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

# 186.813 Algorithmen und Datenstrukturen 1 VU 6.0

## Nachtragstest SS 2013

## 10. Oktober 2013

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: | Vorname:

Matrikelnummer: | Unterschrift:

Anzahl abgegebener Zusatzblätter:

Legen Sie während der Prüfung Ihren Ausweis für Studierende vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie bitte dokumentenechte Schreibgeräte (keine Bleistifte!).

Die Verwendung von Taschenrechnern, Mobiltelefonen, PDAs, Digitalkameras, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

| | A1: | A2: | A3: | Summe: |
|---|---|---|---|---|
| Erreichbare Punkte: | 16 | 16 | 18 | 50 |
| Erreichte Punkte: | | | | |

Viel Erfolg!

**Aufgabe 1.A: Notationen** **(16 Punkte)**

a) (8 Punkte)

Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} n \cdot \dfrac{n^3}{\log_2(2^n)}, & \text{wenn } n > 700 \text{ und nicht prim} \\ n \cdot \log_2(n) + \sqrt{n^6}, & \text{wenn } n > 500 \text{ und prim} \\ n^2 \cdot \log_2(\sqrt{n}), & \text{sonst} \end{cases}$$

Kreuzen Sie **in der folgenden Tabelle** die zutreffenden Felder an:

| $f(n)$ ist | $O(.)$ | $\Omega(.)$ | $\Theta(.)$ | keines |
|---|---|---|---|---|
| $n^3\sqrt{n}$ | | | | |
| $n^3$ | | | | |
| $n \log n$ | | | | |
| $n^2 \log n$ | | | | |

Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

b) (8 Punkte)

Tragen Sie für die Codestücke A und B jeweils die Laufzeit in Abhängigkeit von $n$ und die Werte der Variablen $a$ und $b$ nach dem Ausführen des Codes in $\Theta$-Notation **in die unten stehenden Tabellen** ein.

A

```
a = 1;
b = 0;
i = n;
wiederhole
  i = i/2;
  a = a · 2;
  für j = 1, ..., n {
    b = b + n;
  }
bis i ≤ 1;
```

| Laufzeit | $a$ | $b$ |
|---|---|---|
| $\Theta($ $)$ | $\Theta($ $)$ | $\Theta($ $)$ |

B

```
a = n;
b = 1;
i = 1;
solange i < n {
  a = a − 1;
  b = b · 2;
  i = i + 1;
}
für i = 1 ... b {
  a = a · 2;
}
```

| Laufzeit | $a$ | $b$ |
|---|---|---|
| $\Theta($ $)$ | $\Theta($ $)$ | $\Theta($ $)$ |

**Aufgabe 2.A: Optimierung** **(16 Punkte)**

Gegeben sei eine Instanz des 0/1-Rucksackproblems mit Kapazität $K = 6$ und folgenden Gegenständen:

| Gegenstand $i$ | Gewicht $w_i$ | Wert $c_i$ |
|---|---|---|
| 1 | 3 | 2 |
| 2 | 1 | 1 |
| 3 | 1 | 2 |
| 4 | 4 | 4 |
| 5 | 2 | 1 |
| 6 | 2 | 2 |
| 7 | 3 | 4 |

| $i\backslash j$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |

Diese Instanz soll mittels *Dynamischer Programmierung* über die möglichen Gesamtgewichte gelöst werden. Dazu wird die oben stehende $8 \times 7$-Matrix $\mathbf{m}$ verwendet, wobei der Eintrag im Feld $m_{i,j}$ angibt, welcher Wert mit den ersten $i$ Gegenständen erreicht werden kann, wenn das Gesamtgewicht der gewählten Gegenstände kleiner oder gleich $j$ ist.

Die Felder der Matrix können wie folgt berechnet werden:

$$m_{0,j} = 0 \qquad \text{für } j = 0, \ldots, 6$$

$$m_{i,j} = \begin{cases} m_{i-1,j} & \text{falls } w_i > j, \\ \max\{m_{i-1,j-w_i} + c_i,\ m_{i-1,j}\} & \text{sonst.} \end{cases} \qquad \text{für } \begin{cases} i = 1, \ldots, 7 \\ j = 0, \ldots, 6 \end{cases}$$

- (8 Punkte) Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.
- (3 Punkte) Geben Sie die optimale Lösung an und markieren Sie in der Matrix jene Zellen, die beim Rekonstruieren der Lösung betrachtet werden.
- (3 Punkte) Ist es im Allgemeinen möglich, dass mehrere optimale Lösungen existieren, die denselben Gesamtwert und dasselbe Gesamtgewicht haben? Begründen Sie Ihre Antwort.
- (2 Punkte) Geben Sie die Laufzeit für das Befüllen der Matrix im Worst- und im Best-Case in $\Theta$-Notation, in Abhängigkeit der Anzahl der Elemente $n$ und der Kapazität $K$, an.

**Aufgabe 3.A: Graphen** **(18 Punkte)**

Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph mit Knotenmenge $V$ und Kantenmenge $E$. Eine Kante $e \in E$ heißt *Brücke*, wenn durch das Entfernen von $e$ der Graph $G$ in zwei Komponenten zerfällt.

a) (14 Punkte) Schreiben Sie in detailliertem Pseudocode eine möglichst effiziente Funktion `checkBridge(G)`, die alle Brücken in einem ungerichteten, zusammenhängenden Graph $G = (V, E)$ findet.

b) (4 Punkte) Geben Sie die Laufzeit Ihres Algorithmus in $\Theta$-Notation an.

Beachten Sie folgende Punkte:

- Ihre Funktion `checkBridge(G)` soll eine Menge $B$ zurückliefern, die alle Brücken von $G$ enthält. Gibt es in $G$ keine Brücken, dann gilt $B = \emptyset$.
- Sie können mit `addEdge(a,b)` der Kantenmenge $E$ eine Kante $(a, b) \in V \times V$ hinzufügen. (Wenn die Kante im Graphen bereits existiert, hat das keinen weiteren Einfluss auf den Graphen oder Ihren Algorithmus.)
- Sie können mit `removeEdge(a,b)` eine Kante $(a, b) \in V \times V$ aus der Kantenmenge $E$ löschen. (Wenn die Kante im Graphen nicht existiert, hat das keinen weiteren Einfluss auf den Graphen oder Ihren Algorithmus.)

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

# 186.813 Algorithmen und Datenstrukturen 1 VU 6.0

## Nachtragstest SS 2013

## 10. Oktober 2013

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: | Vorname:

Matrikelnummer: | Unterschrift:

Anzahl abgegebener Zusatzblätter:

Legen Sie während der Prüfung Ihren Ausweis für Studierende vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie bitte dokumentenechte Schreibgeräte (keine Bleistifte!).

Die Verwendung von Taschenrechnern, Mobiltelefonen, PDAs, Digitalkameras, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

| | B1: | B2: | B3: | Summe: |
|---|---|---|---|---|
| Erreichbare Punkte: | 16 | 16 | 18 | 50 |
| Erreichte Punkte: | | | | |

Viel Glück!

**Aufgabe 1.B: Notationen** **(16 Punkte)**

a) (8 Punkte)

Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} n^2 \cdot \dfrac{n}{\log_2(2^n)}, & \text{wenn } n > 300 \text{ und prim} \\ n \cdot \log_2(n^2) + \sqrt{n^4}, & \text{wenn } n > 200 \text{ und nicht prim} \\ n^3 \cdot \log_2(n), & \text{sonst} \end{cases}$$

Kreuzen Sie **in der folgenden Tabelle** die zutreffenden Felder an:

| $f(n)$ ist | $O(.)$ | $\Omega(.)$ | $\Theta(.)$ | keines |
|---|---|---|---|---|
| $n^2 \log n$ | | | | |
| $n^2$ | | | | |
| $n \log n$ | | | | |
| $n^3 \log n$ | | | | |

Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

b) (8 Punkte)

Tragen Sie für die Codestücke A und B jeweils die Laufzeit in Abhängigkeit von $n$ und die Werte der Variablen $a$ und $b$ nach dem Ausführen des Codes in $\Theta$-Notation **in die unten stehenden Tabellen** ein.

A

```
a = 1;
b = n;
i = 1;
solange i < n {
  a = a · 2;
  b = b − 1;
  i = i + 1;
}
für i = 1 ... a {
  b = b · 2;
}
```

| Laufzeit | $a$ | $b$ |
|---|---|---|
| $\Theta($ $)$ | $\Theta($ $)$ | $\Theta($ $)$ |

B

```
a = 1;
b = 0;
i = 1;
wiederhole
  i = i · 2;
  a = a · 2;
  für j = 1, ..., n² {
    b = b + n;
  }
bis i ≥ n;
```

| Laufzeit | $a$ | $b$ |
|---|---|---|
| $\Theta($ $)$ | $\Theta($ $)$ | $\Theta($ $)$ |

**Aufgabe 2.B: Optimierung** **(16 Punkte)**

Gegeben sei eine Instanz des 0/1-Rucksackproblems mit Kapazität $K = 7$ und folgenden Gegenständen:

| Gegenstand $i$ | Gewicht $w_i$ | Wert $c_i$ |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 1 | 3 |
| 3 | 3 | 2 |
| 4 | 4 | 4 |
| 5 | 2 | 1 |
| 6 | 3 | 4 |

| $i\backslash j$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |

Diese Instanz soll mittels *Dynamischer Programmierung* über die möglichen Gesamtgewichte gelöst werden. Dazu wird die oben stehende $7 \times 8$-Matrix $\mathbf{m}$ verwendet, wobei der Eintrag im Feld $m_{i,j}$ angibt, welcher Wert mit den ersten $i$ Gegenständen erreicht werden kann, wenn das Gesamtgewicht der gewählten Gegenstände kleiner oder gleich $j$ ist.

Die Felder der Matrix können wie folgt berechnet werden:

$$m_{0,j} = 0 \qquad \text{für } j = 0, \ldots, 6$$

$$m_{i,j} = \begin{cases} m_{i-1,j} & \text{falls } w_i > j, \\ \max\{m_{i-1,j-w_i} + c_i,\, m_{i-1,j}\} & \text{sonst.} \end{cases} \qquad \text{für } \begin{cases} i = 1, \ldots, 7 \\ j = 0, \ldots, 6 \end{cases}$$

- (8 Punkte) Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.
- (3 Punkte) Geben Sie die optimale Lösung an und markieren Sie in der Matrix jene Zellen, die beim Rekonstruieren der Lösung betrachtet werden.
- (3 Punkte) Ist es im Allgemeinen möglich, dass mehrere optimale Lösungen existieren, die denselben Gesamtwert und dasselbe Gesamtgewicht haben? Begründen Sie Ihre Antwort.
- (2 Punkte) Geben Sie die Laufzeit für das Befüllen der Matrix im Worst- und im Best-Case in $\Theta$-Notation, in Abhängigkeit der Anzahl der Elemente $n$ und der Kapazität $K$, an.

**Aufgabe 3.B: Graphen** **(18 Punkte)**

Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph mit Knotenmenge $V$ und Kantenmenge $E$. Eine Kante $e \in E$ heißt *Brücke*, wenn durch das Entfernen von $e$ der Graph $G$ in zwei Komponenten zerfällt.

a) (14 Punkte) Schreiben Sie in detailliertem Pseudocode eine möglichst effiziente Funktion `checkBridge(G)`, die alle Brücken in einem ungerichteten, zusammenhängenden Graph $G = (V, E)$ findet.

b) (4 Punkte) Geben Sie die Laufzeit Ihres Algorithmus in $\Theta$-Notation an.

Beachten Sie folgende Punkte:

- Ihre Funktion `checkBridge(G)` soll eine Menge $B$ zurückliefern, die alle Brücken von $G$ enthält. Gibt es in $G$ keine Brücken, dann gilt $B = \emptyset$.
- Sie können mit `addEdge(a,b)` der Kantenmenge $E$ eine Kante $(a, b) \in V \times V$ hinzufügen. (Wenn die Kante im Graphen bereits existiert, hat das keinen weiteren Einfluss auf den Graphen oder Ihren Algorithmus.)
- Sie können mit `removeEdge(a,b)` eine Kante $(a, b) \in V \times V$ aus der Kantenmenge $E$ löschen. (Wenn die Kante im Graphen nicht existiert, hat das keinen weiteren Einfluss auf den Graphen oder Ihren Algorithmus.)