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

# 186.172 Algorithmen und Datenstrukturen 1 VL 4.0

## Nachtragstest WS 2009

## 28. Jänner 2009

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: | Vorname:

Matrikelnummer: | Studienkennzahl:

Anzahl abgegebener Zusatzblätter:

Legen Sie bitte Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden.

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

Die Arbeitszeit beträgt 55 Minuten.

| | A1: | A2: | A3: | Summe: |
|---|---|---|---|---|
| Erreichbare Punkte: | 18 | 14 | 18 | 50 |
| Erreichte Punkte: | | | | |

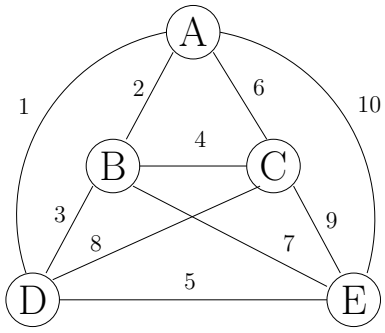
Viel Erfolg!

**Aufgabe 1.A: Optimierung** **(18 Punkte)**

Gegeben sei folgender Algorithmus *WasBinIch*, der auf einen vollständigen, ungerichteten, gewichteten Graphen $G(V, E)$ angewendet wird.

**Algorithmus** WasBinIch($G(V, E)$)

```
1:  Sortiere die Kanten in E aufsteigend nach Gewicht;
    // Hinweis: Sortierung erfolgt durch Merge-Sort
2:  T = ∅;
3:  d[v] = 0   ∀v ∈ V;
4:  solange |T| < |V| − 1 {
5:      (v, w) = Kante aus E mit dem kleinsten Gewicht;
6:      E = E \ {(v, w)};
7:      falls d[v] < 2 ∧ d[w] < 2 dann {
8:          falls T ∪ {(v, w)} enthält keinen Kreis dann {
9:              T = T ∪ {(v, w)};
10:             d[v] = d[v] + 1;
11:             d[w] = d[w] + 1;
12:         }
13:     }
14: }
15: Finde die Knoten v, w mit v ≠ w ∧ d[v] = d[w] = 1;
16: retourniere T ∪ {(v, w)};
```



Graph $G$

a) (8 Punkte)

Wenden Sie den obigen Algorithmus *WasBinIch* auf den gegebenen Graphen $G$ an. Geben Sie jeweils die Kanten in der Reihenfolge an, wie sie zu $T$ hinzugefügt werden.

Tragen Sie den Zustand des Feldes $d$ nach der Ausführung des Algorithmus in diese Tabelle ein:

| $v \in V$ | A | B | C | D | E |
|---|---|---|---|---|---|
| $d[v]$ | | | | | |

b) (4 Punkte)

Was berechnet der Algorithmus *WasBinIch*? Kann *WasBinIch* für jeden beliebigen Graphen ein solches Ergebniss berechnen? Falls nicht, geben Sie bitte alle oben genannten Eigenschaften des Graphen an, die Voraussetzung für das korrekte Arbeiten von *WasBinIch* sind?

c) (6 Punkte)

Geben Sie die Gesamtlaufzeit des Algorithmus im Worst-Case in $\Theta$-Notation, in Abhängkeit der Knotenanzahl $n$ an. Begründen Sie ihre Antwort indem Sie kurz beschreiben wodurch sich diese Laufzeit ergibt.

**Aufgabe 2.A: Sortieren und Optimierung** **(14 Punkte)**

a) (8 Punkte)

Sortieren Sie die folgenden Zahlen **aufsteigend** mittels eines aus der Vorlesung bekannten Sortierverfahrens, das in Abhängigkeit der Anzahl der zu sortierenden Elemente $n$ im Worst Case $\Theta(n)$ Schlüsselbewegungen und $\Theta(n^2)$ Schlüsselvergleiche benötigt.

$$\langle 4, 1, 2, 3, 7, 8, 6, 5 \rangle$$

- Geben Sie die komplette Zahlenfolge an, jeweils nachdem ein weiterer Schlüssel seine endgültige Position erreicht hat und markieren Sie in jeder dieser Folgen die Zahlen, die sich bereits an ihrer endgültigen Position befinden.
- Um welches Sortierverfahren handelt es sich?
- Ist dieses Sortierverfahren stabil? Begründen Sie ihre Antwort.

| 4 | 1 | 2 | 3 | 7 | 8 | 6 | 5 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

Hinweis: Nicht jede Zeile der Tabelle muss zwangsläufig für eine korrekte Lösung verwendet werden.

b) (6 Punkte)

Beweisen oder widerlegen Sie, dass für die folgende Funktion $f(n)$ die Beziehung $f(n) = O(n^3)$ gilt:

$$f(n) = \begin{cases} 2n^3 - n \log n + \frac{1}{3}n^3, & \text{falls } n > 50 \\ 4^n + n^2 - \frac{1}{n^4}, & \text{falls } n \leq 50 \end{cases}$$

Bedenken Sie, dass für einen Beweis die formale Definition und gegebenenfalls auch geeignete Werte für die Konstante $c$ und $n_0$ angegeben werden müssen. Verwenden Sie dafür ganzzahlige Werte $\leq 100$.

**Aufgabe 3.A: ADTS und Suchen** **(18 Punkte)**

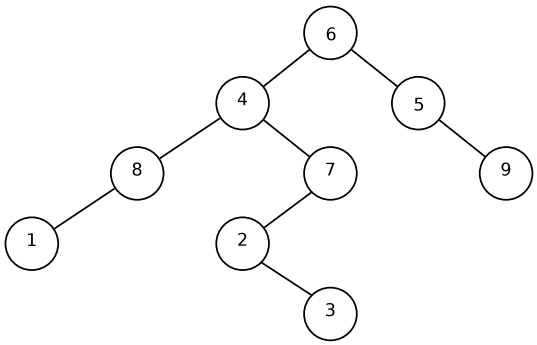
a) (10 Punkte)

- Schreiben Sie detaillierten Pseudocode für die Funktion $blockMove(L, s, x, z)$. Dabei ist der Parameter $L$ ein Zeiger auf das erste Element einer azyklischen, einfach verketteten Liste. Beginnend bei Position $x$ sollen $s$ Elemente der Liste (inklusive des Elementes bei Position $x$) um $z$ Positionen nach rechts verschoben werden.

  *Hinweis:* Sie können davon ausgehen, dass $s$, $x$ und $z$ immer Werte annehmen, so dass diese Operation auch tatsächlich möglich ist (es kann also nicht passieren, dass es ab Position $x$ keine $s$ Elemente mehr gibt, oder dass man den Block nicht um $z$ Stellen verschieben kann, weil die Liste zu kurz ist).

b) (8 Punkte)

- Geben Sie zu folgendem binären Baum die Pre- und Postorder Durchmusterung an.




- Kann ein beliebiger binärer Baum durch gegebene Pre- und Postorder Durchmusterung eindeutig rekonstruiert werden? Begründen Sie ihre Antwort.

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

# 186.172 Algorithmen und Datenstrukturen 1 VL 4.0

## Nachtragstest WS 2009

## 28. Jänner 2009

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: | Vorname:

Matrikelnummer: | Studienkennzahl:

Anzahl abgegebener Zusatzblätter:

Legen Sie bitte Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden.

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

Die Arbeitszeit beträgt 55 Minuten.

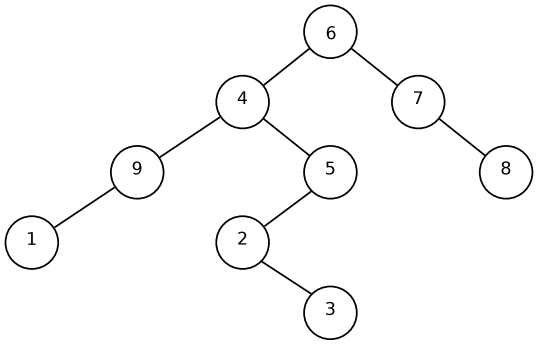
| | B1: | B2: | B3: | Summe: |
|---|---|---|---|---|
| Erreichbare Punkte: | 18 | 18 | 14 | 50 |
| Erreichte Punkte: | | | | |

Viel Glück!

**Aufgabe 1.B: ADTS und Suchen** **(18 Punkte)**

a) (8 Punkte)

- Geben Sie zu folgendem binären Baum die Pre- und Postorder Durchmusterung an.




- Kann ein beliebiger binärer Baum durch gegebene Pre- und Postorder Durchmusterung eindeutig rekonstruiert werden? Begründen Sie ihre Antwort.

b) (10 Punkte)

- Schreiben Sie detaillierten Pseudocode für die Funktion $blockMove(L, s, x, z)$. Dabei ist der Parameter $L$ ein Zeiger auf das erste Element einer azyklischen, einfach verketteten Liste. Beginnend bei Position $x$ sollen $s$ Elemente der Liste (inklusive des Elementes bei Position $x$) um $z$ Positionen nach rechts verschoben werden.

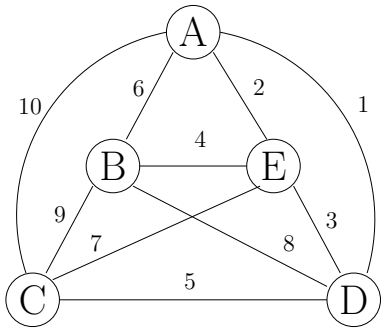
  *Hinweis:* Sie können davon ausgehen, dass $s$, $x$ und $z$ immer Werte annehmen, so dass diese Operation auch tatsächlich möglich ist (es kann also nicht passieren, dass es ab Position $x$ keine $s$ Elemente mehr gibt, oder dass man den Block nicht um $z$ Stellen verschieben kann, weil die Liste zu kurz ist).

**Aufgabe 2.B: Optimierung** **(18 Punkte)**

Gegeben sei folgender Algorithmus *WasBinIch*, der auf einen vollständigen, ungerichteten, gewichteten Graphen $G(V, E)$ angewendet wird.

**Algorithmus** WasBinIch($G(V, E)$)

```
1:  Sortiere die Kanten in E aufsteigend nach Gewicht;
    // Hinweis: Sortierung erfolgt durch Merge-Sort
2:  T = ∅;
3:  d[v] = 0   ∀v ∈ V;
4:  solange |T| < |V| − 1 {
5:      (v, w) = Kante aus E mit dem kleinsten Gewicht;
6:      E = E \ {(v, w)};
7:      falls d[v] < 2 ∧ d[w] < 2 dann {
8:          falls T ∪ {(v, w)} enthält keinen Kreis dann {
9:              T = T ∪ {(v, w)};
10:             d[v] = d[v] + 1;
11:             d[w] = d[w] + 1;
12:         }
13:     }
14: }
15: Finde die Knoten v, w mit v ≠ w ∧ d[v] = d[w] = 1;
16: retourniere T ∪ {(v, w)};
```



Graph $G$

a) (8 Punkte)

Wenden Sie den obigen Algorithmus *WasBinIch* auf den gegebenen Graphen $G$ an. Geben Sie jeweils die Kanten in der Reihenfolge an, wie sie zu $T$ hinzugefügt werden.

Tragen Sie den Zustand des Feldes $d$ nach der Ausführung des Algorithmus in diese Tabelle ein:

| $v \in V$ | A | B | C | D | E |
|---|---|---|---|---|---|
| $d[v]$ | | | | | |

b) (4 Punkte)

Was berechnet der Algorithmus *WasBinIch*? Kann *WasBinIch* für jeden beliebigen Graphen ein solches Ergebniss berechnen? Falls nicht, geben Sie bitte alle oben genannten Eigenschaften des Graphen an, die Voraussetzung für das korrekte Arbeiten von *WasBinIch* sind?

c) (6 Punkte)

Geben Sie die Gesamtlaufzeit des Algorithmus im Worst-Case in $\Theta$-Notation, in Abhängkeit der Knotenanzahl $n$ an. Begründen Sie ihre Antwort indem Sie kurz beschreiben wodurch sich diese Laufzeit ergibt.

**Aufgabe 3.B: Sortieren und Optimierung** **(14 Punkte)**

a) (6 Punkte)

Beweisen oder widerlegen Sie, dass für die folgende Funktion $f(n)$ die Beziehung $f(n) = O(n^2)$ gilt:

$$f(n) = \begin{cases} 3^n + n^4 - \frac{1}{n^3}, & \text{falls } n \leq 50 \\ 3n^2 - n \log n + \frac{1}{2}n^2, & \text{falls } n > 50 \end{cases}$$

Bedenken Sie, dass für einen Beweis die formale Definition und gegebenenfalls auch geeignete Werte für die Konstante $c$ und $n_0$ angegeben werden müssen. Verwenden Sie dafür ganzzahlige Werte $\leq 100$.

b) (8 Punkte)

Sortieren Sie die folgenden Zahlen **aufsteigend** mittels eines aus der Vorlesung bekannten Sortierverfahrens, das in Abhängigkeit der Anzahl der zu sortierenden Elemente $n$ im Worst Case $\Theta(n)$ Schlüsselbewegungen und $\Theta(n^2)$ Schlüsselvergleiche benötigt.

$$\langle 5, 2, 3, 4, 8, 9, 7, 6 \rangle$$

- Geben Sie die komplette Zahlenfolge an, jeweils nachdem ein weiterer Schlüssel seine endgültige Position erreicht hat und markieren Sie in jeder dieser Folgen die Zahlen, die sich bereits an ihrer endgültigen Position befinden.
- Um welches Sortierverfahren handelt es sich?
- Ist dieses Sortierverfahren stabil? Begründen Sie ihre Antwort.

| 5 | 2 | 3 | 4 | 8 | 9 | 7 | 6 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

Hinweis: Nicht jede Zeile der Tabelle muss zwangsläufig für eine korrekte Lösung verwendet werden.