Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Thomas Seidl
Florian Richter, Andrea Maldonado

SS 21
10.12.21

Prüfung
Algorithmen und Datenstrukturen

Allgemeine Hinweise:

- **Laden Sie sich zusätzlich zu dieser Angabe die Textdatei mit dem Titel "loesungsbogen.txt" als Vorlage für Ihre Abgabe herunter, in die Sie Ihre Lösungen zu den Aufgaben der Prüfung eintragen.**
- **Nutzen Sie bitte ausschließlich die .txt-Datei ("loesungsbogen.txt"), um ihre Lösung abzugeben.**
- **Die Art und Weise, eine Aufgabe zu lösen, wird ihnen detailliert auf der Angabe beschrieben. Lesen Sie diese daher bitte genau.**
- **Sie bekommen ab Beginn der Prüfung 180 Minuten Zeit, diese herunterzuladen, zu bearbeiten und Ihre Lösung wieder hochzuladen (alles via Uni2work). Somit haben Sie 90 Minuten extra Zeit, um auf eventuelle technische Probleme reagieren zu können.**
- **Speichern Sie vor Bearbeitung der Prüfung die Lösungsdatei mit dem Titel "loesungsbogen_[Matrikelnummer].txt" an einem Ort ab, an dem Sie sie jederzeit wiederfinden. Ersetzen Sie dabei [Matrikelnummer] mit ihrer Matrikelnummer.**
- **Vergessen Sie nicht, die Lösungsdatei möglichst häufig zwischenzuspeichern. So reduzieren Sie Stress durch technische Probleme am Ende der Abgabezeit.**
- **Notfall-Hotline: 089 / 2180 9325**

- Ich habe die Lösung eigenständig und ohne die Hilfe Dritter angefertigt.
- Ich bin der rechtmäßige Nutzer dieses Uni2Work-Accounts und gebe die Lösung nicht für jemand Drittes ab.
- Ich bin zum Zeitpunkt der Prüfung immatrikuliert und kann dies auch durch ein entsprechendes Dokument jederzeit während des Prüfungsprozesses bestätigen.
- Ich veröffentliche keinerlei Prüfungsdokumente wie Aufgabenstellung, Korrektur, etc. im öffentlichen Raum (Online, Aushang, Bekanntenkreis,...).
- Ich sorge dafür, dass ich regelmäßige Updates meiner Lösungen in Uni2Work mache, sodass es zu verminderten technischen Problemen am Ende kommt. Die letzte innerhalb der Abgabefrist hochgeladene Version wird korrigiert. Achtung: Uni2Work logged Sie nach einer gewissen Zeit der Inaktivität automatisch wieder aus!

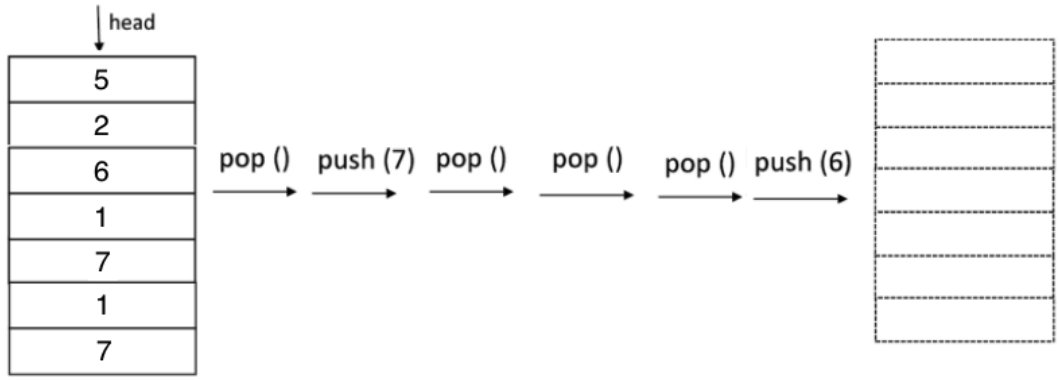
Die Prüfung besteht aus 6 Aufgaben. Die Punktezahl ist bei jeder Aufgabe angegeben.

Aufgabe	mögliche Punkte	erreichte Punkte
1.Allgemeine Fragen	16	
2.Graphalgorithmen	25	
3.Sortieralgorithmen	16	
4.Algorithmenanalyse	24	
5.Suchbäume	16	
6.Hashing	18	
Summe:	115	
Note:		

Bewertung Multiple Choice: Geben Sie an, welche Antworten zu folgenden Fragen zutreffen, indem Sie die Ziffern der richtigen Antworten notieren. Alle nicht notierten Ziffern sind automatisch als nicht zutreffend klassifiziert (Sie können sich also nicht enthalten). Beispiel: Angenommen, die Antworten 1,2,3 sind zutreffend, 4,5,6 sind nicht zutreffend, so notieren Sie in der Antwort-Datei 1,2,3 (Antworten 4,5,6 werden *NICHT* notiert und sind damit automatisch als nicht zutreffend gekennzeichnet). Laut Prüfungsordnung bekommen Sie pro gegebener zutreffender Antwort einen Punkt.
Für jede richtige Antwort bekommen Sie einen Punkt, für jede falsche Antwort wird ein Punkt abgezogen. Für jeden Aufgabenteil erhalten Sie nicht weniger als Null Punkte.

Aufgabe 1 Allgemeine Fragen (16 Punkte)

(a) Gegeben sei folgender Stack `head->5,2,6,1,7,1,7` (siehe Bild unten). `head->` zeigt an, welches das oberstes Element ist. Welche Aussagen stimmen für das Endergebnis nach den sechs genannten Operationen?



1 Der Wert 7 steht zweimal im Stack

2 Der Wert 6 steht zweimal im Stack

3 Der Stack ist länger als vor den Operationen

4 Der Wert 1 ist zweimal im Stack

5 Der Wert 7 ist einmal im Stack

6 Die Reihenfolge vom Stack ist nicht mehr wichtig

(b) Die Knoten des folgenden Baums werden in einem Tiefendurchlauf ausgegeben. Welche Reihenfolgen treffen für die jeweilige Notationsform (Infix, Präfix, Postfix) zu?

1 Postfix: 6 / 4 3 + * 4 5 + 2 1 * −

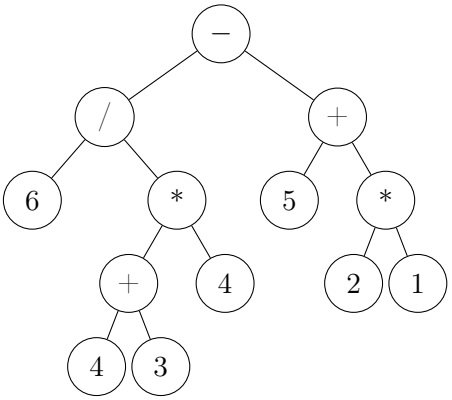
2 Postfix: 6 4 3 + 4 * / 5 2 1 * + −

4 Infix: 6 / 4 + 3 * 4 − 5 + 2 * 1

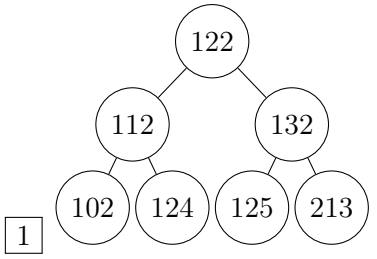
3 Infix: 4 3 + 4 * 6 / 2 1 * 5 + −

5 Praefix: − / 6 * + 4 4 3 + 5 * 2 1

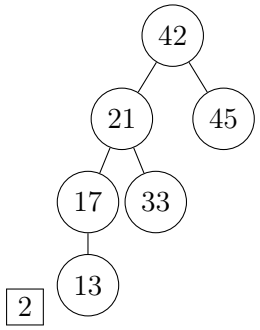
6 Praefix: − / 6 * + 4 3 4 + 5 * 2 1



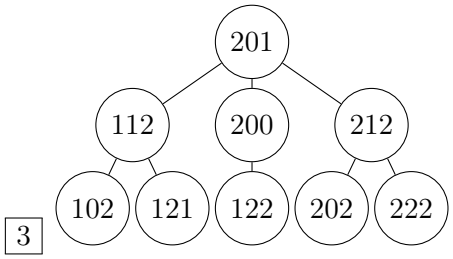
(c) Gegeben sind folgende Bäume. Welche davon sind AVL-Bäume?



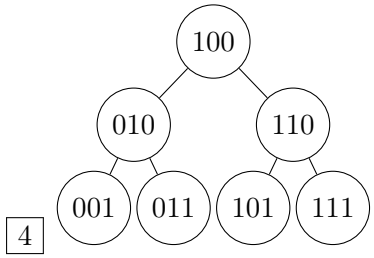
1



2



3



4

Aufgabe 2 Graphalgorithmen (25 Punkte)

(a) Gegeben sei die folgende Adjazenzliste. Welche Aussagen über den zugehörigen Graphen treffen zu?

$A \to B \to F$
$B \to C \to F$
$C \to D$
$D \to E$
$E \to A \to F$
$F \to C \to D$

1 Für alle Knotenpaare $(v, w) \in E(G)$ gilt: $v \neq w$.

2 Für alle Knotenpaare $(v, w) \in E(G)$ gilt: $(w, v) \notin E(G)$.

3 Zwischen allen Knotenpaaren existiert ein Pfad.

4 Der Graph ist ein minimaler Spannbaum.

5 Der Graph ist planar (lässt sich zeichnen, ohne dass sich Kanten kreuzen).

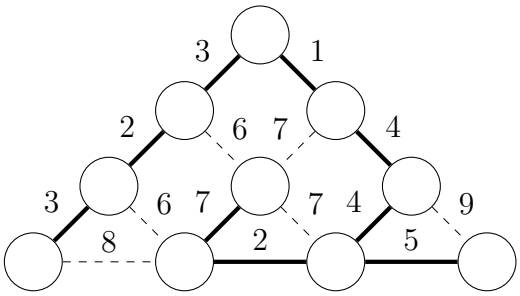
6 Die Liste repräsentiert einen gerichteten Graphen.

7 Knoten F hat 6 adjazente Knoten.

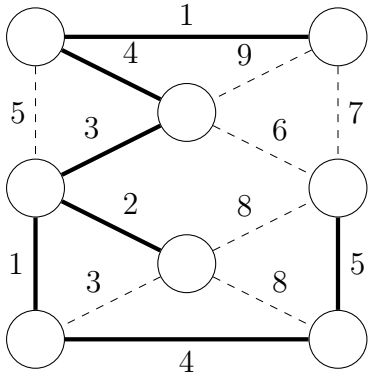
8 Es gibt einen isolierten Knoten.

(b) Die folgende Illustrationen enthalten Graphen mit gestrichelt und durchgezogenen Kanten. Welche der durchgezogenen Kantenmengen repräsentieren valide minimale Spannbäume dieser Graphen?

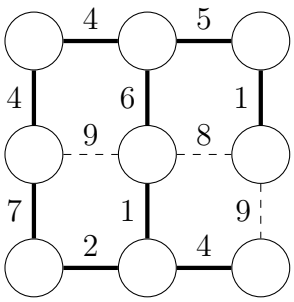
1



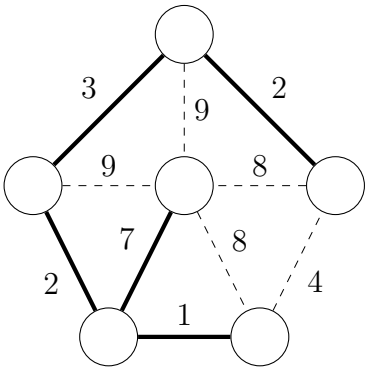
2



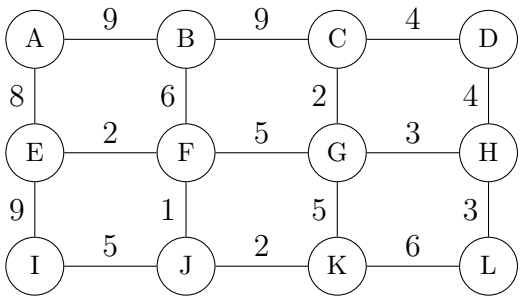
3



4



(c) Wenden Sie den Dijkstra-Algorithmus mit Startknoten A auf folgenden Graphen G an. Welche Aussagen treffen zu?



1 Der kürzeste Weg von A nach L enthält H.

2 Der kürzeste Weg von A nach L enthält G.

3 Der kürzeste Weg von A nach D hat Länge 21.

4 Es gibt zwei Knoten, die die gleiche Distanz zu A haben.

5 Die Bearbeitungsreihenfolge ist eindeutig.

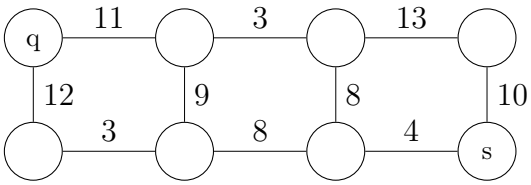
6 Die Bearbeitungsreihenfolge von H und K ist uneindeutig.

7 Die Bearbeitung von D verbessert keine bisher gefundenen Wege.

8 L wird zuletzt bearbeitet.

9 Auf jedem kürzesten Pfad werden nur Kanten mit Gewicht ≤ 8 verwendet.

(d) Gegeben folgendes Flussnetzwerk mit Quelle q und Senke s, welche Aussagen sind korrekt?



1 Der maximale Fluss ist 11.

2 Der maximale Fluss ist 12.

3 Der maximale Fluss ist 13.

4 Der maximale Fluss ist 14.

Aufgabe 3 Sortieralgorithmen (16 Punkte)

(a) Gegeben sei das folgende Array
$[100, 12, 20, 22, 3, 0, 30, 1, 103, 31, 102, 2, 33, 11, 32, 101, 21, 13, 10, 23]$.

und folgende Zwischenergebnisse eines einfachen Sortieralgorithmus. Welcher Algorithmus aus der Vorlesung wurde jeweils benutzt?

i. $[3, 0, 12, 1, 20, 22, 30, 2, 31, 11, 32, 33, 21, 20, 10, 23, 100, 101, 102, 103]$

- [1] MergeSort
- [2] BubbleSort
- [3] InsertionSort
- [4] SelectionSort

ii. $[0, 1, 2, 3, 12, 20, 22, 30, 31, 100, 102, 103, 33, 11, 32, 101, 21, 13, 10, 23]$

- [1] MergeSort
- [2] BubbleSort
- [3] InsertionSort
- [4] SelectionSort

iii. $[0, 1, 2, 3, 10, 11, 12, 13, 20, 31, 102, 103, 33, 100, 32, 101, 21, 30, 22, 23]$

- [1] MergeSort
- [2] BubbleSort
- [3] InsertionSort
- [4] SelectionSort

(b) Betrachtet wird folgender Algorithmus *someSort*. Dabei sind *mergeSort* und *bubbleSort* wie in der Vorlesung dargestellt:

```
def someSort(array):
    if len(array)>1000:
        mergeSort(array)
    else:
        bubbleSort(array)
```

Was ist die Worst-Case-Laufzeit von *someSort* bei einem Array der Länge n?

1 $O(n^{-2})$

2 $O(n)$

3 $O(n \cdot log(n))$

4 $O(log(n))$

Aufgabe 4 Algorithmenanalyse (24 Punkte)

Die Funktion *qsum* berechnet die Summe der quadrierten Einträge einer nichtleere Liste von Zahlen. Zum Beispiel ist $qsum([1,2,3]) = 1^2 + 2^2 + 3^3 = 14$. Folgende Algorithmen sollen die Funktion *qsum* implementieren.

(a) Welche Aussagen treffen auf folgenden Algorithmus zu?

```
def qsum1(l):
    n = length(l)
    s = 0
    for i in 0...n-1:
        s += l[i]
    return s*s
```

1 qsum1 hat im Worst-Case eine Komplexität von $\mathcal{O}(\log n)$

2 qsum1 hat im Worst-Case eine Komplexität von $\mathcal{O}(n)$

3 qsum1 ist deterministisch.

4 qsum1 ist determiniert.

5 qsum1 ist partiell korrekt.

6 qsum1 ist terminierend.

(b) Welche Aussagen treffen auf folgenden Algorithmus zu?
// ist hier die ganzzahlige Division.

```
def qsum2(l):
    n = length(l)
    if n == 1:
        return l[0]*l[0]
    left = l[0..(n//2)-1]
    right = l[n//2..n-1]
    return qsum2(left)+qsum2(right)
```

1 qsum2 hat im Worst-Case eine Komplexität von $\mathcal{O}(\log n)$

2 qsum2 hat im Worst-Case eine Komplexität von $\mathcal{O}(n)$

3 qsum2 ist deterministisch.

4 qsum2 ist determiniert.

5 qsum2 ist partiell korrekt.

6 qsum2 ist terminierend.

(c) Welche Aussagen treffen auf folgenden Algorithmus zu?

```
def qsum3(l):
    n = length(l)
    A = matrix(n*n)
    for i in 0..n-1:
        for j in 0..n-1:
            A[i][j] = l[i]*l[j]
    s = 0
    for i in 0..n-1:
        s += A[i][i]
    return s
```

1 qsum3 hat im Worst-Case eine Komplexität von $\mathcal{O}(\log n)$

2 qsum3 hat im Worst-Case eine Komplexität von $\mathcal{O}(n)$

3 qsum3 ist deterministisch.

4 qsum3 ist determiniert.

5 qsum3 ist partiell korrekt.

6 qsum3 ist terminierend.

(d) Welche Aussagen treffen auf folgenden Algorithmus zu?
pick(n) ist in dieser Aufgabe eine vordefinierte Funktion, die in $\mathcal{O}(1)$ einen zufälligen Wert zwischen 0 und $n-1$ generiert. Für Listen bezeichne + eine Listenkonkatenation in $\mathcal{O}(1)$.

```
def qsum4(l):
    s = 0
    while length(l) > 0:
        i = pick(length(l))
        s += l[i]*l[i]
        l = l[0..i-1]+l[i+1..length(l)-1]
    return s
```

1 qsum4 hat im Worst-Case eine Komplexität von $\mathcal{O}(\log n)$

2 qsum4 hat im Worst-Case eine Komplexität von $\mathcal{O}(n)$

3 qsum4 ist deterministisch.

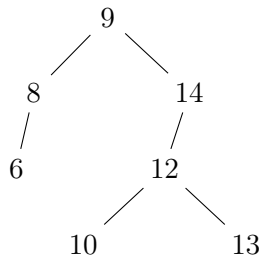
4 qsum4 ist determiniert.

5 qsum4 ist partiell korrekt.

6 qsum4 ist terminierend.

Aufgabe 5 Suchbäume (16 Punkte)

(a) Kreuzen Sie die gültige(n) Reihenfolge(n) an, in der die Werte eingefügt werden müssen, damit der unten abgebildete binärer Suchbaum entsteht. Starten Sie dabei mit einem leeren Baum.



☐ 1 9, 8, 6, 12, 14, 13, 10

☐ 2 9, 14, 8, 6, 12, 13, 10

☐ 3 9, 8, 14, 6, 12, 10, 13

☐ 4 9, 14, 12, 8, 6, 13, 10

☐ 5 9, 14, 6, 12, 10, 8, 13

☐ 6 9, 8, 14, 6, 12, 10, 14

(b) Gegeben sind Suchverläufe (besuchte Knoten) bei einer Suche in einem binären Suchbaum, der ganze Zahlen zwischen 1 und 100 speichert. Nachdem die angegebenen Schlüssel traversiert wurden, wurde die Suche erfolglos abgebrochen. Kreuzen Sie die Menge möglicher Schlüsselwerte (zwischen 1 und 100) an, nach denen gesucht worden sein könnte.

☐ 1 Besuchte Knoten: $25, 46, 30, 26, 29$, gesuchte Schlüsselwerte: $27, 28$

☐ 2 Besuchte Knoten: $25, 49, 30, 26, 28$, gesuchte Schlüsselwerte: $25, 27$

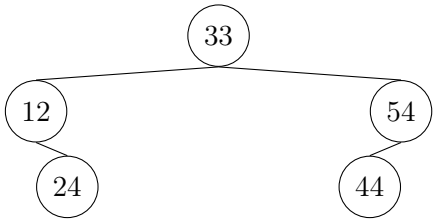
☐ 3 Besuchte Knoten: $93, 14, 23, 20, 15$, gesuchte Schlüsselwerte: $16, 17, 18, 19$

☐ 4 Besuchte Knoten: $93, 14, 23, 20, 16$, gesuchte Schlüsselwerte: $16, 17, 18, 19$

☐ 5 Besuchte Knoten: $25, 49, 30, 26, 28$, gesuchte Schlüsselwerte: $21, 22$

☐ 6 Besuchte Knoten: $93, 14, 23, 20, 15$, gesuchte Schlüsselwerte: $21, 22$

(c) Gegeben sei der folgende balancierte AVL-Baum.



Löschen Sie nun die Knoten 24, 44 und 54 und fügen Sie die 2, 4 und die 8 hinzu. Kreuzen Sie alle verwendeten Rotationsarten an, die vorgenommen werden müssen, um den Baum zu balancieren.

☐ 1 LR-Rotation

☐ 2 RL-Rotation

☐ 3 Nur Links-Rotationen

☐ 4 Nur Rechts-Rotationen

Aufgabe 6 Hashing (18 Punkte)

(a) Welche der folgenden Hash-Funktionen verteilt alle Schlüssel im Bereich 0-999 auf zehn Positionen 0 bis 9 gleichverteilt?

1 $h_1(x) = (2x + 1) \mod 10$

2 $h_2(x) = (x \mod 9) \mod 10$

3 $h_3(x) = (3x^3) \mod 10$

4 $h_4(x) = \lfloor \sqrt{x} \rfloor \mod 10$

(b) Für welche Listen ganzer Zahlen ist die Hashfunktion $h(x) = x^5 \mod 5$ perfekt?

1 6, 3, 8, 11, 1

2 3, 14, 6, 7, 0

3 13, 5, 9, 12, 6

4 5, 9, 7, 12, 3

(c) Gegeben sei die Eingabe $1, 16, 11, 5, 17, 7, 6, 13, 10, 8$. Fügen Sie diese Werte in eine Hashtabelle der Größe 10 mit Positionen 0 bis 9 mittels geschlossenem Hashing mit offener Adressierung ein. Verwenden Sie die Hashfunktion $h(x) = 2x \mod 10$ und nutzen Sie quadratisches Sondieren (c=1) bei Kollisionen. Welche Aussagen treffen zu?
Hinweis: Für alle $x \in \mathbb{N}$ gilt: $x^2 \mod 10 \in \{0, 1, 4, 5, 6, 9\}$ (insbesondere nicht 2).

1 Alle Schlüssel können in die Hashtabelle eingefügt werden.

2 8 kann nicht eingefügt werden.

3 13 wird bei Position 0 eingefügt.

4 17 wird bei Position 9 eingefügt.

5 5 wird ohne Sondierungen bei Position 0 eingefügt.

6 16 wird nach Sondierung bei Position 3 eingefügt.

(d) Nutzen Sie wieder die Eingabe $1, 16, 11, 5, 17, 7, 6, 13, 10, 8$. Die Hashtabelle hat wieder Positionen 0 bis 9. Ebenfalls nutzen Sie die Hashfunktion $h(x) = 2x \mod 10$. Hashen Sie diesmal mit offenem Hashing mit geschlossener Adressierung. Welche Aussagen treffen ohne weitere Annahmen zu?

1 Für beliebige weitere Eingaben wird die Hashtabelle nie volle Kapazität erreichen.

2 Die Positionen 7, 8 und 9 enthalten keine Schlüssel.

3 Nur mittels quadratischer Sondierung können alle Schlüssel eingefügt werden.

4 In keiner Position befindet sich genau ein Schlüssel.