Lehrstuhl für Informatik I
Effiziente Algorithmen und
wissensbasierte Systeme
Universität Würzburg

Würzburg, den 26. April 2021

Prof. Dr. Alexander Wolff
Linus Mußmächter
Vasil Alistarov

ADS - Repetitorium 2021

Probeklausur

Pseudonym: _____

Aufgabe	1	2	3	4	5	6	7	8	9	**Gesamt**
mögliche Punkte	10	10	11	15	6	10	5	13	10	90
erreichte Punkte										

Bearbeitungszeit: 90 Minuten

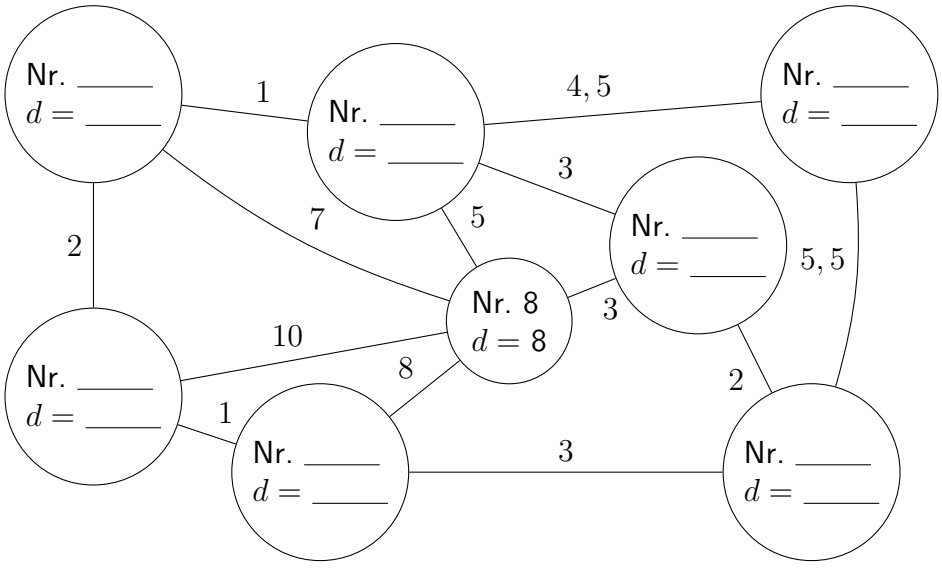
Aufgabe 1

/ 10 Punkte

Auf dem folgenden ungerichteten Graphen wurde der Dijkstra-Algorithmus ausgeführt, doch wir wissen lediglich, welcher Knoten als letztes schwarz wurde (Nr. 8) und was seine Distanz zum Startknoten (Nr. 1) ist. Die Gewichte der Kanten sind angegeben.

Finden Sie den Startknoten, nummerieren Sie die Knoten in der Reihenfolge, in der sie schwarz wurden und geben Sie für jeden Knoten die Distanz zum Startknoten an.

Der Startknoten ist eindeutig, wenn man die Nummer des letzten Knotens berücksichtigt. Ergänzen Sie die fehlenden Zahlen.



Aufgabe 2

(a) Fügen Sie die Schlüssel $4, 6, 9, 1$ in dieser Reihenfolge in die Hashtabellen ein. / 5 P.

- Verwenden Sie lineares Sondieren mit der Hashfunktion $h(k, i) = (k + i) \bmod 5$.

Index	0	1	2	3	4
Wert	_____	_____	_____	_____	_____

- Verwenden Sie doppeltes Hashing mit $h(k, i) = (h_0(k) + ih_1(k)) \bmod 5$ sowie den Funktionen $h_0(k) = (2 + k) \bmod 5$ und $h_1(k) = (k \bmod 4) + 1$.

Index	0	1	2	3	4
Wert	_____	_____	_____	_____	_____

(b) Eine Datenstuktur soll zu jeder Zeit die K letzten Messungen eines Sensors speichern. / 5 P.
Dabei ist eine *Messung* ein Paar $p = (t, v)$, wobei wir $t \in \mathbb{Z}_0^+$ als *Messzeitpunkt* und $v \in \mathbb{R}$ als *Messwert* von p bezeichnen. Auf jede Messung folgt nach *einer Zeiteinheit* die nächste Messung.

Folgende Methoden sollen implementiert sein.

- Insert(double v): Einfügen einer neuen Messung (t, v). Falls zuvor bereits K Messungen in der Datenstruktur gespeichert waren, ist die älteste Messung zu löschen, wobei t bei jedem Aufruf um 1 inkrementiert wird.
- Search(int t): Suche nach einer Messung mit Messzeitpunkt t. Falls eine Messung zum Zeitpunkt t gespeichert ist, soll diese zurückgegeben werden, und sonst nil.

Durch Speicherung der Messungen in einer Hashtabelle der Größe K können wir sowohl für Insert als auch für Search eine Laufzeit in $O(1)$ erreichen. Geben Sie eine Hashfunktion an, die dies gewährleistet.

$h(t) =$ _____

Wie gehen Sie vor, wenn die Zelle, in die eine Messung einzufügen ist, schon belegt ist? Begründen Sie Ihr Vorgehen.

Aufgabe 3

Gegeben sei folgender Algorithmus.

```
MyAlgorithm(Feld vom Typ int A, int ℓ = 1, int r = A.length)
  if ℓ < r then
    m = ⌊(ℓ + r)/2⌋
    MyAlgorithm(A, ℓ, m)
    for i = ℓ to r do
      Print(A[i]) // gibt A[i] aus
    MyAlgorithm(A, m + 1, r)
```

(a) Sei A ein Feld der Länge n. Stellen Sie eine Rekursionsgleichung für die asymptotische Worst-Case-Laufzeit $T(n)$ von MyAlgorithm(A) auf. / 3 P.

$$T(n) = \begin{cases} \_\_\_\_\_ & \text{falls } \; n > 1 \\ \_\_\_\_\_ & \text{sonst} \end{cases}$$

(b) Lösen Sie die Rekursionsgleichung aus Aufgabenteil a). Erläutern Sie Ihre Lösung. / 4 P.

$T(n) \in \Theta\Big(\_\_\_\_\_\Big)$

(c) Was gibt MyAlgorithm(A) für $A = \langle 1, 2, 3, 4 \rangle$ aus? / 4 P.

Aufgabe 4

(a) Nennen Sie eine Methode einer dynamischen Menge, die im Worst-Case in einer Implementierung durch einen Rot-Schwarz-Baum asymptotisch schneller ist als in einer Implementierung durch einen gewöhnlichen binären Suchbaum. / 2 P.

(b) Geben Sie in Pseudocode einen Algorithmus an, der für einen Binärbaum mit eingefügten Zahlen testet, ob er ein gültiger binärer Suchbaum ist. / 6 P.
Boolean BinaryTreeTest(Node $v = root$)

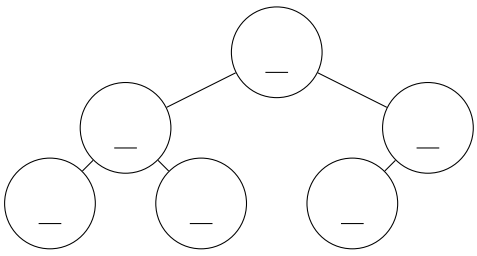
(c) Prioritätsschlangen werden oft durch Heaps implementiert. Nennen Sie einen Vorteil, den eine solche Implementierung gegenüber einer Implementierung durch einen Rot-Schwarz-Baum hat. / 2 P.

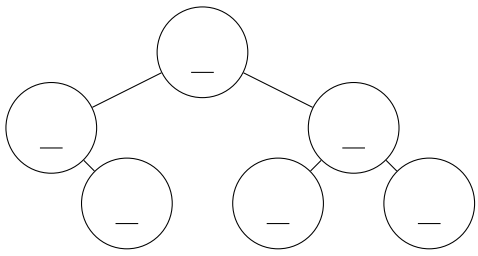
(d) Gegeben sei ein Feld A von n paarweise verschiedenen ganzen Zahlen. Die Zahlen in A sollen in einen *gewöhnlichen* binären Suchbaum T eingefügt werden, der anfangs leer ist. Durch geschickte Wahl der Reihenfolge, in der die Zahlen eingefügt werden, soll garantiert sein, dass T *zum Schluss* balanciert ist (also die Höhe von T in $O(\log n)$ ist). Skizzieren Sie in Worten einen Algorithmus, der die Zahlen in einer Reihenfolge einfügt, die dies gewährleistet. / 5 P.

Aufgabe 5

(a) Welcher der beiden unten stehenden Bäume stellt einen Heap dar? Markieren Sie ihn eindeutig. Tragen Sie in den Heap-Baum sechs unterschiedliche Zahlen ein, sodass ein gültiger Max-Heap entsteht. Füllen Sie den anderen Baum so mit sechs unterschiedlichen Buchstaben aus, dass er ein gültiger binärer Suchbaum ist. Für zwei Buchstaben α und β gilt $\alpha < \beta$ falls α im Alphabet vor β steht. / 3 P.

(b) Zeichnen Sie in den binären Suchbaum die Nil-Blätter ein und färben Sie ihn so, dass er ein gültiger Rot-Schwarz-Baum ist. Markieren Sie rote Knoten mit "r" und schwarze Knoten mit "s". / 3 P.





Aufgabe 6

Sie möchten n selbst bemalte Ostereier verkaufen. Sie erhalten m Angebote der Form

„Ich hätte gerne k_i Ostereier und bin bereit, für jedes Ei c_i Cent zu bezahlen “,

für $i = 1, \ldots, m$. Die Angebote sind nach Preisen absteigend sortiert.

Die Eingabe bestehe aus n, m und zwei Feldern K und C, wobei für $i = 1, \ldots, m$ gelte: $K[i] = k_i \geq 1$ und $C[i] = c_i$.

Sie möchten den maximalen Gewinn (in Cent) ermitteln, den Sie erzielen können.

(a) Zeigen Sie, dass der Greedy-Algorithmus nicht immer den maximalen Gewinn berechnet: / 4 P.

MaxGewinnGreedy(n, m, int[] K, int [] C)

```
maxGewinn = 0
i = 1
while n > 0 and i ≤ m and K[i] ≤ n do
    maxGewinn = maxGewinn + K[i] · C[i]
    n = n − K[i]
    i = i + 1
return maxGewinn
```

Eingabe: $n =$ _____, $m =$ _____, $K = \langle$_____$\rangle$, $C = \langle$_____$\rangle$

Vom Algorithmus ermittelter größter Gewinn: _____ Cent

Korrekter maximaler Gewinn: _____ Cent

(b) Seien A^* die Angebote, die Ihren Gewinn g^* maximieren. Beschreiben Sie, wie A^* aus kleineren Instanzen desselben Problems hervorgeht. / 6 P.

Im folgenden sei $c(i, j)$ der maximale Gewinn, der mit i Eiern und den ersten j Angeboten erreicht werden kann. Es gelte $0 \leq i \leq n$ und $0 \leq j \leq m$. Füllen Sie die Lücken aus: Der Wert $c($_____, _____$)$ beschreibt den gesuchten maximalen Gewinn für alle n Eier. Für alle anderen Felder gilt:

$$c(i,j) = \begin{cases} \text{\_\_\_\_\_} & \text{falls } i \leq 0 \\ \text{\_\_\_\_\_} & \text{falls } j \leq 0 \\ c(i, j-1) & \text{falls } k_i > i \\ \max\{\text{\_\_\_\_\_}, \text{\_\_\_\_\_} + k_i \cdot c_i\} & \text{sonst} \end{cases}$$

Aufgabe 7

/ 5 Punkte

Ein Vertex-Cover eines Graphen $G = (V, E)$ ist eine Knotenmenge $U \subseteq V$, sodass für alle Kanten $\{u, v\} \in E$ mindestens einer der Knoten u, v in U ist. Ein Matching eines Graphen ist eine Kantenmenge $F \subseteq E$ sodass sich keine zwei Kanten in F einen Knoten teilen. Ein Matching ist nicht-erweiterbar, falls keine weitere Kante zu F hinzugefügt werden kann.

Sei F ein nicht-erweiterbares Matching eines Graphen G. Beschreiben Sie, wie Sie aus F ein Vertex-Cover U für G erhalten können und begründen Sie die Korrektheit. Sie erhalten volle Punktzahl, falls für Ihr U gilt: $|U| \leq 2|F|$.

Aufgabe 8

Betrachten Sie untenstehenden Algorithmus. Er erhält als Eingabe ein aufsteigend sortiertes Feld von ganzen Zahlen. Er ermittelt, wie viele verschiedene Zahlen im Array stehen.

```
_____ uniqueNumbers(int[] A)
    int counter = _____
    int index = _____
    while _____ do
    |   key = A[index]
    |   index = index + 1
    |   counter = counter + 1
    |   while _____ and _____ do
    |   |   index = index + 1
    |
    return _____
```

(a) Füllen Sie die Lücken, sodass der Algorithmus das Geforderte leistet. / 3 P.

(b) Beweisen Sie die Korrektheit des Algorithmus, indem Sie zunächst die Lücken der Schleifeninvariante ausfüllen und dann Initialisierung, Aufrechterhaltung und Terminierung bearbeiten. / 5 P.

Invariante: Vor der jeder Iteration gilt, dass _____ die Anzahl verschiedener Zahlen im Teilfeld A[_____] enthält und das Vorkommen der Zahl in A[_____] noch nicht gezählt wurde.

Initialisierung:

Aufrechterhaltung:

Terminierung:

(c) Welche Laufzeit hat uniqueNumbers(. . .)? Begründen Sie genau. / 5 P.

Aufgabe 9

Eine Datenstruktur IncDs verwaltet eine Menge von Zahlen. Intern werden die Zahlen in einer einfach verketteten Liste L gespeichert, wobei neue Zahlen immer am Anfang der Liste angefügt werden. Die Datenstruktur verfügt über drei Methoden:

Insert(k)	Fügt eine neue Zahl in die Datenstruktur ein.
IncrementEverything()	Addiert zu jeder Zahl in der Datenstruktur den Wert 1.
copyToArray(int[ ] A)	Kopiert alle Zahlen in der Datenstruktur in das Array A. Falls A zu klein ist, soll A mit sovielen Zahlen aus der DS wie möglich gefüllt werden.

(a) Welche Laufzeit hat die Methode Insert(k), wenn neue Elemente immer am Beginn der Liste gespeichert werden und momentan n Elemente in der Liste sind? / 1 P.

$$T_{\text{Insert}}(n) \in \_\_\_\_\_$$

(b) Implementieren Sie die Methoden copyToArray(int[ ] A) und IncrementEverything(). Beachten Sie die geforderte Laufzeit. Verwenden Sie keine Leerlaufschleifen um die Laufzeit künstlich zu verlängern. / 9 P.

copyToArray(int[ ] A) Worst-Case-Laufzeit: $\Theta\left(\text{n}\right)$

IncrementEverything(k) Worst-Case-Laufzeit: $\Theta\Big(\mathsf{n}\Big)$