

Klausur Algorithmen und Datenstrukturen SS08, Ing.-Inf.

01. August 2008, Arbeitszeit 90 min

NICHT MIT BLEISTIFT ODER ROTSTIFT SCHREIBEN!

**Es sind keine Hilfsmittel zugelassen, insbesondere
 TASCHENRECHNER, FOTOAPPARATE oder MOBILTELEFONE.**

NAME, VORNAME:

Studiennummer:

Matrikel:

Code

Einsichtnahme
Datum, Unterschrift

Aufgabe	1	2	3	4	5	6	7	Ges.
erreichbare Punktzahl	21	13	8	16	10	10	12	90
erreichte Punktzahl								

Aufgabe 1

[21 Punkte]

Bitte kreuzen Sie für jede der folgenden Fragen entweder „JA“ oder „NEIN“ in jeder Zeile an.

Bewertung: Ist C die Anzahl der richtigen Antworten, so errechnet sich die Anzahl P der erzielten Punkte aus $P := \frac{3}{2} \cdot \max\{0, C - 6\}$.

Nichtbeantwortete Fragen werden wie falsche Antworten bewertet.

(a) Welche Aussagen sind richtig für das Sortieren von n Schlüsseln?

JA NEIN

- Quicksort benötigt im schlechtesten Fall $O(n \log n)$ Vergleiche.
 Mergesort benötigt im schlechtesten Fall $O(n \log n)$ Vergleiche.

(b) Welche Aussagen sind richtig für das Sortieren von n Schlüsseln?

JA NEIN

- Jedes vergleichsbasierte Sortierverfahren benötigt im durchschnittlichen Fall mindestens $n \log n$ Vergleiche.
 Countingsort mit n Schlüsseln $(x_1, x_2, x_3) \in \{1, \dots, n\}^3$ benötigt Zeit $O(n)$.

(c) Welche Aussagen sind richtig für die Implementierung von Stacks?

JA NEIN

- Die Verdoppelungsstrategie kann man sowohl für die Implementierung mit Listen als auch für die Implementierung mit Arrays einsetzen.
 n Operationen auf einem Stack mit Verdoppelungsstrategie haben im schlechtesten Fall einen Zeitbedarf von $\Omega(n \log n)$.

(d) Welche Aussagen sind richtig bei Hashing mit offener Adressierung?

JA NEIN

- Die Suchzeit für erfolglose Suche mit linearem Sondieren beträgt $O(\log \alpha)$, wobei $\alpha = n/m$ der Auslastungsfaktor ist.
- Unter Tabellenhashing versteht man die Speicherung der Schlüssel in einer Hashtabelle.

(e) Welche Aussagen sind richtig bei Hashing?

JA NEIN

- Beim Hashing mit offener Adressierung muss stets mindestens eine Zelle leer sein.
- Beim Hashing mit verketteten Listen darf der Auslastungsfaktor $\alpha = n/m$ den Wert 1 nicht überschreiten.

(f) Welche Aussagen sind richtig für Suchbäume?

JA NEIN

- Die maximale Tiefe eines 2-3-Baums mit n Schlüsseln ist $n^{2/3}$.
- Die maximale Tiefe eines binären Suchbaums mit n Schlüsseln ist \sqrt{n} .

(g) Welche Aussagen sind richtig für die Implementierung von Datenstrukturen für dynamische Mengen?

JA NEIN

- Bei der Implementierung von dynamischen Mengen mit aufsteigend sortierten Listen haben die Operationen *union* und *intersection* für Mengen der Größe n_1 bzw. n_2 die Laufzeit $\Theta(n_1 \cdot n_2)$.
- Bei der Implementierung von dynamischen Mengen mit aufsteigend sortierten Arrays hat die Operation *member* Laufzeit $O(\log n)$.

(h) Welche Aussagen sind richtig für AVL-Bäume?

JA NEIN

- Eine Einfügung kann bis zu $1,44 \log n$ Einfachrotationen erfordern.
- Einfügungen, Löschungen und Suchen haben Laufzeit $O(\log n)$.

(i) Welche Aussagen sind richtig für alle zusammenhängenden ungerichteten Graphen $G = (V, E)$, mit $n = |V|$?

JA NEIN

- Die Kantenzahl beträgt mindestens $n - 1$.
- Tiefensuche (DFS) in G erfordert Zeit $O(n \log n)$.

(j) Welche Aussagen sind richtig für gerichtete Graphen $G = (V, E)$, mit $n = |V|$?

JA NEIN

- Die Adjazenzlistendarstellung benötigt Speicherplatz $\Theta(n^2)$, unabhängig von $|E|$.
- Mit einer Variante von Tiefensuche (DFS) in G kann man in Zeit $O(|V| + |E|)$ entscheiden, ob es in G einen gerichteten Kreis gibt, und gegebenenfalls eine topologische Sortierung ermitteln.

Aufgabe 2 (Hashing mit verketteten Listen)

[13 Punkte]

Gegeben sei das Universum $U = \{0, 1, \dots, p-1\}$, eine Menge D von Datensätzen und der Wertebereich $M = [m] = \{0, \dots, m-1\}$. Wir verwenden Hashing mit verketteten Listen. Die verwendete Hashfunktion sei h .

- (a) [5 Punkte] Beschreiben Sie in Worten die Datenstruktur und die Implementierung der Operation $insert(x, r)$.

- (b) [2 Punkte] Formulieren Sie die Uniformitätsannahme (UF_2) für U , M und h .

- (c) [2 Punkte] Nehmen Sie an, die Hashfunktion h auf U erfüllt (UF_2), und $S \subseteq U$ ist eine Menge mit $n = |S|$. Geben Sie eine möglichst genaue Formel für die *erwartete* Anzahl von Schlüsselvergleichen bei erfolgloser und bei erfolgreicher Suche in der Datenstruktur an. (Die Formel sollte $\alpha = n/m$ benutzen.)

Erfolglose Suche:

Erfolgreiche Suche:

Fortsetzung nächste Seite !

- (d) [1 Punkt] Was ist die Anzahl von Schlüsselvergleichen im schlechtesten Fall bei erfolgloser Suche?
- (e) [3 Punkte] Geben Sie eine Klasse (Menge) \mathcal{H} von Hashfunktionen von U nach M an, die (UF_2) exakt oder wenigstens näherungsweise realisiert.

Aufgabe 3 (Baumdurchlauf)

[8 Punkte]

Für einen Binärbaum T ist definiert:

$\text{TIPL}(T) = \sum_{v \text{ Knoten in } T} d(v)$, die Summe aller Tiefen von inneren Knoten in T .

Mit $n(T)$ bezeichnen wir die Anzahl der inneren Knoten in T .

Beachte:

Wenn $T = \square$, dann ist $\text{TIPL}(T) = 0$;

wenn $T = (T_1, v, T_2)$, dann ist $\text{TIPL}(T) = \text{TIPL}(T_1) + \text{TIPL}(T_2) + n(T_1) + n(T_2)$.

- (a) [6 Punkte] Geben Sie einen Algorithmus (in Pseudocode) an, der zu gegebenem Binärbaum T die Werte $\text{TIPL}(T)$ und $n(T)$ berechnet.

(Hinweis: Baumdurchlauf in einer geeigneten Reihenfolge [Prä-/In-/Postorder], rekursive Variante.)

- (b) [2 Punkte] Geben Sie die Laufzeit Ihres Algorithmus auf einem Baum mit n Knoten an (in O -Notation, ohne Beweis).

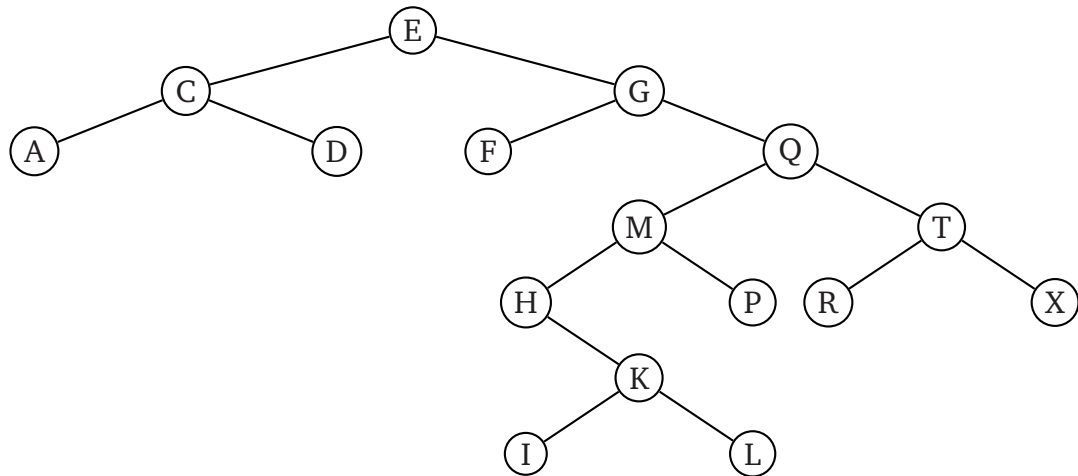
Aufgabe 4 (Einfügen und Löschen in binären Suchbäumen)

[16 Punkte]

(a) [7 Punkte] Schreiben Sie **iterativen** Pseudocode für das Einfügen eines Eintrags (x, r) in einen binären Suchbaum T .

(Ausweichmöglichkeit: **rekursiver** Pseudocode. 2 Punkte Abzug!)

(b) [5 Punkte] Im folgenden binären Suchbaum



ist der Eintrag G zu löschen.

Für welchen Knoten wird die Prozedur *extractMin* aufgerufen?

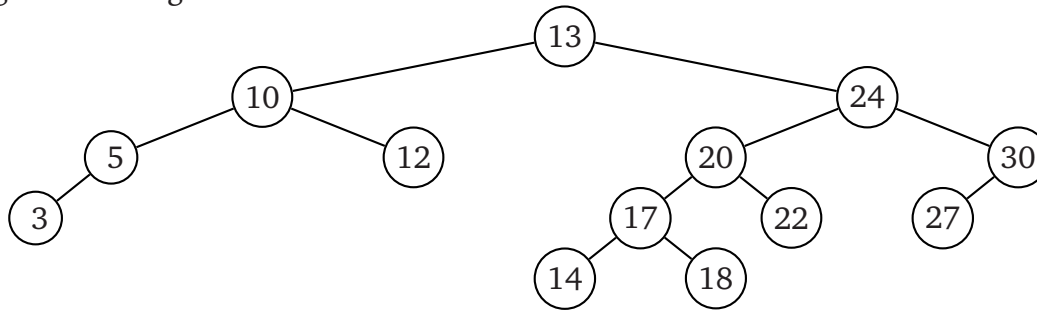
Zeichnen Sie den Baum nach Ausführung der Löschoption.

(c) [2 Punkte] Geben Sie möglichst genaue obere und untere Schranken für die Tiefe von binären Suchbäumen mit n inneren Knoten an.

(d) [2 Punkte] Bei Einfügen der Schlüssel $1, \dots, n$ in einen leeren binären Suchbaum: Welche Einfügereihenfolge führt zu einem Baum mit maximaler Tiefe?

Aufgabe 5 (Einfügen und Löschen in AVL-Bäumen, Tiefe von AVL-Bäumen)[10 Punkte]

Gegeben sei folgender AVL-Baum T :

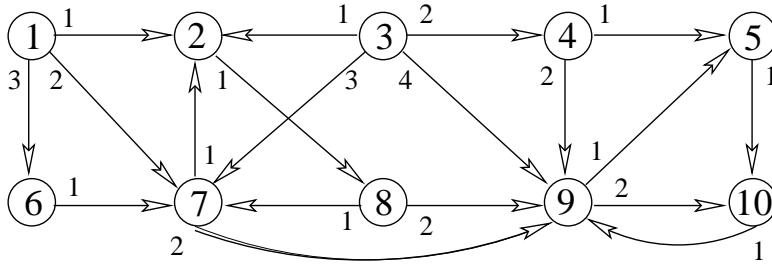


- (a) [3 Punkte] Tragen Sie an jedem inneren Knoten v den Balancefaktor $\text{bal}(v)$ ein. Woran erkennt man, dass T ein AVL-Baum ist?
- (b) [4 Punkte] Führen Sie die Operation $\text{AVL_insert}(T, 15)$ aus. Zeichnen Sie (nur) den resultierenden Baum. (Zwischenschritte auf Konzept-/ Schmierpapier).
- (c) [3 Punkte] Führen Sie im angegebenen Baum T die Operation $\text{AVL_delete}(T, 12)$ aus. Zeichnen Sie (nur) den resultierenden Baum. (Zwischenschritte auf Konzept-/ Schmierpapier).

Aufgabe 6 (Breitensuche in Digraphen)

[10 Punkte]

Gegeben ist der folgende gerichtete Graph $G = (V, E)$:



Die Nummern an den Kanten geben die Reihenfolge in der jeweiligen Adjazenzliste wieder.

- (a) [6 Punkte] Führen Sie in G die Operation $\text{bfs}(1)$ aus. Schreiben Sie dazu die Warteschlange (Queue) explizit auf („enqueue“ per Anhängen, „dequeue“ per Durchstreichen).

Schreiben Sie für jeden Knoten seine bfs-Nummer und seine Levelnummer auf (Tabelle).

- (b) [2 Punkte] Geben Sie an, welchen Zeitbedarf $\text{bfs}(u)$ hat, wenn $G = (V, E)$ beliebig, und $u \in V$ ein beliebiger Knoten ist.

- (c) [2 Punkte] Geben Sie an, welchen Zeitbedarf $\text{BFS}(G)$ hat, wenn $G = (V, E)$ ein beliebiger Graph ist („große“ Breitensuche im gesamten Graphen).

Aufgabe 7 (Heapsort)

[12 Punkte]

Gegeben sei folgendes Array $A[1..8]$ mit Einträgen aus $\{A, \dots, Z\}$ (mit der gewöhnlichen Ordnung):

A:

B	E	C	H	F	G	M	X
---	---	---	---	---	---	---	---

(a) [2 Punkte] Zeichnen Sie den vollen/linksvollständigen Binärbaum, der diesem Array entspricht.

(b) [2 Punkte] Ist $A[1..8]$ ein Min-Heap?

Falls ja: welche Bedingung haben Sie überprüft?

Falls nein: Wieso nicht?

Gegeben sei nun folgendes Array $B[1..8]$, das kein Min-Heap ist:

B:

D	E	M	X	F	G	B	H
---	---	---	---	---	---	---	---

(c) [4 Punkte] Wenden Sie auf das Array $B[1..8]$ die Prozedur `MakeHeap` aus der Vorlesung an.

Geben Sie das resultierende Array und den resultierenden Binärbaum an (Zwischenschritte: Konzept-/Schmierpapier).

(d) [3 Punkte] Führen Sie mit dem Ergebnis von (c) eine Runde der Prozedur `HeapSelect` aus. Geben Sie wieder das resultierende Array und den resultierenden Baum an (Zwischenschritte: Konzept-/Schmierpapier).

(e) [1 Punkt] Wieviele Vergleiche führt `HeapSort` auf einem Array der Größe n maximal durch? (Möglichst genau. Richtige Antwort in O -Notation: Punktabzug.)

Viel Erfolg!