



186.172 Algorithmen und Datenstrukturen 1 VL 4.0

2. Übungstest WS 2008

16. 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:	16	19	15	50
Erreichte Punkte:	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Viel Erfolg!

Aufgabe 1.A: Hashverfahren

(16 Punkte)

a) (8 Punkte)

Gegeben ist jeweils eine Hashtabelle mit Tabellengröße $m = 7$, die Double Hashing mit der Verbesserung nach Brent benutzt mit

$$h_1(k) = k \bmod 11$$

$$h_2(k) = 2 + (k \bmod 5).$$

- Fügen Sie den Wert 7 in die folgende Tabelle ein:

0	1	2	3	4	5	6
22				4		

- Fügen Sie den Wert 14 in die folgende Tabelle ein:

0	1	2	3	4	5	6
		2	10	15		

- Sind die gegebene Tabellengröße sowie die beiden Hashfunktionen eine gute Wahl oder können dabei Probleme auftreten? Begründen Sie Ihre Antwort kurz.

b) (8 Punkte)

Betrachten Sie die folgende Hashtabelle der Größe 5:

Index	0	1	2	3	4
Element	4		7	3	12
Status	W	F	B	B	B

Die Hashfunktion, die zum Einfügen der Elemente verwendet wurde, lautet

$$h(k) = k \bmod 5$$

und als Kollisionsbehandlung wurde lineares Sondieren mit der Schrittweite 1 verwendet. In der Zeile *Status* steht F für *frei*, W für *wieder frei* und B für *besetzt*. In die leere Hashtabelle wurden zu Beginn die Elemente mit den Schlüsseln 3, 7 und 12 eingefügt (*nicht notwendigerweise in dieser Reihenfolge*) und in weiterer Folge nicht wieder entfernt. Über weitere Einfüge- und Löschoperationen anderer Schlüssel danach gibt es keine genauen Angaben.

Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind:

Wahr	Falsch	
<input type="checkbox"/>	<input type="checkbox"/>	Es wurde nie ein Schlüssel gelöscht.
<input type="checkbox"/>	<input type="checkbox"/>	Es ist möglich, dass 1 eingefügt und wieder gelöscht wurde.
<input type="checkbox"/>	<input type="checkbox"/>	7 wurde definitiv vor 12 eingefügt.
<input type="checkbox"/>	<input type="checkbox"/>	Es ist möglich, dass 12 vor 3 eingefügt wurde.
<input type="checkbox"/>	<input type="checkbox"/>	Ein Element mit Schlüssel 5 würde aktuell in der Hashtabelle an Index 1 eingefügt werden.

Aufgabe 2.A: Suchverfahren

(19 Punkte)

a) (10 Punkte)

- Schreiben Sie detaillierten Pseudocode für eine Funktion, die die Rechts-Rotation für einen Teilbaum mit Wurzel u (wird als Parameter übergeben) innerhalb eines AVL-Baumes durchführt. Als Rückgabewert soll ihre Funktion den neuen Wurzelknoten des rebalancierten Teilbaumes zurückliefern. Zu jedem Knoten x werden folgende Daten gespeichert:

$x.key$: Schlüssel von x
 $x.parent$: Verweis auf Vaterknoten von x
 $x.left$: Verweis auf linkes Kind von x
 $x.right$: Verweis auf rechtes Kind von x
 $x.height$: Höhe des Teilbaumes mit Wurzel x

Hinweis: Beachten Sie, dass Ihre Rebalancierungsfunktion so ausgelegt ist, dass **alle** oben angeführten Daten eines Knotens nach der Rebalancierung korrekt sind!

- Geben Sie die Laufzeit ihres Algorithmus in Θ -Notation in Abhängigkeit einer geeigneten Größe an.

b) (9 Punkte)

- Fügen Sie folgende Werte in genau dieser Reihenfolge in einen Anfangs leeren B-Baum der Ordnung 3 ein. Zeichnen Sie lediglich den resultierenden B-Baum, Zwischenschritte müssen nicht unbedingt angegeben werden.

$\langle 42, 1, 19, 10, 11, 7, 32, 3, 13 \rangle$

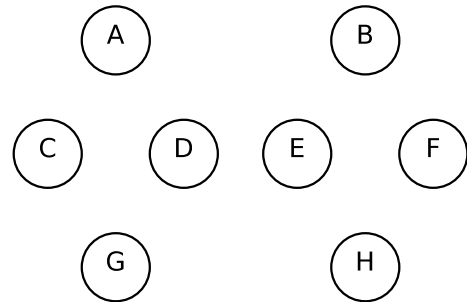
- Löschen Sie nun die Schlüssel 32 und 19 in genau dieser Reihenfolge aus dem B-Baum und zeichnen Sie jeweils den resultierenden B-Baum, Zwischenschritte der Löschoptionen müssen nicht angegeben werden.
- Geben Sie die Laufzeit in Θ -Notation in Abhängigkeit der Anzahl der gespeicherten Elemente n für das Suchen eines Schlüssels k in einem B-Baum der Ordnung 5 an.

Aufgabe 3.A: Graphen

(15 Punkte)

Gegeben ist die folgende Adjazenzmatrix eines ungerichteten Graphen G :

	A	B	C	D	E	F	G	H
A	∞	∞	1	3	6	∞	12	∞
B	∞	∞	∞	∞	9	11	∞	∞
C	1	∞	∞	∞	∞	∞	4	∞
D	3	∞	∞	∞	5	∞	2	∞
E	6	9	∞	5	∞	∞	7	13
F	∞	11	∞	∞	∞	∞	∞	8
G	12	∞	4	2	7	∞	∞	10
H	∞	∞	∞	∞	13	8	10	∞



- a) (2 Punkte) Zeichnen Sie den Graphen G in die gegebene Vorlage ein und geben Sie die Adjazenzliste des Knotens E an.
Die Zahlen in der Matrix bezeichnen die Kantenkosten. Ein ∞ bedeutet, dass keine Kante zwischen den entsprechenden Knoten existiert.
- b) (7 Punkte) Führen Sie in dem Graphen G den Algorithmus von *Kruskal* zum Finden eines minimalen Spannbaums durch. Notieren Sie die genaue Reihenfolge, in der die Kanten in den Baum aufgenommen wurden (die Liste der Kanten ist ausreichend, der MST muss nicht gezeichnet werden). *Falls* Sie einen Startknoten benötigen, verwenden Sie dazu den Knoten A.
- c) (3 Punkte) Ist der resultierende minimale Spannbaum aus Punkt (b) eindeutig? Liefern die Algorithmen von Kruskal und Prim **im Allgemeinen** immer den selben minimalen Spannbaum (bezogen auf Gesamtkosten und Kanten)? Begründen Sie Ihre Antwort kurz!
- d) (3 Punkte) Nehmen Sie an, Sie programmieren einen Algorithmus für ungerichtete Graphen, bei denen kein Knoten mehr als 5 Nachbarn besitzt. Welche Darstellungsform für den Graphen ist aus Effizienzgründen zu wählen: Adjazenzmatrix oder -liste?
Begründen Sie Ihre Antwort, indem Sie **für beide Datenstrukturen** eine Abschätzung des Aufwands zum Finden aller Nachbarn eines Knotens $v \in V$ in Θ -Notation angeben, jeweils in Abhängigkeit einer dafür geeigneten Größe.



186.172 Algorithmen und Datenstrukturen 1 VL 4.0

2. Übungstest WS 2008

16. 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:	15	16	19	50
Erreichte Punkte:	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

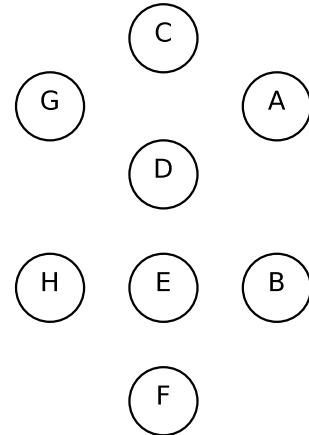
Viel Glück!

Aufgabe 1.B: Graphen

(15 Punkte)

Gegeben ist die folgende Adjazenzmatrix eines ungerichteten Graphen G :

	A	B	C	D	E	F	G	H
A	∞	∞	11	3	6	∞	12	∞
B	∞	∞	∞	∞	13	10	∞	∞
C	11	∞	∞	∞	∞	∞	4	∞
D	3	∞	∞	∞	5	∞	2	∞
E	6	13	∞	5	∞	∞	7	9
F	∞	10	∞	∞	∞	∞	∞	8
G	12	∞	4	2	7	∞	∞	1
H	∞	∞	∞	∞	9	8	1	∞



- a) (2 Punkte) Zeichnen Sie den Graphen G in die gegebene Vorlage ein und geben Sie die Adjazenzliste des Knotens A an.
Die Zahlen in der Matrix bezeichnen die Kantenkosten. Ein ∞ bedeutet, dass keine Kante zwischen den entsprechenden Knoten existiert.
- b) (7 Punkte) Führen Sie in dem Graphen G den Algorithmus von *Kruskal* zum Finden eines minimalen Spannbaums durch. Notieren Sie die genaue Reihenfolge, in der die Kanten in den Baum aufgenommen wurden (die Liste der Kanten ist ausreichend, der MST muss nicht gezeichnet werden). *Falls* Sie einen Startknoten benötigen, verwenden Sie dazu den Knoten E.
- c) (3 Punkte) Ist der resultierende minimale Spannbaum aus Punkt (b) eindeutig? Liefern die Algorithmen von Kruskal und Prim **im Allgemeinen** immer den selben minimalen Spannbaum (bezogen auf Gesamtkosten und Kanten)? Begründen Sie Ihre Antwort kurz!
- d) (3 Punkte) Nehmen Sie an, Sie programmieren einen Algorithmus für ungerichtete Graphen, bei denen kein Knoten mehr als 4 Nachbarn besitzt. Welche Darstellungsform für den Graphen ist aus Effizienzgründen zu wählen: Adjazenzmatrix oder -liste?
Begründen Sie Ihre Antwort, indem Sie **für beide Datenstrukturen** eine Abschätzung des Aufwands zum Finden aller Nachbarn eines Knotens $v \in V$ in Θ -Notation angeben, jeweils in Abhängigkeit einer dafür geeigneten Größe.

Aufgabe 2.B: Hashverfahren

(16 Punkte)

a) (8 Punkte)

Gegeben ist jeweils eine Hashtabelle mit Tabellengröße $m = 7$, die Double Hashing mit der Verbesserung nach Brent benutzt mit

$$h_1(k) = k \bmod 13$$

$$h_2(k) = (k \bmod 5) + 1.$$

- Fügen Sie den Wert 15 in die folgende Tabelle ein:

0	1	2	3	4	5	6
		9	16			

- Fügen Sie den Wert 7 in die folgende Tabelle ein:

0	1	2	3	4	5	6
20	8		3			

- Sind die gegebene Tabellengröße sowie die beiden Hashfunktionen eine gute Wahl oder können dabei Probleme auftreten? Begründen Sie Ihre Antwort kurz.

b) (8 Punkte)

Betrachten Sie die folgende Hashtabelle der Größe 5:

Index	0	1	2	3	4
Element		6	2	11	3
Status	F	B	B	B	W

Die Hashfunktion, die zum Einfügen der Elemente verwendet wurde, lautet

$$h(k) = k \bmod 5$$

und als Kollisionsbehandlung wurde lineares Sondieren mit der Schrittweite 1 verwendet. In der Zeile *Status* steht F für *frei*, W für *wieder frei* und B für *besetzt*. In die leere Hashtabelle wurden zu Beginn die Elemente mit den Schlüsseln 2, 6 und 11 eingefügt (*nicht notwendigerweise in dieser Reihenfolge*) und in weiterer Folge nicht wieder entfernt. Über weitere Einfüge- und Löschooperationen anderer Schlüssel danach gibt es keine genauen Angaben.

Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind:

Wahr	Falsch	
<input type="checkbox"/>	<input type="checkbox"/>	6 wurde definitiv vor 11 eingefügt.
<input type="checkbox"/>	<input type="checkbox"/>	Es ist möglich, dass 11 vor 2 eingefügt wurde.
<input type="checkbox"/>	<input type="checkbox"/>	Es ist möglich, dass 0 eingefügt und wieder gelöscht wurde.
<input type="checkbox"/>	<input type="checkbox"/>	Es wurde nie ein Schlüssel gelöscht.
<input type="checkbox"/>	<input type="checkbox"/>	Ein Element mit Schlüssel 4 würde aktuell in der Hashtabelle an Index 0 eingefügt werden.

Aufgabe 3.B: Suchverfahren

(19 Punkte)

a) (9 Punkte)

- Fügen Sie folgende Werte in genau dieser Reihenfolge in einen Anfangs leeren B-Baum der Ordnung 3 ein. Zeichnen Sie lediglich den resultierenden B-Baum, Zwischenschritte müssen nicht unbedingt angegeben werden.

$\langle 10, 11, 1, 19, 42, 8, 34, 5, 15 \rangle$

- Löschen Sie nun die Schlüssel 34 und 19 in genau dieser Reihenfolge aus dem B-Baum und zeichnen Sie jeweils den resultierenden B-Baum, Zwischenschritte der Löschoptionen müssen nicht angegeben werden.
- Geben Sie die Laufzeit in Θ -Notation in Abhängigkeit der Anzahl der gespeicherten Elemente n für das Suchen eines Schlüssels k in einem B-Baum der Ordnung 4 an.

b) (10 Punkte)

- Schreiben Sie detaillierten Pseudocode für eine Funktion, die die Links-Rotation für einen Teilbaum mit Wurzel u (wird als Parameter übergeben) innerhalb eines AVL-Baumes durchführt. Als Rückgabewert soll ihre Funktion den neuen Wurzelknoten des rebalancierten Teilbaumes zurückliefern. Zu jedem Knoten x werden folgende Daten gespeichert:

$x.key$: Schlüssel von x
 $x.parent$: Verweis auf Vaterknoten von x
 $x.left$: Verweis auf linkes Kind von x
 $x.right$: Verweis auf rechtes Kind von x
 $x.height$: Höhe des Teilbaumes mit Wurzel x

Hinweis: Beachten Sie, dass Ihre Rebalancierungsfunktion so ausgelegt ist, dass **alle** oben angeführten Daten eines Knotens nach der Rebalancierung korrekt sind!

- Geben Sie die Laufzeit ihres Algorithmus in Θ -Notation in Abhängigkeit einer geeigneten Größe an.