19] regarding Theorem 2.28, where they define a graph with corresponding vertices and edges but fail to correctly show the connectedness of their graph for the edges they have defined.

Theorem 2.28. *For a finite alphabet A of size n and $1 \leq k < n$ there exists a universal cycle on the injective words of A^k with length $n^{\underline{k}}$.*

Proof. Let $n, k \in \mathbb{N}$ such that $1 \leq k < n$ and A an alphabet of size n. Consider the directed graph D_n^{k-1}. We claim that the set of edges corresponds one-to-one with injective words over A. Clearly given such an injective word $w_1 \cdots w_k$, since $w_1 \neq w_k$ and since $w_2 \cdots w_k$ and $w_1 \cdots w_{k-1}$ are both injective and agreeing on the indices $2, \ldots, k-1$ there is a unique edge connecting $(w_2 \cdots w_k, w_1 \cdots w_{k-1})$ in D_n^{k-1}. On the other hand given such an edge and the fact that $w_1 \neq w_k$ we can reconstruct an injective string of length k from it. In this way a Eulerian circuit $(c_1, \ldots, c_{n^{\underline{k}}})$ corresponds to a universal cycle on injective words over A by writing down the respective first letters of each c_i, $i = 1, \ldots, n^{\underline{k}}$ consecutively.

It remains to show that D_n^{k-1} is indeed Eulerian. To this end we again utilise Theorem 2.15 as we did in showing the existence of De Bruijn sequences. Clearly every vertex has in-degree as well as out-degree equal to $n-k+1$. For the (strong) connectedness we have to do a bit more work. To that end let $u_1 \cdots u_{k-1}$ be any vertex in D_n^{k-1}. We first show that there is a directed path from $u_1 \cdots u_{k-1}$ to a vertex $v_1 \cdots v_{k-1}$ with $v_1 < \cdots < v_{k-1}$. If $u_1 < \cdots < u_{k-1}$ we are done. Therefore suppose $u_{m-1} > u_m$ for some $m \in \{2, \ldots, k-1\}$. By Lemma 2.27 there is a directed path from $u_1 \cdots u_{k-1} \to u_{m-1}u_m \cdots u_{k-1}u_1 \cdots u_{m-2}$. As we did in the proof of aforementioned lemma choose $x, y \in \{1, \ldots, n\} \setminus \{u_1, \ldots, u_{k-1}\}$ with $x \neq y$. Then $u_{m-1} \cdots u_{k-1}u_1 \cdots u_{m-2} \to u_m \cdots u_{k-1}u_1 \cdots u_{m-2}x \to u_{m+1} \cdots u_{k-1}u_1 \cdots u_{m-2}xy$, which again due to Lemma 2.27 is connected to $xyu_{m+1} \cdots u_{k-1}u_1 \cdots u_{m-2}$, which in turn is connected to $yu_{m+1} \cdots u_{k-1}u_1 \cdots u_{m-2}u_m \to u_{m+1} \cdots u_{k-1}u_1 \cdots u_mu_{m-1}$, for the choice of x, y. Another invocation of Lemma 2.27 gives us a path to

$u_1 \cdots u_{m-2} u_m u_{m-1} \cdots u_{k-1}$. This process of transposing neighbouring letters, which are in a state of inversion can be continued until we have found a directed path to a vertex $v_1 \cdots v_{k-1}$ for which we have $v_1 < \cdots < v_{k-1}$.[3]

For the next step we show that v is connected by a directed path to $12 \cdots (k-1)$. Clearly if $v_{k-1} = k-1$ it follows by monotonicity of $v\colon \{1, \ldots, k-1\} \to A$ that $v_i = i$, which is what we wanted. Therefore let $l \in \{1, ..., k-1\}$ be the smallest index such that $v_l \neq l$. Using Lemma 2.27 we find a directed path from $v_1 \cdots v_{k-1}$ to the cyclical rotation $v_l \cdots v_{k-1} 1 \cdots (l-1)$. Since $v_l \neq l$ and moreover $v_i \neq i$ for all $i \in \{l, \ldots, k-1\}$ we get a directed path from $v_l \cdots v_{k-1} 1 \cdots (l-1)$ to $v_{l+1} \cdots v_{k-1} 1 \cdots (l-1)(l)$, which is connected to $v_{l+2} \cdots v_{k-1} 1 \cdots (l)(l+1)$ and so on to $1 \cdots (k-1)$. Since we have shown that any vertex $u_1 \cdots u_{k-1}$ is connected by a directed path to $1 \cdots (k-1)$ (and the connection from $1 \cdots (k-1)$ to any vertex can be shown in an analogous manner or by relabeling) the graph D_n^{k-1} is indeed strongly connected and hence Eulerian, which is what we wanted to show. □

Example 2.29. A universal cycle for $(n, k) = (4, 3)$ can be found by traversing a Eulerian circuit in Fig. 2 starting at vertex 14 and writing down the first letter consecutively as described in the proof of Theorem 2.28:

`143124324132142134234123`.

Corollary 2.30 (Shorthand permutations)**.** *For $n \geq 2$ there exists a word of length $n!$ over the alphabet $\{1, \ldots, n\}$ such that each permutation π is represented by exactly one of the $n!$ cyclic intervals of length $n-1$ and of the form $\pi(1) \cdots \pi(n-1)$.*

Proof. Let $k = n - 1$. Then any one of the k-subpermutations of which there are $n^{\underline{k}} = n!$ uniquely identifies one of the $n!$ permutations of length n in their shorthand representation. Therefore, applying Theorem 2.28 suffices to prove the existence of a universal cycle for shorthand permutations over $\{1, \ldots, n\}$. □

Note that the preceding proofs are not constructive in the sense that they do not provide us with an efficient way to construct the discussed objects. That presents its own challenge and we will not go into further detail here. For the particular case of constructing universal cycles on shorthand permutations we refer the interested reader to [9, sec. 7]

Remark 2.31. The fact that we require $k < n$ in Theorem 2.28 is crucial, since for $k = n$ the graph D_n^{k-1} is disconnected. A depiction of D_3^2 is given by Fig. 3.

We have seen how a suitable representation can permit or prohibit the existence of universal cycles and sequences. Sometimes though, we might want to stick to a specific representation function. In this case, we may instead relax the conditions for universal cycles (sequences) by not requiring all blocks to be valid but rather looking for a shortest word *covering* all combinatorial objects we are interested in.

[3] To see that this is indeed the case the reader is referred to standard proofs of the correctness of the *bubble sort* algorithm.

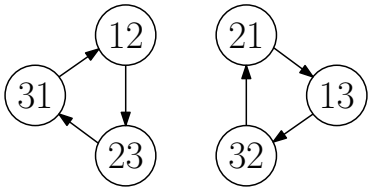

Figure 3: Disconnected transition graph D_3^2.

2.3 Covering sequences

Definition 2.32. Let $\mathcal{F}$ be a family of combinatorial objects, A a finite alphabet and $\varphi_{\mathcal{F}}$ a representation function. Then a string $s = s_1 \cdots s_m \in A^*$ is an *$\mathcal{F}$-covering sequence* if for all $F \in \mathcal{F}$ there *exists at least* one $i \in \{1, \ldots, m\}$ such that there exists $k \geq 1$ and $\varphi_{\mathcal{F}}(s_i \cdots s_{i+k-1}) = F$.

Remark 2.33. A similar definition is of course possible in the cyclic setting. In that case some authors refer to *near-universal cycles*. Also one can think about the dual problem where we require every $F \in \mathcal{F}$ to be represented at most once rather than at least once. Such sequences are referred to as *packing sequences* [7].

Remark 2.34. In contrast to Definition 2.8 there a few notable differences.

(i) We do not require the combinatorial objects of $\mathcal{F}$ to be of the same "rank".

(ii) For this reason we also allow representations of different lengths. In particular it may well be the case that one representation for a combinatorial object is an infix of another.

(iii) There is no direct requirement on the length of the sequence that is related to the cardinality of $\mathcal{F}$. Indeed there are covering sequences of length m covering more than m combinatorial objects.

(iv) With $\mathcal{F}$-covering sequences we typically are interested in the shortest such sequence or if there are multiple equally short sequences then one or all of them (for the packing variant mentioned in Remark 2.33 we look for the longest of such packing sequences).

(The following will be useful later on.)

Definition 2.35. Let v be a factor of an $\mathcal{F}$-covering sequence $s \in A^*$, i.e. $s = uvw$, where u, w may be ε. Then we may say v is a *partial $\mathcal{F}$-covering sequence* and we define the function

$$\begin{aligned} \operatorname{cvrd}\colon A^* &\to 2^{\mathcal{F}}, \\ v &\mapsto \{F \in \mathcal{F} \mid \exists i \in \{1, \ldots, |v|\}, k \geq 1 : \varphi(v_i \cdots v_{i+k-1}) = F\}, \end{aligned}$$

i.e. for a given partial $\mathcal{F}$-covering sequence v the function $\operatorname{cvrd}\colon A^* \to 2^{\mathcal{F}}$ returns the set of combinatorial objects for which there *exists at least* one corresponding valid block. If $\operatorname{cvrd}(v) \neq \mathcal{F}$ then we call v a *strictly partial $\mathcal{F}$-covering sequence*.

Remark 2.36. Clearly with Definition 2.35 what it means for a sequence s to be an $\mathcal{F}$-covering sequence is equivalent to saying the sequence s satisfies $\mathrm{cvrd}(s) = \mathcal{F}$.

Definition 2.37. An $\mathcal{F}$-covering sequence s is called optimal, if and only if there is no $\mathcal{F}$-covering sequence with length less than $|s|$.

Example 2.38 ([9])**.** As an ever present example in this thesis, consider again the case of permutations on a 3-set, i.e. $\mathcal{F} = S_3$ with the same representation function as in Example 2.19 (i). Then an example for a S_3-covering sequence is given by

$$\texttt{123121321}.$$

In fact, this sequence is optimal, which is straightforward to show by using the same arguments as in Example 2.19 (i) but adapted to sequences rather than cycles.

Superpermutations. The problem of finding optimal covering sequences for permutations of S_n in the representation as used in Example 2.38 is known as the *n-superpermutation problem*. It has become rather famous when an anonymous person on the online forum 4chan improved a lower bound on the length of optimal n-superpermutations.

Theorem 2.39 (Anonymous 4chan poster)**.** *For all $n \geq 1$, every n-superpermutation has length at least*

$$n! + (n-1)! + (n-2)! + n - 3.$$

Proof. For a nicely structured proof the reader is referred to [9]. An archived version of the original discussion by the anonymous poster can be found in [2]. □

One of the most comprehensive resources on superpermutations is provided by Greg Egan, a science fiction novel writer [9], on his website [8], where he also proves the following upper bound.

Theorem 2.40 ([8])**.** *For all $n \geq 4$, there is an n-superpermutation of length at most*

$$n! + (n-1)! + (n-2)! + (n-3)! + n - 3.$$

In practice, for larger n, superpermutations are found by solving an asymmetric traveling salesperson problem [15] similar in nature to the construction we provide in Section 3.3 for a different problem.

3 Covering sequences related to subsets of a set

In this section we want to tackle the problem of finding shortest strings encoding a family of subsets. To this end let us first get the formalities out of the way and define the corresponding representation function as introduced in Definition 2.4.

Definition 3.1. Let A be a finite alphabet and let $P = \mathcal{F} \subseteq 2^A \setminus \{\emptyset\}$, i.e. P consists of a subset of all subsets of A except the empty set. We define the representation function φ_P by

$$\begin{aligned} \varphi_P &\colon A^* \to P, \\ v = v_1 \cdots v_n &\mapsto \begin{cases} F = \{v_1, \ldots, v_n\} & \text{if } |v| = |F|, F \in P \\ \text{undefined} & \text{otherwise.} \end{cases} \end{aligned}$$

The representation function φ_P will be the only one of interest in this section.

Example 3.2. Let $A = \{1, 2, 3\}$ and $P = \{\{1\}, \{3\}, \{1, 2\}, \{1, 2, 3\}\}$. Then

$$\begin{aligned} \varphi_P(1) &= \{1\}, \\ \varphi_P(2) &= \text{undefined}, \\ \varphi_P(21) &= \{1, 2\}, \\ \varphi_P(121) &= \text{undefined}, \\ \varphi_P(312) &= \{1, 2, 3\}. \end{aligned}$$

A shortest P-covering sequence is given by `123`, but also `213`, `312` and `321`.

For the rest of this section we will without loss of generality assume the alphabet A to be of the form $\{1, \ldots, n\}$ for some n.

Oftentimes we consider the problem of finding P-covering sequences for special families of subsets rather than in its full generality.

Definition 3.3. Let P_n be the powerset of $\{1, \ldots, n\}$ without the empty set, i.e.

$$P_n = 2^{\{1,\ldots,n\}} \setminus \{\emptyset\},$$

for $k \in \{1, \ldots, n\}$ denote by P_n^k the subsets of $\{1, \ldots, n\}$ of size k, i.e.

$$P_n^k = \binom{\{1, \ldots, n\}}{k}$$

and finally

$$P_n^{k_1,\ldots,k_r} = P_n^{k_1} \cup \cdots \cup P_n^{k_r}.$$

We also denote by s_n, s_n^k and $s_n^{k_1,\ldots,k_r}$ the length of a *shortest* P_n-, P_n^k- and $P_n^{k_1,\ldots,k_r}$-covering sequence respectively. For the sequence s_n an entry in the *The On-Line Encyclopedia of Integer Sequences* exists [25, A348574].

A useful observation regarding optimal P_n^k-covering sequences is the following.

Lemma 3.4. *Let S be an optimal P_n^k-covering sequence. Then both* $\mathrm{pref}_k(S)$ *as well as* $\mathrm{suff}_k(S)$ *are injective.*

Proof. Assume S is an optimal P_n^k-covering sequence where $\mathrm{pref}_k(S)$ is not injective. Since S_1 is only of relevance to the k-block $S_1 \cdots S_k$, which is not valid by assumption and recall that only k-blocks contribute to covered subsets here, the shortened string $S_2 \cdots S_{|S|}$ must still cover the same subsets as S in contradiction to its optimality. An analogous argument can be made for the k-suffix of S. □

Remark 3.5. Alas we were not able to give a similar statement as in Lemma 3.4 for the case of P_n-covering sequences, although since the inequality $s_n \geq s_n^k$ holds unconditionally Lemma 3.4 will still be useful for lower bounds on P_n-covering sequences as we will see. What we can say however, is that for any optimal P_n-covering sequence S where $n \geq 3$ there is a sequence T isomorphic to S for which $\mathrm{pref}_3(T) = 123$. What we mean exactly by isomorphic we define in the following.

Definition 3.6. Let S, T be strings on the alphabet A. We call S *codirectionally-isomorphic* to T if there exists a permuation $\pi\colon A \to A$ such that

$$S = \pi(T_1) \cdots \pi(T_{|T|}).$$

Further we call S and T *isomorphic* if S is codirectionally-isomorphic to T or $\overleftarrow{T}$. This definition is not in line with [31]. What they call *isomorphic* we choose to call *codirectionally-isomorphic* since we believe that the word "isomorphic" is better suited to describe a relationship between strings where the direction of traversing the strings does not play a role in determining their isomorphicity.

Example 3.7. Let $A = \{1, \ldots, 3\}$. Then the strings $S = 1232$ and $T = 1213$ are isomorphic since $S = \pi(T_4)\pi(T_3)\pi(T_2)\pi(T_1)$ where $\pi = (132)$ but S and T are *not* codirectionally-isomorphic as can been easily verified.

Although theoretical constructions of P_n-covering sequences are not the topic of this thesis we want to mention the best known construction to date [20, 31].

Theorem 3.8 ([20])**.** *For all $n \geq 1$ there exists a P_n-covering sequence $L(n)$. Asymptotically we have*

$$|L(n)| = O\left(2^n\right). \tag{1}$$

In fact we even have that the terms $|L(n)|$ and $2^n \frac{2}{\pi}$ are asymptotically equivalent, i.e. $\lim_{n\to\infty} \frac{|L(n)|\cdot\pi}{2^{n+1}} = 1$, but this will not be relevant to us.

3.1 Computational complexity

The complexity of finding P_n-covering sequences parametrised by n or even $P_n^{k_1,\ldots,k_r}$-covering sequences parametrised by $(n, r, k_1, \ldots, k_r)$ has not yet been explored in the literature to the best our knowledge. However, for the general case parametrised by the set P, it is known that the problem of finding a shortest P-covering sequence is NP-complete. Let us formalise this by rephrasing in terms of a decision problem.

Definition 3.9. Let CovSeq be the problem parametrised by (A, P, m), where $P \subseteq 2^A \setminus \{\emptyset\}$, of deciding whether there exists a P-covering sequence of length m.

Theorem 3.10 ([19])**.** CovSeq *is NP-complete.*

Proof. Clearly the problem is in NP as one can check in polynomial time whether a string $s_1 \cdots s_m$ covers P or not. It remains to show that CovSeq is also NP-hard.

To this end we reduce the problem of finding a Hamiltonian path in an undirected graph, typically called HamiltonPath, to CovSeq.

Let the undirected graph $G = (V, E)$ be an instance to HamiltonPath and without loss of generality assume $V = \{1, \ldots, r\}$. Then we let $A = V \cup E$, $P = \{P_1, \ldots, P_r\}$, where

$$P_i = \{i\} \cup \{\{i, j\} \mid j \in \{1, \ldots, r\}, \{i, j\} \in E\}$$

for $i \in \{1, \ldots, r\}$ and set

$$m = 1 - r + \sum_{i=1}^{r} |P_i| = 1 - r + 2|E| + r = 2|E| + 1.$$

Note that sets of the form $\{i, j\}$ are interpreted as single letters here, since these sets make up part of our alphabet A. The tuple (A, P, m) is now an instance of CovSeq which clearly can be created in polynomial time. For $i \neq j$ we see that

$$P_i \cap P_j = \begin{cases} \{i, j\} & \text{if } \{i, j\} \in E, \\ \emptyset & \text{else.} \end{cases} \tag{2}$$

We now show that a Hamiltonian Path exists if and only if the instance is a feasible one for our decision problem. Assume first that there is a Hamiltonian path $v_1, \ldots, v_r$. Then choose representations $\rho^{(i)} \in \varphi^{-1}(P_i)$ such that

$$\mathrm{suff}_1(\rho^{(v_i)}) = \mathrm{pref}_1(\rho^{(v_{i+1})}) \text{ for } i \in \{1, \ldots, r-1\}.$$

Crucially this is only possible since $\{v_i, v_{i+1}\} \in E$ for all $i \in \{1, \ldots, r-1\}$. Clearly then

$$\rho^{(v_1)} \cdot \rho_2^{(v_2)} \cdots \rho_{|\rho^{(v_2)}|}^{(v_2)} \cdots \rho_2^{(v_r)} \cdots \rho_{|\rho^{(v_r)}|}^{(v_r)}$$

is a P-covering sequence and its length is

$$|\rho^{(v_1)}| + \sum_{i=2}^{r} (|\rho^{(v_i)}| - 1) = \sum_{i=1}^{r} |P_{v_i}| - r + 1$$

which is exactly equal to m and we therefore have a feasible instance to our decision problem. Note that there can be no shorter P-covering sequence either, as Eq. (2) implies that any two represenations $\rho^{(v_i)}$ and $\rho^{(v_j)}$ for $i \neq j$ may only overlap on one letter (in particular cannot be contained in one another) and for the way m

was chosen it also has to be the case that two neighbouring blocks pertaining to different representations share a letter.

Assume now on the other hand that a P-covering sequence s is given. Ignoring all letters corresponding to edges in G and traversing the remaining string left to right gives us a path $v_1, \ldots, v_r$. Since $|s| = m$ and for the aforedescribed observation we must have that the blocks corresponding to v_i and v_{i+1} overlap on exactly one letter, which by Eq. (2) must be the edge $\{v_i, v_{i+1}\}$, in particular $\{v_i, v_{i+1}\} \in E$ for all $i \in \{1, \ldots, r-1\}$. So we have indeed found a Hamiltonian path. □

Example 3.11. Let $G = (V, E)$ be a graph with the set of vertices $V = \{1, \ldots, 5\}$ and edges $E = \{a, b, c, d, e, f\}$ where

$$a = \{1, 2\},\ b = \{2, 3\},\ c = \{2, 4\},\ d = \{3, 4\},\ e = \{4, 5\},\ f = \{1, 4\}.$$

As in the proof of Theorem 3.10 we define the alphabet $A = \{1, \ldots, 5, a, \ldots, f\}$,

$$P = \{\{1, a, f\}, \{2, a, b, c\}, \{3, b, d\}, \{4, c, d, e, f\}, \{5, e\}\}$$

and calculate $m = 2|E| + 1 = 13$. A Hamiltonian path in G now exists if and only if there is a P-covering sequence of length m. Indeed as is readily checked G is Hamiltonian and one such P-covering sequence is given by

`1fa2cb3d4fce5`.

From this and P we may also easily construct the Hamiltonian path as

$$1 \xrightarrow{a} 2 \xrightarrow{b} 3 \xrightarrow{d} 4 \xrightarrow{e} 5.$$

The edges c and f of G are superfluous for this particular path.

3.2 Lower bounds

This section contains results (in particular Propositions 3.25 and 3.28, Theorems 3.20 and 3.29 and Corollaries 3.21, 3.22 and 3.27) that have already been released in the preprint [32]. The following observations were already made by Lipski [20].

Remark 3.12. For a sequence S to be a P_n^k-covering sequence it has to cover $\binom{n}{k}$ subsets of size k. Since a valid k-block can only cover at most one new such set, we must append at least one letter for each set. If we imagine, that we are building the sequence left to right, we notice that the first $k-1$ letters of S cannot contribute a new set on their own. Thus we have

$$s_n^k \geq \binom{n}{k} + k - 1 \tag{3}$$

and since $s_n \geq s_n^k$ for all k, maximising Eq. (3) we get that

$$s_n \geq \binom{n}{\lceil \frac{n}{2} \rceil} + \left\lceil \frac{n}{2} \right\rceil - 1. \tag{4}$$

A more involved observation yields slightly different bounds, although not necessarily always better ones. Notice that a single letter $i \in \{1, \ldots, n\}$ can only be part of at most k many k-blocks, which can cover at most k many k-subsets containing i of which there are exactly $\binom{n-1}{k-1}$. Thus i must occur at least

$$\left\lceil \frac{\binom{n-1}{k-1}}{k} \right\rceil$$

times in S. Since this argument holds for any $i \in \{1, \ldots, n\}$ and thanks to the well-known identity $\frac{n}{k}\binom{n-1}{k-1} = \binom{n}{k}$ we have that

$$s_n^k \geq n \left\lceil \frac{\binom{n}{k}}{n} \right\rceil \tag{5}$$

and as before

$$s_n \geq n \left\lceil \frac{\binom{n}{\lceil \frac{n}{2} \rceil}}{n} \right\rceil . \tag{6}$$

There also exist tight bounds for the special case where $k \in \{2, n-1\}$,

$$s_n^2 = \begin{cases} \binom{n}{2} + 1 & \text{if } n \text{ is odd,} \\ \binom{n}{2} + \frac{n}{2} & \text{if } n \text{ is even,} \end{cases}$$
$$s_n^{n-1} = \binom{n}{n-1} + (n-1) - 1 = 2n - 2.$$

Proofs of these two claims can be found in [31, Propositions 1.5 and 1.6], where the proof of the former equation again uses a Eulerian graph construction akin to the constructions used in the proofs of Theorems 2.18 and 2.28. Since these bounds are known to be tight, the cases where $k \in \{2, n-1\}$ will generally not be considered for the remainder of this thesis.

It turns out that the ideas employed to argue Eq. (3) can be used to prove a lower bound for P_n^k-covering sequences having a certain fixed infix, which to the best of the author's knowledge has not been done before. Before we can state this result we first need to introduce some new notation.

Definition 3.13. Let w be a word. Then let $\mathrm{lip}(w)$ denote the longest injective prefix of w and likewise let $\mathrm{lis}(w)$ denote the longest injective suffix, i.e.

$$\mathrm{lip}(w) = \operatorname*{argmax}_{\substack{v \in \mathrm{pref}(w) \\ v \text{ is injective}}} |v|,$$
$$\mathrm{lis}(w) = \operatorname*{argmax}_{\substack{v \in \mathrm{suff}(w) \\ v \text{ is injective}}} |v|.$$

Theorem 3.14. *Let $n \in \mathbb{N}$, $1 \leq k \leq n$ and let S be a sequence. Let $\hat{S}$ be a valid P_n^k-covering sequence, that contains S as a factor, i.e. $\hat{S} = RST$ for some sequences R and T. Then*

$$|\hat{S}| \geq |S| + A_k + \operatorname{clamp}_0^{|R|}(k - |\operatorname{lip}(ST)| - 1) + \operatorname{clamp}_0^{|T|}(k - |\operatorname{lis}(RS)| - 1), \quad (7)$$

where A_k counts the number of k-subsets that are not yet covered by S. If $A_k > 0$ then at least one of $|R| > \operatorname{clamp}_0(k - |\operatorname{lip}(ST)| - 1)$ or $|T| > \operatorname{clamp}_0(k - |\operatorname{lis}(RS)| - 1)$ holds.

Proof. Think of extending S to $\hat{S} = RST$ until $\hat{S}$ is a P_n^k-covering sequence. It is easy to see then, that certainly $|\hat{S}| \geq |S| + A_k$ must be true, since for any k-subset not yet covered by S we have to extend S by at least one letter. It can however happen that appending or prepending letters to S does not cover a new k-subset.

Consider the case when $\operatorname{lip}(ST) < k - 1$. Assume first for simplicity that R is an infinite string. Prepending a single letter $c = \operatorname{suff}_1(R)$ to S will not cover any new k-subset since $\operatorname{pref}_k(cST)$ is not injective and therefore not a valid k-block. In this way the term $k - |\operatorname{lip}(ST)| - 1$ counts the number of "wasted" letters — letters that do not contribute to covering a k-subset not yet covered by S.

If on the other hand $\operatorname{lip}(ST) \geq k - 1$, or equivalently $k - \operatorname{lip}(ST) - 1 \leq 0$, this effect does not happen and we can combine those two cases to count the number of "wasted" letters as $\operatorname{clamp}_0(k - |\operatorname{lip}(ST)| - 1)$.

Now we need to rid ourselves of our infinitary assumption on R. In case $|R| \geq \operatorname{clamp}_0(k - |\operatorname{lip}(ST)| - 1)$, then we have indeed counted correctly. On the other hand if R is shorter, then we need to account for that and cap the number of "wasted" letters that need to get prepended to S by the length of the string R.

In total this amounts to the term $\operatorname{clamp}_0^{|R|}(k - |\operatorname{lip}(ST)| - 1)$. The same argument holds for appending letters to S on the right hand side from which Eq. (7) follows.

Let us now assume $A_k > 0$. Since there are still subsets left to cover S must be extended in at least one direction. Therefore as in the argument for "wasted" letters above, we must have at least $|R| > \operatorname{clamp}_0(k - |\operatorname{lip}(ST)| - 1)$ or $|T| > \operatorname{clamp}_0(k - |\operatorname{lis}(RS)| - 1)$ so as to cover at least one k-subset not yet covered. □

Remark 3.15. Note that Theorem 3.14 is really providing a bound for $|R| + |T|$. If we use $|\hat{S}| = |RST| = |R| + |S| + |T|$ and cancel the term $|S|$ in Eq. (7) we get

$$|R| + |T| \geq A_k + \operatorname{clamp}_0^{|R|}(k - |\operatorname{lip}(ST)| - 1) + \operatorname{clamp}_0^{|T|}(k - |\operatorname{lis}(RS)| - 1).$$

If $\hat{S} = S$ is a already a valid P_n^k-covering sequence, then this reads as $0 \geq 0$. However, in the more useful scenario that S is not yet a valid P_n^k-covering sequence, we have $A_k > 0$ and as a result of Theorem 3.14 at least one of the terms $|R|$,$|T|$ is big enough such that the respective upper bound in their clamping term does not have an effect. For this reason Theorem 3.14 as well as its corollaries that follow

it are useful even though it seems unnatural at first glance to have an inequality bounding the length of R and T depending on R, S and T (A_k depends on S).

Corollary 3.16. *Let $n \in \mathbb{N}$, $1 \leq k \leq n$ and S be a strictly partial P_n^k-covering sequence. Assume further that one of*

(i) $|\operatorname{lip}(S)| < |S|$ *and* $|\operatorname{lis}(S)| < |S|$,

(ii) $|S| \geq k - 1$,

holds. Then, for all valid P_n^k-covering sequences $\hat{S}$ which contain S as a factor, i.e. we can write $\hat{S} = RST$ for some sequences R and T, we have that

$$|\hat{S}| \geq |S| + A_k + \operatorname{clamp}_0(k - \max(|\operatorname{lip}(S)|, |\operatorname{lis}(S)|) - 1), \tag{8}$$

where A_k counts the number of k-subsets that are not yet covered by S.

Proof. If $|\operatorname{lip}(S)| < |S|$ and $|\operatorname{lis}(S)| < |S|$ then we certainly have that $\operatorname{lip}(ST) = \operatorname{lip}(S)$ and $\operatorname{lis}(S) = \operatorname{lis}(RS)$. Although this is not a valid conclusion when we instead just assume $|S| \geq k - 1$, we still have that

$$\operatorname{clamp}_0^{|R|}(k - |\operatorname{lip}(ST)| - 1) = \operatorname{clamp}_0^{|R|}(k - |\operatorname{lip}(S)| - 1)$$

and

$$\operatorname{clamp}_0^{|T|}(k - |\operatorname{lis}(RS)| - 1) = \operatorname{clamp}_0^{|T|}(k - |\operatorname{lis}(S)| - 1),$$

which is straightforward to see by a simple case analysis. Since $\operatorname{cvrd}(S) \neq P_n^k$ we have $A_k > 0$ and by Theorem 3.14 we get that $|R| > \operatorname{clamp}_0(k - |\operatorname{lip}(S)| - 1)$ or $|T| > \operatorname{clamp}_0(k - |\operatorname{lis}(S)| - 1)$. We can therefore bound the righthand side of Eq. (7) from below by

$$|S| + A_k + \operatorname{clamp}_0(k - |\operatorname{lip}(S)| - 1)$$

or

$$|S| + A_k + \operatorname{clamp}_0(k - |\operatorname{lis}(S)| - 1)$$

respectively, so we certainly have

$$|\hat{S}| \geq |S| + A_k + \min(\operatorname{clamp}_0(k - |\operatorname{lip}(S)| - 1), \operatorname{clamp}_0(k - |\operatorname{lis}(S)| - 1)).$$

Applying some more straightforward manipulations completes the proof. □

Corollary 3.17. *Let $n \in \mathbb{N}$, $1 \leq k \leq n$ and S be a strictly partial P_n^k-covering sequence. For all valid P_n^k-covering sequences $\hat{S}$ for which we have $S \in \operatorname{pref}(\hat{S})$,*

$$|\hat{S}| \geq |S| + A_k + \operatorname{clamp}_0(k - |\operatorname{lis}(S)| - 1) \tag{9}$$

holds, where A_k counts the number of k-subsets that are not yet covered by S.

Proof. Since $S \in \operatorname{pref}(\hat{S})$ we may write $\hat{S} = RST$ where $R = \varepsilon$. Furthermore by the fact that $\operatorname{cvrd}(S) \neq P_n^k$ we have $A_k > 0$ and by Theorem 3.14 we therefore get that $|T| > \operatorname{clamp}_0(k - |\operatorname{lis}(S)| - 1)$. Plugging into Eq. (7) completes the proof. □

Remark 3.18. Let $S = \varepsilon$. Applying Corollary 3.17 recovers the lower bound for P_n^k-covering sequences consistent with Eq. (3) from [20].

Example 3.19.

(i) Assume $n \in \mathbb{N}$ and suppose we want a lower bound on the length of P_n-covering sequences. To that end we rely on lower bounds for P_n^k-covering sequences for which we let $k = \lfloor \frac{n}{2} \rfloor$. Since any P_n-covering sequence $\hat{S}$ must cover the set $\{1, \dots, n\}$ we may put, without loss of generality $S = 1 \cdots n$. As it stands S covers $k+1$ subsets of size k. Since $\operatorname{lip}(S) = \operatorname{lis}(S) = n \geq k-1$ we may apply Corollary 3.16 and for the same reason the clamping term in Eq. (8) vanishes and we have that

$$|\hat{S}| \geq n + \binom{n}{\lfloor \frac{n}{2} \rfloor} - \left(\left\lfloor \frac{n}{2} \right\rfloor + 1 \right),$$

which by the fact that $\binom{n}{\lfloor \frac{n}{2} \rfloor} = \binom{n}{\lceil \frac{n}{2} \rceil}$ and $n = \lfloor \frac{n}{2} \rfloor + \lceil \frac{n}{2} \rceil$ can be equivalently rewritten as

$$|\hat{S}| \geq \binom{n}{\lceil \frac{n}{2} \rceil} + \left\lceil \frac{n}{2} \right\rceil - 1,$$

recovering Eq. (4) from Lipski exactly. Alas, this idea did not materialise in a better lower bound for P_n-covering sequences than ones already known.

Note that it would have been just as easy to use the more technical Theorem 3.14 directly here, as the clamping terms in Eq. (7), which a priori depend on $|R|$ and $|S|$, also vanish independently of R and S.

(ii) Assume $n \geq k \geq 3$ and $\hat{S} = RST$ is a P_n^k-covering sequence where $S = 121$. Since $\operatorname{lip}(S) = \operatorname{lis}(S) = 2 < n$ we may utilise Corollary 3.16. Plugging into Eq. (8) we get

$$|\hat{S}| \geq 3 + \binom{n}{k} + \operatorname{clamp}_0(k - \max(2, 2) - 1) = \binom{n}{k} + k.$$

From Lemma 3.4 we know that $\operatorname{pref}_k(\hat{S})$ as well as $\operatorname{suff}_k(\hat{S})$ must be injective for an optimal P_n^k-covering sequence $\hat{S}$. It is therefore reasonable to assume that $|R|, |S| \geq k - 2$. We can now use the stronger albeit more technical result Theorem 3.14 to get that for such $\hat{S}$

$$|\hat{S}| \geq 3 + \binom{n}{k} + 2\operatorname{clamp}_0(k - 2 - 1) = \binom{n}{k} + 2k - 3,$$

an even greater lower bound applies. This suggests that the "pattern" 121 is not useful in finding optimal P_n^k-covering sequences, which is intuitively apparent. However we are unfortunately not able to draw useful conclusions from such observations as we do not know the true value of s_n^k or for which values of (n, k) the bound from Eq. (3) is sharp for example. Nonetheless, we found these "discoveries" interesting which is why we decided to include them here.

(iii) Again, on the basis of Lemma 3.4 it is intuitive to apply the prefix-based Corollary 3.17 on the string $S = 1 \cdots k$. Since $|\operatorname{lis}(S)| = k$ the clamping term in Eq. (9) vanishes and yet again we produce another proof of Eq. (3).

Now that we have found a sort of generalisation of Eq. (3) we turn our attention to Eq. (5). The theorem that follows can also be understood as an extension of a result by Tabatabai [31, Proposition 1.8], where they give a lower bound on the length of P_n^k-covering sequences starting with a known prefix. We prove a comparable result for P_n^k-covering sequences containing a known infix, which is mostly based on the ideas laid out in the proof of their result. We did adapt our result in such a way that it also works for the case where not all letters $\{1, \ldots, n\}$ are present in the sequence S.

To aid comprehension of the proof of Theorem 3.20 consider the sequence S over the alphabet $\{1, \ldots, 6\}$ shown in the following in black

$$\texttt{1234562}\texttt{134}.$$

Let $n = 6$, $k = 5$ and assume we want to extend S to a P_n^k-covering sequence $\hat{S}$ for which $S \in \operatorname{pref}(\hat{S})$. Consider the letter 6. There are currently two 5-blocks containing 6 covered by the sequence S, namely `23456` and `34562` both covering the subset $\{2, 3, 4, 5, 6\}$. Since $k = 5$ and the letter 6 is in the penultimate position of S we can create new k-blocks containing 6 by appending upto three letters in $\{1, \ldots, n\} \setminus \{6\}$ (for example the possible extension shown in grey). Crucially, appending a fourth letter ($\neq 6$) will not create a new k-block containing 6. We can therefore say that extending S by three letters ($\neq 6$) can at most cover three new k-subsets containing 6, so together with the one k-subset covered by S we can cover at most 4 k-subsets containing 6 without appending the letter 6.

To cover the remaining $\binom{5}{4} - 4 = 1$ subset containing 6 (namely $\{1, 3, 4, 5, 6\}$), we consequently will have to have more occurences of the letter 6 in $\hat{S}$. Since each occurence of the letter 6 can be in at most k k-blocks which can cover at most k-subsets containing 6 we need at least

$$\left\lceil \frac{\binom{5}{4} - 4}{5} \right\rceil = 1$$

additional occurences of 6 and hence at least two occurences of 6 in $\hat{S}$ and hence $\hat{S}$ must be at least $7 + 3 + 1 = 11$ letters long.

Note that this is not true in general for P_6^5-covering sequences as is evidenced by the P_6^5-covering sequence

$$\texttt{1234561234}$$

which is only of length 10. This idea of counting additional occurences of certain letters is the key ingredient for proving Theorem 3.20.

Theorem 3.20. *Let $n \in \mathbb{N}$, $1 \leq k \leq n$ and let S be a sequence. Let $\hat{S}$ be a valid P_n^k-covering sequence, that has S as a factor, i.e. $\hat{S} = RST$ for some sequences R and T. Then $|\hat{S}|$ is at least*

$$|S| + \sum_{i=1}^{n} \left\lceil \frac{a_{k,i} - \operatorname{clamp}_0^{\min(|R|,a_{k,i})}(k - f_i - 1) - \operatorname{clamp}_0^{\min(|T|,a_{k,i})}(k - l_i - 1)}{k} \right\rceil, \tag{10}$$

where $a_{k,i}$ counts the number of k-subsets containing i that are not yet covered by S. The variables f_i and l_i count the number of letters appearing before/after the first/last occurrence of i in S respectively (this number is understood as ∞ if letter i does not occur in S).

Proof. The proof is very similar to the argument in Remark 3.12. We show that each term

$$\left\lceil \frac{a_{k,i} - \operatorname{clamp}_0^{\min(|R|,a_{k,i})}(k - f_i - 1) - \operatorname{clamp}_0^{\min(|T|,a_{k,i})}(k - l_i - 1)}{k} \right\rceil$$

is a lower bound on the number of occurrences of the letter i in RT (i.e. occurences of letter i in $\hat{S}$ without the occurences in S) that are needed in order to cover all k-subsets of $\{1, \ldots, n\}$ containing i.

Think of extending S to $\hat{S} = RST$ until $\hat{S}$ is a P_n^k-covering sequence. Notice, when $f_i < k - 1$ then by definition there are less than $k - 1$ letters preceding the first occurence of i in S. As a consequence we can then create new k-blocks containing i by prepending elements of $\{1, \ldots, n\} \setminus \{i\}$ to S, but not more than either $k - f_i - 1$ or $|R|$, whichever is smaller, since in the latter case those are all the letters we have at our disposal.

On the other hand, if there are at least $k-1$ elements preceding the first occurrence of i or i has not occured yet at all ($l_i = \infty$), then we cannot cover any new k-subsets containing i by prepending letters different from i. Note that in this case we have that $k - f_i - 1 < 0$.

Therefore after potentially prepending some number of elements in $\{1, \ldots, n\} \setminus \{i\}$ to S we can combine the cases just considered and say that the number of k-subsets containing i, that are not yet covered is at least

$$a_{k,i} - \operatorname{clamp}_0^{|R|}(k - f_i - 1) \tag{11}$$

where the natural convention, that $\max(-\infty, 0) = 0$ is assumed. Notice that the term in Eq. (11) could be negative. Since this is a count of things, so non-negative, we may clamp this term as

$$\operatorname{clamp}_0(a_{k,i} - \operatorname{clamp}_0^{|R|}(k - f_i - 1)),$$

which my be equivalently rewritten to

$$a_{k,i} - \operatorname{clamp}_0^{\min(|R|,a_{k,i})}(k - f_i - 1).$$

Following through with this argument for appending letters to S on the right side we have that after potential extension of S by elements in $\{1,\dots,n\}\setminus\{i\}$ in both directions, that at least

$$a_{k,i} - \operatorname{clamp}_0^{\min(|R|,a_{k,i})}(k-f_i-1) - \operatorname{clamp}_0^{\min(|T|,a_{k,i})}(k-l_i-1)$$

k-subsets containing i are not yet covered.

Since each occurrence of i contributes to at most k k-blocks, we need to have at least

$$\left\lceil \frac{a_{k,i} - \operatorname{clamp}_0^{\min(|R|,a_{k,i})}(k-f_i-1) - \operatorname{clamp}_0^{\min(|T|,a_{k,i})}(k-l_i-1)}{k} \right\rceil$$

occurrences of the element i in RT in order for S^* to cover all of the remaining k-subsets containing i. Summing up these lower bounds for all $i \in \{1,\dots,n\}$ gives the desired result. □

For the simple fact that $\min(|R|, a_{k,i}) \le a_{k,i}$ and $\min(|T|, a_{k,i}) \le a_{k,i}$ we get as an immediate consequence the following lower bound independent of R and T.

Corollary 3.21. *Let $n \in \mathbb{N}$, $1 \le k \le n$ and let S be a sequence. Let $\hat{S}$ be a valid P_n^k-covering sequence, that has S as a factor, i.e. $\hat{S} = RST$ for some sequences R and T. Then*

$$|\hat{S}| \ge |S| + \sum_{i=1}^{n} \left\lceil \frac{a_{k,i} - \operatorname{clamp}_0^{a_{k,i}}(k-f_i-1) - \operatorname{clamp}_0^{a_{k,i}}(k-l_i-1)}{k} \right\rceil. \tag{12}$$

where $a_{k,i}$ counts the number of k-subsets containing i that are not yet covered by S. The variables f_i and l_i count the number of letters appearing before/after the first/last occurrence of i in S respectively (this number is understood as ∞ if letter i does not occur in S).

Corollary 3.22. *Let $n \in \mathbb{N}$, $1 \le k \le n$ and let S be a sequence. For all valid P_n^k-covering sequences $\hat{S}$ that have S as a prefix, i.e. $S \in \operatorname{pref}(\hat{S})$,*

$$|\hat{S}| \ge |S| + \sum_{i=1}^{n} \left\lceil \frac{a_{k,i} - \operatorname{clamp}_0^{a_{k,i}}(k-l_i-1)}{k} \right\rceil \tag{13}$$

holds, where $a_{k,i}$ counts the number of k-subsets containing i that are not yet covered by S. The variables f_i and l_i count the number of letters appearing before/after the first/last occurrence of i in S respectively (this number is understood as ∞ if letter i does not occur in S).

Proof. Setting $R = \varepsilon$ in Theorem 3.20 and, as we did for Corollary 3.21, utilising the trivial fact that $\min(|T|, a_{k,i}) \le a_{k,i}$ proves the desired lower bound. □

Remark 3.23. Let $S = \varepsilon$. Applying either of Corollaries 3.21 and 3.22 recovers the lower bound for P_n^k-covering sequences consistent with Eq. (5) from [20].

Example 3.24.

(i) As we did in Example 3.19 suppose we want a lower bound on the length of a P_n-covering sequence $\hat{S}$ and without loss of generality let $S = 1 \cdots n$ be an arbitrary infix of $\hat{S}$. After a straightforward calculation we see that

$$a_{k,i} - \operatorname{clamp}_0^{a_{k,i}}(k - f_i - 1) - \operatorname{clamp}_0^{a_{k,i}}(k - l_i - 1) = \binom{n-1}{k-1} - k.$$

Applying Corollary 3.16 we get that

$$|\hat{S}| \geq n + \sum_{i=1}^{n} \left\lceil \frac{\binom{n-1}{k-1} - k}{k} \right\rceil = n \left(1 + \left\lceil \frac{\binom{n-1}{k-1} - k}{k} \right\rceil \right) = n \left\lceil \frac{\binom{n}{k}}{n} \right\rceil.$$

Substituting $k = \lceil \frac{n}{2} \rceil$ results exactly in Eq. (6). Notice that we actually proved a lower bound for P_n^k-covering sequences containing the infix $1 \cdots n$, which is not a valid assumption in general (see also Remark 3.49). However, since the lower bound we get is the same as Eq. (5) this in some sense suggests that it's not totally unreasonable to assume either, in the sense that the bound does not increase by this assumption.

(ii) Consider the search for an optimal P_8^4-covering sequence $\hat{S}$. Applying Eqs. (3) and (5) with $(n, k) = (8, 4)$ we get that $s_n^k \geq 70 + 3 = 73$ and $s_n^k \geq 8 \cdot 9 = 72$ respectively, so $\hat{S}$ is at least of length 73 but as we are about to show, it has to be even longer.

Thanks to Lemma 3.4 we may assume $S = \operatorname{pref}_4(\hat{S})$ to be injective, so we may without loss of generality assume $S = 1234$. We want to apply Corollary 3.22 on S. For our choice of S we get that

$$a_{41} = a_{42} = a_{43} = a_{44} = 34,$$

since there are $\binom{8-1}{4-1} = 35$ sets of size 4 containing the letter i for a fixed but arbitrary i and we cover exactly one of those sets containing 1, 2, 3 or 4. Likewise we have that

$$a_{45} = a_{46} = a_{47} = a_{48} = 35.$$

Since the letters 5 to 8 do not yet occur in S we set $l_i = \infty$ for $i = 5, \ldots, 8$. Furthermore we have that

$$l_1 = 3, l_2 = 2, l_3 = 1, l_4 = 0.$$

Careful calculation now yields the lower bound

$$|\hat{S}| \geq |S| + \left\lceil \frac{34}{4} \right\rceil + \left\lceil \frac{34-1}{4} \right\rceil + \left\lceil \frac{34-2}{4} \right\rceil + \left\lceil \frac{34-3}{4} \right\rceil + 4 \cdot \left\lceil \frac{35}{4} \right\rceil =$$
$$4 + (9 + 9 + 8 + 8 + 4 \cdot 9) = 74 > 73$$

proving that no P_8^4-covering sequence of length 73 exists and as an immediate corollary no P_8-covering sequence of that length. This is to the best of our knowledge itself a new result in the literature.

Note that we could have also ruled out the existence of P_8-covering sequences of length 73 having the prefixes 1231 or 1232 (1233 can be ruled out immediately from being the prefix of an optimal sequence) without relying on Lemma 3.4. Since any optimal P_8-covering sequence may have the 3-prefix 123 as mentioned in Remark 3.5 it is certainly enough to consider only these prefixes and a straightforward application of Corollary 3.17 shows that in any case the length of corresponding P_8-covering sequences must be at least 74.

The last one of the examples in Example 3.24 can be generalised, which turns out to be a fruitful venture and will be explored in the next subsection.

A new lower bound. As teasered above, Corollary 3.22 can be used to prove a new lower bound, call it β_n^k, for P_n^k-covering sequences at least as good as those provided by Eqs. (3) and (5), which seems to have been overlooked up until now.

Proposition 3.25. *Let S be an optimal P_n^k-covering sequence. Then*

$$|S| \geq \beta_n^k \coloneqq k + (n-k)\left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \sum_{i=1}^{k} \left\lceil \frac{\binom{n}{k}}{n} - \frac{i}{k} \right\rceil. \tag{14}$$

Proof. The proof is a straightforward application of Corollary 3.22 and Lemma 3.4. By the latter we may assume that S has an injective k-prefix. Since all P_n^k-covering sequences are invariant under permutations of their letters, we can without loss of generality assume $S_1 \cdots S_k = 1 \cdots k$. We now want to apply Corollary 3.22 on $S_1 \cdots S_k$. Notice that this string, by design, only has a single (valid) k-block and thus

$$\begin{aligned} a_{k,i} &= \binom{n-1}{k-1} - 1 \quad \text{for } i \leq k, \\ a_{k,i} &= \binom{n-1}{k-1} \qquad\quad \text{for } i > k \end{aligned}$$

in the notation of Corollary 3.22 and for the particular ordering we chose, we get that $l_i = k - i$ for $i \leq k$ and since the letters $\{k+1, \ldots, n\}$ do not appear in $S_1 \cdots S_k$, we have $l_i = \infty$ for those. For the case where $1 \leq i \leq k$ we can now compute

$$\max(k - l_i - 1, 0) = \max(k - (k-i) - 1, 0) = i - 1.$$

For $k < i \leq n$ the above term is of course 0 and thus Equation (13) yields

$$\begin{aligned} |S| &\geq k + \sum_{i=1}^{k} \left\lceil \frac{\binom{n-1}{k-1} - i}{k} \right\rceil + \sum_{i=k+1}^{n} \left\lceil \frac{\binom{n-1}{k-1}}{k} \right\rceil \\ &= k + \sum_{i=1}^{k} \left\lceil \frac{\binom{n-1}{k-1} - i}{k} \right\rceil + (n-k)\left\lceil \frac{\binom{n-1}{k-1}}{k} \right\rceil, \end{aligned} \tag{15}$$

which can be simplified using the well-known identity $\frac{n}{k}\binom{n-1}{k-1} = \binom{n}{k}$ to

$$= k + (n-k)\left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \sum_{i=1}^{k} \left\lceil \frac{\binom{n}{k}}{n} - \frac{i}{k} \right\rceil$$

which is precisely the bound we set out to prove. □

Remark 3.26. We remark that Eq. (14) can also be slightly rewritten as

$$s_n^k \geq \beta_n^k = k - 1 + (n-k+1)\left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \sum_{i=1}^{k-1} \left\lceil \frac{\binom{n}{k}}{n} - \frac{i}{k} \right\rceil. \tag{16}$$

Corollary 3.27.

(i) If $n \mid \binom{n}{k}$*, then Eq.* (14) *reduces to* (3)*, i.e.*

$$s_n^k \geq \beta_n^k = \binom{n}{k} + k - 1.$$

(ii) If $n \nmid \binom{n}{k}$*, then Eq.* (14) *can be written as*

$$s_n^k \geq \beta_n^k = n\left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \left\lceil \frac{kr}{n} \right\rceil - 1, \tag{17}$$

where $r = \binom{n}{k} \bmod n$*.*

Proof.

(i) Since $n \mid \binom{n}{k}$ we have that

$$\left\lceil \frac{\binom{n}{k}}{n} - \frac{i}{k} \right\rceil = \left\lceil \frac{\binom{n}{k}}{n} \right\rceil = \frac{\binom{n}{k}}{n}$$

for all $i \leq k-1$ and by means of Remark 3.26 we get that

$$\begin{aligned} s_n^k &\geq k - 1 + (n-k+1)\left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \sum_{i=1}^{k-1} \left\lceil \frac{\binom{n}{k}}{n} - \frac{i}{k} \right\rceil \\ &= k - 1 + (n-k+1)\frac{\binom{n}{k}}{n} + (k-1)\frac{\binom{n}{k}}{n} \\ &= \binom{n}{k} + k - 1 \end{aligned}$$

which is exactly the inequality obtained in (3).

(ii) Let $r = \binom{n}{k} \bmod n$. Clearly $r > 0$ due to $n \nmid \binom{n}{k}$. Since the index $i \leq k-1$ in the sum of (16) and thus $\frac{i}{k} < 1$, we are interested in when

$$\left\lceil \frac{\binom{n}{k}}{n} - \frac{i}{k} \right\rceil = \left\lceil \frac{\binom{n}{k}}{n} \right\rceil - 1.$$

Clearly this is the case exactly when $\frac{r}{n} - \frac{i}{k} \leq 0$ which can be equivalently rewritten as $i \geq \frac{kr}{n}$ and since $i \in \mathbb{N}$ this is again equivalent to $i \geq \lceil \frac{kr}{n} \rceil$. Applying our newfound knowledge to (16) we get

$$\begin{aligned} s_n^k &\geq k - 1 + (n-k+1)\left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \sum_{i=1}^{k-1} \left\lceil \frac{\binom{n}{k}}{n} - \frac{i}{k} \right\rceil \\ &= k - 1 + (n-k+1)\left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \sum_{i=1}^{k-1} \left\lceil \frac{\binom{n}{k}}{n} \right\rceil - \left(k - \left\lceil \frac{kr}{n} \right\rceil \right) \\ &= n\left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \left\lceil \frac{kr}{n} \right\rceil - 1, \end{aligned}$$

which is what we wanted to show. □

It is a natural question to ask when exactly (or even if) the lower bound β_n^k compares favorably to the simpler lower bounds in Eqs. (3) and (5). In order to answer this question precisely we first need the following intermediate result.

Proposition 3.28. *If $n \nmid \binom{n}{k}$, then we have that the lower bound in Eq.* (17) *is strictly better than the one in Eq.* (3)*, i.e.*

$$n\left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \left\lceil \frac{kr}{n} \right\rceil - 1 > \binom{n}{k} + k - 1,$$

where $r = \binom{n}{k} \bmod n$.

Proof. Let $r = \binom{n}{k} \bmod n$. By using the fact that $\left\lceil \frac{\binom{n}{k}}{n} \right\rceil = \frac{\binom{n}{k}+n-r}{n}$ we can rewrite (17) slightly differently as

$$s_n^k \geq \binom{n}{k} + \left\lceil \frac{kr}{n} \right\rceil + n - (r+1). \tag{18}$$

Therefore our proof reduces to showing that

$$\binom{n}{k} + \left\lceil \frac{kr}{n} \right\rceil + n - (r+1) > \binom{n}{k} + k - 1,$$

which is equivalent to showing

$$n - k > r - \left\lceil \frac{kr}{n} \right\rceil.$$

However this is true since $r < n$ and the chain of inequalities

$$r - \left\lceil \frac{kr}{n} \right\rceil \leq r - \frac{kr}{n} = \frac{r}{n}(n-k) < n - k.$$

□

Theorem 3.29. *The following inequality*

$$\beta_n^k \geq \max\left(\binom{n}{k} + k - 1, n \left\lceil \frac{\binom{n}{k}}{n} \right\rceil \right)$$

holds, with equality if and only if $kr \leq n$ *where* $r = \binom{n}{k} \bmod n$.

Proof.

(i) Assume first that $n \mid \binom{n}{k}$. Then we may apply the case of Corollary 3.27 (i) and we get that

$$\beta_n^k = \binom{n}{k} + k - 1 = \max\left(\binom{n}{k} + k - 1, n \left\lceil \frac{\binom{n}{k}}{n} \right\rceil \right),$$

indeed with equality since

$$n \left\lceil \frac{\binom{n}{k}}{n} \right\rceil = \binom{n}{k}$$

for $r = \binom{n}{k} \bmod n = 0$ and therefore $kr = 0 \leq n$.

(ii) Now assume $n \nmid \binom{n}{k}$ and hence $r = \binom{n}{k} \bmod n > 0$. By the case of Corollary 3.27 (ii) we have that

$$\beta_n^k = n \left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \left\lceil \frac{kr}{n} \right\rceil - 1.$$

Let us further assume that $kr > n$. Then by Proposition 3.28 we see that

$$\beta_n^k > \binom{n}{k} + k - 1.$$

and clearly we also have that

$$\beta_n^k > n \left\lceil \frac{\binom{n}{k}}{n} \right\rceil.$$

On the other hand if $1 \leq kr \leq n$ then again by means of Proposition 3.28 and the fact that $\lceil \frac{kr}{n} \rceil - 1 = 0$ we have that

$$\beta_n^k = n \left\lceil \frac{\binom{n}{k}}{n} \right\rceil > \binom{n}{k} + k - 1,$$

completing the proof. □

Remark 3.30. To assess the quality of the bound β_n^k in comparison to Eqs. (3) and (5) we have calculated some values which can be seen in Table 1.

n	k	r	β_n^k	Eq. (3)	(5)
6	4	3	19	18	18
8	4	6	74	73	72
8	6	4	34	33	32
9	6	3	91	89	90
10	8	5	53	52	50
12	8	3	505	502	504
12	9	4	230	228	228
12	10	6	76	75	72
14	4	7	1009	1004	1008
14	6	7	3012	3008	3010
14	8	7	3013	3010	3010
14	10	7	1012	1010	1008
14	12	7	103	102	98
15	6	10	5013	5010	5010
15	9	10	5015	5013	5010
15	10	3	3016	3012	3015
15	12	5	468	466	465
16	4	12	1826	1823	1824
16	6	8	8018	8013	8016
16	8	6	12882	12877	12880

Table 1: The smallest 20 values of (n, k) (lexicographically) for which $rk > n$ and respective lower bounds.

Now we want to apply our newfound knowledge of bounds for P_n^k-covering sequences to bounds of P_n-covering sequences. In light of the fact that $s_n \geq s_n^k$ for all $k \in \{1, \dots, n\}$ we get as an immediate consequence of Proposition 3.25 the following.

Corollary 3.31. *Let $n \in \mathbb{N}$. Then the following inequality concerning the minimum length of a P_n-covering sequence holds:*

$$s_n \geq \beta_n := \beta_n^{\lceil \frac{n}{2} \rceil} = \left\lceil \frac{n}{2} \right\rceil + \left(n - \left\lceil \frac{n}{2} \right\rceil\right) \left\lceil \frac{\binom{n}{\lceil \frac{n}{2} \rceil}}{n} \right\rceil + \sum_{i=1}^{\lceil \frac{n}{2} \rceil} \left\lceil \frac{\binom{n}{\lceil \frac{n}{2} \rceil}}{n} - \frac{i}{\lceil \frac{n}{2} \rceil} \right\rceil. \quad (19)$$

Lemma 3.32. *If $n \equiv 1$ (2) then $n \mid \binom{n}{\lceil \frac{n}{2} \rceil}$.*

Proof. Since n is odd, we may write $n = 2k - 1$ for some $k \in \mathbb{N}$. Then $\lceil \frac{n}{2} \rceil = k$. It is easy to check that $\binom{2k-1}{k} = \frac{1}{2}\binom{2k}{k}$ holds, so our problem reduces to the question whether $2(2k-1) \mid \binom{2k}{k}$. Indeed we can write

$$\begin{aligned} \binom{2k}{k} &= \frac{2k \cdot (2k-1) \cdots (k+1)}{k \cdot (k-1)!} \\ &= \frac{2k \cdot (2k-1)(2k-2) \cdots (k+1) \cdot k}{k \cdot k \cdot (k-1)!} \\ &= 2(2k-1)\frac{1}{k}\binom{2k-2}{k-1} \end{aligned}$$

but since $1 \cdot (2k-1) - 2 \cdot k = -1$ by the well-known Bézout's lemma we get that $\gcd(2k-1, k) = 1$ and because $\binom{2k}{k} \in \mathbb{N}$ we must therefore have that $k \mid \binom{2k-2}{k-1}$ and thus $2(2k-1) \mid \binom{2k}{k}$, which is what we wanted to show. □

Remark 3.33. In particular for odd n because of Lemma 3.32 the bound we get from Eq. (19) is the same as in Eq. (3) as we have seen in Corollary 3.27 (i). Indeed even for many even $n \in \mathbb{N}$ the bound is no better than the established bounds, but we may reuse Theorem 3.29 to give a characterisation.

Corollary 3.34. *The following inequality holds*

$$\beta_n \geq \max\left(\binom{n}{\lceil \frac{n}{2} \rceil} + k - 1, n\left\lceil \frac{\binom{n}{\lceil \frac{n}{2} \rceil}}{n} \right\rceil\right),$$

with equality if and only if $r \leq 2$ *where* $r = \binom{n}{\lceil \frac{n}{2} \rceil} \bmod n$.

Proof. The case when

$$n \mid \binom{n}{\lceil \frac{n}{2} \rceil}$$

is analogous to that case in Theorem 3.29. For

$$n \nmid \binom{n}{\lceil \frac{n}{2} \rceil}$$

note that by the contrapositive of Lemma 3.32 we have that $n \equiv 0 \ (2)$ and thus $\lceil \frac{n}{2} \rceil = \frac{n}{2}$. The condition $\lceil \frac{n}{2} \rceil r \leq n$ therefore reduces to $r \leq 2$ and again follows directly from the proof of that case in Theorem 3.29. □

Remark 3.35. By Corollary 3.34 we can thus conclude that our formula gives a better bound than the previously known formulas exactly when $r \geq 3$. Naturally there exists a sequence in *The On-Line Encyclopedia of Integer Sequences*, namely [25, A042970] defined as

$$r_n = \binom{n}{\lceil \frac{n}{2} \rceil} \bmod n$$

of which the first few values starting at $n = 1$ are

$$0, 0, 0, 2, 0, 2, 0, 6, 0, 2, 0, 0, 0, 2, 0, 6, 0, 2, 0, 16, 0, 2, 0, 4, 0, 2, 0, 20, 0, 0, 0, 6, \ldots$$

Some values for β_n can be seen in Table 2 for n where $r_n \geq 3$.

n	r_n	β_n	Eq. (4)	Eq. (6)
8	6	74	73	72
16	6	12882	12877	12880
20	16	184767	184765	184760
24	4	2704177	2704167	2704176
28	20	40116617	40116613	40116608
32	6	601080418	601080405	601080416
36	24	9075135323	9075135317	9075135312
40	20	137846528849	137846528839	137846528840
42	6	538257874478	538257874460	538257874476
44	28	2104098963749	2104098963741	2104098963736

Table 2: The smallest 10 values of n for which $r_n \geq 3$ and respective lower bounds.

Universal cycles. We have introduced P_n^k-covering sequences as strings covering subsets of a set, but we could likewise instead also ask for the existence of universal cycles in the same context, i.e. with respect to the representation function φ_P as introduced in Definition 3.1.

It is easy to see that for a P_n^k-universal cycle to possibly exist, we must have that

$$k \mid \binom{n-1}{k-1}. \tag{20}$$

Recall that in the context of universal cycles, every k-block must represent exactly one of the $\binom{n}{k}$ subsets of $\{1, \ldots, n\}$ and uniquely so, i.e. there may not be more than one k-block representing the same subset. Let $a \in \{1, \ldots, n\}$. Every occurence of a in a potential universal cycle is contained in exactly k k-blocks. Since there are $\binom{n-1}{k-1}$ k-subsets of $\{1, \ldots, n\}$ containing a, we must indeed have that the modular condition Eq. (20) holds [17]. In fact, we can also see this in a slightly more complicated way taking a detour via P_n^k-covering sequences. Suppose there was a universal cycle u for $n, k \in \mathbb{N}$ where $k \nmid \binom{n-1}{k-1}$. By the well-known identity $\frac{n}{k}\binom{n-1}{k-1} = \binom{n}{k}$ we have that $k \nmid \binom{n-1}{k-1}$ holds if and only if $n \nmid \binom{n}{k}$ holds. Therefore we may apply Proposition 3.28 and conclude that any P_n^k-covering sequence must be at least of length $\binom{n}{k} + k$, but since we assumed there was a universal cycle u, which has length $\binom{n}{k}$ we see by Remark 2.13 that $u \cdot u_1 \ldots u_{k-1}$ is a P_n^k-covering sequence of length $\binom{n}{k} + k - 1$ in contradiction to our earlier finding.

Up until recently it has been a long-standing conjecture whether Eq. (20) could also be sufficient for the existence of universal cycles. Chung et al. have even offered a 100$ reward for its resolution in 1989 [5].

Conjecture 3.36 ([5])**.** P_n^k-universal cycles exist provided $k \mid \binom{n-1}{k-1}$ and $n \geq n_0(k)$.

Partial progress was made in and around that time, e.g. by Jackson proving the conjecture for $k = 3$ [17] and Hurlbert validated the conjecture for $k \in \{3, 4, 6\}$ if n and k are coprime [16].

Remark 3.37. If n and k are coprime then $k \mid \binom{n-1}{k-1}$ holds as we can again use the identity $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$, where both sides are integers however $\frac{n}{k}$ is not assuming $n \geq k \geq 2$ [16]. The converse however is not true in general as for instance $10 \mid \binom{10}{4} = 210$ but clearly $\gcd(10, 4) = 2 \neq 1$.

In a new paper as recently as 2020 aforementioned Conjecture 3.36 was actually proven in its full form by [11].

Theorem 3.38 ([11])**.** *P_n^k-universal cycles exist provided $k \mid \binom{n-1}{k-1}$ and $n \geq n_0(k)$.*

The proof uses probabilistic methods and a generalisation of Eulerian circuits to hypergraphs and is beyond the scope of this thesis. Unfortunately we were not able to do any reverse engineering (if at all possible) on the bound function $n_0 \colon \mathbb{N} \to \mathbb{N}$ and the bounds that Hurlbert [16] determined are only relevant when $\gcd(n, k) = 1$.

For the special case where $k = 3$ we do however have the following result by [17].

Theorem 3.39 ([17])**.** *P_n^3-universal cycles exist provided $3 \mid \binom{n-1}{2}$ and $n \geq 8$.*

Remark 3.40. In fact Theorem 3.39 applies to even smaller values of n. It is easy to see that

```
12345674527635716327415624135217346
```

is a P_7^3-universal cycle of length $\binom{7}{3} = 35$ (found using BeamSearch). For $n = 6$ the modulo condition is not satisfied since $\binom{5}{2} = 10$. In the case of $n = 5$ the modulo condition is fulfilled however no P_5^3-universal cycle can exist, since a shortest P_5^3-covering sequence is of length $13 > 10 + 2$,

```
1234515241352
```

as computed by IDDFS. This is also consistent with the calculation $s_5^3 = 13$ by [31]. We may therefore say that Theorem 3.39 is already valid for all $n \geq 6$ which is the best that is possible. We suspect that this has been overlooked up until now for the lack of computational resources available at the time [17] was published.

Naturally Theorems 3.38 and 3.39 and Remark 3.40 translate into results about P_n^k-universal sequences in a straightforward manner.

Corollary 3.41. *P_n^k-universal sequences exist provided $k \mid \binom{n-1}{k-1}$ and $n \geq n_0(k)$.*

Proof. Follows immediately from Theorem 3.38 and Remark 2.13. □

Corollary 3.42. *P_n^3-universal sequences exist provided $3 \mid \binom{n-1}{2}$ and $n \geq 6$.*

Proof. Follows immediately from Theorem 3.39 and Remark 3.40. □

By bounds on P_n^k-covering sequences we also get the following.

Corollary 3.43. *For $k \mid \binom{n-1}{k-1}$ and $n \geq n_0(k)$ we have that*

$$s_n^k = \binom{n}{k} + k - 1.$$

Proof. Let n, k be such that $k \mid \binom{n-1}{k-1}$ and $n \geq n_0(k)$. Then there exists a P_n^k-universal sequence by Corollary 3.41 and hence $s_n^k \leq \binom{n}{k} + k - 1$, but thanks to Eq. (3) we have $s_n^k \geq \binom{n}{k} + k - 1$ conluding the proof. □

3.3 A reduction to the Generalised Traveling Salesperson Problem

This section contains results (in particular Definition 3.45, Proposition 3.47 and Table 3) that have already been released in the preprint [32]. Akin to how super-permutations can be found by solving or approximating an asymmetric traveling salesperson problem like described by [15], we figured it is possible to find P_n^k-covering sequences by instead searching low-cost tours in a so-called generalised traveling salesperson problem. This problem was first studied seriously in his dissertation by [24]. Since this problem is a generalisation of the classic traveling salesperson problem it is NP-hard. We will give a definition here by [30].

Definition 3.44 ([30], Generalised Traveling Salesperson Problem)**.** Given a complete weighted graph $G = (V, E, w)$ on n vertices and a partition of V into m sets $V_1, \ldots, V_m$, where $V_i \cap V_j = \emptyset$ for all $i \neq j$ and $\cup_{i=1}^m V_i = V$, find a cycle in G that contains exactly one vertex from each set V_i, $i \in \{1, \ldots, m\}$ and has minimum total weight.

The reduction of the problem of finding P_n^k-covering sequences to solving a generalised traveling salesperson problem that we came up with is the following.

Definition 3.45 (Model)**.** Let

$$V = \{1 \cdots k\} \cup \bigcup_{F \in \binom{\{1,\ldots,n\}}{k} \setminus \{1,\ldots,k\}} \varphi^{-1}(F)$$

where φ is the representation function as defined in Definition 3.1. Let $m = \binom{n}{k}$ and denote by $F_2, \ldots, F_m$ any enumeration of the sets

$$\binom{\{1, \ldots, n\}}{k} \setminus \{1, \ldots, k\}.$$

Then we define a partition $V_1, \ldots, V_m$ by

$$V_1 = \{1 \cdots k\},$$
$$V_i = \varphi^{-1}(F_i) \text{ for } i \in \{2, \ldots, m\}.$$

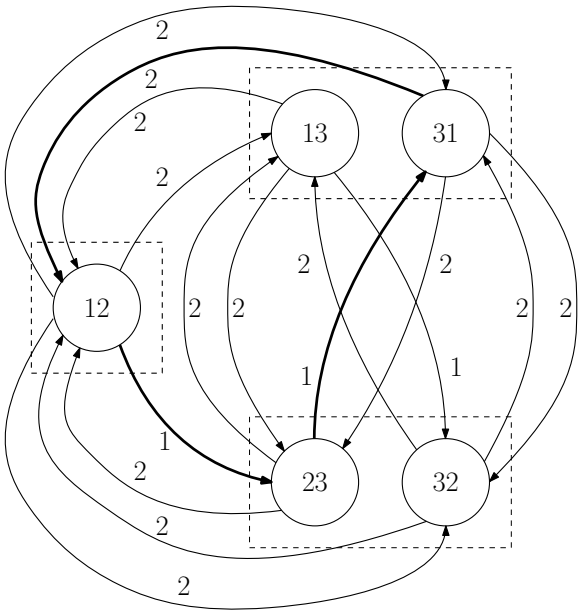

Figure 4: Generalised traveling salesperson model for $k = 2$ and $n = 3$ (∞-edges left out). The optimal path $12 \to 23 \to 31$ is indicated by fat arrows.

Further define the weight function $w\colon V \times V \to \mathbb{N} \cup \{\infty\}$,

$$w(u,v) \coloneqq \begin{cases} l(u,v) & \text{if } v \neq 1\cdots k \text{ and } \varphi(u) \neq \varphi(v), \\ k & \text{if } v = 1\cdots k \neq u, \\ \infty & \text{else,} \end{cases}$$

where $l\colon V \times V \to \{0,\dots,k\}$,

$$l(u,v) \coloneqq k - \max_{\substack{t\in\{1,\dots,k\} \\ \operatorname{suff}_t(u)=\operatorname{pref}_t(v)}} t.$$

Here the empty maximum is interpreted as 0.

The function l simply measures how many letters have to be appended to u in order to include v as a suffix.

Example 3.46. Let $n = 6$ and $k = 3$. Then

$$\begin{aligned} l(145,145) &= 0 \qquad \texttt{145}, \\ l(156,561) &= 1 \qquad \texttt{156}\underline{\texttt{1}}, \\ l(316,624) &= 2 \qquad \texttt{31}\underline{\texttt{62}}\texttt{4}, \\ l(523,513) &= 3 \qquad \texttt{523}\underline{\texttt{513}}. \end{aligned}$$

For better visualisation have a look at Fig. 4, where we show the model for P_3^2-covering sequences as a small example. The partition is indicated by the dashed rectangles and the optimal covering sequence `1231` is emphasised by the fat arrows.

Clearly not every P_n^k-covering sequence can be obtained from a solution to the (n,k)-instance of our model. For example consider the intentionally inflated P_3^2-covering sequence $s = 1232323232323231$. Clearly there is no cycle in Fig. 4 belonging to s. Thankfully though, we are interested in shortest sequences for which we have the following result.

Proposition 3.47. *Every optimal P_n^k-covering sequence can be reconstructed from a solution to the (n,k)-GTSP model given in Definition 3.45.*

Proof. Let $s = s_1 \cdots s_m$ be an optimal P_n^k-covering sequence which by relabeling of the letters we can assume to start with $1 \cdots k$. Assume further that

$$F_1, \ldots, F_{\binom{n}{k}}$$

is any enumeration of the elements of P_n^k. Define the first occurrences

$$i_j = \min\{h : \{s_h, \ldots, s_{h+k-1}\} = F_j\},\ j = 1, \ldots, \tbinom{n}{k},$$

which we can without loss of generality assume to satisfy

$$i_1 < \cdots < i_{\binom{n}{k}}.$$

Clearly then

$$i_1 = 1 \text{ and } i_{\binom{n}{k}} = m - k + 1$$

for the optimality of s. Now if $i_{j+1} - i_j \leq k$ holds for all j, then s is certainly in the solution space of the (n,k)-instance of the GTSP model as outlined in Definition 3.45. Assume therefore there exists j such that $i_{j+1} - i_j > k$. Then the substring $s_{i_j+k} \cdots s_{i_{j+1}-1}$ does not cover a subset which is not already covered by $s_1 \cdots s_{i_j+k-1} s_{i_{j+1}} \cdots s_m$ in contradiction to the optimality of s. □

Assuming we have a solver for the aforementioned model reconstructing the P_n^k-covering sequences from valid GTSP tours from is straightforward. For the particular solver we used, the output consists of vertices given as an array $u_{\text{unrotated}}$, which might not have our starting vertex $1 \cdots k$ as the first element. Therefore we first rotate $u_{\text{unrotated}}$ such that $1 \cdots k$ is in the first position. Then we initialise a string S to $1 \cdots k$ and iteratively extend S to the right by elements of $u_{\text{unrotated}}$, where the overlap to the previous element is omitted. A more formal description is provided as Algorithm 1.

Example 3.48. In the following $u_{\text{unrotated}}$ is a feasible tour in the graph corresponding to P_5^3-covering sequences. We rotate it such that 123 comes first,

$$u_{\text{unrotated}} = (152, 524, 123, 235, 354, 541, 124, 243, 431, 315),$$
$$u = (123, 235, 354, 541, 124, 243, 431, 315, 152, 524).$$

Then set $S \leftarrow 123$ and append 5, since 123 and 235 overlap on 23 and therefore now $S = 1235$. In particular we need to append $\text{suff}_{l(u_{i-1}, u_i)}(u_i)$ to S assuming

the adequate suffix of u_{i-1} has already been appended to S. We continue until we have processed all elements of u and get

$$S = \texttt{1235412431524}.$$

Algorithm 1 Reconstructing P_n^k-covering sequences from solver output

Require: $u; k \in \mathbb{N}$

```
1: function RECONSTRUCT(u, k)
2:     while u[1] ≠ 1...k do
3:         rotateLeftByOne(u)
4:     S ← u[1]
5:     for i ∈ {2, ..., |u|} do
6:         S ← S · suff_{l(u_{i-1}, u_i)}(u_i)
7:     return S
```

Remark 3.49.

(i) The way the weight function w is designed is such that the total weight of a valid solution to the problem is exactly the length of the P_n^k-covering sequence it corresponds to. This is in part achieved by setting $w(u, 1 \cdots k) = k$ for $u \neq 1 \cdots k$, essentially mandating that the solution starts with $1 \cdots k$ and the cost for those k letters is in fact k.

(ii) This also has the advantage, that $k!$ degrees of freedom are eliminated from the search space since the subset $\{1, \ldots, k\}$ is forced to be represented by $1 \cdots k$ exactly. Additionally further degrees of freedom are removed since this string acts as the start of the sequence as already mentioned.

(iii) One aspect of this approach is that the k-prefix as well as the k-suffix of P_n^k-covering sequences constructed in this way are always injective. This is however not a limitation but rather a prerequisite to finding optimal P_n^k-covering sequences as was shown in Lemma 3.4.

(iv) It should be remarked that eliminating the special treatment for $1 \cdots k$ in the definition of the weight function $w \colon V \times V \to \mathbb{N} \cup \{\infty\}$ would allow one to search for (near)-universal cycles instead (see Remark 2.33).

(v) Another consideration that crossed our minds on removing more degrees of freedom was to enforce the infix $1 \cdots n$. This would further reduce the amount of sets, i.e. the size of the model, however we then analyzed solutions and found that to our surprise P_n^k-covering sequences need not have an injective factor of length n as the following optimal P_7^3-covering sequence shows:

$$\texttt{12345136751273142753267436524617456 12}.$$

To see more easily that this solution, which was found using our BEAM-SEARCH procedure with $\beta = 10000$, does not cover $\{1, \ldots, 7\}$ we refer the

reader to the solution checker provided in Appendix D and its subsequent example output. It should also be noted however, that many other optimal P_7^3-covering sequences do indeed contain such a factor of length n, so a possible conjecture could claim that there always exist optimal P_n^k-covering sequences that contain a valid n-block.

Technical aspects and results. For its ease of use, state-of-the-art design and modifiability we decided to use the excellent software GLNS.jl which is provided free of charge and under an open source software license on GitHub[4]. Their solver is based on a novel large neighborhood search algorithm that operates by iteratively constructing and destroying tours combined with simulated annealing to accept/reject new tours found through the iterative process [30]. The problem formulation has to be provided in GTSPLIB format, which is a text-based format derived from the format used to supply traveling salesperson problem instances. This posed some problems for us, as our instances quickly explode in size as n and k grow and a text-based format is not very efficient at storing numerical weights. We therefore decided to modify their code which is granted under the license they supply and this modified code can also be found on GitHub[5]. Specifically we added support for loading instance files that are either gzipped or compressed using ZStd[6] compression. As an example: This way we were able to reduce an uncompressed instance file for finding P_{13}^5-covering sequences which is about 177 Gb in size to a bit over 44 Mb, which is an impressive compression ratio of 4119 by using ZStd on compression level 19. Note that this is indeed a huge instance with 154321 vertices and we suspect much more than what the software was originally designed to handle. We also mention that for vertices u, v connected by an edge of weight ∞ instead the value of $l(u, v)$ was used except for loops where the weight was set to 999 as was done in usage examples of GLNS.jl. We also considered using GLKH [13] but it proved difficult to use. With GLKH, there are many parameters to tune and it was also not as easy to modify as GLNS.jl so that it can accept compressed instance files. We did do tests on small instances where the file size was not an issue, but this yielded subpar results for us as compared to GLNS.jl, so we decided not to pursue it any further.

For our results which can be found in Table 3 we used the following parameters:

$$\text{mode} = \text{default}, \quad \text{trials} = 100000, \quad \text{time\_limit} = \left\lceil \frac{\#\text{vertices}}{154321} \cdot 3600 \cdot 48 \right\rceil.$$

The trials are set arbitrarily high such that they have no bearing on the results and only the maximum time is relevant, which was chosen in such a way that the biggest instance gets 48 hours of CPU time while the other instances scale linearly in the number of vertices. This value is listed in the rightmost column of Table 3. The values in column Δ describe the deviation of results from their greatest lower bounds we found in percent.

[4] `https://github.com/stephenlsmith/GLNS.jl`
[5] `https://github.com/whiterock/GLNS.jl`
[6] `http://facebook.github.io/zstd/`

n	k	Result	$\max(\beta_n^k, \text{IDDFS}^\dagger)$	$\Delta(\%)$	Runtime (min)	Budget (min)
8	3	**58**	†**58**	0%	1	7
8	4	**74**	**74**	0%	31	31
8	5	65	†**60**	8.33%	124	124
8	6	38	†**37**	2.70%	363	363
9	3	**90**	**90**	0%	1	10
9	4	136	**129**	5.43%	57	57
9	5	144	130	10.77%	280	280
9	6	102	91	12.09%	1116	1116
10	3	**122**	**122**	0%	1	14
10	4	229	213	7.51%	94	94
10	5	294	260	13.08%	563	563
10	6	256	215	19.07%	2809	2809
11	3	**167**	**167**	0%	11	19
11	4	359	**333**	7.81%	148	148
11	5	535	466	14.81%	1033	1033
12	3	**228**	**228**	0%	6	25
12	4	549	504	8.93%	222	222
12	5	930	796	16.83%	1772	1772
13	3	296	**288**	2.78%	32	32
13	4	790	**718**	10.03%	320	320
13	5	1498	1291	16.03%	2880	2880

Table 3: Results of running GLNS.jl on our model for the specified amount of time and using `mode=default`. Lower bounds are displayed in the fourth column, where numbers marked by † are bounds found by Algorithm 2, IDDFS (see also Table 4). Bold numbers indicate optimal values / sharp bounds. For some optimality claims one has to borrow the upper bound from [21].

3.4 Iterative deepening depth-first search

When searching for optimal $P_n^{k_1,\ldots,k_r}$-covering sequences we can employ a sort of branch and bound algorithm utilising Corollaries 3.17 and 3.22. More specifically we implemented a depth-first search utilising a stack for storing partial sequences in order to rule out the existence of solutions shorter than a target length, which then is iteratively increased starting from some lower bound (see Section 3.2). This

form of a tree search has been referred to as iterative deepening depth-first search in the literature [18]. It is hardly feasible in terms of computational resources to find new optimal sequences this way, however Algorithm 2 is useful for improving lower bounds in some instances. E.g. we were able to confirm a result mentioned in [31]. For simplicity Algorithm 2 is described for P_n-covering sequences only but it can be adapted with little modification to work for $P_n^{k_1,\ldots,k_r}$-covering sequences.

Algorithm 2 Iterative deepening depth-first search

Require: $n \in \mathbb{N}, n \geq 3$

```
1:  function DFS(p, s)
2:      B ← [p]
3:      R ← []
4:      while B ≠ ∅ do
5:          S ← pop(B)
6:          if S is feasible then
7:              push(R, S)
8:              continue
9:          for c ∈ (S \ suff_1(S)) ∪ min({1, ..., n} \ S) do
10:             S_new ← S · c
11:             if bound1(S_new) > s then continue        ▷ due to Corollary 3.17
12:             if bound2(S_new) > s then continue        ▷ due to Corollary 3.22
13:             push(B, S_new)
14:     return R

15: function IDDFS(n)
16:     p ← 123                                           ▷ Initial partial solution
17:     s ← max(bound1(p), bound2(p))                     ▷ Initial search depth
18:     while true do
19:         R ← DFS(p, s)
20:         if R ≠ ∅ then return R                        ▷ Solutions found of length s
21:         s ← s + 1                                     ▷ No solutions exist of length s
```

The algorithm is initiated with a call to IDDFS which starts off with a single partial solution sequence $p = 123$. This is possible without loss of generality since we do not care to find all (potentially isomorphic) solutions but are rather more interested in disproving the existence of solutions not greater than a certain length s, see also Remark 3.5. The natural number s is the initial search depth and is given to DFS as a parameter as well as the initial solution p. We invoke DFS with ever increasing search depths until a solution is found. Note that in practice we may not find solutions in a reasonable amount of time, e.g. for $n \geq 8$, but the algorithm is still useful even if terminated early since we are able to see with added debug information which lengths for potential solutions can be ruled out. The function DFS is a slight modification of the well known depth-first search algorithm. We

initialise a stack B with the initial solution p which has been handed over. For each sequence S in B we construct some possible extensions for it and check whether the resulting sequence $S \cdot c$ has the potential to become a P_n-covering sequence $\hat{S}$ of length less than or equal to s by applying Corollaries 3.17 and 3.22.

We do not construct all possible extensions as some are non-sensical, e.g. we do not repeat the last letter of S and we only use one letter not used so far in S to avoid generating codirectionally-isomorphic sequences (see Definition 3.6). These optimisations were also mentioned in [31] and can be found in the pseudocode of Algorithm 2 on line 9. Note that there may still be sequences S, T generated such that S is isomorphic to the reverse of T, which makes them isomorphic by Definition 3.6. We found it an interesting endeavour to explore the effect of this optimisation in greater detail by counting the number of non-codirectionally-isomorphic strings (see Section 3.5).

Algorithm 2 may then terminate in two ways: Either we prune all partial solutions at some point, at which point we have proven no solutions may exist of length less than or equal to s and return this info as an empty array to IDDFS or we find feasible solutions return all of those to the caller. Note that we can also terminate and return immediately upon discovering a first solution as there can be many (even non-isomorphic) solutions for a given problem and we are more interested in assessing the length of solutions with this algorithm than to enumerate all of them.

Results & Implementation details. Algorithm 2 is similar to the algorithm described by Tabatabai [31, p. 18], where they used a recursive procedure while we opted for a stack-based approach. Another difference is that they did not use the pruning rule described in Corollary 3.17 as far as we are aware. Nevertheless the basic idea is the same and it therefore makes sense to compare their algorithm and results to ours. They state that their algorithm generated

$$126704677$$

sequences to prove $s_7 > 39$. Our algorithm whose implementational and hardware-related specifications are given in Remark 3.50 used

$$65744476$$

sequences proving the same result which we can hereby confirm. The discrepancy cannot be fully explained by the additional use of the pruning based on Corollary 3.17. Omitting this pruning rule our algorithm generated 90503330 sequences. So the disparity more likely stems from some nuances in how generated sequences are counted. In particular when counting sequences before applying the optimisations reducing isomorphic sequences in Algorithm 2 on line 9 we count

$$126704719.$$

generated sequences, which is indeed very close to the result they got.

We ran Algorithm 2 on all $n \in \{8, \ldots, 16\}$ and $k \in \{3, \ldots, n-2\}$ with a hard time limit of 24 hours. The results can be seen in Table 4.

n	k	IDDFS	β_n^k	Time (sec)
8	3	exactly 58[7]	58	7115
8	5	exactly 60[8]	60	1
8	6	exactly 37[9]	34	590
9	7	at least 47	42	33297
10	8	at least 56	53	14727
11	9	at least 67	63	11967
12	10	at least 78	76	3834
13	11	at least 91	88	2644
14	12	at least 104	103	752
15	13	at least 120	117	45991
16	14	at least 135	134	6414

Table 4: Results of running IDDFS. Only results that either found an optimal P_n^k-covering sequence or improved upon the bound β_n^k within 24 hours are shown here. Tests were done for all $n \in \{8, \ldots, 16\}$ and $k \in \{3, \ldots, n-2\}$.

Somewhat surprisingly for us this was sufficient to find three optimal solutions to the problems of finding P_8^3-,P_8^5- and P_8^6-covering sequences. For most other tests however the observed runtime complexity, which increases superlinearily in β_n^k, is just not good enough to produce fruitful results except for the special case where $k = n-2$, where the problem appears constrained enough and the lower bounds are not too large and hence the depth reached during the tree search is manageable.

Remark 3.50. Algorithm 2 was implemented in Julia and the results were achieved on a server cluster consisting of AMD EPYC 7402, 2.80GHz CPUs. A memory limit of 2 GB was also imposed on the running programs, however this is just a technicality in this case as the algorithm has very low memory requirements. In fact the required memory is of order $O((\beta_n^k)^2)$ as is straightforward to see.

[7] 12345678563874682763485273184271823617541352615246185374 12

[8] 12345678345618745268731548237148236518276415273461823576 1234

[9] 12345678231567418234571836472158643 21

3.5 On the number of non-codirectionally-isomorphic strings

The set of values iterated over in the for-loop of Algorithm 2 on line 9 greatly reduces the search space of the algorithm as also mentioned already in [31]. In this intermezzo we set out to quantify this reduction. Consider first the most naive approach thinkable for searching for $P_n^{k_1,\dots,k_r}$-covering sequences, i.e. considering all strings $\{1,\dots,n\}^*$. Obviously then there are n^m strings of length m.

A simple improvement that can be done, is to require strings to start with 1 since any permutation of the letters of a $P_n^{k_1,\dots,k_r}$-covering is itself one and we can further exclude strings that have consecutive repeated letters. We therefore have only one string of length 1, namely 1 and $n-1$ choices for letters to be recurringly appended yielding a total of $(n-1)^{m-1}$ strings of length m.

This trivial optimisation is already a considerable improvement as

$$\lim_{m\to\infty} \frac{(n-1)^{m-1}}{n^m} = 0.$$

However, Algorithm 2 reduces the search space even more which we formalise now.

Definition 3.51. Let $n \in \mathbb{N}$. We define a set Λ_n recursively by

(i) $1 \in \Lambda_n$,

(ii) $s = s_1 \cdots s_m \in \Lambda_n \iff s_m \leq \max(s_1,\dots,s_{m-1}) + 1$ and $s_m \neq s_{m-1}$.

Remark 3.52. The set Λ_n is precisely the mathematical description for the strings considered in Algorithm 2 due to line 9 if this algorithm did not do any other pruning. As it turns out we are able to establish a connection to a similar set of strings studied in [23] allowing us to reuse their results to draw conclusions for Λ_n.

Example 3.53. We have that

$$\Lambda_4 \cap \{1,\dots,4\}^4 = \{1212, 1213, 1231, 1232, 1234\},$$

which are only 5 strings as compared to all 256 possible strings of length 4 on $\{1,\dots,4\}$. This is also much less than the $3^3 = 27$ strings of length 4 that start with 1 and have no consecutive repeats.

Remark 3.54. Note that as an immediate consequence of Definition 3.51 we see that any two strings $s, t \in \Lambda_n$ can never be isomorphic to each other assuming reversing is prohibited. It therefore makes sense to call Λ_n the set of *non-codirectionally-isomorphic* strings.

Our aim now is to count the number of strings in Λ_n of length m.

Definition 3.55 ([23])**.** Let Λ be a set of strings on the alphabet A. Then we call

$$\rho_\Lambda(m) = |\Lambda \cap A^m|$$

the *density* of Λ.

Let $k, m \geq 1$ and denote by $\lambda_{m,k}$ the number of strings in

$$\overline{\Lambda}_k^{(m)} \coloneqq \{s \in \Lambda_k \cap \{1, \ldots, k\}^m \mid \max(s_1, \ldots, s_m) = k\},$$

i.e. strings in Λ_k with length m and whose maximal value of a letter is k. Then by differentiating between the cases where either a new maximum is attained or an existing letter is reused and by Definition 3.51 we can formulate the recursion

$$\lambda_{m+1,k} = (k-1)\lambda_{m,k} + \lambda_{m,k-1}, \tag{21}$$

where $1 \leq k \leq m$. Notice that there is only one string in $\overline{\Lambda}_m^{(m)}$ for $m \geq 1$, namely the string $1 \cdots m$ and due to the requirement of no consecutive repeats there are no strings in the sets $\overline{\Lambda}_1^{(m)}$ for $m > 1$, which translates to the initial conditions

$$\lambda_{m,m} = 1 \text{ for } m \geq 1 \text{ and } \lambda_{m,1} = 0 \text{ for } m > 1.$$

Since the sets $\overline{\Lambda}_k^{(m)}$ for $k = 1, \ldots, n$ form a partition of $\Lambda_n \cap \{1, \ldots, n\}^m$, i.e.

$$\Lambda_n \cap \{1, \ldots, n\}^m = \overline{\Lambda}_1^{(m)} \cup \cdots \cup \overline{\Lambda}_n^{(m)}$$

and $\overline{\Lambda}_i^{(m)} \cap \overline{\Lambda}_j^{(m)} = \emptyset$ for $i \neq j$ we may now calculate $\rho_{\Lambda_n}(m)$ as

$$\rho_{\Lambda_n}(m) = \sum_{k=1}^{n} \lambda_{m,k}. \tag{22}$$

The eagly-eyed reader may have noticed that Eq. (21) is very much reminiscent of the well-known recursion for the *Stirling numbers of the second kind* $S_{m,k}$,

$$S_{m+1,k} = kS_{m,k} + S_{m,k-1}, \tag{23}$$

where $0 < k \leq m$ with the initial conditions

$$S_{m,m} = 1 \text{ for } m \geq 0 \text{ and } S_{m,0} = 0 \text{ for } m > 0.$$

If we rewrite Eq. (23) by shifting m, k down by one respectively, i.e.

$$S_{m,k-1} = (k-1)S_{m-1,k-1} + S_{m-1,k-2},$$

we see that setting $\lambda_{m,k} = S_{m-1,k-1}$ fulfills Eq. (21). We may therefore rewrite the density $\rho_{\Lambda_n}(m)$ for $m > 1$ as

$$\rho_{\Lambda_n}(m) = \sum_{k=1}^{n} S_{m-1,k-1} = \sum_{k=1}^{n-1} S_{m-1,k},$$

where the last equality is due to an index shift and because $S_{m-1,0} = 0$ for $m > 1$.

Moreira and Reis study the following set of strings in their paper [23],

$$L_n = \{a_1 \cdots a_m \in \{1, \ldots, n\} \mid \forall i \in \{1, \ldots, m\} : a_i \leq \max\{a_1, \ldots, a_{m-1}\} + 1\}$$

for which they calculate the density

$$\rho_{L_n}(m) = \sum_{k=1}^{n} S_{m,k}.$$

We can therefore see that

$$\rho_{\Lambda_n}(m) = \rho_{L_{n-1}}(m-1). \tag{24}$$

They also prove the following.

Theorem 3.56 ([23])**.** *For all $n \geq 1$,*

$$\lim_{m\to\infty} \frac{\rho_{L_n}(m)}{\rho_{(\{1,\ldots,n\})^*}(m)} = \frac{1}{n!}. \tag{25}$$

We can now prove a similar result to Theorem 3.56 useful in our context.

Corollary 3.57. *For all $n \geq 1$,*

$$\lim_{m\to\infty} \frac{\rho_{\Lambda_n}(m)}{\rho_{E_n}(m)} = \frac{1}{(n-1)!},$$

where E_n is the set of strings on $\{1,\ldots,n\}$, starting with 1 and no consecutive repeated letters occur.

Proof. As an immediate consequence of Theorem 3.56, the relation in Eq. (24) and the equality $\rho_{E_n}(m) = \rho_{(\{1,\ldots,n-1\})^*}(m-1)$ we get that

$$\lim_{m\to\infty} \frac{\rho_{\Lambda_n}(m)}{\rho_{E_n}(m)} = \lim_{m\to\infty} \frac{\rho_{L_{n-1}}(m-1)}{\rho_{(\{1,\ldots,n-1\})^*}(m-1)} = \frac{1}{(n-1)!}.$$

□

This should illustrate that it is indeed much better to consider only strings in Λ_n when searching for $P_n^{k_1,\ldots,k_r}$-covering sequences. Even though Corollary 3.57 is only an asymptotic result, we have observed that the ratio $\frac{\rho_{\Lambda_n}(m)}{\rho_{E_n}(m)}$ tends to converge quite rapidly already for small m, which is showcased by Table 5.

$n\backslash m$	3	4	5	6	7	8	9	∞
3	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000
4	0.22222	0.18519	0.17284	0.16872	0.16735	0.16690	0.16674	0.16667
5	0.12500	0.07812	0.05859	0.04980	0.04565	0.04364	0.04265	0.04167
6	0.08000	0.04000	0.02400	0.01664	0.01293	0.01094	0.00984	0.00833
7	0.05556	0.02315	0.01157	0.00669	0.00435	0.00313	0.00245	0.00139
8	0.04082	0.01458	0.00625	0.00309	0.00173	0.00106	0.00072	0.00020
9	0.03125	0.00977	0.00366	0.00159	0.00077	0.00042	0.00025	0.00002

Table 5: Some values for the ratio $\frac{\rho_{\Lambda_n}(m)}{\rho_{E_n}(m)}$ for different n and m. The column "∞" indicates the asymptotic value of this ratio according to Corollary 3.57.

The algorithm we will introduce in Section 3.7 works in a breadth-first kind-of fashion in stark contrast to Algorithm 2. It turns out this has a certain advantage as this allows for a new form of pruning, which we will explore now in more detail.

3.6 Pruning admissible relations

When constructing P-covering sequences in a fashion reminiscent of breadth-first search it frequently happens that there are strings S and T of the same length and for which for all $w \in A^*$

$$\operatorname{cvrd}(S \cdot w) = \operatorname{cvrd}(T \cdot w)$$

holds. This allows us to prune one of the two strings from our pool of partial solutions, since they behave identically when extending them to the right. We could say that S and T are *equivalent* in this sense. In fact there might be many strings that are equivalent and it suffices to erase all but one representative without sacrificing on the explorative nature of the algorithm. Let us formalise this behaviour.

Definition 3.58. Let A be an alphabet and $\sim$ an equivalence relation on A^*. We call $\sim$ *pruning admissible* if whenever $u \sim v$ then $|u| = |v|$ and for all $w \in A^*$ we have that

$$\operatorname{cvrd}(u \cdot w) = \operatorname{cvrd}(v \cdot w). \tag{26}$$

Ideally we would like to have a pruning admissible equivalence relation that has as few equivalence classes as possible. That would allow for more pruning as it is sufficient to keep only one representative for every equivalence class. The following definition allows to draw a comparison between different such relations.

Definition 3.59. Let X be a set and R and Q relations on X. Then we say that R *refines* Q (or the other way around Q is *coarser* than R) if the following holds: Whenever for $u, v \in X$ we have uRv then also uQv.

Remark 3.60. The idea now is that we would like to work with a rather coarse pruning admissible equivalence relation, since then there are larger and therefore fewer equivalence classes. Clearly any pruning admissible relation $\sim$ must refine the equivalence relation $\sim_{\mathrm{C}}$ defined by

$$u \sim_{\mathrm{C}} v \iff \mathrm{cvrd}(u) = \mathrm{cvrd}(v) \text{ and } |u| = |v|,$$

which in itself is *not* pruning admissible. Concepts such as these also appear in formal language and automata theory, in particular the requirement in Eq. (26) is reminiscent of *right congruences* [14], which we introduce in the following Definition 3.61. The overarching connection should be made clear by Proposition 3.62.

Definition 3.61. Let A be an alphabet and $\equiv$ an equivalence relation on A^*. We say $\equiv$ is a *right congruence* if whenever $u \equiv v$ then for all $w \in A^*$ also $uw \equiv vw$.

Proposition 3.62. *Let A be an alphabet and $\sim$ an equivalence relation on A^*. Then $\sim$ is pruning admissible if the following conditions are fulfilled:*

(i) $\sim$ is a right congruence,

(ii) $\sim$ is a refinement of $\sim_C$.

Proof. Assume $u \sim v$. Since $\sim$ is a right congruence we have that $u \cdot w \sim v \cdot w$ for any $w \in A^*$. In this case, since $\sim$ is a refinement of $\sim_{\mathrm{C}}$, we also have that $\mathrm{cvrd}(u \cdot w) = \mathrm{cvrd}(v \cdot w)$ which makes $\sim$ pruning admissible. □

Remark 3.63. As already pointed out in Remark 3.60, condition (ii) in Proposition 3.62 is certainly necessary, but the converse of Proposition 3.62 is not true in general, as pruning admissible relations need not be right congruences. All pruning admissible relations we will be considering here are however right congruences.

The following equivalence relation was communicated to us through email by Alexander Healy [12], which was actually the initial catalyst that led to the creation of this subsection.

Definition 3.64 ([12])**.** Consider an equivalence relation on partial P_n-covering sequences S, T defined by

$$S \sim_{\mathrm{H}} T \iff \mathrm{cvrd}(S) = \mathrm{cvrd}(T) \text{ and } \mathrm{suff}_{n-1}(S) = \mathrm{suff}_{n-1}(T) \text{ and } |S| = |T|,$$

i.e. S and T cover the same subsets and they agree on their last $n-1$ letters.

By definition $\sim_{\mathrm{H}}$ is an equivalence relation that refines $\sim_{\mathrm{C}}$. To see that $\sim_{\mathrm{H}}$ is also a right congruence, assume $S \sim_{\mathrm{H}} T$. Append now the same letter x to the right of S and T respectively. Since we are considering subsets of an n-set, only the last n letters of $S \cdot x$ and $T \cdot x$ are relevant, but as we appended the same letter and by assumption S and T already agree on the last $n-1$ letters and S, T cover the same subsets, we must also have that $\mathrm{cvrd}(S \cdot x) = \mathrm{cvrd}(T \cdot x)$ and therefore also $S \cdot x \sim_{\mathrm{H}} T \cdot x$. An induction argument shows that, indeed, $\sim_{\mathrm{H}}$ is a right congruence.

Remark 3.65. It is easy to see that the basic idea in the definition of $\sim_{\mathrm{H}}$ can be improved upon significantly. Let S, T be strictly partial P-covering sequences and assume $\operatorname{cvrd}(S) = \operatorname{cvrd}(T)$. Then for the same argument as in Definition 3.64 it suffices to compare $\operatorname{suff}_{b-1}(S)$ with $\operatorname{suff}_{b-1}(T)$ for equality where

$$b = \max_{B \in P \setminus \operatorname{cvrd}(S)} |B|.$$

In particular for $P = P_n$, an equivalence relation defined in this way is much more coarse than $\sim_{\mathrm{H}}$. Note that since S is presumed to be strictly partial we must have $b \geq 1$ and therefore $\operatorname{suff}_{b-1}(\operatorname{lis}(S))$ is well-defined. Alternatively one could drop this requirement and instead clamp $b - 1$ to 0 making this well-defined for any partial P-covering sequence. In some cases we can improve this even further with another idea.

Definition 3.66. Let S, T be partial P-covering sequences. We define

$$S \sim_{\mathrm{LIS}} T :\iff \operatorname{cvrd}(S) = \operatorname{cvrd}(T) \text{ and } \operatorname{lis}(S) = \operatorname{lis}(T) \text{ and } |S| = |T|.$$

By definition $\sim_{\mathrm{LIS}}$ is an equivalence relation that refines $\sim_{\mathrm{C}}$. To see that this is a right congruence, we follow the same arguments as outlined in Corollary 3.17.

Remark 3.67. Notice that $\sim_{\mathrm{H}}$ and $\sim_{\mathrm{LIS}}$ are incomparable with regards to the refinement order, however combining the ideas outlined in Remark 3.65 with Definition 3.66 will result in an even coarser pruning admissible equivalence relation $\sim_{\mathrm{W}}$. To that end we now introduce the concept of the longest relevant suffix.

Definition 3.68. Let S be a strictly partial P-covering sequence. Then we call $\operatorname{lrs}(S)$ the *longest relevant suffix* of S defined by

$$\operatorname{lrs}(S) \coloneqq \operatorname{suff}_{b-1}(\operatorname{lis}(S)),$$

where

$$b = \max_{B \in P \setminus \operatorname{cvrd}(S)} |B|.$$

Remark 3.69. We require S to be a strictly partial P-covering sequence here for the simple reason that in the case where S is a valid P-covering sequence already, the accompanying constant b would be ill-defined.

With this definition out of the way, we can define a new pruning admissible equivalence relation as follows.

Definition 3.70. Let S, T be strictly partial P-covering sequences. We define

$$S \sim_{\mathrm{W}} T :\iff \operatorname{cvrd}(S) = \operatorname{cvrd}(T) \text{ and } \operatorname{lrs}(S) = \operatorname{lrs}(T) \text{ and } |S| = |T|.$$

Example 3.71. Consider the strictly partial P_4-covering sequences

$$S = 1231312 \quad \text{and} \quad T = 1232312.$$

It can be readily checked that they both cover the same set of sets, namely

$$\{\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}.$$

The sets still remaining to be covered by both S and T are therefore

$$\{\{4\},\{1,4\},\{2,4\},\{3,4\},\{1,2,4\},\{1,3,4\},\{2,3,4\},\{1,2,3,4\}\},$$

by which we see that for both S and T the value $b = \max_{B\in P\setminus \mathrm{cvrd}(S)} |B| = 4$. Since $\mathrm{lis}(S) = \mathrm{lis}(T) = 312$ and for the aforementioned value of b we also have $\mathrm{lrs}(S) = \mathrm{lrs}(T) = 312$ and therefore S and T satisfy the equivalence $S \sim_{\mathrm{W}} T$.

Remark 3.72. To give a definition for the longest relevant suffix of a strictly partial P-covering sequence w that is more in line with Definition 3.13 we may alternatively define it as

$$\mathrm{lrs}(w) \coloneqq \operatorname*{argmax}_{\substack{v\in\mathrm{suff}(w)\\ v \text{ is injective}\\ |v|<b}} |v|,$$

where $b = \max_{B\in P\setminus \mathrm{cvrd}(w)} |B|$.

Now the eagle-eyed reader may have already complained at this point that there is a lack of potential in requiring that two sequences cover the same subsets when we defined pruning admissability for equivalence relations. Requiring only set inclusion in lieu of equality leads us naturally to the concept of preorders whose definition is recalled here for the readers' convenience.

Definition 3.73. Let X be a set and R a relation on X. Then we call R a preorder if it reflexive and transitive, i.e. it satisfies:

(i) (Reflexivity): xRx for all $x \in X$,

(ii) (Transitivity): If xRy and yRz then also xRz for all $x, y, z \in X$.

Definition 3.74. Let A be an alphabet and $\lesssim$ a preorder on A^*. We call $\lesssim$ *pruning admissible* if whenever $u \lesssim v$ then $|u| = |v|$ and for all $w \in A^*$ we have that

$$\mathrm{cvrd}(u \cdot w) \subseteq \mathrm{cvrd}(v \cdot w).$$

Note that it would also make perfect sense in this context to require $|u| \geq |v|$ (to be interpreted as "shorter and more sets covered is better") allowing for even coarser pruning admissible preorders. However for our use case u and v will always be of the same length and being consistent with the definition of pruning admissible equivalence relations we also require $|u| = |v|$ here.

Changing the set equality to set inclusion in Definition 3.70 we get the following.

Definition 3.75. Let S, T be strictly partial P-covering sequences. We define

$$S \lesssim_{\mathrm{W}} T :\Longleftrightarrow \mathrm{cvrd}(S) \subseteq \mathrm{cvrd}(T) \text{ and } \mathrm{lrs}(S) = \mathrm{lrs}(T) \text{ and } |S| = |T|.$$

Example 3.76. As soon as on layer 7 when running beam search for $n = 8$ (see Section 3.7) we encounter several (strictly) partial P_n-covering sequences that can be pruned by Definition 3.75. For example we have that

$$1232142 \lesssim_{\mathrm{W}} 1234142,$$

since their longest relevant suffixes coincide and it can be readily checked that indeed $\mathrm{cvrd}(1232142) \subseteq \mathrm{cvrd}(1234142)$ holds as

$$\begin{aligned} &\{\ldots, \{1,2\}, \{1,4\}, \{2,3\}, \{2,4\}, \quad\quad, \{1,2,3\}, \{1,2,4\}, \quad\quad, \quad\quad, \ldots\}, \\ \subseteq &\{\ldots, \{1,2\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}, \{1,2,3\}, \{1,2,4\}, \{1,3,4\}, \{2,3,4\}, \ldots\}. \end{aligned}$$

Proposition 3.77. *Let A be an alphabet and $\lesssim$ a preorder on A^*. Then $\lesssim$ is pruning admissible if the following conditions are fulfilled:*

(i) $\lesssim$ has the property that whenever $u \lesssim v$ then for all $w \in A^*$ also $uw \lesssim vw$.

(ii) $\lesssim$ is a refinement of $\lesssim_C$ where $u \lesssim_C v \iff \mathrm{cvrd}(u) \subseteq \mathrm{cvrd}(v)$ and $|u| = |v|$.

Proof. Assume $u \lesssim v$. Since (i) holds we have that $u \cdot w \lesssim v \cdot w$ for any $w \in A^*$. In this case, since $\lesssim$ is a refinement of $\lesssim_{\mathrm{C}}$, we also have that $\mathrm{cvrd}(u \cdot w) \subseteq \mathrm{cvrd}(v \cdot w)$ and certainly $|u| = |v|$. Indeed $\lesssim$ is pruning admissible. □

In light of the work already done, it is easy to see that $\lesssim_{\mathrm{W}}$ is indeed a pruning admissible preorder.

The following theorem is a well-known fact and recalled here for convenience.

Theorem 3.78. *Let $\lesssim$ be a preorder on a set X. Then the relation $\sim$ defined by*

$$x \sim y \iff x \lesssim y \text{ and } y \lesssim x$$

is an equivalence relation and we call $\sim$ the induced equivalence relation *by $\lesssim$.*

For the sake of legibility we will always associate with $\lesssim$ its induced equivalence relation as $\sim$ and more generally $\lesssim_w$ with $\sim_w$ for some word w. For example the induced equivalence relations for $\lesssim_{\mathrm{C}}$ and $\lesssim_{\mathrm{W}}$ are $\sim_{\mathrm{C}}$ and $\sim_{\mathrm{W}}$ respectively. This convention is consistent with this section and for the remainder of this thesis.

Definition 3.79. Let $\lesssim$ and $\lesssim_{\mathrm{E}}$ be preorders on a set X. Then we call $\lesssim_{\mathrm{E}}$ an *extension* of $\lesssim$ if and only if $\lesssim$ refines $\lesssim_{\mathrm{E}}$.

Definition 3.80. We call a preorder $\lesssim$ on a set X *total* if for all $x, y \in X$ we have

$$x \lesssim y \text{ or } y \lesssim x.$$

Definition 3.81. Let $\lesssim$ be a preorder on X. Then x is maximal with respect to $(X, \lesssim)$ if, whenever $x \lesssim y$ we also have that $y \lesssim x$ (or equivalently $x \sim y$).

Finding maximal elements of a partial order or preorder is in general of quadratic runtime, however several algorithms that perform well in practice exist [4]. We decided not to implement any of them for reasons of complexity and scope of this thesis. Exploiting the specific way sets are encoded in our implementation (as bitsets), we stumbled upon a very simple albeit clever alternative: Algorithm 3. It was inspired by [1], although the algorithm they describe appears erroneous as well as its acclaimed runtime of $O(n \log n)$, however their idea of using a total preorder extension has merit and the algorithm can be fixed giving $O(nm)$ runtime, where m is the number of maximal elements, which is especially nice for its algorithmic simplicity.

Remark 3.82. An element x is therefore not maximal if there exists y such that $x \lesssim y$ and $y \not\lesssim x$. Equivalently we may say that x is not maximal if there exists y such that $x \lesssim y$ and $x \not\sim y$ (see Theorem 3.78). For us it will be useful to say, that whenever there exists y such that $x \lesssim y$ then either $x \sim y$ or x is not maximal.

Lemma 3.83. *Let $\lesssim$ be a preorder on a set X. Suppose $x, y \in X$ and $x \sim y$. Then, with respect to $(X, \lesssim)$, x is maximal if and only if y is maximal.*

Proof. Assume x is maximal and $x \sim y$. Suppose $z \in X$ such that $y \lesssim z$. Since $x \sim y$ and therefore $x \lesssim y$, by transitivity we get $x \lesssim z$. Due to maximality of x we now must have that $z \lesssim x$ which also implies $z \lesssim y$ and hence y must be maximal as well. The other implication follows analogously. □

Algorithm 3 Finding maximal elements of a preorder

Require: List of elements X sorted descendingly according to a total preorder extension $\lesssim_{\mathrm{E}}$ of $\lesssim$

```
1:  function FindMaximals(X)
2:      M ← [X[1]]
3:      for i ← 2 to |X| do
4:          ismax ← true
5:          for m in M do
6:              if X[i] ≲ m then
7:                  ismax ← false
8:                  break
9:          if ismax then
10:             push(M, X[i])
11:     return M
```

Theorem 3.84. *Let $\lesssim$ and $\lesssim_E$ be preorders on a finite array X. Assume further*

(i) $\lesssim$ is a refinement of $\lesssim_E$

(ii) The induced equivalence relations $\sim$ and $\sim_E$ coincide

(iii) $\lesssim_E$ *is total*

(iv) X *is sorted sorted descendingly according to* $\lesssim_E$

Then Algorithm 3 outputs exactly one representative for every class of equivalent maximal elements in X *with respect to* $\lesssim$.

Proof. We use a loop invariant. We claim, that for each i the elements in M consist of exactly one representative for each class of maximal elements for which there is a representative in $X[1:i]$ with respect to $\lesssim$. For the remainder of this proof, whenever we speak of maximality we refer to maximality with respect to $\lesssim$.

Initially the array M only contains the element $X[1]$. Since we assume X to be sorted descendingly according to $\lesssim_{\mathrm{E}}$ and for the totality of $\lesssim_{\mathrm{E}}$ we see that for all $x \in X$ we have $x \lesssim_{\mathrm{E}} X[1]$. Now assume $X[1] \lesssim x$ for some x. By the fact that $\lesssim_{\mathrm{E}}$ is a refinement of $\lesssim$ we have that $X[1] \lesssim_{\mathrm{E}} x$ and therefore $X[1] \sim_{\mathrm{E}} x$, but as $\sim_{\mathrm{E}}$ and $\sim$ coincide by assumption we get that indeed $X[1] \sim x$ which shows that $X[1]$ is maximal.

Assume the algorithm works correctly upto iteration $i-1$. Clearly if the inner loop determines for $X[i]$ that ismax = false then there is an element $m \in M$ such that $X[i] \lesssim m$ and therefore either $X[i] \sim m$ and thus $X[i]$ is another representative for the element m, which is maximal, since the induction hypothesis assures M consists of maximal elements only, or $X[i]$ is not maximal as outlined in Remark 3.82. Notice that in the former case, since m is maximal so is $X[i]$ by Lemma 3.83. In both cases the algorithm correctly discards $X[i]$.

On the contrary assume the inner loop ends with ismax = true. We want to prove that $X[i]$ is maximal. Suppose there exists $j \in \{1, \dots, |X|\}$ such that $X[i] \lesssim X[j]$. We consider two cases.

(i) Assume first $j > i$. For the sorting of the array X we have that $X[j] \lesssim_{\mathrm{E}} X[i]$. Now there are again two possibilites to distinguish. Either we have $X[j] \sim_{\mathrm{E}} X[i]$ and therefore also $X[j] \sim X[i]$ showing that $X[i]$ is maximal; or we have that $X[j] \not\sim_{\mathrm{E}} X[i]$ which implies by definition (of induced equivalence relations) that $X[i] \not\lesssim_{\mathrm{E}} X[j]$ and by the contapositive of the refinement property we get that $X[i] \not\lesssim X[j]$. But this is a contradiction to our assumption.

(ii) Now assume $j < i$. Since the inner loop ended with ismax = true and thus did not find any m in $M \subseteq X[1:i-1]$ such that $X[i] \lesssim m$ we may further suppose that j is such that $X[j] \in X[1:i-1] \setminus M$. By the induction hypothesis we know that $X[j]$ was discarded in a previous iteration hence there must exist $m \in M$ such that $X[j] \lesssim m$. By transitivity of $\lesssim$ we also have that $X[i] \lesssim m$, but this is not possible since then the inner loop would have had to finish with ismax = false in contradiction to our assumption. □

Remark 3.85. It is clear by the pseudocode of Algorithm 3 that its runtime is of order $O(nm)$ where m denotes the number of equivalence classes of maximal

elements of X with respect to $\lesssim$ and n is the size of the input X. For our application, which will be detailed in the following Section 3.7 we will however have the situation that there are lots of maximal elements in the input. If we presume $m > pn$ for some $p \in (0, 1]$ then the runtime is unfortunately quadratic, i.e. no better than $O(n^2)$.

The following will be the most important application of Algorithm 3 for us.

Example 3.86. Let X be a list of partial P_n-covering sequences that all share the same longest relevant suffix and are all of the same length. Assume we want to find all maximal elements (or rather one representative for every maximal element) of X with respect to the preorder $\lesssim_{\mathrm{W}}$ (see Definition 3.75). Since all elements of X share the same longest relevant suffix, it is true to say that

$$S \lesssim_{\mathrm{W}} T \iff \mathrm{cvrd}(S) \subseteq \mathrm{cvrd}(T)$$

holds for all $S, T \in X$. In order to use Algorithm 3 we seek a total preorder extending $\lesssim_{\mathrm{W}}|_X$. For any partial P_n-covering sequence S, $\mathrm{cvrd}(S)$ is a subset of $2^{\{1,\ldots,n\}}$. We may therefore encode $\mathrm{cvrd}(S)$ as a bitstring, i.e. a word over $A = \{0, 1\}$, of length 2^n.

For example, suppose we are in the setting of P_3-covering sequences. Then the string $S = 123$ covers the subsets $\{1\}$, $\{2\}$, $\{3\}$, $\{1, 2\}$, $\{2, 3\}$ and $\{1, 2, 3\}$. To get the bitstring represenation for S we convert each subset it covers to an integer in $\{0, \ldots, 2^n - 1\}$ by interpreting the subset as a binary number itself. More precisely for a subset $C \in \mathrm{cvrd}(S)$ we calculate

$$\sum_{a \in C} 2^{a-1} \in \{0, \ldots, 2^n - 1\}.$$

Of course this is just the integer corresponding to the binary representation where the bits in positions indexed by the elements of C have been set. For example for the subsets $\{3\}, \{1, 2\}, \{1, 2, 3\} \in \mathrm{cvrd}(S)$ we would get $(100)_2, (011)_2, (111)_2$ respectively. We then use the numbers we calculated for each subset in the same fashion again to calculate a bitstring encoding $\mathrm{cvrd}(S)$ as follows

$$\mathrm{bits}(\mathrm{cvrd}(S)) = \sum_{C \in \mathrm{cvrd}(S)} 2^{\sum_{a \in C} 2^{a-1}}, \tag{27}$$

or in other words, in order to get the bitstring of $\mathrm{cvrd}(S)$ we set the bit corresponding to the aforecalculated numbers counting from 0 and starting on the right[10]. For the string $S = 123$ as before we get

$$\mathrm{bits}(\mathrm{cvrd}(S)) = (11011110)_2.$$

The nice thing about converting $\mathrm{cvrd}(S)$ to an integer is that they are easy to sort linearly. Also by Eq. (27) it is immediately clear that for sequences S, T

$$S \lesssim_{\mathrm{W}} T \implies \mathrm{bits}(\mathrm{cvrd}(S)) \leq \mathrm{bits}(\mathrm{cvrd}(T))$$

[10] "Big endian"

holds, where $\leq$ is the standard linear order on integers. Indeed, defining

$$S \lesssim_{\mathrm{E}} T \iff \mathrm{bits}(\mathrm{cvrd}(S)) \leq \mathrm{bits}(\mathrm{cvrd}(T)) \text{ on } X$$

yields a preorder that extends $\lesssim_{\mathrm{W}}|_X$ and is total by definition, satisfying exactly the requirements of Theorem 3.84.

3.7 Beam search

Beam search builds upon the well-known breadth-first search, but instead of exploring all nodes for each layer, all extensions from the current nodes are generated and then ranked according to some heuristic function of which then only the best b nodes are kept. This makes beam search a heuristic search algorithm, although for unrestricted $b = \infty$ or b large enough it is equivalent to breadth-first search. In that sense the parameter b controls how greedy versus explorative the algorithm behaves at the cost of memory and runtime. These ideas were first introduced by Lowerre in 1976 in his PhD thesis [22]. The term beam search was later coined by [27]. In this section we introduce our own eponymous variant of beam search, Algorithm 4, which includes for every iteration a pruning step where inferior or equivalent strings according to a certain preorder are eradicated. This is based purely on the work we did in Section 3.6. We will use the preorder $\lesssim_{\mathrm{W}}$ from Definition 3.75 which we repeat here for convenience,

$$S \lesssim_{\mathrm{W}} T :\iff \mathrm{cvrd}(S) \subseteq \mathrm{cvrd}(T) \text{ and } \mathrm{lrs}(S) = \mathrm{lrs}(T). \tag{28}$$

We have separated the pruning into its own function. First let us describe BeamSearch. The function takes as inputs n pertaining to looking for P_n-covering sequences and a natural number b as described above. Note that this algorithm is easily adapted for finding any P-covering sequences, but we chose to frame it in the context of finding P_n-covering sequences as this was the most important use case for us. We start off with initialising a list B with the initial partial solution 123. This is possible without loss of generality as discussed in Remark 3.5. Then, as long as the top entry of B is not a feasible solution, we generate all useful one-letter extensions in the search tree. In particular we again apply the optimisations to avoid generating codirectionally-isomorphic sequences and sequences that repeat a letter as we already did in and discussed about for Algorithm 2 (see also Section 3.5). Then we perform pruning on those newly generated partial solutions, which is not part of the standard beam search technique but our own contribution specific to our problem statement. Finally the sequences are sorted according to a heuristic and only the best b ones are kept for the next iteration, where all sequences again grow by one letter and so on until a solution has been found. Note that success is not guaranteed for small b, as this algorithm may not halt when the heuristic decides to repeat a pattern of letters ad infinitum without covering any of the remaining subsets of the corresponding sequences. In practice however, for b as low as 100 and $n \leq 16$ this behaviour has not been observed.

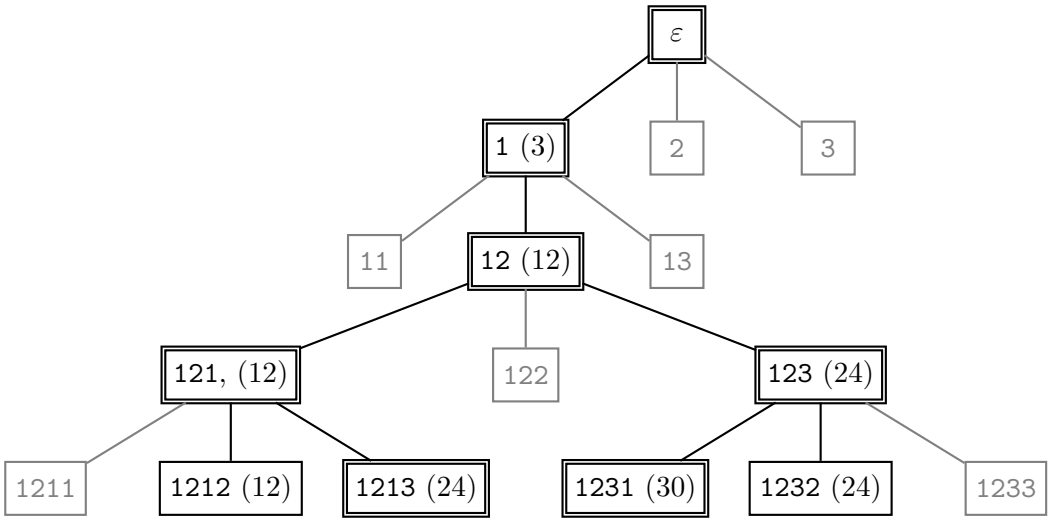

Figure 5: Depiction of the search tree associated with BeamSearch for $n = 3$ and $b = 2$ starting from the empty solution ε. The heuristic value was calculated by the function h_3, which will be introduced in Definition 3.88.

As a small example the search tree explored in BeamSearch for $n = 3$ and the parameter b set to the impractical value of 2 is depicted in Fig. 5. The greyed-out nodes are not considered due to pruning codirectionally-isomorphic strings as was mentioned above and outlined in detail in Section 3.5. Nodes in black indicate expanded nodes, i.e. nodes which were fully generated and for which the heuristic value was calculated (shown in paranthesis) while doubly boxed rectangles mark the nodes that do not get truncated by the small beam width ($b = 2$ in this example).

The pruning step is the innovation we contribute. Due to the structure of Eq. (28) we can get away with sorting the sequences lexicographically first separating by the longest relevant suffix (with itself may e.g. be sorted lexicographically although this is not relevant) and in the second layer sorting descendingly by the bitstring encoding of the sequences' covered sets as we laid out in detail in Example 3.86. More precisely the partial P_n-covering sequences are sorted according to the total preorder

$$\begin{aligned} S \lesssim_1 T \quad \iff \quad & (\mathrm{lrs}(S) \leq_{\mathrm{L}} \mathrm{lrs}(T)) \vee \\ & (\mathrm{lrs}(S) = \mathrm{lrs}(T) \wedge \mathrm{bits}(\mathrm{cvrd}(S)) \leq \mathrm{bits}(\mathrm{cvrd}(T))), \end{aligned}$$

where $\leq_{\mathrm{L}}$ is any linear order on $\{1, \ldots, n\}^*$. This ensures the resulting array is partioned in such a way that partial P_n-covering sequences sharing the same longest relevant suffix are neighbouring, i.e. for each longest relevant suffix present in the array there exists a contiguous segment of partial covering sequences sharing that suffix — within each such segment then the sequences are sorted based on the bitstring encoding of their cover in descending order. This sorting is done as the first step in PreorderPrune on line 15.

We then start with the first element in the sorted array and loop over the elements until a new longest relevant suffix is reached (inner loop). At this point we have found a segment of all sequences sharing a specific relevant suffix and that segment is even already sorted based on the bitstring representation of the covered sets. This is exactly the situation as outlined in Example 3.86 (notice that restricting $\lesssim_1$ on a set of sequences sharing the same longest relevant suffix yields a total preorder extending $\lesssim_{\mathrm{W}}$ restricted on the same set). We may therefore call FindMaximals (see Algorithm 3) on that block since all conditions as given in Theorem 3.84 for this procedure are met (see also Example 3.86). This function now returns all the maximal elements according to the preorder used, in our case $\lesssim_{\mathrm{W}}$, whose definition has been repeated above in Eq. (28). We repeat this procedure (outer loop) on the next block and so on until all elements have been processed.

The resulting processed array (possibly smaller than the input array) is then returned to the main loop of BeamSearch. Another sorting step now has to occur, namely sorting the partial solution based on a heuristic. Finally the best (according to the heuristic) b partial solutions are kept and the rest truncated. We continue in this manner in the main loop until a feasible solution has been found. Note, as mentioned already, the algorithm may keep on running forever if b was chosen too small. To avoid this perhaps unwanted indeterminacy one may set a limit on the search depth, i.e. length of solutions or a time limit for example.

The pseudocode for the abovedescribed procedure is found in Algorithm 4.

Heuristics. We have mentioned that beam search relies on a heuristic that allows for the comparison of different partial solutions to each other. Formally this is just a function with a specific domain and range.

Definition 3.87 (Heuristic)**.** Let A be an alphabet and $P \subseteq 2^A \setminus \{\emptyset\}$. Then we call a function $h\colon 2^P \to \mathbb{N}_0$ a *heuristic*.

An intuitive restriction on heuristics is to require some sort of monotonicity. All heuristics we considered fulfill the property that for sequences S and T

$$\mathrm{cvrd}(S) \subseteq \mathrm{cvrd}(T) \implies h(\mathrm{cvrd}(S)) \leq h(\mathrm{cvrd}(T))$$

holds.

Definition 3.88. The following are the heuristics we considered in this thesis.

$$h_1(C) \coloneqq |C|, \qquad h_2(C) \coloneqq \sum_{c \in C} |c|,$$

$$h_3(C) \coloneqq \sum_{c \in C} \binom{n}{|c|} |c|, \qquad h_4(C) \coloneqq \sum_{c \in C} \binom{n}{|c|} 2^{|c|}.$$

Heuristics h_3 and h_4 were suggested to us by Alexander Healy through email [12].

Note that when using beam search to produce P_n^k-covering sequences all of the portrayed heuristics are of course equivalent. They only differ when there are subsets of different sizes to cover.

Algorithm 4 Beam search for finding P_n-covering sequences

Require: $n \in \mathbb{N}, n \geq 3, b \in \mathbb{N}$

```
1:  function BeamSearch(n, b)
2:      B ← [123]
3:      while B[1] is not feasible do
4:          B_new ← []
5:          while B ≠ ∅ do
6:              S ← pop(B)
7:              for c ∈ (S \ suff_1(S)) ∪ min({1, ..., n} \ S) do
8:                  append(B_new, S · c)
9:          B_pruned ← PreorderPrune(B_new)
10:         sort(B_pruned)                          ▷ Sort according to heuristic
11:         B ← B_pruned[1 : b]                     ▷ Discard all but best b sequences
12:     return B which are feasible

13: function PreorderPrune(X)
14:     B_pruned ← []
15:     sort(X)                                     ▷ Sort according to ≲_1
16:     i ← 1
17:     while i ≤ |X| do
18:         R ← lrs(X[i])
19:         j ← i
20:         while lrs(X[i]) = R do
21:             i ← i + 1
22:         extend(B_pruned, FindMaximals(X[j : i − 1]))
23:     return B_pruned
```

Before running experiments requiring great amounts of memory and CPU-hours, we tested all of the heuristics on $n = 8$ and no greater values for n. We also tested the heuristics on $n = 7$ but the results were not very informative as even for relatively small values of b solutions of length 42 were found consistently across all mentioned heuristics with no improvement even for substantially bigger values for b (an optimal solution to the problem of finding a P_7-covering sequence has length 40). For $n \leq 6$ the optimal solutions were found with all heuristics already for very small values of b. The results of our "training" can be seen in Table 6.

b	h_1	h_2	h_3	h_4
$25 \cdot 2^0$	93	92	**83**	91
$25 \cdot 2^1$	89	**86**	91	87
$25 \cdot 2^2$	**84**	86	86	86
$25 \cdot 2^3$	85	**82**	85	86
$25 \cdot 2^4$	84	**82**	84	85
$25 \cdot 2^5$	83	82	84	**81**
$25 \cdot 2^6$	**81**	82	**81**	**81**
$25 \cdot 2^7$	81	82	**80**	81
$25 \cdot 2^8$	81	82	**80**	81
$25 \cdot 2^9$	**80**	82	**80**	**80**
$25 \cdot 2^{10}$	**80**	81	**80**	**80**
$25 \cdot 2^{11}$	**80**	81	**80**	**80**
$25 \cdot 2^{12}$	80	80	**79**	**79**
$25 \cdot 2^{13}$	80	80	**79**	**79**
$25 \cdot 2^{14}$	80	**78**	79	**78**
$25 \cdot 2^{15}$	79	**78**	**78**	**78**
$25 \cdot 2^{16}$	**76**	77	78	78
$25 \cdot 2^{17}$	**76**	77	**76**	78
$25 \cdot 2^{18}$	**76**	77	**76**	77
Sum	1548	1547	**1539**	1547
Best	8	5	**12**	8

Table 6: Length of the best P_8-covering sequence found running beam search with the specified value for b and heuristic. Best results are highlighted in bold. The "Sum" field describes the total sum of lengths for that specific heuristic (lower is better), while "Best" describes how many times the respective heuristic was among the best ones (Wins / Ties) if every tested value of b was to be looked upon as a competition. With both evaluations heuristic h_3 is victorious.

Based on the preliminary observations in Table 6 we used h_3 in our experiments.

Results & Implementation details. We ran BeamSearch without any time restrictions, however a limit on the search tree depth, i.e. the length of sequences explored of 65535 was imposed. For P_n^k-covering sequences, experiments were run for all $n \in \{8, \ldots, 16\}$ and $k \in \{3, \ldots, n-2\}$ where the parameter b was set as $b = 6553600 \cdot 2^{8-n}$. The search for P_n-covering, where again $n \in \{8, \ldots, 16\}$, was given more priority by letting $b = 52428800 \cdot 2^{8-n}$. The results can be found in Table 7 and Table 8 respectively. The search tree depth limit was reached for the instances where $n = 16$ and $k \in \{7, \ldots, 11\}$ and subsequently no P_n^k-covering sequences were found for those instances. Algorithm 4 was implemented in Julia and the results were achieved on a server cluster consisting of AMD EPYC 7402, 2.80GHz CPUs each with 1024 Gb of RAM.

Table 7: Results of running BeamSearch with the specified parameter b. Lower bounds are displayed in the fifth column, where numbers marked by † are bounds found by Algorithm 2, IDDFS (see also Table 4). Bold numbers indicate optimal values / sharp bounds.

n	k	b	Result	$\max(\beta_n^k, \text{IDDFS}^\dagger)$	$\Delta(\%)$	Time (min)
8	3	6553600	**58**	†**58**	0%	19359
8	4	6553600	**74**	**74**	0%	2978
8	5	6553600	**60**	†**60**	0%	553
8	6	6553600	**37**	†**37**	0%	48
9	3	3276800	**90**	**90**	0%	5666
9	4	3276800	131	129	1.55%	1195
9	5	3276800	134	130	3.07%	504
9	6	3276800	93	91	2.19%	204
9	7	3276800	48	†47	2.12%	29
10	3	1638400	**122**	**122**	0%	1100
10	4	1638400	217	213	1.87%	465
10	5	1638400	265	260	1.92%	434
10	6	1638400	225	215	4.65%	188
10	7	1638400	133	126	5.55%	99
10	8	1638400	63	†56	12.50%	24
11	3	819200	172	167	2.99%	1408
11	4	819200	342	333	2.70%	769
11	5	819200	484	466	3.86%	744
11	6	819200	485	467	3.85%	439
11	7	819200	357	336	6.25%	326
11	8	819200	185	172	7.55%	109
11	9	819200	80	†67	19.40%	18
12	3	409600	**228**	**228**	0%	1390
12	4	409600	509	504	0.99%	623
12	5	409600	821	796	3.14%	498
12	6	409600	972	929	4.62%	510
12	7	409600	842	798	5.51%	452

Table 7 – continued

n	k	b	Result	$\max(\beta_n^k, \text{IDDFS}^\dagger)$	$\Delta(\%)$	Time (min)
12	8	409600	539	505	6.73%	286
12	9	409600	248	230	7.82%	84
12	10	409600	104	†78	33.33%	22
13	3	204800	295	288	2.43%	641
13	4	204800	733	718	2.08%	483
13	5	204800	1336	1291	3.48%	518
13	6	204800	1780	1721	3.42%	541
13	7	204800	1819	1722	5.63%	515
13	8	204800	1381	1294	6.72%	466
13	9	204800	794	723	9.82%	211
13	10	204800	333	295	12.88%	57
13	11	204800	128	†91	40.65%	11
14	3	102400	378	366	3.27%	365
14	4	102400	1032	1009	2.27%	310
14	5	102400	2070	2006	3.19%	414
14	6	102400	3120	3012	3.58%	596
14	7	102400	3607	3444	4.73%	707
14	8	102400	3181	3013	5.57%	675
14	9	102400	2161	2010	7.51%	409
14	10	102400	1110	1012	9.68%	178
14	11	102400	427	374	14.17%	57
14	12	102400	168	†104	61.53%	10
15	3	51200	472	465	1.50%	189
15	4	51200	1394	1368	1.90%	229
15	5	51200	3104	3015	2.95%	353
15	6	51200	5184	5013	3.41%	601
15	7	51200	6711	6441	4.19%	768
15	8	51200	6791	6442	5.41%	927
15	9	51200	5330	5015	6.28%	847
15	10	51200	3263	3016	8.18%	468

Table 7 – continued

n	k	b	Result	$\max(\beta_n^k, \text{IDDFS}^\dagger)$	$\Delta(\%)$	Time (min)
15	11	51200	2317	1375	68.50%	217
15	12	51200	549	468	17.30%	40
15	13	51200	195	†120	62.50%	8
16	3	25600	576	562	2.49%	166
16	4	25600	1879	1826	2.90%	262
16	5	25600	4501	4372	2.95%	486
16	6	25600	8279	8018	3.25%	784
16	7–11	25600	DNF			
16	12	25600	2126	1832	16.04%	224
16	13	25600	702	572	22.72%	37
16	14	25600	224	†135	65.92%	8

Table 8: Results of running BeamSearch with the specified parameter b. Lower bounds are displayed in the fourth column, where numbers marked by † are bounds found by Algorithm 2, IDDFS (see also Table 4). Bold numbers indicate optimal values / sharp bounds.

n	b	Result	$\max(\beta_n, \text{IDDFS}^\dagger)$	$\Delta(\%)$	Time (min)
8	52428800	76	74	2.70%	6749
9	26214400	148	130	13.84%	3229
10	13107200	289	260	11.15%	2797
11	6553600	565	467	20.98%	3428
12	3276800	1112	929	19.69%	4472
13	1638400	2176	1722	26.36%	6124
14	819200	4230	3444	22.82%	12677
15	409600	8237	6442	27.86%	19031
16	204800	16087	12882	24.87%	3095

3.8 Comparison

Algorithm 4 significantly outperforms the reduction of the problem to the GTSP as described in Section 3.3 in terms of solution quality in all but a few instances tested. There was only one instance where the results from the GTSP model outperformed BeamSearch, namely for $n = 11$, $k = 3$ where the former resulted in a length of 167 (which is optimal) and the latter 172. Apart from this outlier there were five instances where both approaches correctly produced the optimal solution for $n = 8$, $k = 4$ and $n \in \{8, 9, 10, 12\}$, $k = 3$. In all other instances that were tested both ways Algorithm 4 was superior.

Due to the breadth-first kind-of fashion BeamSearch operates and the time limit imposed on the solver of the GTSP model it does not make much sense to compare the time the two algorithms took on particular instances. For example for the relatively "easy" instance where $n = 8$, $k = 3$ the GTSP model could be solved optimally in under 1 minute while BeamSearch ran for 19359 minutes yielding the same result. However the beam width was set quite high to $b = 6553600$ and it is highly likely this solution would have been found with a substantially smaller value for the parameter b. We did not set out to find the minimal b for which the resulting solution is optimal for one as this is a computationally intensive task. Assuming the solution quality is monotone in b (which is not true in general, but holds approximately) one could devise a binary search style procedure to find or estimate the minimal beam width required to find on-par or superior solution quality of the GTSP approach.

It is clear from the instances the two methods were tested on that BeamSearch is much more capable in approaching big instances. As we have stated in Section 3.3 the biggest instance tested there regarding P_{13}^5-covering sequences had already over 100000 vertices which is approaching the limits of what current off-the-shelf GTSP solvers and equally important modern hardware are capable of handling. Although an interesting model of the problem in its own right, our results indicate that our variant of BeamSearch is the algorithm of choice for this problem.

4 Conclusion

In the beginning of this thesis multiple examples to illustrate the nature of universal cycles and sequences are given. For the particular topic of universal cycles pertaining to subpermutations we have identified a problem in a proof found in the literature and taken care to elaborate and expand on a correct proof for the existence of universal cycles for those combinatorial objects.

After introducing the relaxation of covering sequences we focus on such sequences covering subsets of a set for the remainder of the thesis. There we have demonstrated that the lower bounds on the lengths of P_n^k- and P_n-covering sequences can be improved upon to what is currently known in the literature (at least for some n and k). In this process we have also provided multiple flexible generalisations of existing theorems bounding the length of P_n^k-covering sequences, which contain a certain infix, from below — the bounds provided by Lipski are then a straightforward corollary supporting the notion that our results are generalisations.

Having done this theoretical work, we then provide practical means of finding P_n^k-covering sequences. One approach, which as far as we know, has not been mentioned in the literature, is a reduction to the Generalised Traveling Salesperson Problem. This novel idea proved interesting, as off-the-shelf solvers for it could be used, however the results did not scale too well, as the memory requirement explodes.

At this point we turn to the algorithmic side of the problem. First we introduce a variant of iterative deepening depth-first search, which utilises the theory on lower bounds to algorithmically improve upon them and in some cases even find optimal sequences. In this same setting we discuss an already known optimisation in the process of finding P_n^k-covering sequences and perform some interesting combinatorial counting linking to preexisting results about related objects.

Finally we introduce the main algorithm for constructing covering sequences which is based on the ideas of beam search. To this end we have introduced the concept of pruning admissible relations and give a theoretical foundation. Algorithm 4 is introduced and utilises these concepts to reduce the search space required compared to a more naive approach. Having presented the results we then compared them favourably to the other methods introduced. Some lengths of P_n-covering sequences have even beat the best known results in the literature to date as far as we know. Based on our results we also believe that optimal P_n- and P_n^k-covering sequences are relatively close in length to the lower bounds derived in Section 3.2.

References

[1] Hugo Åkesson (https://stackoverflow.com/users/21420133). *Efficient algorithm to find the maximal elements of a partially ordered set.* StackOverflow. Accessed: 2024-09-26. URL: https://stackoverflow.com/questions/21560659.

[2] Anonymous. *Archived discussion of anonymous 4chan poster regarding new lower bounds for superpermutations.* 2018. URL: https://warosu.org/sci/thread/S3751105#p3751197.

[3] Daniel A. Ashlock and Jenett Tillotson. "Construction of small superpermutations and minimal injective superstrings". In: *Congressus Numerantium* (1993), pp. 91–91.

[4] Roberto J. Bayardo and Biswanath Panda. "Fast Algorithms for Finding Extremal Sets". In: *Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, April 28-30, 2011, Mesa, Arizona, USA*. SIAM / Omnipress, 2011, pp. 25–34. DOI: 10.1137/1.9781611972818.3.

[5] Fan Chung, Persi Diaconis, and Ron Graham. "Universal cycles for combinatorial structures". In: *Discrete Mathematics* 110.1 (1992), pp. 43–59. ISSN: 0012-365X. DOI: 10.1016/0012-365X(92)90699-G.

[6] Nicolaas Govert De Bruijn and Paul Erdös. "On a combinatorial problem". In: *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam* 51.10 (1948), pp. 1277–1279.

[7] Michal Debski and Zbigniew Lonc. "Universal Cycle Packings and Coverings for k-Subsets of an n-Set". In: *Graphs and Combinatorics* 32.6 (2016), pp. 2323–2337. DOI: 10.1007/S00373-016-1727-6.

[8] Greg Egan. *Superpermutations.* Accessed: 2025-09-20. 2018. URL: https://www.gregegan.net/SCIENCE/Superpermutations/Superpermutations.html.

[9] Michael Engen and Vincent Vatter. "Containing All Permutations". In: *The American Mathematical Monthly* 128.1 (Jan. 2021), pp. 4–24. ISSN: 1930-0972. DOI: 10.1080/00029890.2021.1835384.

[10] Sakti P. Ghosh. "Consecutive storage of relevant records with redundancy". In: *Commun. ACM* 18.8 (Aug. 1975), pp. 464–471. ISSN: 0001-0782. DOI: 10.1145/360933.360991.

[11] Stefan Glock, Felix Joos, Daniela Kühn, and Deryk Osthus. "Euler Tours in Hypergraphs". In: *Combinatorica* 40.5 (2020), pp. 679–690. DOI: 10.1007/S00493-020-4046-8.

[12] Alexander Healy. Personal communication. Nov. 2023.

[13] Keld Helsgaun. "Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun Algorithm". In: *Mathematical Programming Computation* 7.3 (Apr. 2015), pp. 269–287. ISSN: 1867-2957. DOI: 10.1007/s12532-015-0080-8.

[14] Stefan Hetzl. *Automata and Formal Languages.* https://dmg.tuwien.ac.at/hetzl/teaching/afl_2023.pdf. Accessed: 2024-09-04. June 2023.

[15] Robin Houston. *Tackling the Minimal Superpermutation Problem*. 2014. arXiv: 1408.5108 [math.CO]. URL: https://arxiv.org/abs/1408.5108.

[16] Glenn Hurlbert. "On Universal Cycles for k-Subsets of an n-Set". In: *SIAM Journal on Discrete Mathematics* 7.4 (1994), pp. 598–604. DOI: 10.1137/S0895480191220861.

[17] B.W. Jackson. "Universal cycles of k-subsets and k-permutations". In: *Discrete Mathematics* 117.1 (1993), pp. 141–150. ISSN: 0012-365X. DOI: 10.1016/0012-365X(93)90330-V. URL: https://www.sciencedirect.com/science/article/pii/0012365X9390330V.

[18] Richard E. Korf. "Depth-First Iterative-Deepening: An Optimal Admissible Tree Search". In: *Artificial Intelligence* 27.1 (1985), pp. 97–109. DOI: 10.1016/0004-3702(85)90084-0.

[19] Lawrence T. Kou. "Polynomial Complete Consecutive Information Retrieval Problems". In: *SIAM Journal on Computing* 6.1 (1977), pp. 67–75. DOI: 10.1137/0206004.

[20] Witold Lipski Jr. "On strings containing all subsets as substrings". In: *Discrete Mathematics* 21.3 (1978), pp. 253–259.

[21] Zbigniew Lonc, Tomasz Traczyk, and Miroslaw Truszczynski. "Optimal F-Graphs for the Family of all K-Subsets of an N-Set". In: *Data Base File Organization, Theory and Applications of the Consecutive Retrieval Property, Papers from the Conference on Consecutive Retrieval Property and Applications, 17-22 August 1981, Warsaw, Poland*. Ed. by Sakti P. Ghosh, Yahiko Kambayashi, and Witold Lipski Jr. Academic Press, 1981, pp. 247–270.

[22] B. T. Lowerre. "The Harpy speech recognition system". PhD thesis. Carnegie Mellon University, Pennsylvania, Apr. 1976.

[23] Nelma Moreira and Rogério Reis. "On the density of languages representing finite set partitions". In: *Journal of Integer Sequences* 8.05.2 (2005), p. 8.

[24] Charles Edward Noon. *The generalized traveling salesman problem*. University of Michigan, 1988.

[25] OEIS Foundation Inc. *The On-Line Encyclopedia of Integer Sequences*. Published electronically at http://oeis.org. 2024.

[26] Jean-Eric Pin. *Mathematical foundations of automata theory*. 2022.

[27] Raj Reddy. "Speech understanding systems: A summary of results of the five-year research effort at Carnegie Mellon University". In: *Pittsburgh, Pa* 45 (1977), p. 80.

[28] J. Robert Johnson. "Universal cycles for permutations". In: *Discrete Mathematics* 309.17 (Sept. 2009), pp. 5264–5270. ISSN: 0012-365X. DOI: 10.1016/j.disc.2007.11.004.

[29] Joe Sawada and Aaron Williams. "Constructing the first (and coolest) fixed-content universal cycle". In: *Algorithmica* 85.6 (Nov. 2022), pp. 1754–1785. ISSN: 1432-0541. DOI: 10.1007/s00453-022-01047-2.

[30] S. L. Smith and F. Imeson. "GLNS: An Effective Large Neighborhood Search Heuristic for the Generalized Traveling Salesman Problem". In: *Computers & Operations Research* 87 (2017), pp. 1–19.

[31] Paul Tabatabai. “On sequences covering subsets of a finite set”. MA thesis. TU Graz, 2018.

[32] Alexander Weissenfels, Enrico Iurlano, and Günther R. Raidl. *Superstrings of Uniform-Cardinality Set Systems via the Generalized Traveling Salesperson Problem*. submitted. 2026. URL: https://www.ac.tuwien.ac.at/files/pub/weissenfels_iurlano_raidl_26.pdf.

[33] Dennis Wong. “A New Universal Cycle for Permutations”. In: *Graphs and Combinatorics* 33.6 (2017), pp. 1393–1399. DOI: 10.1007/S00373-017-1778-3.

A Alternative proof of Proposition 3.25

Proof. The proof is based on Corollary 3.22 and will be structured in two parts. First, we will derive formula (14) for the case where S has an injective k-prefix. Then we will show that in the other case the bound can only get worse (i.e. larger).

(i) Assume first that S has an injective k-prefix. Since P_n^k-covering sequences are invariant under permutations, we can without loss of generality assume $S_1 \dots S_k = \texttt{1...k}$. We now want to apply Corollary 3.22 on $S_1 \dots S_k$. Notice that this string, by design, only has a single (valid) k-block and thus

$$a_{k,i} = \binom{n-1}{k-1} - 1 \quad \text{for } i \leq k,$$
$$a_{k,i} = \binom{n-1}{k-1} \qquad \text{for } i > k$$

in the notation of Corollary 3.22 and for the particular ordering we chose, we get that $l_i = k - i$ for $i \leq k$ and since the letters $\{k+1, \dots, n\}$ do not appear, we have $l_i = \infty$ for those. For the case where account $1 \leq i \leq k$ we can now compute

$$\max(k - l_i - 1, 0) = \max(k - (k-i) - 1, 0) = i - 1.$$

For $k < i \leq n$ the above term is of course 0 and thus Eq. (13) yields

$$\begin{aligned} |S| &\geq k + \sum_{i=1}^{k} \left\lceil \frac{\binom{n-1}{k-1} - i}{k} \right\rceil + \sum_{i=k+1}^{n} \left\lceil \frac{\binom{n-1}{k-1}}{k} \right\rceil \\ &= \underbrace{k + \sum_{i=1}^{k} \left\lceil \frac{\binom{n-1}{k-1} - i}{k} \right\rceil + (n-k) \left\lceil \frac{\binom{n-1}{k-1}}{k} \right\rceil}_{a :=}, \end{aligned} \tag{29}$$

which can be simplified using the well-known identity $\frac{n}{k}\binom{n-1}{k-1} = \binom{n}{k}$ to

$$= k + (n-k) \left\lceil \frac{\binom{n}{k}}{n} \right\rceil + \sum_{i=1}^{k} \left\lceil \frac{\binom{n}{k}}{n} - \frac{i}{k} \right\rceil,$$

which is precisely the bound we set out to prove.

(ii) Assume now on the contrary that S doesn't have an injective k-prefix or to put it in other words the first k-block is not valid. Thus we have that

$$a_{k,i} = \binom{n-1}{k-1} \quad \text{for } 1 \leq i \leq n.$$

Denote by $d \coloneqq |\{S_1, \dots, S_k\}|$ the number of distinct digits in the k-prefix of S. Since $S_1 \dots S_k$ is not injective, we have that $d \leq k - 1$. Clearly for all

$i \in \{1, \ldots, n\}$ we have that either $0 \leq l_i \leq k-1$ or $l_i = \infty$. Notice that in the former case, which occurs for d many indices, we have that $\max(k-l_i-1, 0) = k-l_i-1$ and in the latter case, which occurs for exactly $(n-d)$ many indices we get that $\max(k-l_i-1, 0) = 0$. Also note, that by definition of the variables l_i, $i \in \{1, \ldots, n\}$ we have that $l_i \neq l_j$ or $l_i = l_j = \infty$ for $i \neq j$. Since we already saw that $a_{k,i}$ does not depend on i in this case we can without loss of generality assume the vector $l = (l_i)_{i=1}^n$ to be sorted weakly monotonically decreasing but in such a way that the ∞-values all occur at the end, e.g. l could look like

$$l = (\underbrace{k-1, k-3, \ldots, 0}_{d}, \underbrace{\infty, \ldots, \infty}_{(n-d)}).$$

With this ordering we have that for $1 \leq i \leq d$

$$\max(k - l_i - 1, 0) = k - l_i - 1.$$

Like in case (i) the above term evaluates to 0 for $d < i \leq n$. Then by Eq. (13) we compute

$$|S| \geq k + \sum_{i=1}^{d} \left\lceil \frac{\binom{n-1}{k-1} - k + l_i + 1}{k} \right\rceil + (n-d) \left\lceil \frac{\binom{n-1}{k-1}}{k} \right\rceil =: b(l).$$

If we rewrite the right term $b(l)$ as

$$b(l) = k + \sum_{i=1}^{\#S_{1\ldots k}} \left(\left\lceil \frac{\binom{n-1}{k-1} - k + l_i + 1}{k} \right\rceil - \left\lceil \frac{\binom{n-1}{k-1}}{k} \right\rceil \right) + n \left\lceil \frac{\binom{n-1}{k-1}}{k} \right\rceil$$

it is easy to see that this term is minimal, regardless of the concrete values for l, when d is maximal, i.e. $d = k-1$ since we are only summing non-positive terms and for $i \in \{1, \ldots, d\}$ the l_i are chosen as small as possible. In this case $l = \hat{l}$ looks like

$$\hat{l} = (\underbrace{k-2, \ldots, 0}_{k-1}, \underbrace{\infty, \ldots, \infty}_{(n-k+1)})$$

and therefore we have that

$$\min_l b(l) \geq b(\hat{l}) = k + \sum_{i=1}^{k-1} \left(\left\lceil \frac{\binom{n-1}{k-1} - i}{k} \right\rceil - \left\lceil \frac{\binom{n-1}{k-1}}{k} \right\rceil \right) + n \left\lceil \frac{\binom{n-1}{k-1}}{k} \right\rceil$$

which is clearly an upper bound for (15), i.e. $\min_l b(l) \geq b(\hat{l}) \geq a$.

We can thus conclude that the bound found in the first case applies to all P_n^k-covering sequences, which is what we wanted to show. □

B Results from Section 3.3

12341578437816584678254628361735471256725362146372481358 12

P_8^3-covering sequence of length 58.

12345768234761247836527134851364852736541283571846231547861378216574265812

P_8^4-covering sequence of length 74.

12345812365841765348726541372861572481638571236487532467138426537

P_8^5-covering sequence of length 65.

12345687124367512486371528134572863154

P_8^6-covering sequence of length 38.

12378642794875162856328957629163946371245968125967146523476381794139245728415379835918 5438

P_9^3-covering sequence of length 90.

12349571496523764938625419286514895763187461279635196584735891628791634257168427134978653418259738294683792458324765219371 57821369253189

P_9^4-covering sequence of length 136.

12345976184579126548319247561239564296734182694152867342936875193675281364578697258914672839459863425896134278531947863712 86351742891732598734 52

P_9^5-covering sequence of length 144.

12345697834569183427568312764153829371846923719865279416328954732195348267159482715369278156429873514 9

P_9^6-covering sequence of length 102.

12391a823645169a25862a794175638962495a763187a453a1428745876259764837425372 1a5693a85394617a34186a423a61289a4897251379185912

P_{10}^3-covering sequence of length 122.

12349673a86917623a1693194871362814376847 53a465318293a17827a39624a86549a1846915732156a9247168571a2679a5268432654179a64135941 2a589635a91854a78915274a15829a746219734a2531a4385241675a26897538a219387295672849a3816a78523498536745923748

P_{10}^4-covering sequence of length 229.

12345a219a35814793681a59681743a879526a34825194a35718a9627198547361a284635a82671a52893547293657a841369a487651a473685297a1389 576a2735164983a67892164289a72854a96531a2495863274156274a591732a49674213856a4739814a6251974a28137625138296a147286549236a892a 3785213967a182943157a59327849613a458162345697142

P_{10}^5-covering sequence of length 294.

123456a729485a2739856a419376582938a1672594a3621857a693217a458967192315 87a9342176495327649385742 61a5834627318a427351984a2719
84a6315743a59816374a9625137a9684512a367584a269174835a6298134629873a14862951436872a59143728915a3298a6714589234a9182764a38251
479a651378

P_{10}^{6}-covering sequence of length 256.

1237a584319846b38ab597651b74a1b4a6b28637423572b94172618b543986a598761b2a5197b56478b932a619a72953b428379421a78291b371542634a
5381a3b2569a8251369ba76b84a296418563a9457812

P_{11}^{3}-covering sequence of length 167.

12345b2a47a9842597b829735b291683261a584ba618b94314a1873a769b83b1978365782465ab7124831298514ab5983a24163b2a6315767185b768437
426b9428b176395b468127a653b841752a4615ba274b32951b3692463b825652a9643a5873b125381a3298ab1963ab586ab78459a61729a4376b4916584
395a74b12a537416b28a67459683ab9652149765237b94a1796289a1285a913546a8913726b5724937ab92768941b3a4879514b751a3728a4

P_{11}^{4}-covering sequence of length 359.

12345281697ab82695b736185673a9754163ab5769318a42b16759a1685b29a148b574682174692a764b893127a14923b78624561b3945625b489a32781
345781a25b34629b8152394785a6473b5297836297b856941b254a91b678a293b62758a94735a71436ab71289453ab9452731496752a968341935a61b47
a3b15274b83259a8b234a781b427986ba45836a125783b931587b19563b82943a68795136247ba26183b7912675314b685a23761a8b7496ab321a5469b5
a143b58461a75b942a8359721a69321b4752491b52638247563a281ab5287a53462ab9736482a19378a19b74a689518a3b481952a7b3158a42791486792
468b3a91462a5b74396b81234a257ab8359621b3742

P_{11}^{5}-covering sequence of length 535.

1239b128ac13b75c1b5397ba951785c94b3ac5369723c7a562345ba9745823579c1837286419a4c2a71c4a8596152a96b6a39c276b32a459315a38943a1
48b9c6129b52946acb412c5678926c7b1a26b85143856bc547198613746b18a9c874c3642716a7368ab248c64527b4a78b2c83bc8

P_{12}^{3}-covering sequence of length 228.

1234852a4362871536b9c78a9435a6127c8b57c42a63572b14a5976a598c2b16cab32c654723a86b174293a7b7519a36ab2895b16a47594b8a537c1987b
4613a8c415bc2938b12a3b59ac2379253ac67312ca8b1458a76b98c6b759c4b53c8174c687521a2756c97364ca186b39758c1ba64287a31482a914bc7a3
4a871642c84913c42bac56349c63254ba72981568349ba8c5439ac72b46978392186c2796136891a763b48c73483cb1794856c175ab297134bac9473bc6
58a19c62b516942516724a182c5a46bc1952c1945629a46873b1924b3c98a3b974ac3187ba96c1a53981b532c8942c57ab135968417ac6837548b5328b4
6132b95c316b95a26731457b32691ab5641c26a8926b5826b78231582

P_{12}^{4}-covering sequence of length 549.

12345a916452abc695a8369127a3594278ca31295b486a295814379c1a279cb427913b48273c6198b3215c73b62871b67345b2c876b39a5c8376ac5b48a
15b935714c82316bc8369cb8371a56c39724ca86b12496ac8b75236ac4b1a97c86179b5a47164cb91a296b45c9614b34a765b3a17954b1729864b3a2183
4abc752468753b1c286371cb295ac23b9478a5c128935b4713c84928b5a37c9ab864c2a7563c1a5b642c738524ca1623471ac3214bc539b6a782934a659
31421a598b1635482916cb1a67cb59a67c253b489ca1543b62919265789cba3864197c64381972c6b8597a3851b4a354692c58ab7518c4ba9619ac43257
915864c125b427c187946238965238c9a24bc8594ab3792ab35c6a2935c4a8342b7698543c261538bc19243cb8297c5a461b5786ca2561274a68973671a
82bc3748ba316578a237691a824519c23ab7621ab4761428b51c38647ca84b9178ab92c6b321784a2957894c57936a28574168b523a8b723b794a18cb74
186a2b486439bc176594a72514c36974a321578a1c2b6538c9143ca7b25168a3c271634a75c193a6cb9738512ab748639b128c5697b4852b6a854379a26
345c7b1a6c85193ab98a13bc4675b293a8475c8269b51c649837ca945c263941a263b

P_{12}^{5}-covering sequence of length 930.

1238153bda2b46a3c4d7a43dc6a154c126a4d68a413b682b6a5428db9538b14279825b4c672a819b8a7bdc1685a3d9615163c87b932b1a768417acd5874
ba31c937812a3986c51da5c71d365b1728ca42c697c3571396bd18c527cd2437643bc5921596ad89acb67d5b7c492cb941d253497a3845d37b2d9461bc8
d4b859716d24652a948c9d567459ad327d83629ba57d91ac23

P_{13}^{3}-covering sequence of length 296.

12345a7984d19745d8743b2a41d6a45b9c61472931b64578c951bc427318ca62357b28c5a4196841521c6ad1bc65d176c8243d59647d47ac3146acb264c
9a18d637bc53612bd4c29d65789bc1a4b71a836972853c76d8a96751b4865bd269b6412968b3598435adc832ad853b49c7d5b8c6178b5298412c8d64517
842b51ad7a3b9ca3b5d169ad43ba1954c68954a893b26a7bc4368c5169328653a7463ab892cb4582a974b5dc967dab693db1c2da5693a5812ab759ca83d
2156b723b8d96b78dc8b1537562b478c94318925c47d2a6879dba7913d84c3dba58732d589a4b63c17593c7263db6824d17c4357d92b19d5681b965c72b
d846721d9a37286137a84c52d783b476ca54d932c145dc369dcab5638712b8ac243ac69257489b5a2631dc6b7d32b5c2634db95ad36542a6b1734d9738c
b4adc398d2675a21d87da9264d925a3c9185dc1a23cd7a5c13b2cd142b9734392517ac253d17b9cd4a8b13452d6c4bd612a741d31a92315a68b29ac7a8b
c7189dc184a6934a827c914bd7c21976a31428135942adb81a67

P_{13}^4-covering sequence of length 790.

12345a183d14257b18cd62a192bd95adc139bd4abc52391b524cd7518ca92784362db14a783c5b97c61a932548cb251d7b253a8b16d724597dc469bdac3
9b6245a8bd26984c59685214cd5b2a4d7b6358932641a39bad29578abc26758bc671853b76a9cb1cd58b1682c7bd54298a16d3b978c6a9d7b432d9b7a31
c2da34562d8451c689d125638125ad6b75a2d49125467ad146b3da265bc1a24b7196b8dac468da917c269813ac46293174d5b631742c9d71386d327cab5
8741b58a2c7635ac2d87b23d574ac156293a78d51a743268a7d9c6123b847256bdc921764c25918b41a59d46ab175342d713a865ac271564b78a659b25a
16827d4b1596ad5cb3a687b2615937a6153b48a2d9765c137d6483976b2584672a954b68261dc783ba94c8314ba9859c1b68729d45381c974a615d798c7
456dc8635d2c936cdba73c9451da3465d1b3a5d68a2b34957896a3c152c891b7c341db957a24817c2ad93672a9b47a35b987d439ca2153d7a62b73cad41
97a1bdc76ab4938d675a214c89b3642ad196ba321d9678bd49265d892b4769c38adb12348bca97812ac613d94678cd4912cb47536d92c653879c5316d48
b19da372b1a6dca74681ca4691c3a8b9273564c83526a98b12a74b16934cb6752dc18374c283d5c61957c683bd5c431b78a94865db942173c9b2617a8cd
54ad56814b26d79b128c35a4b628c24a5db173918495ac7d1c4759124acd915643b8d174692ac637d8ab46cd1bc97d38594b2d81a49cb72185db7a283a1
429657b92a3261d4986157b93412c6b59476c3bd95c3618d723984d573bcd841295abc61375a81bc49273584aca43bdc2857a4bc1d354bcd347ac89d3a4
8612735a67349a8db45b1c46321a793c21b35c9a247c53ad86bc4a238cb15c79285a6b13892b538d9b563cd4829cb5728d7a495367b346a953d2b85197a
65c9d4bc465a3b2c6d7c58a9324cb6982d43c8a6dc3245b26387bdc4268ac6b48971d2cba9513d892c357d9a26bdc9853149cb589617328c652da71832d
a584631bac1d895a1c7b48

P_{13}^5-covering sequence of length 1498.

C Results from Section 3.7

12345671825473861253846275138427163587263415682374158764125736418231784562 31

P_8-covering sequence of length 76.

12341532651726147856483567354728372584268142518361734678 12

P_8^3-covering sequence of length 58.

12345162731587241675834678235682346527438174581245861437521365746281367812

P_8^4-covering sequence of length 74.

12345672184537124687134658273468152367158362145768324578 1234

P_8^5-covering sequence of length 60.

12345671284536871325687413268154372 86

P_8^6-covering sequence of length 37.

12345678912364859137264815329647813659471283657924835712689437165928736914537926145789314275819425687243958617952841365218 9
63478914672538296734521 97

P_9-covering sequence of length 148.

12341526315641789567946894579356834926437845376823582791829142758162715923716931842546789 3

P_9^3-covering sequence of length 90.

12345617426581279684756893578934579645892356923768297482579165382469184369125673915429815732814538197432618743617238614293 1
47521387

P_9^4-covering sequence of length 131.

12345617823649183679583476583926784925683497582347918652749165729381759463719268459127854126735126935182943518642913574218 3
64871324679

P_9^5-covering sequence of length 134.

12345678924517639821467953842675918463791258649327584132695413726851394278135647189352671 2893

P_9^6-covering sequence of length 93.

12345678123956487912436879152438792568413597624 1

P_9^7-covering sequence of length 48.

1234567891a32647951a2836495a728341965a2371864a92537148962571a4398276a4538129763a5824716a93587126a98574163a89254738a15642931
5a48763251a49873651842a79318624a537912586a79421637a892415769a234719635827a1859347a1356482791a4637952834a1298639a458961a247a
51629357a81459372a681759342765a684315268794

P_{10}-covering sequence of length 289.

123456271563189a7896a859a4679567a56845a39847a38754936a28634a2749263792843725382 5a184a164917681537142619 23a1782159314529a12

P_{10}^3-covering sequence of length 122.

1234561374256189a7689a578469a358794a85679456a345893a75468a2579348a29783467a38659a43756a2493678239a1756237a169245a168295a183
24a1362a914827a1468257189247159627a481592364197235a1264718a231954318524193862713528 71386719365

P_{10}^4-covering sequence of length 217.

1234561724581734 98a67598a476593a785469a38569a285793468a25479a23687924687354a92357a6245839276543a82945638247a1589476a15972a6
8194a37691837256817492368 14a53269154732a85136a27189a27158926178a43169a42139842173642a18376a19237a1495321846a512496715326415
8391a523a81659137a5

P_{10}^5-covering sequence of length 265.

123456712839a7621498a567498a35746928a756398a2547683a249673825967a18596342879a134579a234568a139562a487193654a178352a41695723
48a152763a182593418a692514a796314875931a265831427958 1624a37165849217a532471689234168a721369a48571239a4

P_{10}^6-covering sequence of length 225.

123456789213a5687421359876a4239876a1245986a3147589a3147289a5614382a5791346a25817364925a71639258741a6892354a76914253a7698412
3a76851943

P_{10}^7-covering sequence of length 133.

123456789a23416759a81235479a81263549a81637592a416587a3241678953

P_{10}^8-covering sequence of length 63.

123456789a1b3526849ab73152694b7835a264179b832a4517b68392a14b563972a814b36725819b34a672895b3a617824593a1726b48a5192b743a5691
824b3579a6128b57496a23158794a2b6138945ab2368741a9b5684312976ba583417ba2986537a248b5169738ab152764981a2b37915486a39b257a4619
b82743156a8743916a8b273164ab75983a147295a8b71465932b1468a25719ab623489b6251a7639b24516873294a56b2871a354b9612385b4912537ab6
815392a84573964213a86b4752938b56792183b7526a91452837a62b453176b98a74639b15784ab3974a51826a349865a34b217839ab249857a1b652487
91653b749153b846793a41b89a537246813a852b7869ab4721567a291ba36295ba15648a3

P_{11}-covering sequence of length 565.

123456143652789ab78ab679a68b5a7689b569b46857948a5674ab38745b36945a397b4839b2a734b26735928a349264a237b253825a923b182b194a195
819271a261b571836a15247168241b7315481a391632791ba

P_{11}^3-covering sequence of length 172.

12345627894561ab897ab69785a9b47a698b579a48b7a568b459a64b7569b358a465ba867b3a8745ab3986547936ab2587649b27a9387b2598346b28a63
47a2894b37926b5376a295438b24a392563a2573a285369427b168245b17836b15826918b29a14725918a24917843a19b23a17935a17823915b23691a54
3726b193426a14832614b2318946178531672481b372153b4195671a2451374b1a256183a61845719285a1b791643ab1

P_{11}^4-covering sequence of length 342.

1234561234758219ab8679ab5867a4b9875ab4697a5489b3a789456a7b35896745b9a348b6a358b69357b4865a9b248a96378b25a793468a29b7563a9b2
687a346b9278a5346b2a874b26853496a2379b42589b147a3b267534ab25387625ab197834b596238ab1579284673b98165b2348a169724ab13972456b1
78a3259b16a7245b1a68259a1368425a614b9238a15962a7b135a491786291a75239841a7234b1769524a1678541967321b863219875319864b13954712
9b3615a84b12694312874319a24358124a361b72531b847613592418653a21b89a71354b7128a5716285b149a63178b589a3126ba5714623798

P_{11}^{5}-covering sequence of length 484.

12345671289435168a9b7568a9b47685a3b79854ab79368ab74359ab64389ab27589a347689a25b67834596b73458ab236985b246795824679a524b78a1
56b9842a67b1859a3b2486a5b24389a1756b3a2479b13687a25398b1456a372489b1674a25369b14a583624a9b1357a2b196572b38a1745b293765a1497
83b156492b37a162583b1a92837659148a6593471a25849176a82b159472368b1425a69132b8741369a7213869a214876b21587932146b732156ab34125
9a3412679ba152347815239648123a6b74518629b413582a4791b8a4162b53471a64835b92173586917324b81a3759a1287b931a68b92714ab35

P_{11}^{6}-covering sequence of length 485.

12345678123956741 2ab8675934ab678934ab2678945b36729a58b367249a8b56247a9b5623489b5a16487b5a236879b14368a9b17568a942357ba14938
75ab1248965b13789a2b13586a4723589a14675b29836a5719248a3716248a3b15972a3b48571a26b945173a6b82159a6b37124a9635124a968b1275968
31a254b63192847b53612a79854b123764815329ba417832b9415a37281b5463a79185a42631b8742619b35846a9b3127594b6a1734582

P_{11}^{7}-covering sequence of length 357.

12345678912a436789b124a5678b92345a68b193457a8b12645937a826b4913a58674b1235987ab12345689ba2347568b921a34567b912a3586b7914a52
38b716a924587b136a497581236a4b7912365ab791843269138a742569b381

P_{11}^{8}-covering sequence of length 185.

123456789ab21354689ab21475638ab29143758ab12643795ab168479a32b5167894b32516789a43

P_{11}^{9}-covering sequence of length 80.

123456789abc1426378ab5c1294638b75c29a468137bc95a62817b4c53a6928b4157a39c68b1257349c6a815b3297ca485136b79a28413c56b72a419358
6c72a1b938c547261a3bc84925671a38b2c476913a2c85b4791ac2653b491c78253a461b9758ac346b29751c468ba259731c6b4a57839261b5a7c324b19
5a863cb291786a4c5231984b7a6c32984716cba9345862cb7a49563128a7c9651b4a2837c6b952874a1c39b5681c372b549ac812564b7c893a25b4c1896
37a254b837ca164283b5976a24851c962ba3751892ba43716589a423c1756a8bc29743a6b8917345c689b72415a3b964c821b3974651ac389476ac5b392
671c859ab64c251a97b382659c74b3a56287b431562cba187432598cb17263594cb2a31598b4a7c5138b4a5c19364a2c7561b48a39712c4ba89612c35b8
6491527846c31a2758c9314b627c359a426b37549a216b384c1725ab83c9276b1a82534cb1a7892561a3b7c28a4637cb9a18475391b6c785491a6c28357
6a91b5c7249863a1b9426c375a4681273b49852acb796458132c9456b289173c46b58a391c76a8b3924176a92c8473ab92c185b76413986b741a2c35967
82ab416c9785b3142a65b9241c8a79451cb6a3284c596a13c8b642819a57c4183a92746c18a57b937c68a531cb268c9743acb259134a72658ac934b2587
3216c8ab94c836215b73c145ab763485a1b296743b8c524a7b963c24579163a827b1a34915c6729a14c675698b324178562b3a9c278acb4125489bc176b
85c73

P_{12}-covering sequence of length 1112.

12345623561789abc89bc78ba79c679b6ca968c5ab68a59c4a876b57c4b958b47a567ac84965cb3975864ab3854783c45b36c28639c2b73a94736a2953a
c2764b2843b2643a249382735c173c146c192b182a183a1c245a14591a72961b527158293152ab1742163418941bc816a791b32c5

P_{12}^{3}-covering sequence of length 228.

12345627895614abc89ab7c896bac59b87ac9b7689a768cb569c758ac67ba589c47b95a7c46b9c4a856ca45bc36ab478c39a657cb39a468c3ab8459b37a
94b8c35a7489a35b8649c37a6457b368547c3567834ac26b93875b6a8359c28a7349b27c6348c279534ab289635b4679546c2387b2a96c2536a4289346b
258c17b268c19b257c16a25cb18a25743bc169237c1a92cb1a954c239a18c2ba17936c15a29817a26b15834b16825918b23c14a25b1972ac139425816a3
2846718b42761ba325719c247a1584c1723a1497263178253916b32741b2391b4c216437154b2351492651784215c94186324a9614c531a769183a2c381
4a5b1382a615317b5

P_{12}^{4}-covering sequence of length 509.

123456178953618abc978abc6987b5c9a86b7ca56b8c47ab9658cab498c7569bc4769ac486957a8b457ac389ba5768b45c79a4857c36b9845cab387c645
9ac36798a3679bc3a7459bc2469b537a9c258a649b78356cb439ac248b935a8b2c79834cab2598746acb298a653c9b268c453b8c257b46389c267ba4369
b278ac63b8a26794538ac26785436c9257a46387b256a4b27953c4a653b7c2a563749c235b68247bc169a245bc139a6257c438ba1768b2397652489c17b
a296c15ba2739c15a8734b526c814b762389a17c8245ac168924ab1c37249a1568297a148c326ab1987245963ab714983256914ca325b9167a2358b17c5
2349b1ca6243ba15762349a1b58423a59176c2459186a247c159b347c618b4327a146c2358712b94517a3c2b6137a824b1739a45b31298ac12b75316ab4
819a52819b6251c863419c58319ba231c98b516a2381cb23418a2751486231c567b12387c143b65142873915273418 5c7a14367912367c1a2451b79a321
4c537149c281496b13876124bc16398a461235ac129746182ba5416c3ac185b2c19534a158321496ac5

P_{12}^{5}-covering sequence of length 821.

1234567128495a16bc987a5bc986a7bc459a8b6479cab3689c574a8bc567a9b456ca9b487c56a48bc3957a846957c346b9a28c7b64589cb734a9c6837b9
a625c89a3b578c26ab78349acb25689a345b7c2948b6c357ab248c7a3548bc26978b5346cab2978c5349b8a257c96b35a8c24697a35469c2b58a6724cba
5367894c36785b236a987246ba38279b546378b2459ac1368945276a93c27b654a38c6542389ac167b9253acb1897a25389c1467b293ac4526789154bc2
6359742c389b147ca253698b15c7a263bc197a83c2697a185b923687c15a692437bc18a674328ab91678c2459b1a368429ac715389472ba1c46823bca19
47b532c84b157c62349b1657a324cb1685a423b95c1a8b2456a132b9841a7b625189a62c1894b637ac8129b7a5129c87314ab862139c75213ba852136c8
5213b9c4217b8c431a967451ca83241c9563a1bc453278c1542a9713c4a671328a671529b641739b561c24a961837abc251749a3251cb832715a8427139
b62c173b6241853a624173b5621a84c76135892713549621 7c489a1354b621ac58691423ab71546ab192c534168b725148b3761ac23461875b31c6845b1
a29483175a42317c45931b8a76951324b85971cb86291a3b461c5a345c81923ac17b9246c913726538b1739a5bc7a69c41b8a725c916784

P_{12}^{6}-covering sequence of length 972.

12345678193456281abc976845bc97a6485c3ba97856c3ab47569c3a8b4795368ab7c4563a9bc45837a9c645837abc24698abc25678abc24569a8b275c9
a84376b9a84356b9c87246b9a5c247b9a8c236b9a75248cb9a1568cb4a2369c75b2389a7c145b9a73246b9c1578ab4625789b146ac837259a6c147b9a3c
256b4738c95b2637c89b1465ac72348ba9175c68243a5bc12689a571468bc235498a167b49253ac8b16957a2436897c14356a2b17c9862537abc14859a2
c178b3524697c15386a2c175b48293c7b4615a83b2679ca1348792b1a6c352478a61c39b84625ca1837b956142c7b361a894c2367ab14298c65134ba862
179b5c3418ab27354c9a1236b87512a9bc4316a9b52317c86b5319ca724318c9524718ca524318b9625317a9642c13ba9826137a9826413ac756b123c84
671259c346172ac359162874b316297cb413529a68c71b24a587136249a51b3742a9813c5b2941865c237149a5b821c936b7a1583927541b6382517a643
8bc12a45793614b52c81793b241a6c85714b39851246ba9c58741325ab74831c2a73b19c2487a69834c1256b941365a8913b67c5

P_{12}^{7}-covering sequence of length 842.

1234567891a3b457289acb6574893acb6758249cba758346c9b7a2358c9b6a42578c396a25b7c346825abc3469287bc3459a86b23c4597a6823459bc1a2
86795bc138679a2b435679c2a84367bc1285a694c138ba7954613cba9724168bca974168bc592416abc5834179bc582314abc672514abc892316abc8742
519ac8734615ba983241c7b963251cab7982315ac79624185ba97634128ac765312ba9748361bac9523416bc85237169c8542317b98652314b7ca563142
79c56b1a87243591a682b4391ca762b3851a7c4236518a7b42c3691854b726381ac5497b1623a49c715a6842c91b753a489712ac83b59142a759631b48a
57c618243a95c681279ba5614c6b53789c12b5a341678c9

P_{12}^{8}-covering sequence of length 539.

123456789a1bc425368a17c4b5968a27c4b59683a2c475b1638ac975b12486ca73b92586c17a49358bc21a69458b371269ac5b3472689c1b3475a8291c3
675b82a14c3965827a1bc364927a1bc35694721a83cb96425183ab9674251c83a49b7215c384279b1635a427c9168b2a743598c61ba43579a1b28436c7a
b5

P_{12}^{9}-covering sequence of length 248.

123456789ab123456c879b1234567a8c91b342567ac91b346287ac91b354728ac91635b478ac692b15478ac26b5394128ac567b3

P_{12}^{10}-covering sequence of length 104.

123456789abcd12457389b6cd24a578139bc264a7d1539c8b26ad14597c8263ad45b791863cad527b9486c1a53279d86b1a4c3529d6b714c832a9db517c
6438adb215c69347d821a65b934dc821657b34a9cd187623ba5c9481d63b52c748a1693bd5c7a82964b13d7ca2964518d37abc24985631a7bd4285c19a3
7462d85b1c97a423d58c96172ab45c36d78ab29431dc6a75b82194d6acb32159764d8a2c1379b645d8a1c7b2463d95a8cb1263d97a54c8b3d2719a4586b
c3d19a2854b3d79c6a84b32175d9c6a3128479cb531da276cb549d82a6c541bd98a3765c421b9a87d53c2164b8d79c2563a41d8b7c39a52d6b78349c15d
a68b329174dc6852ba973615d8c4b7a6159823c74ad56b918c72d43951abc678342a9d6b1c73a5429bdc71a469b327851dab4c32856a9d17b245683a9b1
725c6d498a172c5db83147629c3ab814756d23c81b4956273a189d5b237a8c9d4b12538ac4d1625b93841d762c3b85716d92ac5471d396c824ba15739c6
ba85143c729b865d34ac219d5384a76bd2314a57682dcb4a97528631da947c6b35a829cd16543b2d7a9645c817b23a46c192bd8ac7612398d4ba6259341
bcd86a9324cb5da71683c594dab163c58479dba31c857d429ac816b37d95814276c3adb58476ca5128d346bc259d8631b2a5769cb2da53619478ba32915
47bc3d219875cab21698da72c143b5876cd923b475a893c614bd2a58c374d6b51a38dc9b56a4128db759a634cd217b68a9435c7286a4b317c5d64b287c3
d9561248ca5d972684c3d512b9ca7d346129c8b57d3a621948357d6bc943825a719dc86b47129c56a8d741c9b653841a29db3874ca915632d78a4931b7d
54c9213ba6d974182bcda394867c12b3a456728bca91673d81c492d68b3ca21d567b98c4a137d9c482b5163c7a4bd39678c5a14b972c5318b7d9246c35a
9b8146325d8a9b627314adc56b812a4769b513d4a87b25d4698327adb16c5974238bd19c738a1b6d742a51d9b64832adc61857ba36d29c17a5d3b984256
abc941375b962718abc4d3572894ac63b918526d4a5981327cd95ba483c9215a48d7c2ba14635cbd1a8795b48c163257ac9b6d3a58cb29d376451b8d7c6
9451a7269345bc87a196cd4b72a93c18d5276b34ad295873acb62d1843a67d589c126a47b5ca142d796538714b9da35c7b21d854769b2c35d46917bc2d5
6a793c468d1a35974dc26873ab952c468b352da78c31629bd4a86c53d1a9b6854cd923a147bd5a26198574ad369285c4ab697d812a369b47dc1368a42c7
b34952d17b5a6d827c514b92a6c75d32b8149c7163b427851a632b785c931a467b8a5dc91b2a58376cbda419386a7cd42631a7d52c4b3d1ac6895b27361
d2c58467921a38bdc1453289dc267ba193c25674a98b36d2c596418739da2b319d856b1394d6c5a179b82d4a53b1c78249dc176b8d5917cb4a359d128a7
b654217d3b564a81732d6b9a47d386c9a24d153c2a7843dc5b128d36a2c1843bd9c279435627adbc45893

P_{13}-covering sequence of length 2176.

1234563789265abcd9badc8bd7ca9d8a97bc98b7ad6c97d86ac87a69d58967c5ad4c867b58ab685cb6d756b49d3bc49a5784d65974ac3984bd5c648a396
47d3c547b3854b3ad2c836d29537c2a459c2b634d2b53d2834c264ab27d1b2943a25d1a263a7293725c195b182b39174c183d196b135a136c1b4271c325
17826184258192a81d4516ab173412ca1762d164941a9b1dc

P_{13}^3-covering sequence of length 295.

1234562347891abcd9abc8dab7cd89bc78ab97da69bd5ac987bd6acb59ad879ac68bd5c9867ba58cd698a5d7c6b9857ac65b976d857bd48ac39db4acd38
bc4d975c867a956dc47ad3bc756da498d3a9b47c965874ba865bd36ac459d36cb458c37d84c9b37d6498b36a9458a37b64a79368d45ab387645d739c647
95648c3675b49635cd2ba654bd27a846bd28c934dc29b635ad289735cb28a634bc2a8935ac278536d29b4358d27c346d2ca438b29ad17c26ad18c259d1b
c26843db259437a528934ad1c9267d18b26743ba279c18b357428d16b239d16c54d328a16c23a547c16924c539b17823cb19826d13ac7d9248b16725d14
b6258b1ad247b1a625c139a25b13d5248c1a9265d1c49a15c72b916827b15a24815923c17a23d1492ac14b52794186327916a423814a2b1d723b16c4239
6147d318523cd1247a158d912375124c781a3261532b41a567185cb1734b195c4136a7193451938c1637814d6cb3a15d791568adb5164267a31475

P_{13}^4-covering sequence of length 733.

123456123789a1bcd9a8bcd79abc6d9ab5cd986acd789bc67d9b58ad9768bad768bcd576abc856abd4c89a76c8d579ab65c9da48cb7a58dc47abd397ca5
6d9b4c7d9568ad4b987a569c487ad369c785bd647a9d38bc946dac39bd6849ba73cdb657c9b36ac9458bd367cb459dc38ab9647cd35a9c837d954a8d35c
ba4678d546ac837b9586c7459ab3479d638c745dab36897456dc349ad28cb6457ac348bd27ca854bcd259a834b9d26ca7498d357abd297a635b9c27ab54
68c357bd43cabd289c356ad24c97358ac29bd53689c27d86349cb28a9634dab267d5348cd26b87536bc25a97346bd25a86347cb269a4538b927a85469d2
3ab58269b435ad27695b478d2a6953478b2a6c345db239cd18ba376c5248ad17b9256cd1a9b524acb189d2758463bc8257bd16c92a476359d4825b3687a
23cbd178c265a437d624cb1689275cd13ab4279d168c247ad15cb248ac17b846a91d78349c528db15694278a169b23a891c7b238da16b8249c1a6b345c7
238d157923ca159d238c14d9258b14cd2368b179a2368c1a5b6279c15d8264ab15782369c17da236bd14ac2359b176c234bd1958234ad15682349d15672
349c1867254dc176a2347c1568429ab148d3256c149b2375d1469325a7c139a245d3617b5246d132a581d4723ba1c2d6714bd2739145c238a142b671a9c
2d1b96371ad6381a75423b1d5a2c1859bc1659871459831c7d231c674218ba741cb53218b73d14c62a81dc25317a2561b843217b43518a47315c8b31d97
4b152a431678423c19524719652d139524816324d153a4c1536a21c75813692718d629143b621ad936157a93157bc28149a56147a2b1d684c17236bc129
83b15a8921b3c461a295b1489da14689713bc9d831ca6518743a9164b51d27b3a1947c2d1845cad7b5613cd2a14938dab49c26bcd4

P_{13}^5-covering sequence of length 1336.

123456789a436529bcda789bcd6a79b85cad9b48cad579bc6d58a9b6d57a98c6b5ad87695bac7685bd9c467bda8469bca7489bd67489cd6a478cd6b459c
ad3b89c745adbc379adb4589cd367b9a458cbd349ac8d367ab85469dab357cd8b369cd574b6cad358b97465acd7389ac56379b8d26cab9357da84957db8
3469da734bcda289bc76389a5d27bc8456a9b38567d4938bac7269bd5348abd267ca8356bdc2978ab346c7a9356d8c4379bd258c9b346dc75348dca257b
8c34a5bd63478ad2567ba4396cd2479bc536879d254b87d26a9c5436bad2596c74359dc2768b5439c8d265798cd657b3486cad734569d2845c7d23a8b6c
247ab68259acd14b9ac238b9d17c8a5b29c7a4d26bc3754abc23789d16bac4827d9ac168bd24c986534acd6237bad18c9b264ac98267a54d26b934786c2
5abd1468ac2359bd167a93458d6239abd1589a24b7dc1689b2756cd14a8b2739a5827cdab1579c243da98246b7d1598c2637ad1489c2537b9425a7b38d6
a9147c6523da8c1679b2485ad17c69245bc3d2a97b168ad2346bc17a89234cbd1876a35bc84297ad1836c5248a6d3579c14d5a923678d145bc2638db175
ca2438bc1569d2347ac15689243bad156792438dc156ba2435dc1b96a254cd1827a3c18b672358a47269d1c37b296a8154c92367d1b29c3d18b492da6b1
328a9b136ca2579b1468a235b916a87234cd162897415abc3269a1c58d237c91ab8d2734bc198752349d1586b2437a918b57324c6a1358b291da652c19d
7523acd1475b2d38c1729a3c478123db4916ca73b19c85371ab46281c936258bc142ad9c135ad94712cd654328b71943dc71648bc7135db2619c4d25ba8
1435da61427cb51439cab1527d8b1427d861527d46319ca2451b893261dcb39517da463198d2534b9a1534bd719368ac12356d7142a95316c795a138c94
61578a4c123ba56172c853146bd582149d65ca149853216a795213d864219bc74d813a942615cb3281456b721cad6391547d321549b631d58a647153b62
4175c864132c864d152a743619274581b3da2518736421dba74c12359671b32dc7158bac14357891645a371d928a631b489a21cb645a128cd9ba1748392
17b438d21a438c971432ab71634c58216735a2186bcad215c37ad1263b98174a2b915683b17a28c1bd52487613c2b4d1952ac6714d986a413257bacd316
54b2139ad423c6b17583d912b5c4916ab58271bcd4621a59782c1349d8

P_{13}^6-covering sequence of length 1780.

123456789a2456319bcad8679bcad58679bc4d8a79bc56d489ac57d4b9a68c57bda684c7b5d368cab54978cd3569bacd47698ac35d7b98a36d7c985b3ad
c9658b4adc35796abd34589cbd37a6859b37ad4695c783abd56c289adb4567cad34869bcd257a8b63457da8934bcd7a268b9d73468c9d27ba8c45369b8d
25ca9b6475d8c3b47a98c3b6579d24cb6a73945bcd2389abc16789d5436c8bd2759ac64378b9c2a678b4593acd24589bc16da895b267ad9534b6ac25789
b4d2ca8793d2bac863754abd2c357a6b249c8ad157b9c2a368d4b25a98c4627ad9c34b6ad28579c34d2b697c435a89d6245bc8d179abc24578ad169b7c3
d269abc15789a3465dc72386bad147c9b2356acd1748ba3529d8673529da6c158bda47298cd136ab92457cad13b8796245bc7d1389ab642d7c8b1a4d9c2
538abd126c8a4d159c8b23479ad156b7c23459bd1876ac2348bd5726c8b4a197c862539bdc14869a23578bd1496ac273b8d6c17598d2437bac5286d34c1
5b7ad36289cb14768a253c79d1456ab2834d9c156b782435adb16c47a352bd6c1378ab2d19c856243acd1528ba4c15679a24358cd127ab93815ca672d19
ca832479ca152db974356d8a14359c2a168b594236c8b1457ad62349cba15379c2645d8ac31479b5238d6a1925bc3a16897b2c16d9472358cb1627ad891
5b67d2435b68d124abc531d89b4261acd37248ab9132cd7b41369d82417b9a5281d7b65324cd769314cda652149da83214bcd63517ac846519ac843619a
d742518c9742518ad374b19ad35241bc965271dab3924867d3b124ac936158ba7241dc9b63217cd845216cd935218abc34617da832c17b9d35217bc8435
16a9782316db972314ab952371ac9623517ba623417c9863215dc7a96148d723514c8762513ac842d13b9862a135b9627145bd62a137cb6541238975d14
2c98351764ab981572ac461537b9ca18256d7413296a7b13452a7618b3c976214adb95c1473ad521b4867dc124356d912b873561d98273c156942831b65
4d9813764b29518674b9d152c764381dbc2a36187425391b84c62d13845b2c17a893421da693c41b56d8a4721c38657a14928c6d154783b21d5c4ab6915
7438a621b47ca319d54289a51632d79a153b879415a3287db64213c9b8da341b692da54361b7dac236418ab53c187d49ac2651b3d9c68a5421c39da847c
d531684972c8b1d4375926dc37b4219d36a542b718a95cb628317a4965b7c1ad3859c142a7d346819b258d3ac41b57cd2

P_{13}^7-covering sequence of length 1819.

123456789a23456719bcd8a6574bcd896a37bcd486a59b7d4c3a58b9d746a38b9c576d38bac5769d34bca8679d54ca8b79d345a68c9734b568dc934a5b6
8d2c9a5b6734cd29ab578634cad27b896c5437ad98c5246bd9a7c2458db9637c25abd849167abd8c2369abd572489cbda15789cbd23568acb427569adc1
378b9a4526c879d1456abc3d24879abc16587dab234689cd1a3578bc426d9a8b352479dc6b13589ad72635b9acd14678ba5239cd7b1546ac93d2586a7c1
4bd5893c167ba5d24389bc7a264d8bc1a367d85c24369abd157489a3c267d89a1346cdb9813a6c57d2b3468c197b54a3c298bd156c49a2735bc8d14a79b
c2583647d9b283657c1d28ab9371d65c4a2837b6d145983c2764a5d18b367a2cd1b965a82d4796c158a4b93d27ca849152dbca38156d97b2c158da46239
7b8c1425d9a3c1478d652934bcd1a2568b391d4c75a2b198d74c3625bd14a37c62b948a7152bd98371426bac531978b6d2534a9b612d8ca94315bdc6724
18abc572619ab8342719cd8352719ac836251bda9372618bcd372619bc845231ad7945231cba8642319bd7542619bac743218da9756314cb97523148db6
52713acdb542619cda7b356192cd843517ac9648217bda634518cda9674318bca9526184cda325187b96a32158db743216c9b8a452136d894721a3bd846
92153bd4c781259d4367128cab3451728a6345b127dca435167b8c24319ad652c431a9762c8514376d2a8159346cd21a75964bcd321759684b21a5d7364
2c15b7384ad1762b94358167a942cd16ba392c8156b472a9153bc2d841a75c2386d179c5246391a2cd78359b1a24d86321ca4b396715c2a49bd13826ad1
945837cd64b18c35927a1658bdc271943bd65a1c329d8651723bca618254b637a1cb9284d537a68b145d726c931847b952cd314769c2b3d1a64378cd9ba
54618c9d25a37b491d27a68c59d3

P_{13}^{8}-covering sequence of length 1381.

123456789abc1425783dbca9657834dbc9a672548bd9c6a2358b79c6d2458ab9c7d23486ab97d534c6a8b7d23596ac4bd235798ca4d632578bc4ad32679
8bc4ad12798bc6453a279bc6d13a589bc4d13768a9c5d1b768a9c2534d7689c153bd7a894253bd6a7c1485bd693c2485bd193764ac5bd192786ac3d1547
98ac3b26479d8a152c6b7d48193acb7d251849ca6d2713589cad2b13678c5a241d6b9c752348dabc1523698abd1423679c5814ab6d972315acb8692d15a
7b8642c159d7a642318cb9a642513cdb96784513adb97245168dbc3247159ab86243157cd962831b7ac9654d2138ca9746213bcad87654213acbd954271
38adb5624138cad7526b4931dc8256a7914b3cd567218b394a56d17b23489561ad2347c9815b23a764819b2d5a3147cb289da65731b48cad6193248c5ab
163d4875c1b9624d83a5192764c83b195a762c31b954a78db216ac43985d16724cab9316d8724ca591d36247cb51ad926387bc15d2938b6c74a15382bd9
17c4b23ad891746c2b358da74b6c1935a7d2486927acb8512934adb6

P_{13}^{9}-covering sequence of length 794.

123456789abcd34562891acbd4567398a2cbd56479138cbd5a672934cb8da1657934cb2d165897a4c3d125697a8c4b13569da7c824356b9ad18247c6b9a
d1324876cb5d132987acb451d23679ab851423cd9ab7514236d89bca1725348d9c6a1253b8d749a16c328b7d95461a2cb37d5841a62b3974d81ca256b74
d38ac21657b948231c657d98b42ac367195b82d4a3c6157829c4db37a6812453bac891d263748159c2ad7b6

P_{13}^{10}-covering sequence of length 333.

123456789abc12345d6879bc12345d76a89c12345d7ab86c123459d76abc1238d597a6b41c32d897a45b1c3689d7a42b15689cd72a4b135689d27abc468
935d2

P_{13}^{11}-covering sequence of length 128.

123456789abcde13526894abd7e1c358246ab7e913dc524a8be6173d5924c8b61a3ed927584b1ca3d627e58b9c143a67d52b9ec438a6d5179bc238e6d45
7a9c21836bd4e5ac712369d8e4b57c2a6918e34d7cb2a915e63478b2da9c56e417b8d39a526ce7481d9a5b6c3e7821da45693ceb271d468593eab2741c6
8d59b3e7a4c6d289b1e5a37c649d2b15a87e639dcb4512ae893c7b46521ed9a8c37425ebd6a18c47923b5adec876129b43da8e51c2649378da5cb1264e9
7da351b6ce8924a731bde589c4a326bd58719ce2a4b537169ced42ab3815679c2de3ba1465c8723db9a6e4815c739ab26e81cd45973a2e1cb8d974625e3
c8b1a74d25e9c18376bd54a9e23c61db4579eac6832d415e7bc8a2d69153be4acd872193b6e4c8d527a169b483dc75e6a124b3c9586ead1723b5849dac6
e13b4798a52ced643178a2bed9415387a2e69d1bc532a764e8915d23ac7eb684d291a7365be4c1d9238a675c4912eb3865acd724eb983a61c7e42985b61
d3e274ac96bd8e25a7491bd3c285a617b4d3e5926c8ba473d52ec89b1a4d3526c7b1ea84d62593c1ead4762b85ca3916d78eb5a4c91682dea3541768b9e
ac5d1783246cb5d97e1248b3ca9de7651b28394a6d1c27b589ae413c65d789e123c45a8b76e394d2c8b71e354296b7ad1834c6e297bd1354ce9768d2534
b1c7a98d2e3b56a481973dcba265e4917dcb6a38e214dcba756914d3ec8a72569bec48a7261dbec58437a9e16c583d49a1ebc62374d8a51c96b234e857a
d9c612e7358dbc64a2e35897cb416ead5729c368b415e27c8d63a9b472c81ea356d9b481ec376db281ac4573ed6ba8921547dc6eb2a97815e3d64a29c51
8db4a63e7952b18d3e97c5b68124397dbec8265a91347ceb2a95d8476bc13a29d4e7861ac5be239867dcb52a493671bd5248c69a73b5dc4e9a175bc36de
2a1498c3675e2ba139d8c574e6b29dc53a8e79b4d61a35ce24db61ac98e473162bad584e93271dca4569b382dce154ab83729651eb8a3dc7254b9136dce
8a241695b7eacd349682e157a4bc986152d37ae846c95d73b2e6a491c7235eba8d1c765be824d173cb8546ead9178b534de267813ba9ed516c2a483eb59
c12da4e8b73596de428cb9a51d7e2c349587e6da2c938b1e64dc728953ab4c76d1382acb7691e3d24b7a918ec5d34697b82e1d5a493e68c1a27bde56ca3
97d2851c63b9ae4528679cbe453182a67945bc8d316a7e5928db413e52a9dc487e2361a9bd8e56c71342e96a8dc7b4519e2a7c6834d9125a7b38ce425ad
b79863145acbd972614859c3eab14298cde75b21364e8a5c7126d349be17584a6dceb9712a3546cb91e87365b24ec819d6a7245eb9d632c41a7ed83569a
b12dc497ea6bd2834c1a765b24d938ea17249db53ac178ed54267c9b135d8a42bc731a9856e23acb9875d346caeb8295317ed4cb8965ad2e318954b6e7d
ac19354287da6b51e32c4da7968e123cb4da9e873b214d65ce971485db63a78425cb3de81624b9a315c26db83a14e7628ba1457d3269ae5bc182d465b31
e7c9a8541edb67a841dc9673b24ec61897a5be6419d73a6e5412873cb6592a4d3c89e2765c183adb276c48915e27cb861d9e3b24ac1e6d8592c3a76e1b2
9c35864a9d17c523b864ac52e973b456a21debc9386a2b1c9473ed86b5ac7e2d615349728bd5c93ea4bd5196c34e78a2b5639ea1b47638da251c473dab9
6c24537aedb92153a8e7bd15ac6e47b213956edcb43185d679bea3c8415d2967b83a2ecd913865e429c7be84263ad1b7928cae35b741c2d6978a53bd21e
ca59437bd8e29413acd5e79461a2d8ce73b46d2a5c8b7193e5a48cd723b16c45793b1a82ed653cab768194dea368179dce2ab84173c9d2b165e879a46b3
cd879241beda963c85ba126e973ca426deb38914a57edb9812a34d76e1cb84da9275816b4cd9e3852764ac1b9d386ce5b21d4a867ec3d49518be36cad71
5264ec9ab74d582c36a1945b2cd3a17e4bc92d81a735b629dae48c152976b8543e9d728615ed47cb968132dcea5b8269e1c5b7a249eb861a432ed8571a9
c63e5128b7de6932c148b59ad672c81395aed82476a93d518c2a376b4dc5a378964e21d7c8459a6becd873145e9d62c857e3bdc2497536a21ec457368c1
bea4d379cb526ed971b48ec26a915dce7263491e7da24b9c37e68a12547b3d28594ecb135a4298ce36d4a9b218c6435e97ac1bd5e932c784b59e321a746
5e8bd17c5e6a83d91c4b87da563c2bd4e8759c61d43aec572481b6d79234a85d61cb3e9657123c8dab4532e7a6c4b5e18a6c439ba78de23641c5d296b84
3e172c9da8561b9ea84d6b2c1a7e3b216a74d5ce2b8736a54de813ac9b257d6c48932bd16589ecdb27869a2c354b7eacd21639a48b275ed398ab2e76c59
3d817acb64219de538b279c4163bde2a9783154ae96723bd5ac89e7b4615d73c9ae2b1845971ca234e8dc967ba132d4c7398b6e5ac4317e652a8394e1c6
2d85a19e28cad419573a2d8496e5b21ad763589b1a74dc635b21a8e79dc6425813de7b2a594d67ba5e1d3728b6e1dca98b4526ce8d3b4915c62a1e7b4d5
89c36124597edabc8623d95ec41b2768315adb87214ac9d536bea4871c36d5e729c81b53d7c26913ec75b9a41e638d7a94c8b321547869ca574239ea68d
134cbe9623acb71d2953b478c6ed5a429186ba7de913b5da648c75be291a748ed532ca879462bde7398b421e3a5716cbe48327d5b461ac293e5876b9143
a7dc89b258a349c17582adce4936a25d1cb736ea95dbe61a732b4d8e167542d81e9cbda23689174dcb59e86a42c5b3e64917cade4b697d3c458aebcd389
247c561b927ec8645d31e7ba8c2673ced18ad765234e18c574a6b91d85ea926738dab2e594c361ab9d27458e6a32179eba42cd157e9283175adc6b81245
a769ceb276a1c9ed35b9ca682ed9b156bd78c53792846b93

P_{14}-covering sequence of length 4230.

1234562356789abcdebc9ebade8bd9cae8cad9e78bc6ed7ae98cd69c7be69a78c6ad89b68ab7968d5ea6bd7ce49c5bd49a5dc4b85eb4985c76e5ca4b67d
956d4a857d4e865c47e39b5a746e3864ae3d84c63b759e2bd3ce2a93d54963a54e3bc2983ad2974853b457e2d637c2a738b2953c82753e2643d2843c267
ac3942734a28e1a25e1926b524d1a3b187d1b24e1d25c1ab2391b72dc174b16281a561c241852173d16a2381973416751bec19a419d513e613c4159618c
b18da71e2

P_{14}^{3}-covering sequence of length 378.

1234562347893abcdeabc9edb8cea9bdc7eb89dca7bed6cba89ce6abd78cb97ac8de9a8de79cd6ae978da67bd96ebc5dea78b967de59ab68ed5bae86cd5
8bc69ad57c968ac59e867ac5d9867be56a978ce4bd867cd49bc57ea49ce3db958ad4cea58be49d75bac48de39ad47b956de4abd3cae76cb48ae37c856ac
47be3ad567e948a579e3b8756ba479658e46c9587e3d6748b65d874c983bd54ae35d946bd37c6489d37a846da38ce26dc39b548c36be2ac937b54ce369d
2bc657a46e539c457e368d2ca54be35cd29a645de29a73cbe27ab369458736ac2b9e56c43a9547d358e27c536ae2d85467e2db34cd27a538cb2795367b2
8a639c2ed436e28b536d2ab438d27b435d26b439d25a437c289e1ac279435e279d1ce47d25b64e926843ad1e934852ab38652ce16d24be19c24ae1bd289
367428e17a28e347528c1ab35c18b49a28d19b26a17d38c17926c4db15c23ed18b29a17c24e13c45d17b25938e1cb25e13a24d17e32ad1cb429d16c238b
16e259c16b329e16a427e15b248a17b32cd18a23c19a326918c627d16a52981be324d815723d16b471ce581ae2549e15627a316723915b321eb7918abd1
3ca51b42387124c5167829431d524183521a4361e74816328159264e1862b1a945137c51d9a7419cb715a32c48156391b674156d8139d41e5ad146c31b4
9615b871dc4a1597314ba6197e7c61a83721de5624198ac2

P_{14}^{4}-covering sequence of length 1032.

1234561237895abcde9a8cdb9aec7dbae8c7d9be6a8cde97abce69adb789ce67adc968bdec68ba976dcea59bde87abc6d9e87b6dae589cbe5da986c7be5
8dcb769de578acd579ac85bea7689eb568dab57cde4b9cd56abe489db578ce49ad8769be47ac965bde768cdb4a9e765cde469acb47ed6589cd46ae8d4ca
e856cbd479e8567ae45bcd39ae874dae579bce38ad7569ce458ba74d9ce38db746ace3dba9847cad389ca45dbe379d846bed398b7569ad37bc846adb38c
a765bec37ab8659ae378cd459eb36cd8579ce35ab9648ed37c9b457de368a9457dc369e7458ac369bd458ea3bc98467ad35c9746bca359de2bcad36ec84
bade28bc547eb8369de27ab9458d764ce538ba6459e8367ce2a9d645ac736bae28cd5369ae25bda34ecd29bae349bd27ce5a49dc28ab934cdb289e734cb
e279a835dec269d735be6478c564de356ad27c9a348eb27da5486dc25ae734dbe268cb3579e268d7349ad278c9367bd259c834abc26da8347eb269a7358
bd2a6e7358d92acb5638e45679834ec92b78546bce258a6349de287a634cb8296c534d9826ac534be256db435ae24b96357ab248de16bc259e6347ca258
de1abc276d5b398e257d463ce24a98e2c793468d239be1cda248c5367be15da346eb238de19ac245b6729c53479b23ace18bd27496a2c7835694c287b34
59e237cd19ab254de169c4b2786e19bd26495b286d5348b25a64e28a3456e27485927684357a26b843dca147b259e187d25cb169d25ae189b24ad158c26
4b73aed147c263de17ab23de159a724de139ac253d7426ab18cd239ec1d8a23cdb198c236bd1a8923bce178a246dc198e245cd1e7b234cd1e6a235bd1a7
c235be197d284ce16a93b7d156c243958267c189a526ec179a236c5eab146e3258b17d9326e8a9bc135e724cd1879425ad1cb7236e1ba8275cb1869234a
b197c2459d1a78325ce168a3259d16ac2e19643c8715ed237b516ed271c8b32ae419d2348c1a7e4239b61ac8561ea732498617ad432e87146d23ae148b7
914db231cab451c98451ce8321dc95a1bc7651ace4b1d6735296b15d8421ec7351d97641b98341bc9531ba7541db8631da7231da9231be82519ce4716ca
4218b73619e74215ab6325d8164c57126b341d873241b95231e94261a94351ca43268b1546a7124d56129c3612dce3145b231a48526e1752be149568127
4631ca82917b6841236851a3762d1ba364185324c19752d134a52713e4251876329174563b1c743d16983a1e5473185e461ad2e4618a914d352a186dc31
89d5ecb1283971ea29341b759a31be78c41b625a1c7324591a24bc1d67e3186c92b1e682c1356a91785d31c94a7139bd51c24891e3567192edb138ad14b
7e5916c739158bae218d67ba1847526c419d82ec1385b1e4382a163eca91635db21734961d5a7819bae571cd2b8d4e62cade53

P_{14}^{5}-covering sequence of length 2070.

1234567189ab652cdeba98cde7a9bcd68e9bcd78a9be6cda8be67da9c5e8dba769ced57bace6879ace5bd9a867bced469abe758cade479bcd5679bec568
dae9578cdbe4a9cd867b9ed568bca9578dbe459cdab489cea65bdce497abd5869cb7586ace4789dbe46acd8749acd6578be9467cdab468cde7569ca8457
dea8469de78569ae7458cde369bcd458abe37c9ad28bcea549bc87a46be5d3cabe8467dab5498ec657ab98467dce359bdc847ebad36ceb754acde379abc
4679ad5468bce3d9ab6e57dac38b9de27abcd349ab8e3d7cb645ade3689c745ed9a387bcd29aebc358ae7645cdb7398cd4659bad378e96457ace298da63
79be5467dae349bd7638ecad278ec6459ae834dcbe269dac3589ed437bce826d9c7358bde247acb3689d754b8ec3976a8459dc368a54ce6d358b9e267cd
a346bce259db8437eca8269bc735aebd267aeb3489ce256ab8437cde2487cb635ae9b287ae9346dbe7358cdb269aec3568b9735da8b27c9e5368bae2579
bc4368de245ab8736eacb257de936a8bd249ca75369de24acbd529ace4358bc92a78d5639bae2479c6538dae2598da3459ed2678ca3469dc2578e643ba9
c2578a94367be25c8ab435cde2467bd35a69c248ae6537dae2567c84396eb2587de345b9d2768b4539c8d256ba74358ed26b598473b65e249cb65349ced
23abce189ad3c28e973468db297ae6345dac2379ba5436cdb2478e5926ad7345bce2349ba72689b4c2a8d63459ac238bde17acb2638d7c245bde19acb26
457a3e29d7c3458ae2369cd18eba4d279853e46ca2d57b4e2c8654937ad2469be158cd243ea872659e3b2c7d6345be28367c5248ade179ba254d96825ad
ce16bad2439c7e18d9b2c38e4627a8d34beac156de2738b9e2457ca326dbe174ac2639b8425c97346dea154bd2639a582479be168c9273ebd924a863759
ea167bd253aeb4268a5e17bc92438d5726cae187bc2539e7428a9e15c8a264de1c57a3829e54367ce1489d7325ebd17ca83945d72ecb61947a5239b7456
d9e1c4ad65279ad18bc95e17d8645ebc163ae425689c1ba78234ebd19ca8253bcd179ab6427ce168db4e1c97d28eba1649c3e18b97256cb184e92637ae1
469d238bae15d9c2478abd169ea243cd19a68235bd9178ca236ed17a5b362ce1857b362ae174892635de176cb4325acb1679d423bc78195ed2347b85cad
91458b26cad1859b2367c149db273ace1458d293ab8d153ac2e179a4238bd1459c2638d19e672459bc16ad32457e168ad2e19ca7342ec69157be3c19d76
2349ac186bd3e178692ecdb1329e8614abd351ec863247cd1b56a4237b68149ae251cda7425b8d176a92c1d875246dc138ea2c1bd78234cab185973d1ae
85273ce6512bdc4819b76351cd862435ce127bd361ce9a381cd76421cb963a1dbe4623dc1a768235cd76139ed241acbd391ae68341bc9a351eb86721dca
8431de97241bda7341cde82519ca63817c96231bca5421be973418eb63918cd5431bea72318ec7231ad87421ebc53618da52319cd57418ba5291db65321
eba65814eb73216d98431cdae2618db94613ed82419ba82415ec73412da93516ba92316eb74c158a97316cb82413ab82614ec825319ec28b19ea23518e9
43218d956a17b85621e785361eba95831bc92561ac84e312bc768142ae5714dae98712de5461ad752361ab723541ba7258136972514c862715ca6421dbe
859614cd52314ca673148d56234b951264d87b13246e71329d54176bdc912435a91b4375e1962b74159c8321a945623ecd17238691423bd5172469a1384
657913b25781432cb81a35c62d19eb34c17982345e19c24631d5a48e135d26a1735e26a14537921ac4e6915348a2d19654b21c753428165a723149d5e13
4a26b157d4391aeb42817c3596128b3541c9a7531d82c4917bd349167ecadb1c546e21b85c7a1b3987d16348c51bd9a47138be27159bca621743eda125b
3c91725d681e475918d3bc65718da2be1594a821c6547a12de984c61db23ec416b3a871592a8714cde2617eb54613e987a415c36128e7acb416d3ae5167
3dab2197d4e123d751c28bd14a687ed416b583ea1795d6e9ac231a6859b1a74b316d95c23486b5d1ec567891425eac31478bc9e46219b3845d21a7b649d
1a8ce9d328e56acd314b6e93b517c69a82e9bc14d5aeb

P_{14}^{6}-covering sequence of length 3120.

1234567819abc256deabc897deabc697d8ab5ec9d8a6be7c5da9b8e47cadb8569caed45b9cae687d9c5e687db95ce67ad9b5e68cd479aeb6857cade468b
ca975e8dbc4a978ed36acb9754dea89c37bde8946cbde7846acbd75489cebd3689aec4b769de5847bade3597bc8645adbe3697adc5469beac3d78b6a49d
c85e37abc9d26eabc85479aec3d58b9e4675cd9b3678aec546bd8e35a79cd28eab97548e6cb3798ea6547cbde3659abdc3869abe4d389ca764589ce367a
bd94567eadb458c9ae346bc9d258abce345d9ba8346ced9734abcde278a9c564b7ed35c6a8e29bcd78465abe79348dbce267adb85369cde825a9cbe3487
dca63597dce246badc35879ad4358eb9c2768de94357bdac2689be7536acbe2579da86357cbe8a3659bce247dab98267dac94368ebad259ceb34769de25
87cbd3458ace2379bad52689bc45378ebd2a49ce563789be256a8d7e24b9a85376bde2489cab35478dac2569eda435cb8e2d39ab8e257dab64358cde246
9bac7258e9a34678ed23abc9e158dac4e268b9d4356acb824d9ae68257bc9634a7bec2865d97438c6bd297aec64357ade62489ca53648d97b2568dc7b24
a8d65e347a98b236ead9c1b8ead374c69e235acbd179eac24679da325be98d16abce29487dce23b69ad4258bce1697ab5436ed9b254acd9327b8cad156e
9c3a28be763549cbd2358aeb42768cd3e2a7b693548e7d239c8be17a6cd2537aec42568be3c276a894e257bcd364578c2d3a6e84257acde149bac27369e
d5248acbe137d9c5426be9c1d5abe243cda8e1679bd2438e9ad1768be3d259c8734abe59276cde14589a6b247cd5e1689cd2437ba862459ead1b879e243
6bdae1578bd2435ca9e184bcd263798cde147b9c2536dea4257bd93628aced146bac25379d8e164a9d38256ab7e1489ad2c36b98e157cab24368c7e1b59
a62738ec942568bde14a79b2538ac74265bde137ac9542678ad154ecb37258a9b16c7de34269a8d157c9e243bad9c1678ae24359c7e1468d5723beac146
9be352479adc156bea2438cbe12ad9c3e16ab9c2346ae7d159b8c24379be12d68c3492a7e83546c9d81a47cb5e164ac823569a4b235c89a14bdce25348b
76e159d4c2367ade12489b5d167c8e2536db7a1958cd3427ae56384bda1c389e2d1a7c8e2436bc7a159e6c287ab91e38dc65247e9a1538be27456d9a137
8ed5624ca8d1327bc681497de25389c174bad26539b8e1456ad273b8c1d5e7b263498572c6a9183dbc6527ae9cb127ae693187ce5423adbc1678ad3426c
e7b135ade2c139bde2c189bd7351cab873619dbe53c19db68457edc31469b87215aed93245bcea1248bd67315a9d42368e95174bc8e513adc8641bad783
41bed76291cea87341cbd572918ae6b241dc9a5236bec1527da96148bec271d8b653279e86143cdb8216acde3418ac97241ebd87291cb685234de7b9c68
1357ba2d19c84726e9ad134be9a8512cbd97415abd872149bd65213acd74516bc97321decb6341adc962418ec96231bea864715eca6321deb85421dce96
4518cb973418ec953726dec51329db85134cde2815ac67321deba68514cea93218cd75341bca953216de953817bc65324ebd1567a9281d5679381ab6472
3d9a81457cd2314ab892316dbc54219eda372148dc673149dc85213aeb742159eb73261a9b74321dea865913dba7c214eb967215ea9643158d9742356ea
81234de9716348c9b1235ad871246ca351d4e9b65138ca76413de9862134ce9d25176ec3251bd8a32619cdb43e16ac8b32165cda4e215cb973a12db9436
125e983412cdba834257cb81d456abc12349a781235de46127b8a45132e9b4d18362b4d175e6823479b581627c93415eba938c17245e3817a65b241ca89
65217cd9a85312bc74691dbae3c51276b4351d9c6832154ca7321548d732614eb8329156cd4231597de4c12659b43182d9764152ae7391426a8351479a6
35e147da3521948d6521eac763d192a475631de8726b135a9482137b9648512cad4761258ad4316597b428c159763281e47d52b314c6725381b467d3521
46a87321eb4563c19a4762318ce7bda12643b5a1d629435c1872a45d1b92e8541c3ae2716489a357126e495317ba64519c723a81b6d5243a1b78e2514ab
8c3de126485b71e346c5d183e4761ac592e18d7a469e12375b4a18e36db712c453681eb25a941c7e3a5d16c79e8412a6dcb7319e6c4127d93bea21493b5
c61a8e23c41b8629ae31548b7d195c4be761538d2947518c6e2b571a4e8c6d1b59a84271ec3b49d71a3c96b1e523d7c6512a49ecdb2a51978e2c51347ea
219b8d6732a1e64d3871ca2b3576c4192b6ad319c58bae1d249658adce19723815e6a713b952438ca16275db831e7956c83d791456e7a9d4281b6a53e12
8d67cab14e5d39124c6aebd251c689e34a1b69e285673a42d8e15792ac13b47ad51638ecb7461d938b7ac5196b23784e12bc6a758b9ce6d32a89dbc145a
938de71c8ae692714b258acd34e9867dea1b7894

P_{14}^{7}-covering sequence of length 3607.

123456789abc34652debc9a875debc968a7debc469a8deb5c67a9deb4568a9ce74bd86ac57e4d89ba5c37e89da645ceb98d367cabe84597cade346b7ca9
e386bd5a79486cdeba35789dcb4657e98da4c37be9da56378cbed543a9cbed56487abc9d348e7bc96358edacb495678dce43a69bdce24789bad63c579ba
d438679eb5a3468cdbe27a98cd5637be8ad45379ceb8d257ace964358ace9b24d8ace7635d9ba8e26cd9b74356ced9a247bced59348bace563479ac8e26
5dbac98276deab53478ceab26598deac237b9dea6248b9de3a268bcd95346ab87e239cdab67259ec8a4d267be954a26dbc784539eda78256ced94725bac
ed8179bace26478db9e235acdb86245aced832b69ae748356cde72a4589d6b237ac89d16e7acb45d638ae2795bd8634e9cbd2537ae9c6248bae57624cb8
9d3725ebc86914adbce26379dbce15679da4e25bc9876d25ebc38467aed25349ba8e25369bc8a25769de34c28b9a53746dbc2e39867ba254cd89e16b7ad
82546c9a7d234be9ac1578dbe3425c9dae148bc9d63257acb9d1487eac25369deb425869ce3a4258bc7e14698ad352c6b87d3429cba6d1758e9b2437ace
d15b98ac6e174db9c25387deac16589de24367ab9e15867cd2435b897e1c45bad378c26ead354c28e6b37d4589c23aeb67d1598ae627439cde1a587bc4d
23a8eb7c165a9ed2b168c9e73526abde14587ac932468bde1594c6b3e2579c4836d2a9ce1746d8b2537eac49256b78e342c9ad681b7c9a53426bd9c51a4
6bd937e1ac8653247becd1a3869e4d157c8a64d1bec8723d4ab6c17859db3e178a9564e278d95c1e3b8a964257deba1469c783b246ace139758d6423ae9
8d1547cba9e136dc7a45298bd7a1546cbe3d1289cab3d1689c5b3427ae8d136b5ae24c367da9e145b8cd2e197ba5d234897ae126bcd97514abe8c2356ab
e9c1258adb37196cad425379ba68145ceda732698cde13457b968124cdea63159cdba231e9cd752348e9c672158bdea3217cbde53419acd853e17ab6823
5de9c1627abd381469cb73e148ac6b3e17d8c492b17e8a562349eba1526cde3814b9a7c263589d4e127ab8c4315dae86241cbda872419bda672419bed6c
3517bae46351dba986271deba84251cdeb3a42179cd654318ebc972618dbc572319eac862347edb591236ecb47125ad9c3714bad982314bce952714dbc8
63219eda7834159cb76321aecd893b156e7d83419aec782315bec962413adc872513adeb947138dcba45126e9b83412aceb756314cdb95821aed7cb9521
4dec765214dea9635218ecad64512bce8674513da9872516ebd8c3512abe863412dbe98765c4132bda965173cda864129eba735129de8736125cba93416
7eda534129cda874125cda9671234eb9671524ebd3a6128ce754312dce8564912db857341ae9b653841adb9523816db974231cde96428137cba6421359d
b762134cd98561274ac98b512437da691845eb732168c9a753126dbc7451328ea9451328dea4516972cb4e315a8c762419e8c73d61ba5c8e24137ad8652
14cea9d7621534ed962153ace74b1823ae7546128cba5731289ca5631428d97631425be86317495e8631a29be8671423bad5c1723eb98514263c9751428
ab6351c48a962731ebd8427519ba8634712de85637912e85cd743a129ec56431d2bc9785136a2db48156ae379812cba6395147d86a2314e7c6952d41738
b6952a17ce6453219be7d6c81243a576b1824ae9d73154ba29cd17364b925c13d6a49521c836d7521cae43d8b165743de291a574c63b2915a84637c1de2
8493715b26a497813cb6e512d37ba98e416c37a2541edc6b2751ea4862753e941c2856ad1be9c32851b4de3a86127cb458319d6eb5471c3a962e147db36
ac1e5498326b715d498a2ec1b638425cd19738b24a513e627c8549b16dc23a9481cb2eda9186234ced1b92586c1eabd5246739a15b4876e1ad235c46981
3a5d2467a1eb9c357d148e6253d7bc81a395eb247689dc1327489a57e126d93b4a1c27d68514cb736851ec4ba291637ce5a687bd132c487a531b9d46e81
72c5a9be183d257964be3c1a82bd6e13a94527eb18936ca72d1e49c6a257b1d3ea4621b893cd4a5827391bacd9e863c15db8a274e3cd519286dac514e79
2836a5d47be1c3586da7e2cb58136ea479d6bc1e2734968cad71e59364b78d1a324bc53e892da647951b6d2e419ab68de37bac4682

P_{14}^{8}-covering sequence of length 3181.

123456789ab2345671cabde895746cbde895a46cbde387a96cbde354897cbad356e89cb7a4d6e389bca5d6748e39b5ad768ec349abd67e28c4ab9d6e257
8abcd634579abecd625789baec435689bde2c37689ad54c3e78bad56493e7cabd546298eabcd14798eab56347c89ade23567b9cde24587b9cde2a4578bc
d3926a58bcd4e23798acb652479daeb3825c79aed68234b9acd5827649ace5d23b79ace6528479bda3526e8cb7934d2a6ecb854723adebc68195ade7b34
6289cdeb71658acde342678acb9e14678dc5b2346a9ec7d2358ab9ce12478abd6e13978cbad263578ce9ad1647bce95a2367bc4ed52397a86e452dbca79
e1546dcab392e478c6d539247acb8d13697ace5d13698bce254369adce1b73569d8e24735b9cde12768a9c4d157b8eac3624579bde16285a9bde14378a9
b6e154cd8ab7e2356cdab179856cd3e24b98a6d73245c8a9e13d467bac2e15d89b7c3e154d9a8c2e175b9a6c348d2bea7915468bda3e157698db234576c
8ae124579d8c3a245b9e8d1327cbae451687dba32e16c987b52436aecd152789ae3b1d4c89a62e1dbc85492637ecd54a183be6c75429da8e3c172b6d893
e1457ac893e165bad472e16cb8a473e259b68ac134d7eb625834ac91d267e4a5c1db983472e16d9854b371ca9d6582437eacd1924b7ea861325dceba413
698dcb251a678d4c2b15a9ed683714bcea582314bdca7523189dea754318bdec752316bde9c52416ade9b32415acd9762315abd984231eacb9762418ecd
a763214ecd9853217bcd984621ebcd9835216bec9743216bdc9743218bac9764315bec9a624135ecd8724136bea97241368ce975412b8ca9754162bcda9
751346edb927135aed8642139bce8a62153bde8c671254bed8631275cea9486135cbae79812654cdb39128a67c53912ed6a7534128c9d6734128abde534
712acd85643719ced85642173cebd9461a82ebd7593412cbeda83912657bca38126e79b54ad1632c89e5416b3a79d2e148bc7563d1429ab785e13624dab
78519364abc271539e8b42615a9d8734215ced97a486215bc9e48a2361c9b5da43716ce58a923146b7e582a1436d9e5271438dae6c925174db83c261e4d
b59a231ec6d874ba1395268e4a179c368b25147936cda2b174e359a2c1d6b83542a179d6b534e2a1d8c5396421b87de35641cb2a98d751462ca837d1954
cb687a12d9c43e72a18b539c76d1e2543bc871e256934cd1ab26e54c813b2a657e14389cb67de521a849b3e5c1da873465b1d29c784eb632a1c794b562d
18739e24c51da38672e5c4b31a89d564e3c18a7db523e14879a6bde15c47982a63be15978d46ea12c97384da16b2549e38d16ca5be3127956ae43c1b726
85deac4b391825d6ecb391728ad64c1b8ed326a9c5184b97e63c58d126493a7521cde4b689172a5b4d36917852cd3a1b9e8523c1a76dbe29841a93d25e7
c4ab6185deac29b37d1568c493e2bd175c34a681e27bda8c34791bea67c9db243586e1

P_{14}^{9}-covering sequence of length 2161.

123456789abc2345671dabce8956734abdec986534abde2789c35abde4687c39a25d46b8c79e25d46abc79e23548dac679e23458acb7de23468a9b7dc52
4e68a9b7cd35468e2b9adc3567e289bad45c17689bad43e15789cbd43e25a698cbd1347ea98c5b6137da9ec548127db9ace561378d9ace561478bd9ae56
3278cbdae1465978bc23d465978e1c2ab6d573e148cab96d273458ab9e12d3768abc452e3d79abc152648e9a7d1c2658b9e3d1476a8b5e2d134ca9b8e2d
1647c9abe251687cda4e235697bdc1423e897b6d5124ac987b3d152ace98b471635cde9b47a1268de3cb5a1269784cd3a156e98b4d2c176ae3b4d291578
eab3c6d19278abe43c6925d8ae71364c95bae71324d8c6ba51327e8dcb691453a8dcb29175e638cbd91254e6a8b3712c4e698a3d12b47ce563a1298dce4
6371b9a8cde457213a8bce9672514adceb639271a8ced5b42613ace87542613b9ad754c213b9ad6875213b4cea8d62159b34ce86215d7394ea6c12d5397
eb8a16d53427ecb86913d427eb8651c34a729edb15364729ecd1538b47a9621c38547d96eb21ca4875d3e621bc94a53e182d6c9754a1382b69e5471a382
6dc5419e3826d4bc1597e3284d5b19a63e27451b98c3a264d1eb853ca9127d4856ea9314c2b7d8561394a2ed78c6b5912347da6cb81459e7a2dc1365497
e82ac3154d9be238a1c5642b7ec93d185a427b61de3ca547b86d2a9ce51bd472a83619e75ad2b8c369147abd26c3519782a6b3de578c1a46e3b25947dea
8b3

P_{14}^{10}-covering sequence of length 1110.

123456789abcd2e1465879bcade3465879bc2de3465a891bcde3467a289bcde35617489abde32c16789abd4e253678cabd142536789ced1a4536b79c8e1
24a56b79cde123458b79cda1e23586b9c4d1e235a7b896d1e23457c8a961d2e457cba83162e4d7c9ab31526ed78cab413296d78ca5b1e4392786a5dce41
32986a5bc7e13d984a25bce631749a28d5ceb1346a298d5cb137642e8d5ab1947326ed5ac19b872456da3c19b784256ea31c7984be2d6ac37594b2ead16
357c492ba1de3684579b2cae31578d269b1c5e7d48239a61cdb783e592

P_{14}^{11}-covering sequence of length 427.

123456789abcd123456789abed123456789aced12345678b9ced12345678abced9234567abc1ed823459abc61ed823459abce71d62389abc7e14d6238b9
a7c51ed4689abc57e12d489abc76e513d89ba76c543e2

P_{14}^{12}-covering sequence of length 168.

123456789abcde1f2457389b6cea124d57f3869ce1b2da5f3486c7e912d5b3a48c7f9612ed53a8b7f4962cd153ea874b62fcd593a187e4bf6d293ca857e
bf14d632a958ebcf4d736a9152b8f4ed3c6a1729bf548ed6c1372bfa49e56d8c13b7af9e2658c431bad9f7e25c643bd891f725ac4e6b391d782acfe65b3
1d4972c8ea5631fd9b7c2e486a1fd5973ce4b2a8f1567dc94bea328f167c45be9a3fd682741b5ce93f6a2d48b1c795efa23d481c6975bae2d8f1c39654a
27edb8c3f51492a7dbc638e41f9a5bd6723ecf894ad167b3e5f2c894d67ae1325bc89d6f7a4e352891cfba64de573289bf1a4cd56378eb9f4c1256ad837
bf9c2e156a87db492f3e16c5a74b283e19d56f742ac3b9d58fe714623b95cf8a714de2b956cf73ade81b24cf573a9861ebdf27c5984136bfe2cda971435
b86feadc17459268fbd3e715ac946823db17acf594e36d812cba5f479e83612ab54d9e8c3726af51bd4e3926c7518bad4f2e635c81b9a47edf5286b194c
7afe385d26b1c4fe739d86bac14e52973fdc8a451e9b32fcd8574e6a912fbcd8743a61f9ecd5b78a21394cd5fb76e89a4c3d52761ef8a93d27b546f8ac1
3e295d768bc43f21a5de67cb3f184aed962b7f15384ad6e2c7f93581d64be7cf9a281654e37b9a286fcde53b942a6fc1d3758b426ec9ad3187f42be5acd
69317825afce4619d28b75c3e4f612da958b3f4e276da9158bfe3c6a742851ed3bca4f968517eb32cdaf49178e26b5cf43a17de2895f436abe127c8df49
36ab157cd24e9f8ab65c3741df2ba965e8c1f7243a9e6bd851cf23769b48e1caf56d2b431e79ac682b31fde4c952ab76f8de194c532ab6d19e74c8523fa
be716cd248f39a517edb4c3928f7a65d41e3cb8975f62ed3ac74b16295f3d84cbae75f369142c8ab5de63794f28a5d1c3b96f428d1e3c7b96fa45d2c8e9
1fb647adc25e3b849a617f2d3ec8a61745fbd9e32ac7184d6f53ba92e18d6734bafe1c2689d5347abe2f189635ace7d4f9b38a157dcfe69482b3a1dfe67
942cb5daf8e76431c2b98ead7451f3b26cda48759136e2acbf87d951624e3afc9b7521d3a4ec86b591f73e28d64bc951fea284739db6cfe827495ab1c3f
68de95a273b86f1d9c2735a6be81f4d3976ce5b124fa38675b19def3c6248a5be17cdf6425831be97adcf5326be149ad3856fc72b1a4e538c76b19d245f
ea67b8192453dfb7c6a82143e9d57af8b1c6e4d25879a1cbd6f34e98a5c26d7fb143958ed76caf91325b47c89adfe516738c42bda91e3fc854b6972afdc
51e8374bfd658ac192b37e6f85a4123ce6bd59487f3c1a5e9bd826c3a5f94d187b2e5c9fa6d8b3159fc746ea12d83f5b74e9268a3cfbde754216af3987c
e5d12f39ba6754dc1e82a6b5d437c918eabf367d298caef53d4619728fbc4ad965372f18da96c347b18da296c45efba3d28c576ef9ab4d85216fea379cd
8b4f15697ae2c4f8193b762a4d1589ef32a7cd4b5e68329cf174dbae389c14d2bf57a98e142c73bd5689af74e2dbc658f742a1c6d9e7bf8a23c15e74fab
d3926e54f1cd8732a64915b8edfc2743961eabf5289dc176a4be925173ac86b9fe2715a4368ef2b59dc14a362b8ed7945ca3861df29b7ac3d58e42f6bac
d957e1648ba2f93cd165e7829abc4631f7de9bc846a5d237fbe1c852a7d43e9c615fa7bd39e6c841adf293ec5b61d7243ce9fb1a6458d9cf7e1632589c4
b1dae635974fbc2e83617a5db29ce314af5b8c72931ed6f4c87ba1e32d645c81f97e2db35c41a789265c4f3da791e2b85df74ac29165bdfea72c14563be
98df17a6b245e3f7d86cb145ade862c7b3594a82cf16b5e7a4893c2de657ab8c23145dfe9acb63274df8aecb13675fa924ceb1876f24359eacd1867253b
9fdc1a8346bf9e5128c46bfd91e2a543fb9c8ea74f1623c58ba94e217fcb53aed9824b1f3e5cd74629a85fd3162b7e953fa862bd1e5c987643f51cbda8e
76423cdf1b874a356df92e8c3abd694285e1f73ad2b4e851c93a7f462d153acf84e6b57213cdaef4921836bed49725f1ce6d87b9a134fc8e795b6a1d438
e7ca6f5b42839de6a7c2bf35814de62b7af18956e34ca2db17f85c4ea961d3857eb2f1649c7eba3518d2749fe31b8764fd91c5a2e76f39db45817ec629b
da8f3e1547d6823cf1e7a9d452cf8b63915ea827fd96c53b8e4afd6c237b948e56a2f3b1c98d75ae3629b1cd4e8f73a192cdef654b371a268dfe495731c
6dbae2357cfd496ae38752f6d3be91ac2f7689ed4b1af3679d8cb2e13f5a9647ce3f2bd5189c6fa342b197cfa8652b9d37cef681429ad35fbc619273d54
8cbe76a3245cfe7b691ac3e54bd761cfa45b9286cf157bd398ec52bf43d86c95ab1dfe87c523649db1a8f7324be965c1ad274f3916b5ed278c43a6fdbe2
543178cdae2491378deb2f94c53ed68a1b5fced4286793ca5b4de839fc564bd8273af54bd9c83afe65d79b2f41ea659d2f71ae6c34f92d1e85ac673db4e
2c8315d974fabc2d138964fcbea32586491db7c2ea4815df726bec5a138f67db9a43ef2851dc47b3f9e6a827bd5419e6f8c7b3519a248cd7b593612ca8d
f7539bce1d8f39725ad1eb6cf927a3d64c1f8b72e956cd413b82796ced4a5f129b63854d71ab9c3486e21b95ac7e84d1b35a9e8cd6b1f2a4978d16fc259
743ea168b72dc3f1568ed29a73c8be4f1957cda68f1b497c563efd12c74b6eaf1d2c4596ba381f52946de73a28c5e467f1da9c854f7b236ad8971e5b46a
9d2173bf5a8e42d9176fbca3482ed7f15b683edc2759a8f4b6d2793e8f65412bac3e7f81954a63edb8745a6c31f284d95b73e14f2db87ae39f4dcb27569
f4edbca286f497eb1d8a467fc5de2a43961c782b3fe1a9475b8f1d3a46987bcfea13d54762ef8a3d4b2956fe314a7286dfeb9a51726c3eb9f85adc3b641
e792cdf8b635a4e17cb9d56fe817ac2b9e685f4c2a19e673d8c4f5e1b6a379dc5eb428791f36e4bc59231da6fb5734c89e26a15fcb9842e3fdca9b76e35
2ca9fb657834eda9c17825fbaecd67413b982df75abc38169def2ab45178c6d3afe124c6758d932a1b6c5d329ae18764c2bd9ae8f1c25437a9fb8615e4c
32f87695d41a2ceb86347fac5ed8239f64ae1d875b62fea473c1295fde7b84a92fce64bd8a5976ec138bda946e7582ab164cd93278beca563d1f97c824a
53bed126f5a48de9c621387d4a9f26c58d3b24e75c9fdb14e2a65c93b14f27c65abd82f1965e74238f91ac54ed36b2517acef89db4615789ab2dc15f74b
a8e21357dfe986ca1b7d4239e5c6fd48a92b16e7d8c94a135fb7e891cdf43b26ea517f9b32e8df5c913e7d4fbca532d4816f37c9b4ad57268e3b957ad18
fe63b95da487f35ec12ad67b3e4a1c685be7d9a42c36be91f2a735edfb6841237ed6fa9b14e832fca7596143ed2c571839e4b6cda5f3e912c47b83f96d1
ac4ef9b5328adfc476e298345dc1f6be28a735d6f9b872516f9a4d35e8f972c46ab8371f9ce245d1b6f374edc12b35a7c4e68fa9d2c5e8b1963d2f457a8
e1c94f56d371a28ef9bc317da5f2e68b137daecf5689d1be2735c9861db7ea2396cf14a7b3d528196c4fb7d8a39e14bcd58a921f47de683af79451b2c8d
76f5e4b217f63adbce7521984dcf63572b49a16cefb3d42a168f7c3ab29ef64713ca89bf451d672c9a34f8d5bc73e148abcd76295ae4f37d2b869ae3c74
28ba1fcd3497861ef2dc4935b8172c9ea5b31fd496e2ac3f5894e162cb8597a3d421c56b7f39ecd1248f537c16a98237ec15649237dfeab8415c2a63d48
f7925134e8cdaf17b948ec532b6847a1f53268e4b7dc93182a56bc9ef147d8a2e51bc96df2458bacef6928b3a1546fde832cb964d78a51c62f7dabc38e5
fa2d9bc13486f5e79a42bc1fe3d78c592e617d43b5efc71489d5e2b7cf89a416e5dcb89f176ebda4f58c162ed54a91b83ed5f4c6ae27184f56b97e1382f
64c9b5a7def1b28935de62a4bc7fe85a9241cb65f8392c475dfe19ac4768de195a348fb2cd573a61ef842b973af58c6974be3d1a2c89e536afbd249c875
a6239bdcef876ab4d2139f76b45c28d93fa1b569743cf1b52ad96cf34e8ab691d35a4c78f26b34a51c6e7f9a8d5426eb73cf4a28de135679a4cdf532e76
91fab3c854691df3ae4679b5df12e4a76bc58fe91a326d85fa1c237db9fa5ce376b4a52c6de83479ba62d1c3e4f56a7d9ce13824a57bf6d819aeb56d28a
97f1eb36d4c8ab925fe41ab753d694c71283a65bd1c87e245a63fcb921576dacfb8419de6cba815efd3942a86bedf34a9816cb27f3495dace68453d2b1f
c8e362a417cde3569fa21cd7e549683ac1e25f4763cdea892476c5eb4138fdcea7b1568fdc39b465df87134b5c2ea7f6941873af5d912f64b3ce927d681
f3e92d648b75ef94c312ab79f8d4ec5ad87214e6b593ac8d762f4513e9ab72f51d3948ebcaf321d895fc7613ed92bf6534cd8a516e4bd92c715a4bdf27e
16cb3d24157e9fba4321e9685732bfa6e14c397b86f2e4ca7d9315ce8b79614a253879e24dba36851e2bfcd9673251b486e39f2c5b7ad4832c5f6abe48c
2956dea382b9f4d17632a9f45db2c1a87f3d9eb52a18d67e2cfab157e2d6c4981b2f6ac371e8b24597d6ae1cf3298a6b54172f8db6134a79c5e2316a7cb
238d156c9a4bed8132795bcd41fe78ba369fdc82abe5479d12b3658fd94b7ca681ef4d39bace71f43895fb71c6e82f574c218a3eb5fd162c438b1ad62e7
f93cd85b14ef6735db819ace5b4df9a13c56b79824a15b7de69acf37d1b89c2635f7a41cd286be4f3c1728654e9f8ab36ec4d2fb815c496afe2dc5b678a
f3259edc163b25e98a7b1234df85ae6174cb83a92e4cd8b37f129c5a3b7f4e62c139dab742f619ced8274bfa6c8157df2a6351748de9af2134dc759816a
ef2378d53fce9742a153bf6948d1cae7583b42cf9ad875ef46d21b8ae765c94f8e2a6c7b931d5e78fb3c9a5d6f174ce958613f7cd284139cef67bad192c
4f58e36c79ad8e6c21459837aecb4d1f6a9e5fbc736a2ed91473685cbdfe76328a45f1e8d2369bfe42d53169b4a25f7eb96a2c457e36a84192eb7c14f8a
635b9e4df1a7b3ed5281c6a3bd19f58ac2d6f4798cb26ad5143fc785269d4a8b2756d3e9fb4862d3c5aefb163542e79bfd46a5e39217d58b923fc1d63e4
a9c76f8b15edac647195b82cf314a67db153c497fd2a5e8164cf2bae97dc65b4237ae1f45cb9d238a45efc6298175e2f3b81a6d73e2496f1a87265ef3ca
4d7189b3c5e42adb1fe9c75d824f359b1e2c43fb2ae8695fb8ca29714368dc5e9abd34875a1fe382476af9b3d7ce1265bfa7d4c3895a46fdc159487e6cf
b5da179bc6ea8fdc21957e3ac8d17b9e4af812e7d5a34bc67f1d398756ab2e8cf135926d8f24e7c5bdaf834691cbe5d839467ac2d38f64ae1bc986fda73
9e62b1dc7f6927cb8e63fd5c2b7136854f9731a8c25d49e173abc58217e39af4562cbe3a4c8159bd423c7f6ea45129bf3d457ac296b48e5a13df927e8a6
d54392acfe1658ba9cd74e8359fadcb872563df1b8ea6f19c3d4e28c49a1b35ecd92f4657139deb5a78c41ef295836acd47195ae2fbd8396c2b5d81694e
3bfa786192d4378fe16db45c876df29a8b73162fe5db7a412f5d36a42e795f1c3bad2796b1a2c8bedf96a5ce7b94386d7b41e293c675d81abf4c3ad96eb
2785de4c62391f4ac2ed76b9cf3581d6a894cf3d5ba291fc654ab8236f5ce217486f5a1e2b746dc28bf91358eacb7f453b8d214a6fcd3675b849ce6

P_{15}-covering sequence of length 8237.

1234562356789abcdefcdafbceabdeafc9bf8cd9fe8df7be9da8ce9af89dbfe7db8ae78b97af6ec79e6ac98e6da7bc6df5ab69f5bd68a7cf6be5f869a5d
87cd5e76b85c76d97f5c967f4ea5cb4de3cf4d95ec49e3ab49f3ac4d75984e65a84f56c8be4785694ad37b48d39746c3b648c3df2e54d65b93a54bf3845
c37f2c936f2b836e2c74a63745934f2a735d2e436d2a837e2a43d2b53f2853eb2938e1f29e1c2b34c2a57b264f1a6725f187f1b2a1e3f1d3b182d185e17
24d1c32e1629a1d723a1b429152c81428619d2c61382791452981a741351deb19f614afc15761bd615b7139c1a57c14e826abc1

P_{15}^3-covering sequence of length 472.

1234562347689abcdefacbefd9cef8dbe9acfdbaef8bcd9aef7bce98dfa9bdf7cead8ce79df6ced7acf89bd78fba7dfe6bda89cd7bea8cbe6ac97fe89bc
f69ba78fc6adf59cd68fa97eb86ade5bf978cb69ef5dbc67ef5cde4bfc5ad978ed67cf58db6fae58bf67ad58ce69db5af968ef4da879bc5ea769e57bf48
de3bfd4cba67c95abe49f75ac869b75de94afe3dca49df3cbe48c75bac78d69ae867b589a47d958ef39c84ae65fd47ce56dc49bd3aec4fa856bd48cf3a9
758e47af38bc56af459d67f859ce46bf37d8569b47de39ba46df3ab7489e3cf56be38ab4f9657cd38f647ac38e647bc36ef45bc39d6489f37c946ac3bd7
65ab36de45ad379e45cd36a845be37f8459c36f754bd35ef29d83b9546b837e6587a45cf2bea378546cf2ad8369745e639bf28e43a9e2cf43de2af639a5
46f359e2df537ce26b735d647cf29a835ce2bdf1ec34af28c734be28c635ae28df1ae3648c56934fe27bf1cd347f2dce19d438f27ae18d534ad29be18c4
35f24de1bf538e279f15ed27a634bc2d9534bf25a734e926c439b72ace17d926bf18cd26ef14bd2586f29835b62acd19f23be15cd249f1ca927cd1ba438
9726cb48926ad19bc289e17bc258437952baf18e267534e258f19e235af17e246f13ca259c17f236e1ad257f13ba249b1ae2376824ac18b275e16d247c1
ab2d65a9e167428f14d637c256f17d26f19b2ac16b28a17d48916a78e159247b1982bd17825b16a28c15a3c918d27b16925f14a32d457c16932cd16942c
e16823bc1e42a714923d14c23e14b23915a62e5b913a624b5a816d32791af23c817b32df15c2481a9235b1d8451b842d153728a31f47eb618342f135861
fc4913825a146571a3d912c3512ef319e46127451da4e183b41c687136c21b3621ed3712589173e51ab751967f2c1734c1562d831fd57184631542af61d
5941a237d16b4da81b3d251bcf8d5e4178f12bef9

P_{15}^4-covering sequence of length 1394.

12345617849ab6cdefbacde9bfc8adefc9abef8dcbe7afcd98bef79dbca89efc7d9eba8d9e7abdf98acd7b8ecf6bad978fed69bfc7a9df6caeb879cf68a
de7c89de6afb978ace67fbdc69aef5bdce4fbda87fcd68ebc97aef5c9de678fcb67edf59bea689bf5cade49cfa67bcd58aef769bce5fda769bef4acd95b
af8679ce58df967ae859daf48bde57acd68bfe57cd968acb579fe48bcf579de48fac578efc56bef47adb689ef567cb859ace4b9d876ced48abe579af4bc
9a57bdf4c8ea56dfe478ab56cdf498be569cf47b9d65abe47dcf587de46bcf3daeb678da567fb85daf468e9c47af856edbf78a956e8c47db8569df478cb
a4de7568cf45abdf3ecb746ecf39bdc46aef38db9578eb46acf35de964ade594cb8567df459caf3be9745cef37dc946afb38cef2adce39adb469ce38adf
2bce756bdf37acd45ebf389da478c9567ea459bf37aec458df379ca65fb8476dc598fa456cde3b9a746f9e38dc9458de379f845eaf3c8da456ef378ca65
7b9468cb34df9e35bcda39fb647acb36def29bce36adf27eba389e7648fd36ca8457be369df25ced37bcf2a9e8475cf367bd459af3689a457fe34dcf289
d645eb837aed2bcae359cf2edb837cd54fe9a378f546dbf27ae538cbd29af835bde27cf6459b836aeb2f89645cb736adb289e5468c734bef269cd35abf2
6ce735dfb29ac835dae268fc34aed28fb6458d7369f574baf28cb965fae269cb38ad6479cb34adc279f856dca28be635cb927df635bac279e635db926af
734bde29ab5647ef258ba349ed278ce34f9c2d8e635ad92ce86379af24ce9375bf267ea348fe297b836d9a278d934be827ac6359ac26ed734abe258c934
daf257ca346de2987645f3cb4e29ac348de259f8347df286a745d639ba278f63579c268d53978c25af436be259a8346bc278fb3498e267b9346cf258a73
469f25786a35fd248c6357db26ec54bdf9c386d4c3578b2d5c634ef256a9437ce25a864e3af248ce1dfb249df1cbe25798436fd257e83459f23dce1abd2
56e7345de2f49735ab486de15fc2b4a5c2d763459c2d4b6375ad2678354ce2a794e268534bf27396c258df14ae25697a25b638fe17ad3b2c7df16b928ad
19bc247de19cf23bde19ac246df1abe238df19ba246ef18bd345a62cbd16ac28eb1a7f359d1efa247cf19ad245cf16de23acf189b254d7925bf18cd239b
e1c8f237ed18bc239ef17da2469758dc17af236ce1a89264c73f269be158a2d397b2489f1ae8245b3c2ef156d9238ae17cd236bf179d435f7269d1c8a24
bd168f32abc179e345a723ebf16ce2478f1dac2389f17a8246be189d253b7489d17ba236ef17b823fed18ab235fe198c245be1c7826bd158e936ac158f2
349b5eda169b35ae146c235af1bc7256b43af916cb324ea715cf324db71e9a237d196e2379c16e8327df15cb268ad15ce234dc719ab3c1f9b325e4639d1
4ce2d1b9867ab149e238c71bd8426fac14ef358c147e235d48bc1a59432c861bae491cb8523da617cd361eb73f1d284a1d78524fb61ac9481eb7423ca71
49e523a681c97423d9e146d328b615cd321ec9521fda321fb8421fcb451fe82719e83b1fa8691feb251ad9751fac2715eb341cfd261cf7e418d7241fa92
31fea6812db731cea5618cf49167f8519bf27138e421fde4a136f471a89421de76a135c241da937156e231abe5c8176b423a871b96423a9615f8431cefa
82134e621fa59236b7158e231bced16b95217d6524f3159a24156f3215d94261da43258913ac4571bc428167a241b9832476e193c6f142978316c5427f6
15237c61239c518274c1327a651b274531ab75f138a5416ba2743f197235d614852a31476c512d834169852467d1b5286f14a83d1625738145962185369
18d5421b38426719542713b94f175e421bf68d713624f51da3b4162a3c1d45781ac9d51ba9e615bfe41932861e532c81e354b19d3851ba4dc1ba98f217e
6841b5df31b5de3147dfb143df915c6e7135ce712b6c4193aed1297b41fa5e912b36d1f92e81d3cf7183bcd157a9614dc8614ac831f2c7915b32d19c38b
a14c2961f37e21ac5413976d51a476fa1589761cbe4d152ea613b9c514e8a9e751d34289dfea7135b62e13952a619a2715f491836bef8147b351c46387d
a2bf1869ecf7b946e1586dcea34f1

P_{15}^5-covering sequence of length 3104.

1234567829abc38defbc9adef8b9adef7bcadef78bcde97fbcd6e9fbc8ae9f7c8bade79c8adef679cdaf86bedaf5cbeda69cfe8d79abcf6e8d9b7f6ecda
87fbe9a67dfbe59cdfb689adf57ceba869cdf58becd67abef589acb768def579bce6879fcd5a8bfc697aef58c9bda678ecd5a9bef48acd759eabc67f8da
569fce4adbfc65adef94cbdef39acd856fbce4a9dbf75cae986dcba479edb56acef479dce65b9fac48bef657adce489afb578cdf4b7ecd5689bf47aedb5
87acf469dea5879de658b9ce46dfb9587ecf658abd9478efa6598ef4db98c756fabd478cea569bcf4879ab56879df5687ad9f468cde39abf784edb679c8
a5679fe4a6dcf38abed45cfbe3adcf7659bdaf389cd7568bce457fdb648afed37cbfa459def378bca469dbe37fac954bedf36cbed489ac7f459ecd378be
a567dbc456aef379bcea37dbc9468adf379cea458fde369cf854badf369be845dace38b9f764aebf3689de457afbd3c9ef578bd9647cef836abe7849adf
35bcae467dcf35aeb8937adc549fbd7389ef645dabc369fae83c9f7586ec9457bef34cad9b38fcd5469fba358cfe465bcf7468eca359be7648ade367fca
836bde5478cbe369ac7548dbf359cad6389bc746adbe3589fd475a9f367bde539af86749df365cedf348ced537fba6459ebf348ae96378af5467ed8359b
c7638fbce27adf8437bea5489cef2bdac8467dfa356be7c359db7a38f9b547ace384bd95a37ed9465acf3489ba6378cdf29eab6458fe73a5cb67489ce35
6bf87459aef28dbc934afed536acef25bdce349abf28d9eb346cdf25abe8c2dfae6349dcf27abd6358ebf27dce5468bcd347aef269dc84569bc345edf26
8be75469bd3748e9635ad8e29cbd5736ecab29fd78369cdb27e9f6358ceb26dfce18adb439fce25dab7349ead2bc9f7438cbf269ac834d9ef2b8ac6547e
a8359cf643bdae2798d54679cf258ad9437cef24a9c7d28eab6439f8e27ac9536dbf249ce7b26af8534dcba279ef4367cbf258ae6437bde268cf5476da9
345efc2798e543acbf259ed736cbaf426deab1cfde47359c8d2a7b94638dbf24ead7835bcd268aef15cda643ecda26fbc5347fbd269af5437ebc259adf1
7ecb4a29df7634acf728bdc5634bf7a298db5e279fd354ac8f267ad8346cfe23dbcf19ade2679ca436bfe24cdbf16ace2539bce26f9b5348c9b267feca2
89f6375ba8437cdf248ae9b25df8634ec7d269a85347df2869ce3786bd4539cf2a769b8357ecf169de3548bdc23afed1b9af257de8624adbf189ed3f28c
e7435abf2489a7356e9b285ac734b9f258ed6435abd8249feb1c8df2649eac2678ef345bde2698f543679e2a5cb7924de8f17bcd264eb7f25dc9e17afc2
569e7824dcaf17bde284cba526dc9e18abc274da8527beaf187ce254bf6725dbca189fd256aef14c9de237acf148be392acd67349af2587e6435fc8724b
dae178cd3567fe239bdf168ce2a57d692b8f7356ac8249ebd178af346de7238cde16a9f237bcd19ae86735bcf2479d835afe176dc823aebc179af2468b7
a246cbe1789c265bef138cd9245eba326cdf179be263df9c2aed156bc3829f76438abe174dc5628b974358ce2469bf13ec9452adfe1c9ab2658d932bafe
16d8b2759df13abe2478fbd15aef237bae16c9b2458e96247dfe159bd2468cf179ba3568ef139dc7256f9e17dac2548bf326ad8f15eba2649cf18dae245
8df13ace2467f85246cbf18a9e25d8ca379be164dc92368b7459dc174fe253cdf16a9c2587df14ae9526dbf1458e263bcd198ab243fde1b89c253abd149
fe236acf15bde2349c782be9138af425cde139ab254df3628abd167ea283d9f14abc2379eb68f7d193ac726b8f1759b6237dc54faeb195cf4267ad145bc
7238fe147ad2538cf1479d253acf1869b2a38fe1579d362ba75469f3258ab1c6de234cbd1689e235daf167be2348df127ce4a19d87265cef148ad327f8a
16b9f235ebc1689a237dce15b8d2349f5728bc167ad345896237fbd159ac4328feb169ad2437cf15a9e2348a657e9c136be4527a93456ea2379bd158ae2
749bf158ce2639d4528cfe1978a2c4ed1a38c264978523a9f176bc234fa65279ce168ab2456df137cb24689af147b8256fe1489d263ae1758d4f6e71849
c2537af146de253f87916de5273bf1568a273bd4527c8346bf172de3f17b8923fad415ce63278ca156bd234aef129cb4e18ac5236bfc157d8243cef12a8
b3d17e9823acb154fa278be154ac2639f174eb2d19c68537df126ae931cba8491cda75b1ecf428bea15c9b4d17c893247d6b1c9f82b1de69245fd7169bc
5d18e6a234dcf12a9d561cf97324ba9158cb491ae674259ef136ab25f79148be631dfb5234cf9127bd89123bf581acd63e1ac7625be8312cfa461bd8326
7fa1589e2c1df58236cf415ae7234fbe126cf731ead5432e97518da62539db12fc73564d8235e7c146b9537fe1628d471ba98351ec96432ba5719c86341
dec82a1fd57234ca18796532cd461f985423be6519a87461df89432bd1765a24378f15c9d231fbac2e19b5832674f15b69241eca6571be9a241dc76243e
8561cd792e1fb75246da187c62459d136ab471cf56234ef152ba9d138bc521efdb6518e9c321fda7341bed8321cbf4371dcab261dca9241fde53b1af952
c1ea75623df1549a86312bec8316f9a4315dbc2415eb9341de86321db57324fa1637c95124ed75123be78163ad57126ec341ad975218fcb3517e86241ae
853217c96431db85421db963a15e973814ce73214c965318ca64315ca74218fa97514ebc27148eb2614fc72518ed92315af84931bc84231ad85461ab953
418ea73915fa62317ef82d135e9261cfbe23146f82314dc92b316fe4281dca46213ca56419ef32a17de56f13ce524a18d73461eab34c179a6321dea6451
7d963c127d8351c4f829617b85431fab78612d9738164e9321a7e9461d2f953167b942517b932547e831247693125b864175ef23165bd471f2abd3176c8
451d9a824571ab84591ef8254137b852194386251f9b3c2618f754316f7523918c45231f8b46c1289a3412ec65d1374b2615c97421a35e421768a53172f
8941b27a95162483b16a5e8dfc49a162b43f1d68523a4961d5234691f7236d1c453e6197548c1e673bf19423bc1592a4671c3be492173e5621abf48d1c6
2534b1a2d58193fc5d4126753c14b6538147a5361f8a23d916f54271c6542efa6b518c73ab125c7ab12de3471f2b84a1367b251e864af157b643195cd82
1b46d9815f2a7e1439ac516289417de62f19453e71d683cb179435d6187b32dc1eb4a251dc7be46152a31879bced13f876419a8257169f3c781e54d3819
6eb21a78e61972ad4f132c5981372ca193fe7b4915deac315d29e418c2a6d514ebf69ade3b1965843cba6197d43216fca7d4125fca37419ec5fb124d7c9
8641bacd3726a4853a1b78cd921f7c8d95ef623819e4af532487e1d2ca3f81bd57eaf8c6157b89ea3d2156fb4dc318ae26913ac89f61e57c8b41d9a7bf6
d13a4cf91b386f154b3e6d51b9c723681ae4d239128f5b4ea9f5c713d8e56b127a8dcfb93a1458c2fd76c8af4172f9a8d612fd9ecf3b724896735812ce5
7abd41ef6c243a59d1

P_{15}^{6}-covering sequence of length 5184.

123456789ab45c38debfac98debf7ac9deb6fac9d87efacb68def9c5abdef976acdef57b9cde867fbcd9867aefbd589acfb7689acef57b8dcef569abde8
576fcdab5798fceb567adfe8569bcdf48a9bce678dabce5798dafb679ecdfb568aecd9568aecf7569becf4ad9be785a6dbef479adcb6579aecd4b89aef5
76bdcef478abd96578acf965d8bef467adbce469afd8e479bfca5679def458cabe7469cadf37e9cba857dfce645abfce8479afce3bd9af7568bcde459af
cd3e89bfa6c48dbf7e36acdf7489bcef358adce7459bdfc378ebd9647cfbae369dfb87569cdeb35a9cd8746bfcda398efc764a9bfe57869d4ea78b6c4ea
fd7b36cef9854bcdfa735bcdfe4869cdf357ebcd465aef98467adbe386cfda5479bef358dcb9f368eda7459cfde3679bfa548dec973b8fda6457cef936a
8dcb5469efdb547acef3867dce45869df375eab9d4678fae356dbcae3879fda645b9cfa378be96a458bde9365cbd748a5c6f398abcd7369aeb58469cfb3
578dcf46579dbf365aecbf348aedbf2c9aed8456bfce3d49bf8673cae9d645bf7a836d9bca4578fd9c354bfda9356bfed435acfb8745defac8496bdaf34
7cbd9f28aecb6378adcf269bed7538abc9e268fbcd346afe85749bcde278abf9436edbaf297edba3589fde26cbaf9347cedf25abce9438adbf259eacd34
89ebf27cade5638ca97456bef37859ce637d59b8467edf3a4c89b7635ade9f248cde9f278cbdaf35ec94768aec5496fa7e349bcfd258ebca347debf25c8
ad64759eaf3487cde29ab8f564e7ad35968ef437bc9ad268bce573af698547cea93587def26a9bd7348aeb9726cdef5348bdef2679cab3568df4e3c6987
5f3ad94785cf3e4b6a75839cd67f28bec9543adebc2897ae6538b9df275cbe84379cdf246bae8f259cde6438bcde2f78ac6439bdae267cbf5438aecf257
9be634a8cbd257afc63489bf7536adcf2478be653c7a9f256ecb74358de9a247bdf96258cef4b2dace7463fbd8e276abf8437debc259fad84376cbf28ad
79e3468adf2749ec6d287ab59348cdb726afce4357dbc9248fae6d259bc8f3479e56f28ac9d73568cb9f23adce6824bdf95827cba64537def265bac8926
7fda43589eba24f8ce7524afbc6358df74639ace82679bc54368ebf245adc96347bafe23c9db7824afcde1b9afc2857de9f236bde8527acd9f18baed256
8fb74356dfcb238adef169bcd243afb9e168dfc254beda7259bce4f269ab84f25dae73c28df97345aebc2389def14cbad3f2e7c864539bde2478af9526c
daef159bda372e9fca4356ebc2478ae5b267dcf53489bd62758fa46395ef24c7abe925d8ca4365de9f234bcea62589dc4a27b9f65437eb9d286fab53478
ecf13bade2469bf7e246cdf732abcd6f15aecb2739fda6257bfd4c239ead6824bc9af167dec253afb79258dab4925fe78639abce1689ad5426cfa9e17bc
df2359bea42679fc3546edf2387bc942eadb86437ecd26b5af943876de9238afc5429bdef1469cdef17abc9325efad179bcf2648ed75269cead17b8cf25
3edb76248cdae1679cf2546bda72389cef147dac83269eaf178bce263fad85267cadf18e79b3f26ae874539adb26589e7b236cd9f187ace3428bdaf1678
eb3a269cd45738af2b57e84f269b538f2e9b5634ad9f2537ce6f1abd9c2537bfd4269e8af15bdce342fabce189dab3f189eac2538bdf14a9ec2736abe52
47df8e16bcda2457fc8624d9afc168e9d253acf74298bedf139acb42589df3427ebcf1589de7425cb96f18adcb293feda42679bd382e7a9f156dce2835b
daf1479ed3526bc9e178adc2437fe96283bfc4627ad894f27ea36458bd72349ecf158ade3426bfde1758bf3627a9b84367a9e2546fad3257bcaf148ebc7
d16f9b82546ce93258ad7f13bced2645bc782369dbe1a57cf3864d9c5326fabd15ec9a24678bf13dc9e5428bcdf1579be2436cdf125bac7436db9f145ad
c73268abf154ecd283a9f7b186acf24359d87246fae352bdcae145bdf23689abe176f9d2547cbf1398dc4f16bea83f59ce174abd3628cfed1579af2435b
ea82479cde185bac64289ebc147dae2f189cb7356fde1739bc5e178f9c2347bd6e159fac23489de716c8fb25479d632f5cbe16879f25368ebf127ace3f1
5bd98324abe91578ac34269d8f157bae4f1d8bc6257aed1648bf3527ea9813dcea24589cfa137bed2f1c9a8b237cef1529dc7e158adc2e17ba86259cba1
468df2537c8ad156bce481d9bf73428cef1569bd243f8a761c4dfe2538adcb1598ef43269bcf1458ed263a7f54268c9f147d8c2539bf4e187abc3428dfe
13c6bd4e159ac72345efc167ba95238cf645e2897c4365a9824f367e19c85623abcf1658de73c19ad68327efa1469cd27348ef1b56a9e2345da8b19c67f
4358e67249bcf1328ae6c157dae42398bf7156eab4d198c7a2f1dbe58246ace1d29ba87165fcb4d179ac6235def14689be312dfca831bed9c6238fa5917
bcd64238ec5617baf42386db7f148a95276c8e14a9fb2d18ec3627fa89165eb7324a9dc1587ba2436ecf128abe3f169c782d1be958237bef1549ac8615f
9b74c15df672389ed146af92537bd681a975d234ca6b17d9ea2618fdb5271cfd85234ebc96138efa421bde975218dea73f18cd67452feb67149ce8351df
ae43b1cd987324adb57169cd84a1f9b67285eaf136dbe2a159d6e241bcd9a3416efd7241cefa6251bec9a2318deb76319fde54219fde7b315cfda2617cf
b42357da68134bde251caf763b1cea76245cea371d9f46352ead168bc92d17fb65248d9e15678b2495a7632c4f5d18769b325afd16478a25394b67253ae
b18497c35286bd13ea9c651bea87351deca6341dec95231dfc95421afc96321efda98513efb92a13efc92615ef89241ceba82417fe982314cef86317cdb
86315fe9a2418df95371bdac5341fda872315fea74318fd967513ef8c2b14ef97621aefb53419eca62314fda72314fdb72814eda93612bda93415fdb634
1afc863b14ed972614efb823156eb94217cfa95d128fa96415eca7291befc74615ef872316efb52816edb53417ef953218ec964b21dfc94361de8a5423c
bd18567c2438b91d65c832496ad157f49236a78e1459bc231da976251dac79231dea65421cdb65341ceb75231afb864219fb85461aef854237ed65123fb
e4d175fba4c125d9a4713cf98674215eb93721ceb93851cfa74651cdb732819fb74361dfa854c16da83251cfb63451ed8c24b17ea93421ced963841cea5
3271dfe83491bfa726438ea72514bfa62315efc62713ed865412ce9753146fc725138eb72614dec532149da763219dfa43c12eb673419eba62813fd7624
13bda879136dba5218ecb63921edf863517ea94831cba96241ec7835619ea732b1dfc6e23549bd71325ce6a137f984521efb743512edb74312ea85631be
a982317cda6853419fe657324fd815326fc7814ba956218da754219db874c153db94f123dc84519bda64213bca862154fea321648d7351bfa9823147dc6
23519fc6423158db42319cd8527149ca652134fc8521b3ac756129e8743125dc846315dfa729134fa825134eb82613ca954713cba62e1374fc2516ab893
5127cb963f124e96831acef452137ac89b14273c961427ab3581de9a24516fd9247136f984517ced239146ba75219efb648c12d965431fa86542b1ad764
251fb9cd4e1a358c9416bd75241eca872b13695d8417be652431bc8765a143b97521fdec73b1824ad65319eca857361dfa24536cd719456e8712a4ced68
127bf365197b83451ad7634281fbd36721cb854327cb651429386714f9ac36714ba85ef714ca852714fa639e1278b53619c784261ea7543291ab8462359
ad1c62389b451726a8354716d8923516ad8493175de4621af87536219dc743a18965472c18f56342d8a1c35467b18925a7c413e269574f1a3b2d89164b3
f251946e3c71d4b29ac1357d8261c5b987d13f5428961a3f2c781ed46235bc1e8435927681ac34259e173864b217f93eab51428e6359af126de5ba13748
2f913c7e82514c37f8b145d623718c46b59182ca4671e352d87491a2b534816ef725a1bd973c51f8ba371925db3617a9548e1c625739d812495c63182fe
5d4bac127634a5e1289dca6b14e32975f1c48623bf1c4a65b2317fcd59641b3cef2d8a41593edf41c82ab15f9ced74f561b927ad8143e75c619fd2b5314
9cf2eda143bf5c91786aedfc918257f613cdbae716498aec2b31896f2ed14b8ac5316b7a2e5193daf6c142d3a5971b462cfe1b8926a1c4d725f18b43a92
61e5b48dc13f56a7213ed4f79b16e2d7ab18f2476915caf32197d8eb1fc9483e2d16cb397e586cd13965f821a94e7d61394becd51784eb321af78ec9215
bd87a41c9bef371895abc31f486d5fa9e726c14ab35d12cab7f89e6d315c498fdbea684137cdfab12c67dfa85cb631e7ad48c13edb953f47182d6f5b1e4
96d32fb19a6354f1e83296f715842b937c41e6bf37241acfd3e517a48f6ec359281a6db54e7cf96d82574c9e6d514a2e6b53fa1974236ec7816a3b8de95
c3718b4d6ce7129f8d3a1f245cb71a34e9f21ac38e7d9154cb2e7daf15c28b1d4698febc3a156d2c79abf4d617b298c146a397fcd18eab95741ef268ac1
d453896ead7b345681e7a231f6ace59bf138256a4bf15c78643bdcf1a4891be28adf5

P_{15}^{7}-covering sequence of length 6711.

123456789ab34c618debfac978debf6a9c8deb56fac9de758bfacde6789bacd5fe789ac65fe78bdc6a59e8bfcd469eabfc756d9abef756dc89bf75e6d49
cf8abe37d6fca9b487defc965b78afd965e4bcfda8675ecbd9f478acebd5498fceb764adfce58694bdce7af39bdce8a4697bdfe358cabdf4759eacb8476
9dfce3578abf9e4678adcf39b6ead7c5f48ebda7564cfeba85679defa8457bdcef3489adcef368badc9457ebfac349debf8a359debf6837cdea96547fcd
e9386abfc75498def6a359bcdef27a9bcd6457eafbd3468ecf9b357daec9b368dfa79456bcfad3479bcfe6358dbcf7469adeb5846cadef35789adc64578
9fec3a67bdf954867adef34869bde7358a9cfe4678abdc35679aef24dc8bae376f98cad574eb98c356daf98b457eac69b458fea6c347dbe9f28cadbe563
7cafb958467decf3549abdef237cabd8f269ecbd53879fcbd258e9af7436cbeaf2589cbe76458afbd37469cf8a35b69df4c387aeb9f25d8ace73496bacf
28e79dac4358bedf26a79ceb345df9cae328dfb7c5648dfe793648bcf793a4ebd8f2579ecd68357afbcd2698aef5436bdcef25a79dbc3845feda63798be
cf246dabec3f487dbe65348cfab76e29dfca57438deb9c2678fed43569acef2487bdce6394abd8f267caed5348b9ace27d689b5f347aecdb26879fe534b
adcf2587ead943687ecf249bad785369bcfd2458eac9d2478facb5269efad4357becf246a8be953c768af25be7c9d3856fe7c2db8a9e7536bf89e245cdf
a9b238cefd95268becd3a269fec853479bdf256c8ae94357cdbf249e8ad6b259c8a7346bde9f2387cda9625e8bf74635aedbf2476aed835947cfe2a65d7
b84f39ac75e2bdfa694538beaf24798cba53679dfbc235ea9f867b43ae5c6f249b8dc57269fbec4386adbe245c9afbe2687daf4536c79ef24d68bc735f4
9d8e27b5a9d4637f58cd246ba9ef3287badef169cbad34567cea834569dce23b7a9d8f2567bcde189afbc2738eaf562d47ebc98236fedc45279afb8364c
d9ef2358cbde14fa9cbde24689cfd5247abecf158d9be3627afdb53648acdf2397eac8625d7ebf36459ac87f1bd9eca26478bdf32967abdf138ce9b6427
acdef1689bcd3726aebc5d178febc25468af9c2d467bf5a239cdebf15a9cde2436bfca527489df63a2b8e9df146abce2537dafc62459dbef1348adbcf16
79aed2538fcae42579fbc36458df9a2347bde95236abdcf1579abe3428dca963475edf2864a9e5d23bf67e9425abd87f16caeb825769cef145adbc92368
aedf14c79be2538adc9f1678bde2439afbde175c9af3628ecb754369efca1857bef29348acbd25679ef3d24bc9a8f165becd324a7fe8d196acbe2438fd9
7a2c468def137cbad542689cbae14d8f9c3526bdf9e1578bcf24397ae8b2459dfae138bc9d4f168bce7243f8ba95267cf4e39268cd5ef1479abd5328c9e
7d16b8ac5472de8acf1379be84526fab9d1578eaf26349bec52837adef165c9bd3426af987c1ebadf362589ec43d768af2534ced872f36cbd475298cf6b
157dea826359bcae1678dfc243b7aef129dcba5e189dcbf35ae97d4236bac9f158d7ac3f168deb52439fdc6a1579ebc263a8de45267abc8f1369edc2f17
bae8c26497bf53268acdf1459bac372689eaf1756b9e3824cfd9e1378bcd2e16fab982746dcef137a9bd46257ac9e182bfad5e168b9ad24358bf9e147d8
bf25369adce1478fab532cd79ae168c5f92a37dbc16489ef3527c8b9f146acde25347acbf1568d9e3724c8abe137fd9a5426cbf7814a9ebf25367ce9d15
86abf3425c9a8f14d7bce23658abd14e7f6982345bcef193a8bc5d1967fa82549bcf17358ae46279d8ab1467cfd2354bacef1358dbc472af69d8154aefc
3d2679a853f24ebacd13978ba5426e9c7d14bae935278dcba146e9d8253b7acf126eb987a15cde68234bcaf156e9b8472f6e5ab38c29df176e5cb243789
ef12c5bad3e179cb6f2348dae917bcf563de28bf1675e492836d7ce1459bd8632a7ce5f146adbe27189adce3512fbdc8361aebf742568dbf1429cae3b16
79da82e45bcdf139678c4523dfa8b19576de243c59ad17b6ecf25347ad89c1657eb83a2469d75f184ebc92d18aef5b2469d385724fae36d19bc785463eb
fc128a7e64f15dc8762439acdf17486ec5f239b8765423acb9f14768ad253976cf14859db3f16eca7524869fd172beac49178fe564327cfae14689bd732
4c8a9e163bf8c2a169ef5d2436bea581c4db9a271f8db56374fcda16279bf4a1568ec724359df8a146bed72358f9c1724dbf5e137cad82e159cb683427e
fdc13569ef4a138cdeb47158cf963425dac86174edaf25367eda91546bcf2d15e9ba63427edf153698e4c15a7b8d3426caf87145ecda36179fce8341d9b
f782436def1b2589d3f14eac79326f8dc1537bea48156dcb732459ef68124cbde65187abd42f1e8ab672359def146bc582f19dab482e19df74a3e18d697
5243c8fe17b469a2537e4bf126d7ec3518abf943e1cab7294358f1ce94a6325ce9b14678ce3b15fda8c241bf9e572d1bfc93728ab64f17d389b6f1725ae
d3b1689a7f3426be8571946cad5f13e89a64715dbfa62c19de847253bfae16258bf97143acde26187fc5a291ebfc367249ef13587bd261cf8e572436ad8
57149caf6231edcb95231adfc95261aedbcf231aedb742816fdc954317fb8a5241cfeb73291adeb45231cfea46231dfca752619efb48371cdab465e1fd8
37b62e9453d1bfac47291efa845291bfda36491cfd738451efba93271cde856342cd985a143efbd72186ac953e17bd6492f1c5d7462e1cfb859431cebd5
24713fea954218dfc736415dfb972315efb964a12dfe96341bcdf82a317ec985241bdfa37241bed963241fec953241fdca53841bec657382feb65c134ae
9d6218bca976318ebd542739af156487d3e12f9c7ab56149ed7a2b154fe8d3216cba85321defa958b12eda963b512eca893517eda84231ced9854613fda
875213fdba658219df7463219dcb874216efa973218bfe976314bfac852137edb846137efd954217ecd953217fca843219fca843516fba783216efc7534
1bdea893714dea65231fdc9754b12dc9673415ed9872614fcba8e2316fb9842317fcd652413ecd752f13ab9874215fcb834215fda864213bed98f7a3156
dcb42318adf547218ecb96df1257bae3614dc987231bdf697ec1243fb965174cdba231947e862319dba852713efa654213efa956241efcd539817bc6534
28abe17394cf2d1a786592e1b74f5d3268c7b541269f8a53124eb9853167eca42316fbdc548123efab46125eca834175dca62314beca67213deb9564713
dba8624157fa962318cda9625134fbe625174fca638124fba935164ecd825136eca842517dca942316bda752316fc98425136fd9725a14ec9623518be9a
436128ec974a513de9824516dfa724381df9a24351cdb9824617cdbe258134dba652137cba652147ecb362187fc93b6127dca4361528ea9741536fac721
936fba821743cb96251dafce7836125fba834165db9734125fab976312eab95674321aed9c4531276de8531a467e9231cb8a7629143cba758163efb75a1
439fdb281635acf241679ac8521437dc9861257b4d8319a65e74812afdb645e218fca6457219cab635472f9c1b5342a8c761532ea984615bf7c234168f9
75341d2ef9687132b59a87142cb9657312dbce4f761328d9a47163c8bd521649ecd8725319ca846251deca46791b85fc3621b47ac5923617ad8e239175d
bf84e612ab98435172fb6439d1a27f6b3c14987e52f1638a74b52136dc78451b369ce2f18d4639e51274fa8635d12948c3651b2ae7468315cd9264715f3
8c249715d83624b179ea52f3dc648129e3fdb7c8125ad34b71285f39be7142835a971d2bc5a891643be27518a6fe347159862f37ad1e56289371ce642b9
51a76382c59e1673d42bf1c5a63ebf841c27ead591b342aedf1563b2eda1c8492b3d1c7f45a326981bc4a73261894b5ca21d648e75b12c34685f1ec3728
41d65c937f8d651ac39fe24b1d38f6a9c1453f6ed2174a5398f1ce2a4b5fd8132cea6bdf48713a5bec49183a76d4125fb39da1c5248f673c1bde4f9a7cb
d1652e7349d1b68273e5a41fd796b152cae731db856924a1d5e3692f1a5bc397ae1826fd5371ba649f3718ae6b4351928ef4731d928a57ce134b8265f1d
7e8241c6ab9d584176ba2d8e13645c7a81db9fc2561ed9bc2781fb346c52918b7e354fd1695a374fe8b1235c97e1f8dc257916f3e284c19f326d8941ac3
bd6e14579cf6e8b5412ad9f7ce15243a6dfc1e9452db38c1a9726eb531479c8fda127e6fb945827c61b9e832a4176fb53819746de2b183c75916af8e3c4
d179b6af8c5d4913a62fd8ceb16a4571d3f892ae153cd84f1297b4d83ac167e2d39a1542e681cb53e9f41c62bad957fc32417a98f52e1c36d4725bf9163
cd8ebfa719642d5f1e8942b6731ef4cd21ab5e83d6ac92145f3c6d92e71bfc54daeb1784ce965d1a273c54d69b1ce4f6389a1e24d587361cfba21579e34
c2b1d74efc8a51d397bc6a2fe5138ba69c4e5fb23de49c81bf329cae1673dbf4a2869c351e486bd9f1a5e7896341b25ae9c8fd267c5fe3b89d27516ec4a
8d2713b49fcd3e1af28c794f6a25187bed34a671f9eab2d651932acfe673a4b1cd357f28ac156b7e9d83c1457ed9b32c867dab45fc682e7b13af59becda
8b9fcea8bdfce9abdfce824b51

P_{15}^{8}-covering sequence of length 6791.

123456789abc3d5467efbcad8957efbca6d987efbc46d9a8efb547cda9efb387cda9e65bf78da94ecb568daf479e5bcd86f7a49bce86f357abcde684f59
acdeb386f9acd75e3b8facd47e698fbac547d39efba56478cdef35697bcdef3568abcd97456eabcdf34589ebcaf34789dbec635a89dfbc475689dfeac34
67bdfea8346c9bdfa7358c9bedf4a358c9ed76f3abc89547ef6bd83c47ae9f65d387be94af26dc8be7543adf9bc82e7adf9bc5687aefd4356c9aef7835d
c9b6fe4378acd6f25be8ac76439debf8c25a9debf46358adeb946735adeb98f267cdea9b34568cdef2375abcd84ef236bcda8572f9beca653489def7a26
b89cdf472ab8ecd63479fbac563479dfeac25689bdf4735c89bef26a578dbe4f3c59ad7684e35f9ba78463e5cfbd2978a6ec54fb3976ae85f249cdbe8a1
57fc9de4628abf9d534678becf2349a8be7c25df6a9b4c2e8f6a97453cebdf27496aceb3d27f9ac865347bfae295d68cf3749e5a8d237b9ce68f15abdce
942768abdcf1359eabd4726c98fe54b26ad9ce54f278bd9e54f27acbde59168acbfe34578acdfe13b98acd472659fecd3a28b7fe6c54398bf2dae457c89
36f2adeb758964ad3fe2b9c657d438bf29e765d8a3cf247be9586c24abf978d16ecbf9a45278dcf963b2a7ed894c15baef892367daecf14b98dae36257b
cdaf18e97bcd2548e9af7c256b89af3724ced98fa167cebd83529acfb7d1659acfe382b79ac6e3f28bd7965c1efbad792456ebcaf13978ebd4625fc9ebd
1478fcae5236d8fba9157ce8df264a7ce8d1369fac75248b9eac3d256b9ef83d25abce76f1458dce96327fbdc891a67ebdf35248c9adf13e7bc986425da
becf14689dbe3527f8ad9e1567fbad4c23e9f87ad146cbe9753a28cdef4715abcd9f24367abd8f1e467abd5f239c7e8d54f239acb68e1479dfca3624e9b
f8d1576acef3425d9bc7f1865aedc34278baef1365d9cb4e2368adc9714f6ebdc27345afb8d273956feab247856cde13ba87fc6245eb8a9f1367cedb284
f679ace1d548fba69e17c58bdf2463ca7b9d2568e4acf13d7b98a651ef4bd9a2c8365be79f243d8ae76b15cd8f94263ae7bcf129d6eab57138cdfa95e14
7cbdf382569ae7b14c8d6fe392458bcdaf12e98bcd463257facd914b67f8e3524acbd8e1f657cbd38246a9cf5b137ead9c2f158ebd9a42378cef6a14d9b
87c5326afbde14379cfb862457daef1836c9bd2e18a7cf49362bed785f149bce7a25649fd8e1376b9af5243e7d9af1268cebd5a1478cfbe2916a8df5734
92acdf16b3789c5f2463baecd194765ae3f284b67dae13c589fd46b325ae9cf41687ba9c2435e8db9164c5ead37f169bca832456ebdf13795aecd24389b
76cd145af89e23746cfbd15e3a89f6b4257cdef61439bcde2716a9bf583426adcf5e1387bd9a452368cfbe1725a8cf3d1469aeb7253f6cd9ba152e8cdf4
b135e8c9a2d167bfc4a2e189d7c562e4a398c1d6b5a4e273c9b56f12d7ac893e156bf7893425abdef134c9abd27186f9e5c23478fdeb126978cf3e146ba
8c53f196da84e2f179dc8564237aedf154697be23f18cad76e2359caf781645becf2391a8bde642c358fde164a37cf5d19468a7e3f12cd9b8a57146cde3
92f18ba7c3d2649baf13758d9e264c35badf1694758b3e26af17cb35d9268a7f45d12b8ce763a14b9cf582a16edcf452379bdea14258cfd67134abcfe25
167abd8435267fc9e1458a7dc3914b8ef5723946ade15b38c9f26e14bac5782d16bfca482357becf41268dbe73415acfb862319adef542637bdef14258a
bd9f134e8abd2f1579ea68432df79cb14568eda32b1fc978534a2deb7f1c384aeb2f165dca4b2f179ed85634279edf15368bce29157fab462819dec7462
38abfd71426eacf381574bdce2a1659fd4c231be89a57431bdfc9852316bfec9d421abf8c75326d9afe741238dbfce519637bd82c14f9eb78261afc9b74
321edcab753416efdc27341abed86c2951abfe467231dcfa954271decfb532a19dec6483a19fe57482d1abc69483f17ea5b49231fdeac6b29157e8d462f
15ab9d483617ce9a54281dcb9a356728cef193467da8f1259ec467812a9de563b1487fa5d321cbe98743f15de9c273418af9b57231def8b46591adc8724
359b6f1de8a27b4365f1cae2479d68f135b49de261fc7b95a2813debc572914dfb8652a17ce9b36241dfe9853267bfd9a54132bcfe854173bca9825146e
fb9325176afe842319bdfa742315edfca428136edca957213fdcb654213aefc96d28137ecf654213adcb854621adef354821bcea7653819cf4b26381dcf
79264b1dce75362f1bca495d236e1cba9425678ef139256db7e14395fca261749bd58321efa79683d125eca78341b69cd253418eba9653218fdba643912
8fde6537128cdfba3412e978b634125deab7849132fcd6a591834ed7c5621b98f74cd3156ba7892f146eb78c2513fea9462513dfc846271adebf59231a8
dcf65291eba84762c15ead34b8c16f753ad491ec2586349b1ef7a85634c19da726348cf15e96347c215abd396721ecaf8b5321e9c7a8b36129eda743c15
9fbd2467318fa9542631cedb8472931cdba6724513edbc9a271536eb8924153adc7624158bea732c165fd9734216cfe87d3a2164bc89537124ecb856317
2df98564137aef9c2b41563fa872b1549da63f2158ce97632158ca97642315fdb764e2a31c5b967423d15cb8764321acf9785461e3fbc7529148facb637
e5412cd86a375129de684371f25ec9438162dbef59417238dea9416527fb8341692cae853b1462ad87591364cfb725a136de8f245a7136de945c7821adf
436cb8512d97a43b612ce8a4576129dfe453a61829e754a31c28ed756a142f8b3c79a15d62b8cf7415d238ba67154e2fb36817d542bcf317e6924ac3d18
5629b4a31ce569b4f731d2c965a4f1382c96a7e1bd3256948f1ca2756b3491a8f2d637e1a482cd69b1f3a27459b1c3d287e45b1392f786451d3ab729864
1cfda275e8139c4a76521b8ed37a56f2891bd357f6a914c3e72fa51983c426751f9e3ab6dc78124956dc7318425e6dbc19375e4d281b97654e321f897d4
c5a12eb73985c1de6f23b5a18e6c23745b189f2e634c1b7d62534a817bd325fa41d89375c2f1486ae92b371f46ae29c5134b8a2fd791362e4bad9128734
6ebc1982a43eb17f8259d3ec1482f56b9a314cd58e6a9b213fc45da682913ced75f6a4b31e957f8cb62135ae8f4c6d7193b2fec4d1a36274f98c5b613ad
2f7cb6e9241538d6fab79e41d523a87cb1429de56c71348fade2913b4ca7ed21f65439e8a216f54cb7a918de5364792a18ecf349d7b516ac894e75b1d68
3c47a561ef9c328bd415f96c3271b5ea4f839db172e4638d9b15a72e89cd6bf4213ac759befd8316c2a7f9d1485236ca9fbe3451cd8f397eb6152f8c731
e4d5fb2c3a6e17d9524b3a8e16725dfbc918a34ec56bd18f4723cb6918dea37452189ebcd374af9d126eb85afd7c64215be3af76928135b4f6ade832c19
54fed63978ba4c25ef67a84195c3bd4f6718a3b5c47f82e19ba4657c9fdb1e65a239fbd4618c2ed7fb31a428d96b571e3a86dbf2193e4cd6a79b2f14a58
c63e97f2d5a61349d57c2ab164e3fcd879a31bc268d975a3f4682d7ec351f4ba8de2c3b5694acd83f25ea19478fcbae46d2c37948eb5f26714d89e3b5c1
627f89ced51246a97b1ed86a27ecf3528d1b47ac3

P_{15}^{9}-covering sequence of length 5330.

123456789abcde14267fabcde89564fabcd789e36fabc5789d4efa6bc759d3efa6bc78493efdabc685739efabd465783cefad46b783cef5d4967a3bcef2
5896d7ace4b58f9d3ace4658bfd39c74a5e8b6f93dc4a758b26edc9af7583b46e9acd5f2b78e9ac65f437bdeac8925f67bdec8945367bdef294a5c7bde3
8f2469acdb7e15f69a8cd47e23f9abcd86725f4abcde8139f6abcd58249f7eabcd1685f9e7b4c3ad289f7b6ec14ad8f9b7e2c364af9d78e2364bcaf9d15
3e78cabf4259678edc3f25ab79e684d3cf2bae685d4937fbe2c8ad469f17bc8aed5249f6bcae81579d6bcf482e53d9abc68472fdeab56c3748f9e256cd3
b8af149e7c6db35289afc7ed136b9af587d436eb2af89d4573e2cfa96db148e7cfa56d23b9e78f5a146bcde9f23758a6dce14975fbad238e49cfba15687
d9cf2345a67ebd92845f6c7b3d289a56efb147d89c5af2346bd98ce15a47bdf8c236e5a9d7f1846e5cbd2af198e6c7d43a25befc968713abefdc92487ab
5ed39c2f8a76ed35b2c49a8f17de3b59c284f1dabe569c274dfbe569c138afd7b649258ace7f16d35bace7826f35bd9e17a4836cd59f2ae178bcd396245
7aecdf138467bec9253a468def127cb96ad531e8cbf974263aedcf815496b7ad2c3f456e9ba283c756f9e14ad3cb7f2e645a8cd1f3926bae8cd14359fbe
ac812679dbfe4832569cbaf1734d6e9c2f1758bad6e342c58bafd137965cbef248736ac95e14b7d8f3c5a2467be18dc3596f7a2bde35946f18bdec29437
68adf12594e7cb3f1d589ae46273b58fae162d79cb84516ae7df382b45a9ecf1d23768b9ae512c7d8fb9341a6ced75b241f89eac65241dbfe9a37281dbc
f56e324a9d1bc768f35294bdce1f36a957b4e183caf67b2d159c8ef7a34259bcdf16384e9bac217658deb93f1267cadeb3415f89ad7b2615e8fc94b3615
e8af9db2317ceaf9642b18dcaf764295e3b8d7a14c65f3ed2819ac5674d2839abf1c57d46e329f1ac8b6537249dfae1836749bdcf125a3e8bd6f12479a8
ce35126f9bade4381269cfde537128bcf9a643518ecfba6245189efbc7a35149dfec7826154dfeca782134bdefa6521348defa7521349cefa8521349ced
a752136bceda8721469ceda832145bceda9381275cfeab691237dcf8e4512b7adf8e65934217abefd9854127acfdb95612837cdfa461925bcde743162fa
cd9e4713825f6bd4719e35f8a62b4d179ec35682b1ad9e47562f1cb98473a2e16d98574b3a2ef18cd463597f18ac4635bdf17e2a695b4f13e786a5c4213
efdb89c751a346fbd8e512347cfba8512346cfdb8512347acde85123679fdba4153289fde7614358abf9726135ecbfa4271968ebf532719cae6d53241bf
ae8976231b4cefa7682134becf967512a83edb9641258f7ea9643b1258ce9746312fbcde87451632cdf97451682adcb9341568adf7b324156adc9837214
6facb837294516fea37294815cdf376e2b18ac5d9432ef1acbd6832451a9ebd637421aceb95637412fdab98564127cdb93584127fdcb93564172deb89c3
61a2457d8f361b942cde85631947bfc8523d9614bca7523961dbe87fa4351cdb872a46315ecb8724d3196a5fc247813b6d5ac924e817d5a396b421ce87f
39a4b215ce86f3742d1becaf5736219deaf58c3642b17a9f5edc4b36a12f9d8e4c5b13f267ade54139286c7ab41e935f27dac149368f25da1c4368e27d5
f19b3842e7c5a1bd392e6c7a8153b926cf48e1739db2f45c61789eb23fc51d7a892bec641537f2a8e64d5179c23e64a81b5d2793e4681fb257a391d48cf
2763519b8e42d7f56c1a38297f45c1ea82bd34971f65a2dc8913b64a75cef123d4897ae51b2368a9f475cb123d69ef45c1387b69f24ec153987d2fac16e
584bd32caf156e94d2c87ba1369d428feb1a7c2358d6b1e7fc9a3421b857e963cda4157b26e3f1ac849d5b1627ef4a31dc6287eb4531d98a76f2be41953
d8627ab1e93c852dfb1ea394628f513bda74e291c56bd83742e15fbd79c34ef6a8b1257c49da387b1e64c527dae89364157fc2abe19d45867fbec13d489
7f625a34b7d8c1f26a5b3489df72615b3dce4f7219b3ea5cf48d2691e534caf7d68e1c3972f46a1de8b9cf524736ed185f9b2a3e4d6c195b74af2813bd6
7e45a891cb3fd24a7915dc6fe2b781349a6f5dbc4178e39a6cd2fe4b1356a78ced4b2593fe78a1d2bc9f638e15a2df4c3967badf15248ca69b5ef1324c6
89ad3eb5712ca46fe978c5d1263fb49e7a581263ecd4b1a9fe6724da835679ec2

P_{15}^{10}-covering sequence of length 3263.

123456789abcde1453279fbcde865a347fbcde896a2457bfde398a6457bfced93a6457bf28dce9a457bf28dce3a9674b28dcef3a657b289cef3465ab289
def34c5ab289d6fe745ac23bd68e975f1a4bc68d9e3f2547c68abe39f247d68acbe13597f68acbd23597f68ac4d2e3579b8acfd1e3679b8acfd453627b9
eacfd4156389bace2f15678dbace2435968db7cfe145968dab73f1c5e68d9a73b2f4c6ed5897b134caef68d92534b7acf8e12965dbac74f13968dbec54f
1237ad9ec84b1f35ad9e78c4612bfdae79c34168bf5edac271968f5beac42d178f9bea642c17d59fb8643e12cd9fba8634175cdfeb923a1678cdfe493a1
576bcde4f291587abce4629158fabcd7423169eabcf5d241678ebafcd231578ebaf4d26395ceb7f481236acde9f7514268acdbf3714928aedbc3615f49a
e87c2365f1dbae89c275614fbea8c392715fdbae6482319fdbca7652e139fbdc8745263efbd1a948267cf3b1de59a46c73218def956c4371bde2f985a37
41bedcf85a93216ebd78f495a126b3dc87e49125b6fd87a3e145c9f26ba37185c9d26f4b31a7e9d8654231abecdf67492318ecbf567429a138ecf67459a
231edc6785b42913efad76852413cefda67589213ec4b6ad85912734ef6b85a12c394edf8b7512369cdef4ab12835cd9e4671283afdb95471623cadeb95
71426c3fda957148ec23fa6b71548d23f9ac7e54b12368df9a5471c2b3df695ae7124b3f6c58a91d7b24368ea591fd24c683a591f72edc864a51379ebd2
861cf3547ab98e216d543f798c26bd14357ae82f69d41cba783265ed1a4f7c8239561edbf47a35281cd4be7f36912ca4b8753e9d21c8a6b43d5f12ce987
b36541caef2793b54681dce2a97b34816fe5c2ad378419f6e53ab27819fc4352dba9671fc8e42a5b9d17f6ce2345918afd2cb3e64591a2dfb8e637c549d
812b6e759f3c4a1b2e7d38f9cb15624da73f9ec58b6a1d749fce2386759dca4b2e618375bdf4ca69e83d514b2f7c68a95214de3b976ac5d12f38ba5ce17
9d2b6834fcae7d6b25f13ced847956bcefa87956bdefc87a56bdef9c7a58bdef9c6a135247c8eafd96b7c58efd96b7c5aefd96b8c5aef79db8c6aef79db
8c6aef597b8c6def5a7b896cdfea7b896cdf5e7b8a9cdf6e7b8a9cdf6e7b8a9cdf5e6b8a7cd9fe6b8a7cd95f6e8a7bcd5f6e8a9bcd7f6e8a9bcd7f6e8a9
bcd7f6e8a9bcd7f6e8a9bcd7f6e8a9bcd7f6e8a9bcd7f6e8a9bcd7f6e8a9bcd7f6e8a9bcd7f6e8a9bcd7f6e8a9bcd7f6e8a9bcd7f6e8a9bcd7f5e6a8bc97f5e6dabc97f5e8dabc97f5e6dabc87f5e9dabc87f5e9dabc86f7e9dabc86f5e9dabc78f6e9dabc78f6
e9dabc78f5e9dabc68f7e9dabc68f7e9dabc68f7e9dabc68f7e9dabc68f7e9dabc68f7e9dabc68f7e9dabc68f7e9dabc68f7e9dabc68f7e9dabc68f7e9d
abc68f7e9dabc68f7e9dabc68f7e5dab96cf7e8dab96cf7e8dab96cf7e8dab96cf7e8dab96cf7e8dab96cf7e8dab96cf7e8dab95cf7e8dab69cf5e7d8a6
9cfbe7d8a69cfbe7d8a69cfbe7d8a69cfbe7d8a69cfbe7d8a59cfbe7d6a89cfbe7d6a89cfbe7d6a89cfbe7d5a89cf1b43279ad5

P_{15}^{11}-covering sequence of length 2317.

123456789abcde21456783abcdf9e456782abc3dfe956718a24bdcfe56793824adcfeb67918345dacef672b13958acdf6e2713498abdf6c25e13978abfd
645e1c978abfd625e349c8abf1d25e3749ca6f1d85b37e4ca6f1285d3b7e4c9f1a256db7e8c391a45f6b7e2d391a458fb6e2d3c17a489fbe26d315a4c9f
7b8263d154c9afbe278d1546c9f3eb7281d46c9faeb372854169dace3b72845f619ad3e72c85f4169bd37a2c5e4189b637afc2d54189e637fa2c1b458e6
3d972a1bfc548d3e29a1b6cf4853927dba1cef83462d9b7e5c1f83a62497d5fb8e1362c94a75bf3ed61c24975f8ae36bc129f547d83eba1c92568f73db1
a42c5e6f37d84ba1269cd5f48e1b279c34adfe8561279adfcb85463e1

P_{15}^{12}-covering sequence of length 549.

123456789abcde123456f89abc7e123456d8fab79e123456d8cfa79e123456d8acb7fe123456d987bcfe123456a79bcdfe123485a79bcdfe126835a79bd
cfe1246839a5cbdf7e481693acbd5f78e4926cbd3a5fe8946cbd127f3ae98cbd627f45a1

P_{15}^{13}-covering sequence of length 195.

123456789abcdefg1246378a5bce9fg2d467183a5c9fbg24e61d8357c9bga24f6d83e5c19g7a2f6bd3485e917acf2bdg3865941aec27dfg3b86519ae274
dgcb365f198e7a4gc23d5b169e7fa4g82cd5139eb7f46g8cda1295bef34g78c6da25be1f497gc63a28be1d457fgc3a962ebd8517fc43a6ge2d985b71c4a
6fe239d5bg7148a6ec3fd95g721846bcafe3d972584gb6cfe1da937284b5cf61dag3e7289b4cf1d5gae762893f4b1dgca752693e48f1gbad756c93e8f21
b4agd76953f8c12abeg7694df3c521bge8694afd5271b3c8g946ad5e71f328bg96c45a7e13d2b96c84f5ae1g32b7dc894fe56g21a3dcb98fe7g45162d3a
b8fec745g29136abfed85cg94317ab2f856dg4ec37192fb5ad8g43e1c627b95af8gde3641bc29a87fde6g1b534c9afd28ge1536c7ab9fd841e52g37a69d
b48ce2gf53761abd4c9eg83267f5b4aceg19836f7d4ac25e9gb367d81afc425e39g6d8bf7c24ea935g81b7dfc64295a8e3b1f7g62d9c5e8b4a31f62gc75
894ad3fbe2c1756g89a34f2de7b65c1a98fg4d37e562ac8bfgd9147532ae8fgcd69143b7e528cf69ag134db78ec65afg139d428e65ab7g31dc4826f9eab
3g5cd74869f12ae3cg7b469f52d1e8ca73b94f5g6182ad7c394e5b6g81d27c39efb6451a278g9dce6453fba71g8ced453f297618bdca5g42ef768b9da53
41c2g7e8b9af3d6c251e87b4ag9fdc56e18b432gf79dac51b42e63g7fa8c95421db6e3f7a9g54d2b86c31af57e49g268c1bfa5e9dg327c86fa14edg529c
3b6f1874eda9gc5b621f43ead985gb72c1f6439a58bgce71f2d94583bce67f1dag29538c67b1ead243g85cfb19e72468da3gbcf57e2418ag6db9c7f2358
ag6b1ce9d75f8346a1ge2db5f897436c12dbafg57e643c8912fg5d6eb43ca82fg71d9e4ba356gf87d1ceb42a3gf96d17c5ea84326gb1975efad83cg12b4
69ea573g81df2ce6ba4791853fegdb627ac931f5de467g8b93a2f156d4c7ge9b2315ad84f7gcb23e69d85f14gb3a6e9d728c513g46feda7b8251g4c9fed
b73526819agefb73cd426a9g5bf387dce214ga56bf8c3e2dg419af67c5e23849bgf6d75132e98agbc6f7452d83ebg1f9c6724a5begd198c734a6e5g1d9c
2f347b68aed51cg239b7f6a5d4c1e8b92fag53d6c4819e2ag57fdc8691e3ab74f2g85d16c3baf49e25d76g13ca49b2d78e1g6c4f3b29da8gec5437f91ba
d826gec357b41d2896a3fc54bdg721e6f89c5bg43de1a687f2b945cead63g7f8925cbd61g4e87935b26ag4cfe813db97g2465c1fe3d7a84g5691c23baef
784516d392agce7f46db1985agc34fedb2917a6g8c3edb95f14ag738ceb96f2d1473g5b8c6a9df17e43g2b5d69f781a3c24b5edf618gac27b5df63148c7
e2ba956dg1f4738be9ad612c4785e9bgd3124a68f5e7dg9bc1a38fe6257b49d3c8fg6aeb1594cd7g82efb561a397gc2f8bd14ae53c6f28g94d7a1bce35f
9g28a4b7cde361gf94278a5d36bg19e2fca5d6b874132gafde5cb691382a7efgdb4953612ac7db8ge5f19432a76decg1b5f24a98ecg67b135f482e9c716
agfb483ec9d267f15abg4c8e6d723fa954b8e6d12379g4ac8fde6521749c3fbg8a2e5794c6dbf1g82a543c7df9eg82b6a45d19fc837b2ad461f9ec578ga
23b194ec7d85af2639b4e7g1acfd3b5289g167ea4db5f2c19g7e386524dc9a1fg7862d35ea1c9b784fg536e2abd8471gc6293fd8e45gb7c6913fd845bac
6e9gf731528aceb94g63d18af5927e6bg4183a9527cd6b418a9e3fcg2d56874e913a2c5gbf6e7931d4cagf2b7e93d85cag6741fe2d89ca364b175egdca2
96f41b85e3gd7c249a86bfg13c725de489bafc1673e4d2b9gf8156a7e3db2cg8954f6abd31gec29865adfg47c236b89ea5g4d723bf68e15g97dab2f3c45
86egad7b3f1248c6a5793fe42g81cba53d79f4e6cb82d573fage1694d85cb7fa2e163g95d4fc82713g95e4dfcab78135e46gc2daf1395bg42ce7681f53g
942dea8715fgc39642ae5d78gc9614e32a5gb79fc14d38aeb75f29cgd43ab87f56c9ge4a2173f6dcg548b19376dcge5bf497a836g12dbfc479ae3g81f2c
574bad93g8f6e251ca743bg6f5291ec34da6gf7b812459e6cfad7g8159432bdc6egf7198b2a5c3def7486b59g1ac3de86b59f71a4d823c5bg76a4e82c13
dfb67a498g12cdeb53674ag21f9b3d86c57ge1abf2948d731egcfb2689d37ag51cfe24d68g79ab1e23fd85c9ag76b23e15c8f4gbd2ae953c718fb4ea6gd
c9712b435aefc891gb4276fea5319d8gb4eafc73629d51g8ea4c2fd37159b64c2eg38ad1b5f74c9e86a2b1g5d3fe7869a14cgf2e7b69a35d1c8febg6354
297c8f1aeb6dg9542f87eac6bdg54f213ac7e986df4g325c7e96dfba14c83g296d57b1c3ef4269d785c1agbf369d5e12a4b76c39fdgea18b7265c34e189
bd2g63a548c9ed1276b5gf98ea41dc6g7b38ef596a2dg741c586b92afd47315egb9c6fd21ae84376g9f52de8cbag34572f186dcg3a9b7e45c6fg23b789d
a51e64f38c9b5217ega6c395d2b7e41fc685g2ade4f9c31gb6875a29dfcb3eg514a786df2eg319a754d62b3e8gc17f9d5436ac2bf198de74g52bf38197e
6d4cb2ag195764eb8d2ga37f65cb1d82g49a357b1efg6842a7c3be96d1f842a5bc93dg71fe258963bgcfae194683g25beca169d872f5gc4a16e3b87dg24
f1c9658dbg743eaf928c67bd453af2eg967db5cfa1298g347d6a152e8b4fcd3ag5186e42b73a9dg16e28c45a7d1b326f845gac73de1249f5a76g8dbc43e
5f9ag1b8d723e49g6fc817b5aed3f62478gb9ced6a2f714cgb836af79215gb4deca8261579f4bdea638519cgfdea7458b93c1d6g24a5f7839ec2bg167d4
35c92fbe167d8c93gae42f16bc95dg4a17fe82cb9g43a6f71dc5e248gb69f137da52c8g9fe1db46c539g781dfeac236g574d98feb326g1ac487d3e695a2
cfb4137g6ed2c5f94183ab267fcdg14853ba27e9gdc13546ab8fd721c59eg843f6bcd25ge789fa3c2dg6b49817ce32f6g481d9ac7be5f362491dbegaf38
62c49d7bgf85213ae769c4g2d153eb6a9c48f732deg5ba84f1792e56gcba4f19d238cga7b4195ed82a3fg41ce7b58d36942gbeac75df63812acge57968b
3a1gc2f75deb43a298c156gb4fe931d862c54fa73e8612bcd49fg3ae65872419adbg53e74961fc8b52ge69a743d185efb9a6427g583dbac6294efg8d1c5
a2b6f4ed813257bg94cd8a6f2beg5738cad1f9g5b2e46a3d98f7gcb15342869af7dc31be4g9285ad16fe3cb2748d5a69cf1e37842gadf9bc578e34d9g6c
a7f2518d4b6g37efc5a94g127eb8fd34a169c28fd5g3749bce86d5g134fc29a7d613e8gfc5ab4976321ef54gdba7286c93f5b7gde642918a5gf34bc2e97
da156g3f294dec7a86g2bf4d71985e6cbg3a27159d68geba315fd4c7e8g91b3ad2ec46517bfg9ad3c548b7fag6e951c437d286befc514a927b86eg34ad5
21bf698eac3527b4f8d693ce5g217bfd9385eg2ca674135f2beda798gc1364dba87g5fe341ba96d75ef283c4b91d7g6e5f2c84abg6d5f321987c654faed
g37c8b4f65d9ge7ab428f659g3ac4e71d6b58f39427gcb6af59d3e7gb16a4c529g378a1dfbe2c536947fb1adc6g9574f18a23ec76g914ba53ec768fg9d2
1bc3856ae4b1gf7d69ae4c3f15b862g7c9fe1485a2d63cebf41a275gd3b6e84f97ad5b26g4c38afde956b2438efa51926b74835ga1c62fe9dg378561eb2
9cdga5746e18cb3f275g94de1caf7586924ecdgab36f714e5cg9a86df1b42g376adc19bf32eg85d49bcaf6e715g283da6c7948gf31ab26d94e8ga3125b6
4c78aefg126d945738cfg614a2b953ec8g7fd2a6439b87f15eg3cdb69f178g4e3cb2d1785g46e9a2df718364e25gdfcb3768aeg125d74fc9ag28b5d31fc
9ga7e8213b59df4c6g23187b9e46d35fac82e916357cda8f9412ge6cdb8f53a961c74dfeg2b6ac1854df32bg9ae1c36d725fae8g1cbd34a965g1fc23de9
a618f3g72e95bcd1fg8a4e25b7cd1g3f42eb98a751gf264b9d7ea5381c627de49351ac7bgfde9634cb182e59gf4ab3d7c89e154g6fa32deb98c765243da
9b81e5c3g42af7ed1b34568gc9d2ab1756ef3c92gb81a5def297cb46ad3g589f4bec1d5a6g2f7ec834a19dg5ef68b7a1d93c2487eg51b62fa3479gb8edf
613297bc5ad8e4f297bgc1ae5f327b94d156ecgf89a45d23ec8b1a4gf79d36b528ce49gf37b1d8c4a29357dg1e84ab6cdf7e325418c9bagf6de35179fc2
4g86ad519ce42fbad831cg7654e2df8c971a5bg6ed84fc1795b682dgcf4e63798a51fbd6g47c2835f6149bca72dg318456ab9c32f8dg41bec27968gafdb
3142e97gfad38546e91bc73agf48e65c7d1g28ae639f45cg71a3e8bd5cg67f13eab9dgc853f6a271bc8e93f521gda843c62e79f1a34cbg65f9e237a8db1
549g628a3bfc57269g138fcd74ae6291gf7543bcade9182g3b6a7dc1524b8fge916c2a5f3bdg471ea265fd4839cabef2gd839a16c4ebg89517c4afb8369
ge5fca7239bg4e516d289cab4g572136cfaeg94bd382165g7acde931824f76a53dge2cb768f4a51gd9276b3e85g4a72c1965d4f7a823b9e16f482cbda59
6e8gf73ab2c6e1d493g8cfa1564928bdfea5g3c6897fbae53g4c79d128ab569g37d14ea26cg8b574df36c1geb5a928f37dgc4e5b623a719gde86fa24c1d
3e58ag26f9bd3e47a2g6b1fed9c8a7526b3f941c8a762ed3gb1fc8276943g1eac5892d36174bea5fg2c39db71ea52f4dc63e187g29dcb6e45f8371gb9a6
e4d875129aceb47g8df1695ae78gbd39c4672fg5db83c1e9g425ab3d87c6ef59b2871ga46dfce98bg3a427d1cbfea5639417g8befa2c5391b4ed78af2cb
g19457efdcb829457a6efb38d24gca96b187efc54ag1392f7dc85e1gba247cd65e3fb12g79adc35f614be728dg961c53ea2g96b1df3478ag6cb92f5d71e
86a24bc5d37ea284b1g6f3d59c71g28fed43a576c8gfb49d25e3186bg975a4ec86d32f5ba7gc892fd1a54bg2c3879a1bfed2g865793bf1ced4ag395bc6e
7a29f41bc3d672g5a8fb1e72346a59gfdb3816c74ad59ef18c36754b9dag8f126359be7fcg4819d52bg7c6aef18d3g2976bc14fg8235dac1fe48b395ga2
c6fe817d35b6c24feg139d865a2bge738154c9ad6b7e2f398ga75b4ecd921ag6f54732c1b896edg7a43518cbg6fda493e725bc1ag6fe98275341dagb6f2
54e81ca976f2bd158g4ef2738b59d1e6f4g738ba16f547d32eg18c57b39afg14e2b538a617dg92ebfa568d71cgbe3a46f97c5e2g864917daecg2b3f685e
a19g4bf632ec8a9g16423dec59fag71362bc5agd13ef4879561aceg83425bd1c6e8g39541dfca72g36e489ab7c5f21643d89c7fea41bd6g57c3a948e21f
c365bd9e78a21c4gbfe9682d5c37f1ba296ecg7df51b4862gcef1395a72edbc1895f26a4egd7bc3968ag2d7eb5136c47fba2dg3e9c4bf51g36ea9d8c425
7gbf9ead86c54g1f9d2ecba8451f96e3gc472b1893a54e7g2681bfa93d5241c67gb5e8293d674bf5ge193d62c78f4g1ad25c96fg4a7be38195g2cbd4f37
86cae1dg239584ce67b23d91a4e58bf721649adgbc758f26ae1bd879g43a26b1e5cf7g936ba1487ecd2f6g13975b4cdfage291784365gde9a21b84cf3gd
6b2485e97g1afd364c8b5afe6g49dbc18a5f34b2ged6a9f312cgb4e53ad618cfbe53a9d8642g153bf7da8g4e6c3b17ag85d2f936c1ebad4g32fce51764d
9gcf8e362da1758f3ec4972ga5f31b4d798a6235e14dfc896b7e315gd6ca4f938b2761dag54f37e26ab8cdf3eg675b421fd3c867ge94f538a2cgbd6f4e3
1a7gb2f8d4e569g21ac3e47bf52ag1c839465fe7bd1234gca65e78d9f13a4562ec78b3fg91427d8e3g6a5bf7d249eg8abfc7391d25bef6c784d15b3ca27
89efb14a62853c7gd1eba68c95fg716ade48b5f329gc48a5df321e97b6g8f2dac371b865fg4d3729c58ebfg3d76182e5gcdf37a862g95c4ed7f26g3c95d
7b1f6aeg5dc419b675ae843d26gc51e4b83d2ca916b5fd3e82a7cg134b69e82adf5cg4e8b396dace521fb89437c6gd21ab8459g736edf12a4bc6ge7f183
a4b6c2fd5813g47cf9dea2138469fg72e13cb45fd89g63e7abcf9153g286cbd91gf7583aecbd7f2g94a6b8d3f71ec425gd7a83e942b156gc3fa24eb79d1
62c48g57fd1e9b3gc7682ed5ab31g4768f5dbg1943c27a65bedf34196c57gafde8945g12f7ea3dc68549b2ae176c58bd9fa73c4dg829b7fae643d81bcf9
a7gd854269f1ebga3824f6ec791b5386dafg2e519473f2beg86d759a31ebg6c85fd9eb7a3g5c2df1b67a482dceg95b47d23a185e4c6b9f25718ce3d49f5
ag2c7819bfd523ge617c4f2d981736acfgd5b746ace123d9b8746cge329d8a415gc7be96da148fc75g9db1683ec52ab694d587f13b26cg9a174efb5cd6a
3124789cge5bd62198efg3a4db67e2fca1g8db764215e39gad76c2f1eb8d43962f7gb1e9c25634fa8c7251b49f3a6758g1edb24a957cf38d42ab6f53ecg
4dbf7a96g523d4ca9e8f5763bdace598fg3a1d4c7592gb631dce5a74g8f9d13657gae9bc23468fg15de2978a61b5dgc39e78652bdf13aec8gb274fe15da
b6923ge845f16a7934gd58bcfa296e3748b5d92a3ef4g67d8b92afc47g653b281afe9d4g73b682914cfa7g86e91cd534876ef2abg31d86c2e9fga4b6572
319fa8c64gb1d2978fe6c34b719g6fa84dbe97c32gf5a91bc8eg2fd574a38cbe62d79f3bgec64d723f1e865d4c2gf18697534adce8g7f5612dca83e5gb9
f16a2d8e4g19f675adc2e148gb9af3cd417b56ag2edcf15932bgd4c7819523dab768cf1452deb87ca9635e1d47cgab3965de2c17gb86ae5c972db8f3c6g
5429bf1dgac3e2457f69gdac1e84b372g5cd981ea73fcg28b45a71f32dgbe96c2847df1be9ca34gf17289e3dcf1g56892e3c4fa16gbd732f59864cdg1ef
5ba7d8493f2gba6c175f98da3647e158cgf3964bd25c9feg43db81a2c54g679e12a34bfgc59de7142gfb9857ea14db39f8g7e4d2ab58f3g7c9e24f1a65c
3be4gf1da26379ec1df42b653a89c7d62g54ae8f3b697da24gc1963f48agdbc21e467g3ac89b45f2e6178b9c32d5fe694a13d8e7gfbc694d23eg8b16a9d
3cgf7b4518e9ca32bd416f78a9b351c648297gabd6efc843g19b27edc43af62957c31ga86fe5db1c7a824fd96e1752bf36g14e827fc59a1deb3674f5cd8
a6ge9354c1f8ga29e5d4c6fg73b59ad1472fg6859be17dga2f59ec471ab253ef9gd6a1b425c9e73fd6a58149gfb7ace56g38df12ac75e8bgf641d973c2b

P_{16}-covering sequence of length 16087 (Part 1 of 2).

6f8597a4eg283fc5beda462139feb4c781326a59be4df8gc52174f6ae8b51cd296gf8a3ce4976b28df1a5c6e4937dgca8bfe426gd37b45a86gc1e27945b
gaf23674c1e9gdb25fc8176g4b59de37ca21b968g7c54ed2b63fe1a48d56b32egc1489f5b37cd2a6f9b543c286a9g1dc4fb87e35d2ga1c7bfe35gd4ac1b
6fe9532a1g648fe7d2b5g9a31874b562dg137ae5b4f9c678agd3b92c6f1g4739ec8dab6g5943fa71dec648b35ga2e98641bcfd752g63e94ac825gfe7b1d
89c24fgb67583d9cg6bf51a82e7946f1b83e57cd26f8913ebg4a851d726bc81e47ga5f68b9d2ga436bfcd7g1235fe8b9d46g51a73c2e48d16739ge5ca82
4d61b3g97efcad8b491365ea7bgdc98135a2f4e7d589a32cbf1dg453e8729gda51bce239d458c1b62eaf387c5bd1g432e657bc9243af5e68c1947gb2ae3
64c7dag51f8b39c6eg1d74283bg95c16a438f7g596dce847b2f1ad938egbf452c319g7f8a2346b71c9f2dga5b4c78d9a312be67d89a1feb36784cadf259
36b8ceag7932d8514679aefc35gb261ea45dg8f6239b1d5caf4238e16c7952f4813ca9begf5d1642798a35e1fc6g278b4e5cd9f764a8ge91fcd64238bge
fc14a6d2598g73bea649dc5g3ea27f9cb6g8d725baf936c18d5fb4a9278c3db6e9f75g34162fdeba941c86d73gf4b9865e3172c4fbde692g51ba43cf72e
gd95a1c7328bead59g4f7bc8a231547g9cef82ba754d1ge9b65c4d3f829e157c3daf2985c3gdef6b785149d3b7ace25f46g91cb8ef739d421eafc85dg9b
17a6ced928bg375a62194ecbg53a16f829g4ae3bfcd75423g9ef61574ac829f3ebd16g87549fc2g3abde65c82gf9b167ed834gc579b82e1gf63aeb8421c
d3g9a5168f273gc415d6bf9g87a1ecb45d28f61eb435d271f8ba394e5d76fag814dc27b3a91ged4cb8562e9a7dg431862caf5e6d382gabf4e7935dg1fba
743258g1bcd628f7a13gcd6e24bc385a9d6e21f7ga4bc3261e9a8dcfg3b9561a43d8c5g2eb73df49a6bc23871ga5ec42bf16378g9c2e5badfg69237edab
f8g652479cda1f238497cgdf1eb429817ac6dgb4f83561cg27bd65e4gaf3c7961g5df2b746ag83b9d57fa1g3e9c5b7a6f2e8g51c49d6a3gb1e5c8f4d61g
3928f45cg6a319847fe6d534291ge863a72b9fe15367g4b9acf8d62eg9c15fba46c82d75e1a49gb28567f4ec3d1g58fa6c7de59b812afg476e9c2a15g86
4bd3c9f786g2a5c49bd3efagc296b1d8f43gea12bcf5384a9gdb1f3827c64baed8g751c3e4f69b27g8a5946c127a5b93e6gd281f49ba6328fd75bc63924
1dbca65437fe1bg46938defa21746bd53a1fe8gd592417fbca638e12945a31cf76bg59e4af135d6a729f18ecg3d9627g415bea6387d2g4fc96a317bg8dc
2e9f3147c8ab9fe326cd1b7eg48356c291fd5837ce2gda819be746gcad25fb6e3g927a5c8bd2g46a9f5cbed7g3654a8e9d2b4gc157f3284gca7d91385e6
gf7ac4d2eb5g3af7d1e42c9g65db34f2789egad1643c7g8b91a4627g3c1be49d68gca2e457f9b8cge621394ba7c68152d94e6cfa8bg1e26d3af5g9c1e67
d2a45f8g71293ed46f71a93ec45d861gea5943bfd28cae795df3624a1c5e3bd27ag1f6e8cd3b4a62efg9571b3284agbec59f631da8bec7f5328g6d7b19c
54ag7fb39c128e5f94db67gc52efgb18362cfgd475eb8263agf79d5b2e6cf74a82bd96315c2478ae96gbf2d718ecg596413f72eg6da1bc8f27469agc1e7
23df5g981aeb2cd6g4897f3aed5219c836gd4e5f91cagb3678fc54gd32ef7a489cg13bdfc6a7g158bf96ca352876gefbac5921e687df15cg3ba8fe5246g
1cdf52a34g9d15eb82a7g61cf8b927g4e6cf8g7dba94ecf1g65eb24gd79a1e6b8745g91a2fc6753g8defa465bd1e83g97fba2cde83f4g9261a5df47be8g
52d4a1c869ef42531cg6ea2543d1f97ec86dgb35172e9fcbd48673e2c9a4d671g23fae657b9432af15b7c29d48e1ag6c74b25e1g976af38cd95ba17egdf
96c437a28dfbg95e738c4b196d785acfg394bc28efa31974gedb2cf371564gde9785cb1f429egc3bd4a58e279fdb81g6a43759cbg18e435b928a7dcg4b1
f932dga4768e593bcf47285ed1g34c9f587dg3cbae69f158ba3724c1g895ad4cf6be1892673ef5db1ca86937dbe18g452abfc786193a2cd5gbae642781a
dgefc63254d71ef8gc239ab5461ce73a25d491bea625748b9de325c74bgfe37d8ac1g9b456387d29fe8a53g14de6c9a5g8f726d41eba3cf6249da7ef1c8
42d579f613c2db4f5891e3d6fc72bg5938f6dbe7c5a1f425ge378a415fc9gd6ab2138f4d79be6g2cf8a7e5g62dc387be16g2487359agcd41b879f36ge4a
87cbg9fde1285bc739g6258a3db417g6f5ced21384gbf59ad172cb569g8ad71f63e9ac8g715fd2896ge731a2b4gcef89513bda4fc295e38d4gba285c63g
47d918a5ce3b918625fc79d3e68af75b1dce432f769b13adf47g236b9f5cd3a4e8615cg27a69b54f1g8e97526d3b1eac729fgd8a5e2c63dag5b721d9e83
46cb1fg2a5eb794c13a26d875b9c46g32dafb47e98531g6a4fe7b6c928af4edgb6175c39846b2fe5ca7d4612b95fa8ce41329ga6f5c3b1e2g59fa684ebg
513fac275d8gb49c6af1d3g5b8ea716934b5fde2ga14cb982761cga384fb267e3g5c8b4dfe1532g68a1dc5f7368a1cdb29f83gedac96fb478g12d65ea7b
4gf8c53e619bg2d5843b7fe1dc46539e8ab762d4g95fe7a63c9d245g81fcad263b541faeg73b2c6f54ga3d7bcf8g13a2d79453fb2ec97gd4f5318a92g6c
4b1e9f5a3d768e5c92f6gb31e4fadc6283e54f1gcebda984123bc9e76f1d8cg9a3421efbc8a93gd64eb923a6fd4871ab65dc938fe627cgd193eb465c7df
e8g96c43afe8bg92de6a148bfd5g736a9fdeb8756acgb31d28e7af6c932b8ge157a24d3cgb79fa61e253c4d817a2be953c4ae7dg859164c7dfa53b682df
4g9ace76b43dfg9eb56a8142ef6d5c913g284e97d1ca568e9d1bc5ga7e3d94185af2cbge783f5246gde8cbf3456a9c2eg8f4d572cb6a31e7dbgf2657agd
c849e1f368749de5b6834g17ecf258b6g1c7ef39654g8d73c2e94617dg893baf1c56749ag1f8c72d35a6g41bec7928f3dab5g4179285edf347ab6egcf45
a8736dge419fb53cg74a2e1cfg6db93a7c85ed294bg3a1cfe96732g15bcd4a8962e3d7c14g26fd8a5eb743g19dfae6427cf539621ead7bf9c1d3a8efb59
de2ac873fd94g61b28d5f3a72g9c48f1ae5d7294cb38d1eag675c431b8a92c41dfa2397865bfdgc1a749bf2gdc8769ae2f4c61937245ecg68a3247ebgda
5c826f471eab9d384f5g197d2c6bae45f2cd3g79e1ac642b8gf1ed6573ca18fbed96c31fg4bdac29e1gdf6a5b8c9fg3145a267g8bc41a3d76ge8f4596d3
17825e6afg37cb91d86f54a3g285e6bdf4g72ce81954f7bg693e8af467dg935b1a79f24835dg6ce7b914fde32ga985761fb29e54671fbd8g129cab3756f
841263b7d4c8a65e9f8g2b64397c1g42f5ad9c18437gbdfa689415bg2f738de19b264531gbdea2843c5d7ebf21365edagf9762ced48f1g6953e47cb162a
5d3841e7b6da5g92cabfe179gd2a5c4b87e2gd6fac2be8351a7d6fb918347e29b6a8g452fbd79c8621f3deg7c592b16a8d75gf136ace4g2f57a83b946fc
53e9ad46gbe3f8dc56ge97a423be5d18a27cf46b5d39e81g46cd37812ba4d79c581g6abe7f3942dg8caf357e1cbg6a59d32f6ge48f923caef714gd8abe9
136fgad81529eaf7c38g12e65bc96f7gdae258bcg4172d849aef58bg132c69e45173ab2ef89g73516dca4e37b9gd21a3fe2b5g674df1328ga4bcf29a354
8ecgb637ec5814d3b62cg538b7a4fde3c91b6afgd8c134e65ad8f7915ebcadfg12cd8693g72c4eb9gf7845a6c2d1e697f8214b63fe1a82c54961f35274a
c19g5e3fadb49853fc71b4683gfcbd47e693ga185bc4ef3g7826a1bdf594e2a3d5gb978c236agf571e4g23b9f5876egc1fb27ce4a89b231dg7e5349bc1d
57628fe4cd3bg8ea7d6f3c7e15a6f8d2914aeg826145be89g7d1b36597e1f3g45c27e6d5g4a63892c1384bf9gc8b27ega

P_{16}-covering sequence of length 16087 (Part 2 of 2).

123456341789abcdefgdebgdcfgaefbgceabfdabg9df8adg8edacg9ebcfe9ag8fca9dc8bg7de9cf7gc8ef7db8ac7eg6dc7bf6eb7ad6fg5eb89f6cg5bc6a
f9g87e98d79c87b96ce5f97ae6d9bf86bd5f87ag6ba5g76c85g96e85dg4e95bf4da5cf4g86a95dc4bg3fe4ag3c958b49g3ec49f3de5af48967856a75b67
f45e67d56g47956c47e39a48e3ba47d3c756f3c846b3d945c3b745d3af75e3a854g3d84763ca46938g2db45a36g2b835g2f735f2d638f2eg1b539b2e64f
264de2a73g2743f2943e2cf1e2843d29g1d2a43c2af1g2cd1f2b4e29cb268d27e1c4g1c2837263e197g1a8e1b3f182a17391c52b1629165271d529a1e52
41a32517b2a61d42c614b32981c728514g2a5136e149d13g519f617f418g618bc1ad813ced1b927ab1f5

P_{16}^{3}-covering sequence of length 576.

1234562347689abcdefgcdbfgcaefg9cedgabfedacfbega9fdg8cfe9dgcabedg8aef9bcge8fdc9afb8cga9beg7dfb8edc7gfa8dcb9eda8bce7adfb9gd7b
ef6dgb8aec98df79cg87feg6dcb7age6cda7bf98gbf6cgb7fce6bd97ceg5fda6gf97edf5cga6beg5dc96ge89bda6ec87dg68fc7agd98ef5bdg4cfd69gb7
8eg5afb69ca78e96cfa89cb78de69b7a8d97ag68bf59cg4bea5cde4gfb5cef4agd59bg4acf58g967fd58gc67ed59fg48ce59fa68cd5be86abc59a87g9e5
ad968bc57eg48db67fg56ef49dg3bea79f85aef68da5bg67ef48de56cg47fb56cf48bg3cfb4adf3ceb4dfg38d957cd4ae967cf49ba67eb49c867da48e95
7gb46ea57bd49eg3cd94af75ca67f856dc48b956gd47f956db4ca857ga46fg39ce4ag8569c478g649e567d948a657f469a75dg3efd46eg3af849ga56df3
ba95678b5gc4e79b45eg37ba46ce3adg29ef38cg2bef45de3bfg2ecd3abg29fc3ab856eb39g746da38eg27f846d93ae7458f39ac45fg37df2agc36dg28f
b37ef28dc37e846bc37a945fb39c875ef36a874de38b745cd3b9785db389745ga36cf2bd548c657g36ba54dg27c645fa37de2bc837ag26ea35cg26f937c
f2ae548g369845bg34cf2de938ag2ed6378g2bd638ce29b637a546e735ce27b6459g27bd34fb2ad53f9d28e635fc2g9835eb28d5379e28ac34be2a9635b
c2d9736fb29a536c927bc459d2ae439f27b534dc2ab538g25b934ce26f83d7456834da28f436g25e437g269743ba27f43eg1cd2gf1ae2cf17e26bg19f25
a84be1dg34a72e8435f26d539g1df25b67cd1fe259436b82a9348729ag18f7ae19d26c435g24a538c429e16f98c52dg18c26a43fg17b59e32gc1af24eg1
ad23fe1gb24df1bc29af1be23cf19b24cg1bf23ga98e1ac27d16e257g18e25d17g24f6bd15f37826ad1bg32d546g17c28f159823de1ab62c37652c91bd2
39c1ed248g16f237e1cb329g15f249d1fa627f14g268d1ca4298f1d782a438b17f52ga14e238c1ab247e18f324c73g14d23c16a47c13e62935c19d346e1
g9423g15e32a56f18b426981af321ec4527d415g631ag691b723d18e671ac527ad1435728b1e9361be5713d6215b4238a14726d41c9821a7326a19b4615
b32617c5g1286519b8254178352c169251386241c3a2157921379241b32817561ecf51bcd61fcg917de51c8f419ef51a9d81abg51d3f917ag518bge71b3
f716bf413aef41c8e41582f4168c71943g8193ba154fa17bd41e3bd185e91763417bc61g2e51a3dc1549817a5d1964c1b86714gb4a193561c9a21483e1a
68a516dgc136fe9b2d184e261b87fcdfga

P_{16}^{4}-covering sequence of length 1879.

12345612789a6bcdefgbcdeafgcd9efgc8defb9gdeabfcd9agbfc8dagecb9dg8ebfc9g8efac9bd8ecag9f8deab9g8eabc9g8adf9b8acfg79edf6gced7bf
gc6aefg78ced9afgeb6dfgab8dfg7abef69cge7adcf69bge7cfbd6aef98bceg68fde7abcg68aef79bdg68cfd79acf68bde79agf68beg7fdce69abg78cdb
a79ceb6agdc7b9gf87bdg59cfb789df6a8be97a8gd69acg789de67bfg59def4gcdb869egf5dceg4adfe5bgdc67feg5acdf4bged78acb67dgf5abeg49fce
789gb67ecg59bef48dge69adb5ecfa78bec59dfb67cgf58ecd67afg569edb58cg967efd58bga67ce958bfg47ceb56fge48cdf57bcg48aed678fg56cdea5
8fc967bda89cef579dga5bcd967g8af5bc8967eag589ba768dca57edg49cb8769df587cd95afe768cg5a9fb678dg4a9ce56adg47bef857bde49afc579gf
4bceg56bcf4aeg958dcg4abfd56abg489ef56abe759af84ecba58dfg3bedf47cbd568gb75acgf48ac9578eb65dgf47aec869ag756fce479fb568ae956bc
a48ebg39dfb489da7589de47abgd49fc6589fg468dc94bad865fed46bfg39ecd47af968e5dg4b6ec857abf469eg3acde46afg3dceb49afg38bde45afc64
dge3afdb749cg567df459eg748fc974adbe39fcb45age648fbc37eg8569bg476fc845efg37def2agdbc3fg9765cda46bef3acdb45afg37be945dgc74ab9
657daf38egc54beg36adf45cga648bdc37ae946dfg34cedf39ca657feb38cg745bdf37ced548bfa39db756eg4759bag38ed746gcf38ed9a56f87a5gde39
af8749df837gd946bcg37ad9648ec93adg857cb6489ga37ef645bea389c745fgb3acfg28ae746cae35dfg29bec36fd847acf36bd9547fb839gd6c3ag964
7be839gc645bg937cd8459fb368ea459fg368cb547ef398cdg37ba459ce34bfg28dea36bg8459cd36eb9487ef35ac9b37fa645cf9378e654agd35cfg24e
db37ac9648ad537fcg2def638cdf2begc35beg2acd835bef29cd7469fa34cdg29ec645dg8369e745ac8647ab836de7548df359eg28bdf34aeg27fd936ef
g25bd983abce26gdb34afg27bd645ab8349ef2gcb936ad745cef348ce739g8f27ade348ga53cbfe28cga34bcg27adb53ga9458eg36798456fg349cf2aeb
735cde279f863ac78659a478gb357fg26dcb439eg2a6d9835cbd29ga7458gc347eg258fb349d8g27cd634fcd2aeb9358fa4d9c6547dg259f763ceg248c9
g2fab6739d5687b45687g35ea8736c9f25de6487db348gf26bc734dfe29acg1edb534ed92abc635ed92bfa8e2bcdg1afc7b2ed8573ae6f29bc837fa928c
e634gb928efg16dc534bfc289d734ace27gb635ef82dab635fab279g534fd928ag635e9a28bcg19df268c934abe269g538fca2bdfg18ce6f2bd7368f527
ed436gc729ea635cg926db537ag825cdg17fcd26ae435cg268b7359ad278e5349dg27e95647a356df24beg17fb634d9c2a6g734af826beg15ca436g9b25
edg1fce278g6459bc278a6539ba258eb437fg296a8347cf25aeg1bcf258d436af27e6b3479g268ef1bcd278f6439ad24cef1abg265f7348bc25adg19bf2
87c543f8926e7534g8d25abg18ed2c69743ae8267d438cae25bcg16ed3g2fc6534ef27b98365dg23aef19dc257bg19ed267ae5269af18de357ca24bdg17
ec26afg189de23bgf1ead257f4a2edg1a8e492ab6439c725gaf19db278g492fad534gb72acd179e358ca23fge19cb268a5436bc249eg15fc32bef16cg24
a98357fa23bdf189c256b83459b823edc1fab362fd736gf19ce2589g14fb267a5b269753486c537g24abf18cd264eg187f4569dg15bf248de1bca239ef1
7bd2589f17de23acg18db247cg16ad248cf17ea248bg19ae5b8g179f345a927cf16bd3c9g186e254dg179e246bg17ac265eg13bd286g435ab178f246e38
2c76435d7269f1e5c2fde169g245ef189b57ge14df239cd1abe237gd16be253df17ab283fg158e247bf19ac246fg17ad245cf169d28bc1a6f25bd14ag67
de18af259e168d237cf19be245ag178d3abg165f4d2g67e18cf23de19ba2g8d16ce248f16cd23gbc1a9e346f5239be17ag235ec189d24bce16ag238dc17
bg238ef14gc239bf168g254ce18b9247fg13cd256ag189e237cg159f263cg14ae236dg14cb2379g165d283af16bg239af15ed97cg185a427b8e1acd324e
g13af254g67258g139f246a8f157b283c542bg186c5dfb148e26db1a89253ef149d238f1b5c396e14bd325ac1b7f326e7438g174b62398c17a9236g145c
268a159g234bf159d237ae169c427df15c82469d17ab325gf14ad236b752ecg138e279b138a4d1c8a24gd185cae13bd591ce7425c918ad361f98247d51a
cf423gd9147f32ec9d16ac324eb8615fbe415d8641cg8235f79168da21ge6325c71deb2g1f8432ga184f523bc1689257d16f8237g615ce761fad321fe78
91db6425ab16e92348e19ad581bc9571ea8671db9321cg9421fe6521fea4b1c87425b9716eb3215ea321cea451dce831ac9523a87159d6231cf62719ga2
51dfa24189a731gb94a17e9431cfa2715gd37425g314b8623gf14239c617ab9315d8721ge5431ad9451g86351c78351ac7324a97651ed73914fb6751486
2357b186a3257e1f462798315e9421a69425f1a78423d514f986b1dc53a169b4231gb528147e321gdf3e162d431ga965123f641cb96g1475923678149c5
31476a2345b71ec3627415e6381bd42519372f19a76315gc2a1439561cb748132b651ead6g127d63192d7413g928136f7412ab34169f531a7d8f1356d24
81c3feg47a156c27194b8a12374b1a58271c34285b31ec4g68175g3624c175f2b941386c219468375219a3824b17e28f15d64e185924361c7d4a16c2bg1
953ad1e7465281d34cb1524e316a45196b281fd4761c8af941cdb3271gf5db1ag4261b73e1g92f3a41c9735f1ba2df614a8cd2g154b6a12538eb159a726
1e4c9g613b54819d34e1a62371dc254a1f73821g479613bfc412de5317db491ea2871b9c31fd4c1e28g7951cf29e1387bd156afbg813d6c145831cb75g6
21fbda9125be1c2df391865egd13abc6fea1c74f5132c879abfe13761ag359d815eabg32f1b7c21gfe8dbc196a85gf4d1786f9gd12bcag8e76a1de491ag
839472682be143d6571c39e76ba31472e5gcfbd1e8579fdg84b91567fac8f9d24539af2g1

P_{16}^5-covering sequence of length 4501.

12345671289a5bcdefgabcdef9abgce8fdbga9e8dcfag7e9cdfgb89cefga8bcdf9ag8bdec9ag8fed9ag7cbf9a8dcge9bf6adgec7bfdge69cfdg78bcef6a
gbce79fgbd68cfge7dabcg69daef78bdce6a9fgd78aceg698bfeg6cdbf978edcf6ag9be78acfg679dcfa68edgb798cfg67bdef968adeb978cdge67acfb8
69deg7f6cdba897fgba86ecd97a8eg697abefd68g9cbd7a8egf6798cef679bdga689bdg5acefd769abce769afg58cedf4abceg57adfbg59cedb679cfa86
7begf59cdag48fedba59fedg4bcfe9a78cgb6a7efg58baecf59bdeg47acdf58becg46afeb859dfec4gadf958cgdf49begc687dfeg58adfb678cde59agbc
859efg74dbce975dgcb94adef857cgde48bcfg56abde75cfgab48cdf975bgeda49cfg75baefg47bcef65dgbc849fgb768adcg56ebdf47ceg985abdg785a
fec748gde956cfge49acfb659egf46dcea759dfg47abdf948ecg659abec489dgf56aebg478cfd569aeg478fbe567acd956gbfe48acg9678bec56dafe479
bcd58a9ec658fgbd469fga847def658acde479gbc658deba479dg865aecb748dgf657bde946cfdb578agc6579fde468cfb579adg657cfba489dcb64aedg
5679ed856afb9857cgba46dfg7859beg4a8dbf749agb8749eda6789g5a67bfg45cdef37bcdg648efa9568bca947dfcg38befa946cdge3abcf9587abf469
gce756bg984dafc6758gbe467cdb54aefdg39bcfd458egac3fbdge586bag45f9ca748df9674egf5dc7689ae57g89da56b79c48g6fad7e58g69f45bdeg39
acdbg45a9ef3bdag945cefg34bcdef36bcda746gcbf379gdc45aefg36bdce459bfg38debf29cged38acgf458bdg647cfag36bed954acbg398efac39deb8
46acdg35febc468dfa7569ef847ace936dgc8549def37acdg25feca649bd8567fga458ceg369fbc458efg379ce846adg93fcedg2abfe746abcf389eba45
6deg37bca9567dbga36cefg2bdce8546dgf37abc854dafg389def27bcde359fgab38cdef29gac745bfdg349ec8576aeg457dbe845adbf389ad7548gfd37
baeg268fca457efc638bed7649cag378def26cbg9457fde36abf8457cdg389abce389ag745bcea378fbe649a8f765b98f467dae39f7b8469adf378ceg26
dfc7459ged37c9f845e7cg36b9df27cge6459dfc34beg963acfdg28bcf7468ega379bdg28ecb9645efg37c89d467ba9f36gda5489eg734cdef2abcgd359
beg24afce358dfg29beda389cfd638bgc456aef735bdcf26agde35cag9836cdea28bfg7e3adcf546bdg3789fe63b8adg297ec6459fa837cde6547gfc369
8eb5a79gf368deg29abdf348aeg27cdba359cfg28edaf358cdg269fea34dcge25bfca348gfb637adc546bedg25afeb349cdg28afcd348eg9637dfa459bd
c638afg5639ef75869df358beg279fd8546bfg3589ca736bef54869c7548gab3678ed53bagf493bda8564cfe359bcg27a8f6e37gb8645cgf3687dg2c6ae
7458bc934fe8g279dc6547bge356adbc25gaf7369cef27dac934bgde27acf9348bef279ag6487db9638ae7469bga347efg26abc7539adg268be9457ac83
4bedf26abe9347dfg259eda34g9ef28cbd9358cfg24bdc835abge289cd753eba84759de367b98a35ge9c2bdfa5647bdf365cbg734defg27bce8359adf26
8gb9347agf269bce375adg4369fde25acg7349dce26bad8476cg9348adg257cfd346beg2ca9d8637cf5e289db573ag96548eg376ace8347bfg259ac863b
9f75468af3579dg248bfc3578bg4359eca278dg563cabed249gb8536ceg258acbf279ad634cega289fc653edbf258eda3479fg284de9a28bdc734ab8f26
9gc5378ead269be75389dg267fbc346dge257bfa346ebf298cd653afcg278ba6539ce745d8f3749ea837bd6c28aefg16bce354dfg237bc9d2aefg346acf
2789e6537gde2698g74569bd345fag2649ea5367gf4839dbg258fa7469dg8346cfg279ba843dcgf249bc7368ad5736cg8a2b9fe574baf2598g634abd75c
8af9436edf249ceg16abf4538cbg247ad8926bcd534aec9287bf6359adc267ef534acbe286dc4539fbd287eg543bade279cb6847ecd258be6439abf267e
d8436aeg2479b8536eaf258bcg1adeb4c2fda9435bcg24adef18bgd3549fg237cef19gbc3ef8g6537b9g258ef6347dbe25c9d6347cef286ag543c98a24e
bfg17aed265bge427dfag189ef265cg8927ac65389de275fg3468e97346gb8c29agb3ef9c7438gdb27a9eg15cdf2e3gc89267bf435cdg246bca527dfbg1
9adf2g3be8a269b7435aeg267bce346adf23cbeg18dcf256bd8f2a9e537g8a629cfd167ec8249dfg17ecd284af73689af276db53489g275ec6348bd7529
fb8357dgb23aceg1bfad273be9g267ca4538fg2659ab427cfg196ad4538eaf2759c4368df529cbd185eg3d28fc7345ebf23adeg19fac2e39fag248cef1d
bgc293df87269aeg1cfda256cf472aeb6539gdf187ag4c29ba7536fde19abg254de7a259dbf18ace256dfg14abdc238efg169de254cfg16dbe2489f7526
gbcd18agb2659d7438a96547e8923bdeg178cf264bgaf27e64359ag2478bac269eg17bcf253ge8627bd9e1cabf263egbcd5fag326bdf17ceb25a9c6437b
ef249abg18f9c256be4729d86547cg2369df18cbe246ac8725bf6g18abe2579fg14bde236cfg159de348fcg15ead2469fg15abd2479cg16ade238fbg159
ef246ab8527ea6345cge239fbg178ad264cf9726gb4357dc826fbe1c4gd652c897356f8924adce1bg9d265ag7426ceg139fca248edf17abe345dg762abd
g179ce254ag837bfd245acg13def2458gd327cbae15gfb2a9ce175fg392de67538abf197cd36fge174cf3a28d95436bf8247cdg169be253adg189bc2468
ed937b8c425afe168dg234cef1dabc3469f723acde18bac2369fg17ace253dfg168ce2379dg15bec239daf18c9e2658c4376ad2587fg16adb2489ag1c7b
d246cef19bad235fb9c1adg7436ed5248fcd16egf245bc639adb164eg3529cef158gd2649cg5278b6459gd8167bc5298eb1a6fc372beg189cd2548be326
9c75efbd168cg2739ef16acg234bdf17e9ba35gc84269gb378cf15adb823fc9ga5efd1749ef2368cg14afe237dcf14ged69bac154fd623gcbd197af4539
e724cad17b8g264fd75gb9c146ge273adg14c9f2378e4629cag158ec243bfg157ce236abd189eg243abf1789e2547fd326cbe1897d236ca849egb138cf5
627d84356fe248a7c1bf89245bde17acf2587ed1c9fb237agf15dbc23e8ag1d67e24a9gd178cb2369af1c78e253dbg1689e23a7cg16b9f245cd3827b9g1
5adc2389ef167ad245g97368ce157bg2349de15cfa2368g742d6983457g2839f642a89753fcgb14aef8519cdb324ae9b16fad3b1ef9d235cf9178ag235c
8e1a9d7325cfe149bd2367ef159ab2378eg16b7f234cde189fa2358bg1679a254e8g16a9b42dfag139de2a1gcfb23age1789c245abe139dc481gef7234g
ef15d6b238gaf149eg5268bf15d7g2349ce168ad324egb8179ac325b9g148fd2357bc42egf1378e5426ga814edg768af1456be327a85439f7b145da6239
eg146af2537eg149ac23gde156cg2348f7526abf158ed632ga971b5ef26gbd175ag4328baf167cb2439a752gbc163dg8a516fg8459ab136gf4278ed169b
f4328edb16ace5g4bf17e8a295ge167ac3258fe16b7a243gfc17da9425f76348ba153gf2678e145af28d7g142eb9c16de83a1cb9832agd154cea9158ge2
d1bf97243cd871ae96254fg13abe291fg78234cab1869c324daf1689b5276eg1389b462df89137ed451gfac341ega7253eb491cda8254bf7619dea361bf
87325bag1678c2356eg12d9c5g16db42579d1c86a2547eg185b9237df162cg4a17be8391cfea2g1bc7945dgb128cf491db87243ac591fd87425ceb817df
c2417eb8561dfcb2e1gfc8231gdb7243eca714fe8359g7164fe3259db134gc571df86324eb9615dcb421gdc8321egc8341fd96524cd7613beg521fdb739
1cfb6324dgab312fd9541agb8521edb7621gfb7351deb9251fcba421fgd7351feb9841fda7351eca7851gd67235g9d167bc351fd97241ef96271cga8321
deba521fde6451cgb7391agbc6519ea7361gc98241gde9c418gb7321fec7641bgf82e1cdb7521cgd73619eb8531dfb8421ega93246ed9157afe9135gfe6
142dga6153fca7156gb231cfd9g21afb9351ced8391aec6351dea6751dfa83614fbe2317gc8421bg96341fc965a1dc96235dg147b93268g571cab9231dg
fb2534ce8159da321cf96841bfcd381ab76345df126ae4819fa5628ed146cb3258d97165ecb91327g6a149cb2514egc27185gb4217fa83219ga6341dfe9
5613feb7ag125ed431gca6523e7d814ac9581bfcg5218fae3g1548d2635c7921geb3481gd925a1gcb35e14agc68153efa217db95412fb93618fg92b13e5
742b8e6132dc97163fg2b146cf5324db86129g5841fba53246fa7134ec261ad87321ec963415cda7g63425ea6147de231cb86571dc86231bda56217ca94
31bfc74912ad8531gce76b1539e4817c96421fca64517ea86231gc975a18fgc394182ea3418g973418fb72a136dc4215ga83615fg74216fbc8g719243e8
56274d35a2768354ab6235fd1a9682543d6g12f596347a816c57342fa971352e96147fc52138fb52136ac4215fe783416cebdf41a9b7631g9d47b31ecf6
2a15bc82619da3421fed73g128fa4319gf85b71248f391bd863571gba34e1628f541eb972813bde4921cgf3d41ae968b174cg63154fcba31ef74561d3a7
241dba82715f9832156g84213dcb8g6135d9241fg8357126g93541fa925c147ba2513e892517eg9f847153ga2614d9732168d7431gb56423cf1756a8914
5ec261348b5613d8924157g926417ca254619c8532417gc35216fc73416abc78913af2547189b3241gab6291435b72f1e8b32517fa326147892631ea856
413e972651b9g3412ca8731596a7315cd26715f9324d1ea732468c1a35b6f1239g58174ab681529c3417d58321ab68324g591632bc71432f6718c3b4215
a964218a769415b87361d25g4763198a4236d914835a2917b645281cf65e217dg9e841dbc37a12gf345b162eg34186953248b1d576392a1e4592b3641d8
a2g14786c25143e82g9a1478f6231c75462e17b86235a941735a2417e365b1cad2e135fd6gc219b754a1c83249675814c6dfa1e36g9425b6f91582df316
b74231d5b42186fd4b123e6781f5342871dc4be21a6g54f19723a814d5c9136a457d12f46581e97624a1b3759d14g235b1c49fa6128e5a3cd17e5ga3417
d629f1845cbe1d36c9b1758c36gb418d3c6f18e3cb4198ad7fcb5164d9c21a856d192b8g31a9c275d18ae29f17c6g52418efba517e28g46d158cg931d85
49c17e36dg1a485297348b1e5a63f417bad34159cg2e1dfc3a819b5da7219ed5a41f7g3251ecad671b4e5g16897fb516ed2c41873gaf6127eab8521d643
e81df2aeb1c486e913df78213c97561ba42931fe4ac81f64ea1c23gdb9146gd7f1453ecf1g2463ae9152c6a197f341dcaf97g13ebf4591e73bd812fg69b
e71c4d567e184ab2c6193fa581cd2b67g381de54f61ca5321d7c495f16738ae1b4639871d4b9a156geb83c15dbg89de21c5gd3241fbce5gf6794b1286ge
f9bd82763abc127dag9315c48gfe1da4gbf9c6e174gb5721e48cdg2a1c8f5d14bcg6ef13285g7a69d13b276598c174fg89dc13a45bge6da41c37426a839
15b2c6872g15fad63bg5a2d4186g93725f368b416e2c749ae13fcg5324e9g1a57b8342fce17456g83d1f2age18c37gae8bcd4295g8db67g4ab28917fd46
159eab7cf2dg169a8cg57ea1d47eb9df5623e1

P_{16}^{6}-covering sequence of length 8279.

123456789abcdef315428gabcdef9657348gbcdef9a67258gbcdef3a67498gbcde5fa634789gcd2ebf564789gacd2b3f65789eacg243bf6789decg153bf
6a89dec427gbf6a89de3c257bg6a8f4e3d2579bgcaf46e13879bdcag56f1e879b4dca53f6ge29b78d1acf6g45e9b328acdfg46e957b3acd2f8g4e5679a3
bd28g4ef57ca36d29g8ef5bc1a47d69gef8235c4ab79defg185c46ba9de23f7g546b8cdae2f379g548bcae62d379g458bcf16e3a7g9d5b8f4623e7a9cdg
18f45637abc2e98gf14d67ba5ec3g2f1d98ba7e56g243c9fbdae17g2568cfb493de1g578caf2b94de1678cag3b594def127cg8a6b95d3e42fg1c7b98d56
4a2f3c7g9bde6418f3a5c9bg27e1684adc95bf3g1678ade9c5b241g78adefc3b261g79adefc45283679abgf145e2c689ad3g1f475ce8b6d2a1fg95ce847
62b3dfg1a5e97824bd3g1fca978564b3e1dgfca8954b26e17fgcd893452ae17fgbd6c4352e19afbg8764c251dafegb3864291cafedg5368491c7aedf256
83g1c9beda472f568g19bde4a735f26g1c8dbe4a3572g19cfdeb6457231agcdeb894572316gfedc9ab425167g8ecdf3429167agecbf5342169dgaecb534
21689dfega573c4619bfdga283746c1bfdga28579436c1efgd2b75893c1aefg64d258b91cafe6732d84g19cbe5673df8g129ac5467ef8b123gad5976f4b
123cadeg78645123cbdaf79864125cbedfa78341629cegfab8531472cegda98561437bfecd9a8215367gefcb98214537gefdcab816932g7ef5c4816a329
dbef74815632ag9efb471d635a8cefb271g43569cdab271f4g5683d9b2a1ecg5764d93281efbga6c4752d831fgbac67925d3184bgc6f925a3184bgdc769
f25314eagc879f26314bdgea59f827c134debg65827a134dfeg956c27a134bef895d2c713g6be894f5da71263ecg8fbd4592137cga8be465d1293g7abe8
46f152d3c9ga8b6415e23dc798ga4b15f3e6279dag81453ef26c9b7814a3d526ecfb719a345dg82f6c1e97a53db2f8g619ea4573b2dcf18e69573a42dc1
fe8b673g429ca1f5bd6387291cea4b5f3826917cg4dab3e6289175cg4dfa36b28157eg9d4f6a2b5183cge946d2b57a18fc943ed2gb6718a9345fcdb2ge7
a1948f326bdc571ag8f3429bced16a7g5329cf84ed1abg6735fc289d4167b3e5ca28d61g479fb3a25d6c1gf479b23e5681afd79bc4g25e613a89df47g2c
3ae16859fbg37d26ec845f1973db62g5ef148d73ba2g5f6e19cbd34a85216g9cb374fa85d629ecb1g75f8346dae9bg517fc3462aebgd17f89c3265ab1fe
d947g256c8b13ag7e49c5d681af4g3b9ce7251d8agf4cb3e2586197fdagce3468129bg5fd7ac38e9124b7g5a6cfe381dgb2945a76ef1c3d2gb5a74891e6
cgbd2a39587f146cbg325efd817bc4g3965fe8a2d14b6c37f9egda8b1437fc29e8agb456fd7e89g13256c4fdagb9e7658fdcgb9a765efdcg89ab65ef7cd
89abg5ef67c89abdge567c89abd1346ge72

P_{16}^{12}-covering sequence of length 2126.

123456789abcdef2g416738abcdef9g4567283acdebfg4591627ac8defg459b167ac3d2fge59b164a87cdf3g59e164b8acd27g59f3e46bacd28g59f1347
baecd6g58923f4bae7dcg51829f463bade7g1529f863cabe4g1d29f5876abc3e1g294d587a6b3cfg294d157aeb38fg629d1c7aeb58f3g642197acb5ed3g
6421f789cb5ed36421f78ac9beg364512f8dac9eg37461258dbc9fge73412a586dcbgef74392158dcabef6g3972481dabec6f35724g18dbae6f937524c8
1bagdf6395724c1eab8fd396574g1eac8fd326957g1bec84d3a2f96751becg83da2967451fegc38bd296451aefc3g8db297564eaf1cb8325g7d6e4fac1b
825g79d364fe1bc82g5a93746eb1d8cg27f493a61eb58g2df4c796b1a58g3d2ef9416ca7b28eg539f4ad17c8beg62f349751ce8d6a2g37f5c41b68a2d93
e57cg4b619df387e526gca143d89b7gef2c6a45981gefbdca67935gefbdca8679321b4dfec5a2398b7dfg64

P_{16}^{13}-covering sequence of length 702.

123456789abcdefg23456718abcd9fg2345671eabc89fg2345671eadb89fg2345671cde9b8fg2345671ac9d8efg234567bac8d1efg23459ab6c8d1efg23
4579abc8d1efg23679ab4cde81fg2379abcde5641f8g79abdec52614f89gbdeac756134fgbde9ac7682513f4beg9ac8d56372

P_{16}^{14}-covering sequence of length 224.

D Solution checker

The feasibility of the previously displayed sequences (and also of those previously appearing in the thesis) can be verified by the following Julia code.

```
const alphabet = "123456789abcdefghijklmno"

function check(S::String, n::Int = 0)
    n = n == 0 ? length(Set(S)) : n
    @assert n <= 24
    txt2bits = Dict((alphabet[m+1], UInt(2^m)) for m in 0:n-1)
    sequence = [txt2bits[c] for c in S]
    covered = Dict{UInt, UInt}()
    for i in 1:length(sequence)
        subset = UInt(0)
        for j in i:min(i+n-1, length(sequence))
            subset & sequence[j] > 0 && break
            subset |= sequence[j]
            covered[subset] = i
        end
    end
    k = 0
    cnts = Dict{UInt8, Int}()
    for subset in sort(UInt(1):UInt(2^n-1), by=x->(count_ones(x), x))
        if count_ones(subset) > k
            k = count_ones(subset)
            cnts[k] = 0
            println("\nSubsets of size ", k, " found: ")
        end
        bs = bitstring(subset)[end-n+1:end]
        if !(subset in keys(covered))
            println("[ ] ", bs, " is missing.")
        else
            i = covered[subset]
            println("[x] ", bs, " at position ", i, " as ", S[i:i+k-1])
            cnts[k] += 1
        end
    end
    println("\nStatistics: ")
    for k in 1:n
        total = binomial(n, k)
        println("[", cnts[k] == total ? "x" : " ", "] Subsets of size ",
            k, " found: ", cnts[k], " of ", total, ".")
    end
end
```

For example we may call `check("12345136751273142753267436524617456l2")` where the function automatically infers that $n = 7$. If we want to explicitly set n we may call `check("12345136751273142753267436524617456l2", 7)` for instance.

The output from those two calls is identical and looks as follows:

```
Subsets of size 1 found:
[x] 0000001 at position 36 as 1
[x] 0000010 at position 37 as 2
[x] 0000100 at position 25 as 3
[x] 0001000 at position 33 as 4
[x] 0010000 at position 34 as 5
[x] 0100000 at position 35 as 6
[x] 1000000 at position 32 as 7

Subsets of size 2 found:
[x] 0000011 at position 36 as 12
[x] 0000101 at position 14 as 31
[x] 0000110 at position 20 as 32
[x] 0001001 at position 15 as 14
[x] 0001010 at position 28 as 24
[x] 0001100 at position 24 as 43
[x] 0010001 at position 10 as 51
[x] 0010010 at position 27 as 52
[x] 0010100 at position 19 as 53
[x] 0011000 at position 33 as 45
[x] 0100001 at position 35 as 61
[x] 0100010 at position 21 as 26
[x] 0100100 at position 25 as 36
[x] 0101000 at position 29 as 46
[x] 0110000 at position 34 as 56
[x] 1000001 at position 31 as 17
[x] 1000010 at position 17 as 27
[x] 1000100 at position 13 as 73
[x] 1001000 at position 32 as 74
[x] 1010000 at position 18 as 75
[x] 1100000 at position 22 as 67

Subsets of size 3 found:
[x] 0000111 at position 1 as 123
[x] 0001011 at position 15 as 142
[x] 0001101 at position 14 as 314
[x] 0001110 at position 2 as 234
[x] 0010011 at position 10 as 512
[x] 0010101 at position 5 as 513
[x] 0010110 at position 19 as 532
[x] 0011001 at position 4 as 451
[x] 0011010 at position 27 as 524
[x] 0011100 at position 3 as 345
[x] 0100011 at position 35 as 612
[x] 0100101 at position 6 as 136
[x] 0100110 at position 20 as 326
[x] 0101001 at position 29 as 461
[x] 0101010 at position 28 as 246
[x] 0101100 at position 24 as 436
```

```
[x] 0110001 at position 34 as 561
[x] 0110010 at position 26 as 652
[x] 0110100 at position 25 as 365
[x] 0111000 at position 33 as 456
[x] 1000011 at position 11 as 127
[x] 1000101 at position 13 as 731
[x] 1000110 at position 12 as 273
[x] 1001001 at position 31 as 174
[x] 1001010 at position 16 as 427
[x] 1001100 at position 23 as 743
[x] 1010001 at position 9 as 751
[x] 1010010 at position 17 as 275
[x] 1010100 at position 18 as 753
[x] 1011000 at position 32 as 745
[x] 1100001 at position 30 as 617
[x] 1100010 at position 21 as 267
[x] 1100100 at position 7 as 367
[x] 1101000 at position 22 as 674
[x] 1110000 at position 8 as 675

Subsets of size 4 found:
[x] 0001111 at position 14 as 3142
[ ] 0010111 is missing.
[ ] 0011011 is missing.
[x] 0011101 at position 4 as 4513
[x] 0011110 at position 2 as 2345
[ ] 0100111 is missing.
[x] 0101011 at position 28 as 2461
[ ] 0101101 is missing.
[ ] 0101110 is missing.
[x] 0110011 at position 34 as 5612
[x] 0110101 at position 5 as 5136
[x] 0110110 at position 25 as 3652
[x] 0111001 at position 33 as 4561
[x] 0111010 at position 27 as 5246
[x] 0111100 at position 24 as 4365
[x] 1000111 at position 12 as 2731
[x] 1001011 at position 15 as 1427
[x] 1001101 at position 13 as 7314
[ ] 1001110 is missing.
[x] 1010011 at position 10 as 5127
[ ] 1010101 is missing.
[x] 1010110 at position 18 as 7532
[x] 1011001 at position 31 as 1745
[x] 1011010 at position 16 as 4275
[ ] 1011100 is missing.
[ ] 1100011 is missing.
[x] 1100101 at position 6 as 1367
[x] 1100110 at position 20 as 3267
[x] 1101001 at position 30 as 6174
[x] 1101010 at position 21 as 2674
[x] 1101100 at position 23 as 7436
[x] 1110001 at position 8 as 6751
[ ] 1110010 is missing.
[x] 1110100 at position 7 as 3675
```

```
[x] 1111000 at position 32 as 7456

Subsets of size 5 found:
[x] 0011111 at position 2 as 23451
[ ] 0101111 is missing.
[ ] 0110111 is missing.
[x] 0111011 at position 33 as 45612
[x] 0111101 at position 4 as 45136
[x] 0111110 at position 25 as 36524
[x] 1001111 at position 14 as 31427
[x] 1010111 at position 10 as 51273
[x] 1011011 at position 15 as 14275
[ ] 1011101 is missing.
[x] 1011110 at position 16 as 42753
[ ] 1100111 is missing.
[x] 1101011 at position 28 as 24617
[ ] 1101101 is missing.
[x] 1101110 at position 21 as 26743
[x] 1110011 at position 8 as 67512
[x] 1110101 at position 7 as 36751
[x] 1110110 at position 19 as 53267
[x] 1111001 at position 32 as 74561
[ ] 1111010 is missing.
[x] 1111100 at position 23 as 74365

Subsets of size 6 found:
[ ] 0111111 is missing.
[x] 1011111 at position 15 as 142753
[ ] 1101111 is missing.
[x] 1110111 at position 7 as 367512
[x] 1111011 at position 32 as 745612
[x] 1111101 at position 4 as 451367
[x] 1111110 at position 23 as 743652

Subsets of size 7 found:
[ ] 1111111 is missing.

Statistics:
[x] Subsets of size 1 found: 7 of 7.
[x] Subsets of size 2 found: 21 of 21.
[x] Subsets of size 3 found: 35 of 35.
[ ] Subsets of size 4 found: 25 of 35.
[ ] Subsets of size 5 found: 15 of 21.
[ ] Subsets of size 6 found: 5 of 7.
[ ] Subsets of size 7 found: 0 of 1.
```