

Technische Universität Wien Institut für Computergraphik und Algorithmen Arbeitsbereich für Algorithmen und Datenstrukturen



186.172 Algorithmen und Datenstrukturen 1 VL 4.0 3. Übungstest SS 2008 27. Juni 2008

Machen Sie die folgenden Angabe	n bitte in	deutlicher	Blockschr	ift:	
Nachname:		Vorname:			
Matrikelnummer:			Studienk	kennzahl:	
		Anzahl al	ogegebene:	r Zusatzblä	itter:
Legen Sie bitte Ihren Studentenau Sie können die Lösungen entwede				oder ouf 7	/ucatzblätto
schreiben, die Sie auf Wunsch von mitgebrachtes eigenes Papier zu v	der Aufs	sicht erhalte			
Die Verwendung von Taschenrecht Ausarbeitungen oder vergleichbare	,	,	- /	Büchern,	Mitschriften
Die Arbeitszeit beträgt 55 Minute	en.				
	A1:	A2:	A3:	Summe:	
Erreichbare Punkte:	15	18	17	50	
Erreichte Punkte:					

Viel Erfolg!

a) (5 Punkte)

Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} n \log_2 n^3, & \text{falls } n \text{ eine 2er-Potenz ist} \\ 3n^2 - 2n & \text{sonst} \end{cases}$$

Beweisen oder widerlegen Sie:

$$f(n) = \Theta(n^2)$$

Beachten Sie, dass für einen korrekten Beweis auch geeignete Werte für die Konstanten n_0 , c_1 und c_2 angegeben werden müssen!

b) (6 Punkte)

Seien f(n), g(n) und h(n) Funktionen mit positivem Wertebereich. Sind die folgenden Aussagen wahr oder falsch?

Aussage	wahr	falsch
$f(n) = O(g(n)) \land h(n) = \Omega(g(n)) \Rightarrow f(n) = O(h(n))$		
$f(n) = O(n \cdot g(n)) \Rightarrow g(n) = \Omega(n \cdot f(n))$		
$f(n) = O(g(n)) \lor f(n) = \Omega(g(n)) \Rightarrow f(n) = \Theta(g(n))$		

c) (4 Punkte)

Gegeben ist der folgende Algorithmus:

Algorithmus 1: WasBinIch(A, i)

```
Eingabe: Feld A[1, ...n] mit n ganzen Zahlen, Feldindex i
1: m = i;
2: für j = i + 1, i + 2, ...n {
3: falls A[j] < A[m] dann {
4: m = j;
5: }
6: }
7: Vertausche A[i] und A[m];
8: falls i < n dann {
9: WasBinIch(A, i + 1);
10: }
```

- Um welches konkrete aus der Vorlesung bekannte Verfahren handelt es sich beim Algorithmus **WasBinIch**?
- Geben Sie den Aufwand von **WasBinIch** in Abhängigkeit der Feldgröße n für den Worst- und den Best-Case in Θ -Notation an.

Gegeben ist eine endliche Menge $\mathcal{M} = \{a_1, a_2, \dots, a_n\} \subset \mathbb{N}$ sowie eine positive Konstante $K \in \mathbb{N}$. Gesucht ist eine nicht leere Auswahl $\mathcal{A} \subseteq \mathcal{M}$ von Elementen aus \mathcal{M} , sodass die Summe der ausgewählten Elemente möglichst nahe an K heran kommt und die Kardinalität von \mathcal{A} möglichst klein ist.

Analog zur zweiten Programmieraufgabe ist diese Aufgabenstellung mit Hilfe dynamischer Programmierung und einer $(n'+1)\times(z+1)$ Matrix \mathbf{m} zu lösen, wobei der Eintrag im Feld $m_{i,j}$ angibt, wieviele Elemente aus den ersten i Elementen mindestens ausgewählt werden müssen, um in Summe j zu ergeben. Der Wert z gibt dabei die maximal benötigte Spaltenanzahl in der Matrix an. Des weiteren bezeichnet $n' = |\mathcal{M}'|$ die Kardinalität einer Menge $\mathcal{M}' \subseteq \mathcal{M}$ all jener Elemente, für die $a_i \leq z$ gilt. Die Felder der Matrix können wie folgt rekursiv berechnet werden:

$$m_{0,j} = \infty \qquad \text{für } j = 1, \dots, z$$

$$m_{i,0} = 0 \qquad \text{für } i = 0, \dots, n'$$

$$m_{i,j} = \begin{cases} \min\{m_{i-1,j-a_i} + 1, m_{i-1,j}\}, & j \ge a_i \\ m_{i-1,j}, & j < a_i \end{cases} \qquad \text{für } \begin{cases} i = 1, \dots, n' \\ j = 1, \dots, z \\ a_i \in \mathcal{M}' \end{cases}$$

Lösen Sie diese Aufgabenstellung für die konkrete Konstante K=7 und die Werte $\mathcal{M}=\{1,2,3,4,8,10,12,15,16,17\}$. Beantworten Sie dabei folgende Punkte:

- Erläutern Sie, wie z im allgemeinen Fall berechnet werden kann und geben Sie eine gute Wahl für z im konkreten Fall an.
- Geben Sie die Menge $\mathcal{M}' \subseteq \mathcal{M}$ der tatsächlich zu berücksichtigenden Elemente $a_i < z$ an.
- Lösen Sie die Aufgabenstellung für Ihr gewähltes z und \mathcal{M}' und verwenden Sie dazu die zur Verfügung gestellte Matrix (die Initialisierung wurde schon entsprechend vorgenommen). Beachten Sie dabei, dass nicht notwendiger Weise alle Zeilen und Spalten benötigt werden!
- \bullet Geben Sie eine optimale Lösung an. Markieren Sie alle Felder der Matrix, die beim Zurückrechnen der konkreten Auswahl \mathcal{A} von Relevanz sind.

$i \backslash j$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	∞																
1	0																	
2	0																	
3	0																	
4	0																	
5	0																	
6	0																	
7	0																	
8	0																	
9	0																	
10	0																	

a) (12 Punkte)

Für den Datentyp der dynamisch disjunkten Mengen (DDM, Union Find) benötigt man ein Array father, das jedem Knoten v einen Vorgänger father[v] zuweist, sowie die Prozeduren $\mathbf{makeset}(v)$, $\mathbf{findset}(v)$ und $\mathbf{union}(v,w)$. Um die in der Vorlesung behandelte Verbesserung $Vereinigung\ nach\ H\"{o}he\ zu\ realisieren$, braucht man zusätzlich ein Array height, das zu jedem Knoten v die Höhe height[v] speichert.

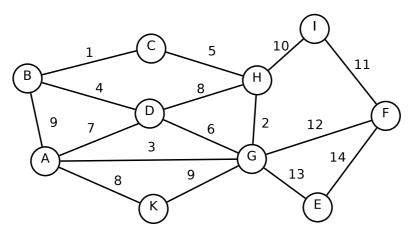
Die Prozeduren $\mathbf{makeset}(v)$ und $\mathbf{findset}(v)$ sind dabei wie folgt implementiert:

Algorithmus 2: $makeset(v)$	Algorithmus 3: $findset(v)$
1: $father[v] = v;$	1: $h = v$;
2: height[v] = 0;	2: solange $father[h] \neq h$ {
	3: $h = father[h];$
	4: }
	5: retourniere h ;

- Schreiben Sie detaillierten Pseudocode für die Prozedur $\mathbf{union}(v, w)$ (v und w jeweils Repräsentant ihrer Menge), der die Verbesserung Vereinigung nach Höhe realisiert.
- Für welchen aus der Vorlesung bekannten MST-Algorithmus ist die DDM-Datenstruktur eine effiziente Möglichkeit, Kreise zu finden?

b) (5 Punkte)

Führen Sie in dem unten abgebildeten Graphen den Algorithmus von Kruskal zum Finden eines minimalen Spannbaums durch (die Zahlen bei den Kanten bezeichnen die jeweiligen Kantenkosten). Zeichnen Sie den Spannbaum direkt im Graphen ein und notieren Sie die genaue Reihenfolge, in der die Kanten in den Baum aufgenommen wurden. Falls Sie einen Startknoten benötigen, verwenden Sie dazu den Knoten A.





Technische Universität Wien Institut für Computergraphik und Algorithmen Arbeitsbereich für Algorithmen und Datenstrukturen



186.172 Algorithmen und Datenstrukturen 1 VL 4.0 3. Übungstest SS 2008 27. Juni 2008

Machen Sie die folgenden Angaber	n bitte in	n deutlicher I	Blockschr	ift:		
Nachname:		Vorname:				
Matrikelnummer:			Studienk	kennzahl:		
		Anzahl ab	gegebene:	r Zusatzblä	itter:	
Legen Sie bitte Ihren Studentenau Sie können die Lösungen entweder schreiben, die Sie auf Wunsch von	direkt a	auf die Anga	beblätter			
mitgebrachtes eigenes Papier zu ver Die Verwendung von Taschenrecht	erwender nern, Mol	n. biltelefonen,	Skripten,			
Ausarbeitungen oder vergleichbare Die Arbeitszeit beträgt 55 Minute		nitteln ist un	zulässig.			
	A1:	A2:	A3:	Summe:		
Erreichbare Punkte:	15	18	17	50		
Erreichte Punkte:						

Viel Glück!

a) (4 Punkte)

Gegeben ist der folgende Algorithmus:

Algorithmus 1: WasMacheIch(A, i)

```
Eingabe: Feld A[1, \ldots n] mit n ganzen Zahlen, Feldindex i
1: falls i = n dann {
2: Abbruch: fertig!
3: }
4: m = i;
5: für j = i + 1, i + 2, \ldots n {
6: falls A[j] > A[m] dann {
7: m = j;
8: }
9: }
10: Vertausche A[i] und A[m];
11: WasMacheIch(A, i + 1);
```

- Um welches konkrete aus der Vorlesung bekannte Verfahren handelt es sich beim Algorithmus **WasMacheIch**?
- Geben Sie den Aufwand von **WasMacheIch** in Abhängigkeit der Feldgröße n für den Worst- und den Best-Case in Θ -Notation an.

b) (6 Punkte)

Seien f(n), g(n) und h(n) Funktionen mit positivem Wertebereich. Sind die folgenden Aussagen wahr oder falsch?

Aussage	wahr	falsch
$f(n) = O(g(n)/n) \Rightarrow g(n) = \Omega(f(n)/n)$		
$g(n) = \Omega(f(n)) \land g(n) = O(h(n)) \Rightarrow h(n) = \Omega(f(n))$		
$(f(n) = O(g(n)) \land f(n) = \Omega(g(n))) \lor f(n) \neq \Theta(g(n))$		

c) (5 Punkte)

Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} n^2 \log_3 n^2, & \text{falls } n \text{ eine 3er-Potenz ist} \\ 2n^3 - 3n & \text{sonst} \end{cases}$$

Beweisen oder widerlegen Sie:

$$f(n) = \Theta(n^3)$$

Beachten Sie, dass für einen korrekten Beweis auch geeignete Werte für die Konstanten n_0 , c_1 und c_2 angegeben werden müssen!

Gegeben ist eine endliche Menge $\mathcal{M} = \{a_1, a_2, \dots, a_n\} \subset \mathbb{N}$ sowie eine positive Konstante $K \in \mathbb{N}$. Gesucht ist eine nicht leere Auswahl $\mathcal{A} \subseteq \mathcal{M}$ von Elementen aus \mathcal{M} , sodass die Summe der ausgewählten Elemente möglichst nahe an K heran kommt und die Kardinalität von \mathcal{A} möglichst klein ist.

Analog zur zweiten Programmieraufgabe ist diese Aufgabenstellung mit Hilfe dynamischer Programmierung und einer $(n'+1)\times(z+1)$ Matrix \mathbf{m} zu lösen, wobei der Eintrag im Feld $m_{i,j}$ angibt, wieviele Elemente aus den ersten i Elementen mindestens ausgewählt werden müssen, um in Summe j zu ergeben. Der Wert z gibt dabei die maximal benötigte Spaltenanzahl in der Matrix an. Des weiteren bezeichnet $n' = |\mathcal{M}'|$ die Kardinalität einer Menge $\mathcal{M}' \subseteq \mathcal{M}$ all jener Elemente, für die $a_i \leq z$ gilt. Die Felder der Matrix können wie folgt rekursiv berechnet werden:

$$m_{0,j} = \infty \qquad \text{für } j = 1, \dots, z$$

$$m_{i,0} = 0 \qquad \text{für } i = 0, \dots, n'$$

$$m_{i,j} = \begin{cases} \min\{m_{i-1,j-a_i} + 1, m_{i-1,j}\}, & j \ge a_i \\ m_{i-1,j}, & j < a_i \end{cases} \qquad \text{für } \begin{cases} i = 1, \dots, n' \\ j = 1, \dots, z \\ a_i \in \mathcal{M}' \end{cases}$$

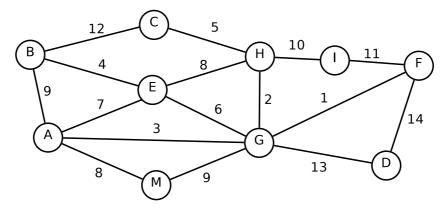
Lösen Sie diese Aufgabenstellung für die konkrete Konstante K=7 und die Werte $\mathcal{M}=\{1,2,3,4,8,10,12,15,16,17\}$. Beantworten Sie dabei folgende Punkte:

- \bullet Erläutern Sie, wie z im allgemeinen Fall berechnet werden kann und geben Sie eine gute Wahl für z im konkreten Fall an.
- Geben Sie die Menge $\mathcal{M}' \subseteq \mathcal{M}$ der tatsächlich zu berücksichtigenden Elemente $a_i < z$ an.
- Lösen Sie die Aufgabenstellung für Ihr gewähltes z und \mathcal{M}' und verwenden Sie dazu die zur Verfügung gestellte Matrix (die Initialisierung wurde schon entsprechend vorgenommen). Beachten Sie dabei, dass nicht notwendiger Weise alle Zeilen und Spalten benötigt werden!
- \bullet Geben Sie eine optimale Lösung an. Markieren Sie alle Felder der Matrix, die beim Zurückrechnen der konkreten Auswahl \mathcal{A} von Relevanz sind.

$i \backslash j$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	∞																
1	0																	
2	0																	
3	0																	
4	0																	
5	0																	
6	0																	
7	0																	
8	0																	
9	0																	
10	0																	

a) (5 Punkte)

Führen Sie in dem unten abgebildeten Graphen den Algorithmus von *Prim* zum Finden eines minimalen Spannbaums durch (die Zahlen bei den Kanten bezeichnen die jeweiligen Kantenkosten). Zeichnen Sie den Spannbaum direkt im Graphen ein und notieren Sie die genaue Reihenfolge, in der die Kanten in den Baum aufgenommen wurden. *Falls* Sie einen Startknoten benötigen, verwenden Sie dazu den Knoten A.



b) (12 Punkte)

Für den Datentyp der dynamisch disjunkten Mengen (DDM, Union Find) benötigt man ein Array father, das jedem Knoten v einen Vorgänger father[v] zuweist, sowie die Prozeduren $\mathbf{makeset}(v)$, $\mathbf{findset}(v)$ und $\mathbf{union}(v, w)$.

Die Prozeduren $\mathbf{makeset}(v)$ und $\mathbf{union}(v, w)$ sind dabei wie folgt implementiert:

Algorithmus 2: makeset(
$$v$$
)

1: $father[v] = v$;

Algorithmus 3: union(v, w)

// u und v jeweils

// Repräsentant ihrer Menge

1: $father[v] = w$;

- Schreiben Sie detaillierten Pseudocode für die Prozedur **findset**(v), der die *Methode der Pfadverkürzung* realisiert.
- Für welchen aus der Vorlesung bekannten MST-Algorithmus ist die DDM-Datenstruktur eine effiziente Möglichkeit, Kreise zu finden?