

Aufgabe 1 Allgemeine Fragen - MC

(18 Punkte)

Entscheiden Sie, welche Antworten zu folgenden Fragen zutreffen und kreuzen Sie diese an. Mehrere Antworten können ausgewählt werden. Eine richtige Antwort bedeutet einen halben Punkt, eine falsche Antwort bedeutet einen halben Punkt Abzug für die jeweilige Frage. Jeder Block gibt mindestens null Punkte. Sie können in dieser Aufgabe minimal 0 Punkte und maximal 18 Punkte erreichen.

- (a) Welche Aussagen über Komplexitäten sind korrekt?

$f(n) + 100n \in \mathcal{O}(f(n))$.

$\boxed{2^{n+a} \in \mathcal{O}(2^n)}$ (mit a konstante positive Zahl).

3 Es sei $g(n) = \sum_{j=0}^n j$ und $f(n) = n^2$. Dann ist $g \in \mathcal{O}(f)$.

$f(n) \in \mathcal{O}(g(n)) \Leftrightarrow g(n) \in \mathcal{O}(f(n))$.

$f(n) \in \mathcal{O}(g(n)) \Rightarrow \forall n \in \mathbb{N} : f(n) < g(n)$.

$\boxed{\mathcal{O}(n^2) \subset \mathcal{O}(2^n)}$.

- (b) Welche Aussagen über Sortieralgorithmen sind zutreffend?

3 QuickSort arbeitet im Worst-Case schneller als alle anderen Sortieralgorithmen.

SelectionSort wählt im k -ten Schritt das k -kleinste Element und tauscht mit dem Element bei k .

Mit HeapSort lassen sich nur Arrays sortieren, die die Heapeigenschaft besitzen.

*einfach
zuerst Häufel*

MergeSort arbeitet nach dem Divide-and-Conquer-Prinzip.

BubbleSort kommt im besten Fall ohne Vertauschungsoperationen aus.

Terminierende und partiell korrekte Sortieralgorithmen heißen stabil.

- (c) Welche Aussagen treffen auf Bäume zu?

Ein Binärbaum hat stets mindestens so viele Blätter wie innere Knoten.

3 Ein Binärbaum der Höhe h kann durch ein Array der Länge $2^{h+1} - 1$ repräsentiert werden.

In einem AVL-Baum der Höhe h gibt es mindestens zwei Pfade, die länger als $h - 2$ sind.

Bei gegebenen Schlüsselwerten ist die Anzahl Knoten n des zugehörigen B -Baums der Ordnung k eindeutig.

Ein B^+ -Baum der Ordnung k hat genau k Wurzeln.

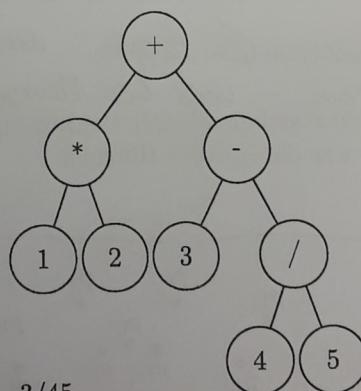
Beim Einfügen in einen AVL-Baum kann sich nur die Balance von Knoten ändern, die auf dem Suchpfad liegen.

(d) Was gilt beim Suchen/Hashing?

- 2
- Binäre Suche beschleunigt die lineare Suche durch paralleles Suchen von hinten nach vorne.
 - Beim linearen Sondieren wird jede Position der Hashtable getroffen. nur wenn $i = 1 \text{ in } c_j$
 - Doppelhashing reduziert Clusterbildung.
 - Eine perfekte Hashfunktion kann beliebig viele Objekte kollisionsfrei einfügen. nur bestimmt viele
 - Binäre Suche findet ein Element immer schneller als lineare Suche.
 - Beim Hashing mittels $h(x) = x \bmod 3$ erreicht quadratisches Sondieren einen Zyklus nach 3 Schritten.

(e) Was trifft auf den minimal spannenden Baum $G' = (V', E')$ eines Graphen $G = (V, E)$

- 2
- zu?
 - G' ist ein Subgraph mit gleichen Knoten $V' = V$, aber nur einer Teilmenge der Kanten $E' \subseteq E$.
 - Wenn zu G' noch eine beliebige Kante hinzugefügt werden würde, wäre G' kein Baum mehr.
 - In G' existiert mindestens ein Knoten mit Grad 1. wenn $E \neq \emptyset$
 - $|E'| + 1 = |V|$.
 - G' ist stets eindeutig.
 - Wenn man eine einzelne Kante aus G' löscht, so ist G' kein Baum mehr.

(f) Welche Graphtraversierungen passen zu folgendem Baum T ?

- 3
- $\text{Preorder}(T) = + * 12 - 3/45$.
 - $\text{Preorder}(T) = + * 12 - /345$.
 - $\text{Postorder}(T) = 12345/*-+$.
 - $\text{Postorder}(T) = 12 * 345/-+$.
 - $\text{Infixorder}(T) = + * -123/45$.
 - $\text{Infixorder}(T) = 1 * 2 + 3 - 4/5$.

Aufgabe 2 Grundlagen

(3+3+1+5+2+3 Punkte)

(a) Was ist ein Algorithmus?

3

ein wohldefinierter, schrittweise Lösungsverfahren für ein allgemeines Problem. ✓

(b) Gegeben sei folgender Algorithmus zur Bestimmung des Schnitts zweier gleichgroßer Mengen der Größe n , die als Arrays vorliegen.

- Die Eingabe besteht aus zwei Arrays P, Q .
- Initialisiere $R = []$ als leeres Array.
- Für jedes $p \in P$:
- Für jedes $q \in Q$:
- Falls $p = q$, dann füge p zu R hinzu.
- Gebe R zurück.

Beispiel: $P = [1, 2, 3, 4, 5]$ und $Q = [2, 4, 6, 8, 10]$ liefert $R = [2, 4]$.Geben Sie die Worst-Case-Laufzeitkomplexität in \mathcal{O} -Notation an und begründen Sie Ihr Ergebnis knapp.

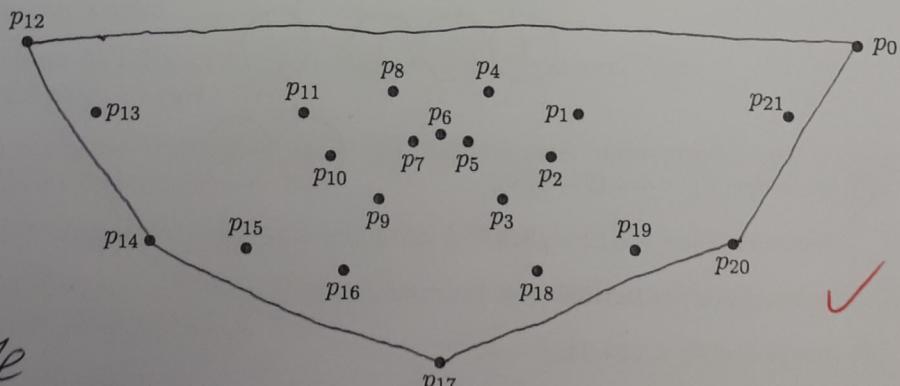
3

Angenommen, die Arrays sind sortiert. Wie lässt sich der Algorithmus dann beschleunigen?

($\mathcal{O}(|P| \cdot |Q|))$) jedes Element aus P wird mit jedem aus Q verglichen ✓
 $\mathcal{O}(n^2)$

zwei nebeneinander liegen: nur die nötigen Vergleiche durchführen - wie bei Menge-Schritt in Menge-Sort ✓

(c) Zeichnen Sie für folgende zweidimensionale Punktmenge die konvexe Hülle ein. Beschreiben Sie in einem Satz, was die konvexe Hülle ist.



kleinste Teilmenge einer Punktmenge, die sie vollständig einschließt,

die konvexe Hülle besteht aus den "äußeren" Punkten

- (d) Benennen Sie für jede der folgenden Komplexitätsklassen ein Beispiel für eine algorithmische Aufgabe.

i. $\mathcal{O}(1)$: *einfügen auf Stack* ✓

ii. $\mathcal{O}(\log n)$: *Suche in AVL Baum* ✓

5

iii. $\mathcal{O}(n)$: *lineare Suche im array* ✓

iv. $\mathcal{O}(n \log n)$: *Sortieren einer liste (z.B. mit Merge-Sort)* ✓

v. $\mathcal{O}(n^2)$: *Bestimmung des Schnitts zweier gleichgroßer unsortierter Mengen* ✓

- (e) Geben Sie in eigenen Worten (1-2 Sätze) die Bedeutung der \mathcal{O} -Notation an.

\mathcal{O} -Notation gibt die Obergrenze der Laufzeiten von Algorithmen an genau

1

- (f) Bestimmen Sie mittels Master-Theorem die Komplexitätsklasse für folgende Rekursionsgleichung:

$$T(1) = 0$$

$$T(n) = 4^3 T\left(\frac{n^2}{2^6 n}\right) + 3n$$

2

$$a = 4^3, b = 2^6, g = 1$$

$$a = b^g$$

$$\Rightarrow \underbrace{\epsilon \mathcal{O}(n)}_{\mathcal{O}(n \log n)}$$

Aufgabe 3 Sortieralgorithmen

(5+5 Punkte)

- (a) Sortieren Sie die Liste [5, 2, 4, 1, 3] mit SelectionSort. Tragen Sie dafür den Inhalt der Liste nach jeder Iteration in die Tabelle ein.

5	2	4	1	3
1	2	4	5	3
1	2	4	5	3
1	2	3	5	4
1	2	3	4	5

5/5

- (b) Sortieren Sie die Liste mit QuickSort. Das Pivotelement soll stets das erste Element sein. Geben Sie jeden Verfeinerungsschritt an. Markieren Sie alle Pivotelemente sichtbar.

49	99	69	32	11	42	64	84
32	11	42	49	99	69	64	84
11	32	42	49	69	64	84	99
11	32	42	49	64	69	84	99

5/5

Hinweis: Es gibt mehr Zeilen als benötigt werden.

Aufgabe 4 Hashing

(4+4+2 Punkte)

Tragen Sie die folgenden Schlüssel in der angegebenen Reihenfolge nach dem jeweils angegebenen Prinzip in eine Hashtabelle der Größe $m = 11$ ein. Verwenden Sie die Hashfunktion $h(x) = x \bmod 11$ und geben Sie für jeden Schlüssel an, wie oft sondiert werden musste.

4, 24, 5, 11, 15, 16, 26, 2

(a) geschlossenes Hashing; lineares Sondieren ($h(x) + j$)

Index	0	1	2	3	4	5	6	7	8	9	10
Hashtabelle	11		24	2	4	5	15	16	26		
Sondierungen	0	✓	0	1	0	0	2	2	4		

8

8/4

(b) geschlossenes Hashing; quadratisches Sondieren ($h(x) + j^2$)

Index	0	1	2	3	4	5	6	7	8	9	10
Hashtabelle	11		24	✓	4	5	16		15	26	
Sondierungen	0	0	✓	0	0	1		2	3		

3/4

(c) Nennen Sie wesentliche Vor- und Nachteile des offenen und geschlossenen Hashings.

7/2

das gesallo offene Hashing braucht zusätzliche
Datenstrukturen ✓

im geschlossenen evtl Clustering

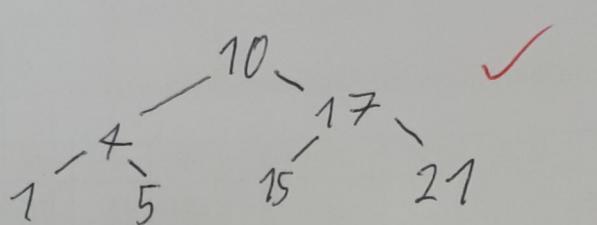
0(1) Zugriffzeit in beiden Fällen evtl zerstört

(3+2+3+4+3 Punkte)

Aufgabe 5 Suchbäume

- (a) Geben Sie den binären Suchbaum an, der entsteht, wenn man folgende Elemente der Reihe nach einfügt:

10 17 15 4 5 1 21



3

- (b) Diskutieren Sie den Unterschied zwischen einem binäreren Suchbaum und einem Heap in max. 3 Sätzen.

z.B. g.p.c

In einem Heap müssen nur direkte Nachfolger kleiner sein als der Vaterknoten.

Im Suchbaum müssen alle Elemente der Teilläume größer bzw. kleiner sein als der Wert im Vaterknoten.

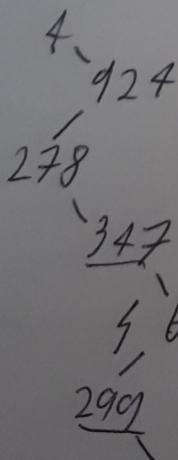
↳ zu ungern
→

0,5

- (c) Angenommen, ein binärer Suchbaum enthält Schlüsselwerte zwischen 1 und 1000 und wir suchen 363. Warum kann folgende Sequenz von untersuchten Schlüsselwerten nicht möglich sein?

3

4 924 278 347 621 299 392 358 363

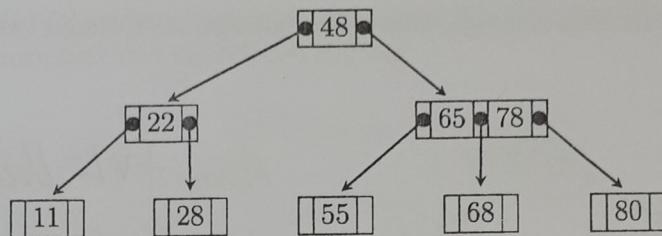


✓

der Schlüssel 299 stünde rechts zum Knoten 347

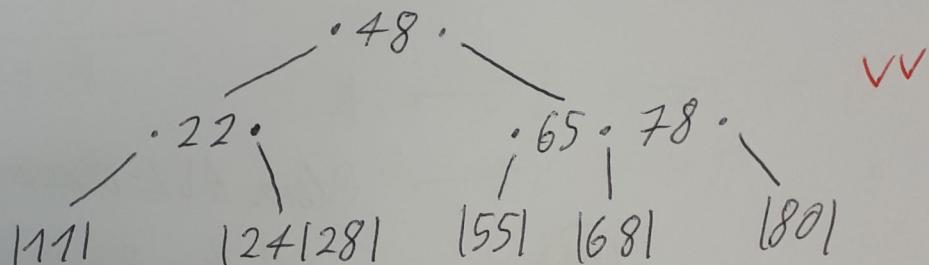
→ Eigenschaft des Suchbaums verletzt

(d) Gegeben ist folgender B-Baum der Ordnung $k = 1$:

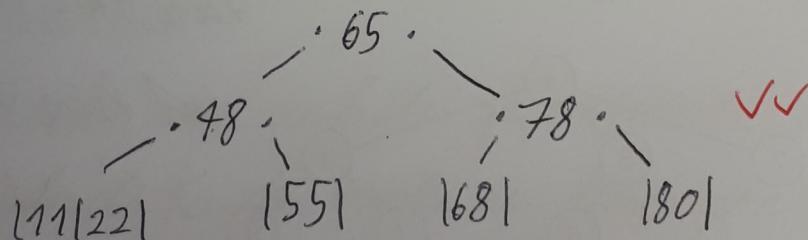


Geben Sie in den folgenden Teilaufgaben lediglich das Endresultat an und starten Sie jeweils erneut mit dem oben angegebenen B-Baum!

- i. Fügen Sie in den oben gegebenen B-Baum den Wert 24 ein.



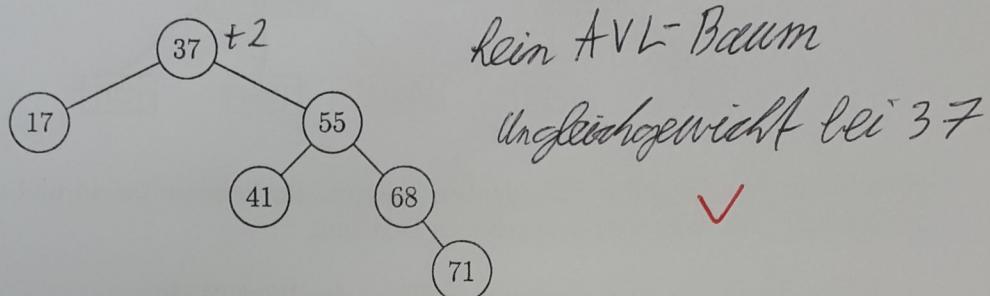
- ii. Entfernen Sie aus dem in der Aufgabenstellung angegebenen B-Baum den Wert 28.



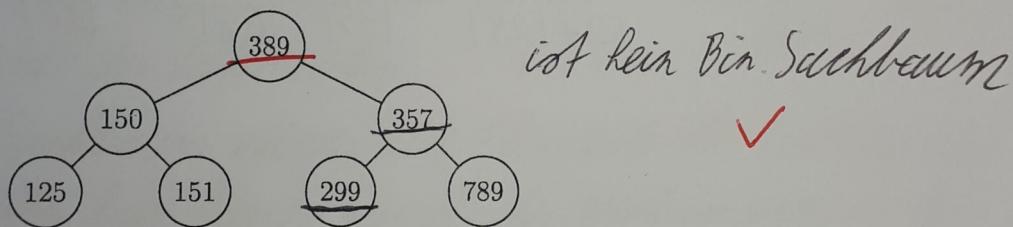
- (e) Gegeben sind die folgenden Bäume. Entscheiden Sie jeweils, ob der gegebene Baum ein AVL-Baum ist oder nicht.

Begründen Sie Ihre Aussage, falls es kein gültiger AVL-Baum ist!

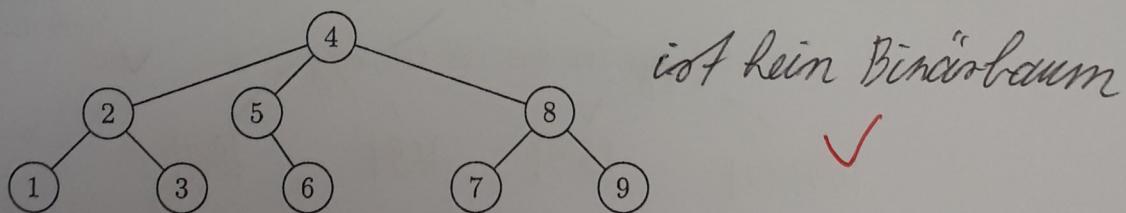
i.



ii.



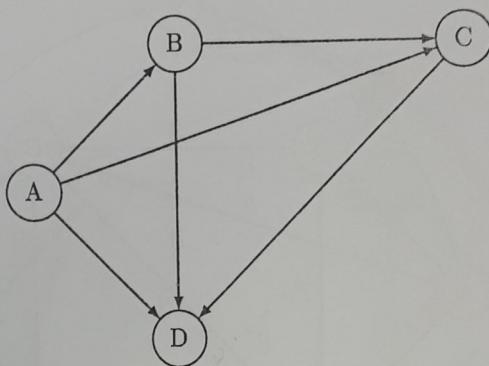
iii.



Aufgabe 6 Graphalgorithmen

(3+1+6+5 Punkte)

- (a) Betrachten Sie den folgenden gerichteten, ungewichteten Graphen. Geben Sie den Graphen als Adjazenzmatrix und als Adjazenzliste an.



nach

	A	B	C	D
A	1	1	1	1
B	0	0	1	1
C	0	0	0	1
D	0	0	0	0

von

$A: \rightarrow B \rightarrow C \rightarrow D$
 $B: \rightarrow C \rightarrow D$
 $C: \rightarrow D$
 $D: /$

3

MP

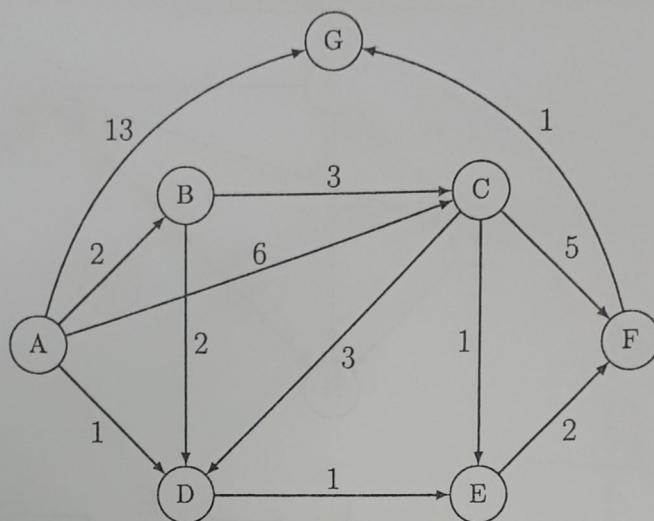
- (b) Diskutieren Sie knapp den Unterschied zwischen Adjazenzmatrizen und Adjazenzlisten. In welchen Fällen präferiert man die Matrixdarstellung, in welchen die Adjazenzdarstellung? (max. 3 Sätze)

Die Listen sind einfacher zu initialisieren, aber eingehende Kanten werden nur langsam gefunden.

Nur eingehende Kanten wichtig: Adjazenzliste
 mehr Information gewünscht: Adj.-Matrix

1/2 MP

- (c) Betrachten Sie den folgenden gerichteten Graphen mit Kosten gemäß Kantenbeschriftungen. Haben Sie im Folgenden in einem Schritt eine gleichwertige Wahlmöglichkeit zwischen zwei Knoten, so gehen Sie *alphabetisch* vor.



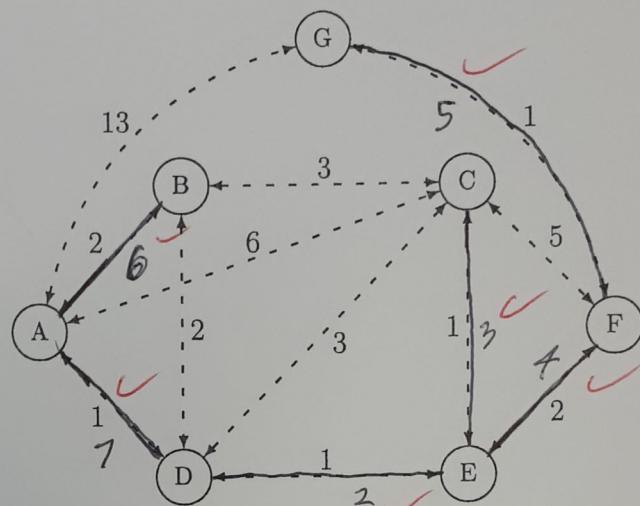
616

BR

Ermitteln Sie mittels Dijkstra-Algorithmus die Kosten der günstigsten Verbindungen von Knoten A aus. v_k bezeichnet den in Iteration k ausgewählten Knoten, $D_k(X)$ die bisher kürzeste Distanz von Knoten A zu X.

- (d) Ermitteln Sie mittels Prim-Algorithmus den minimalen Spannbaum des Graphen. Beginnen Sie mit Knoten A. Zeichnen Sie die Kanten ein und nummerieren Sie sie anhand ihrer Einfügereihenfolge. Nutzen Sie die Ersatzvorlage, falls Ihr erster Lösungsversuch fehlgeschlagen ist.

bei 4
mehrere
Möglichkeiten



5/5
RR

