Lehrstuhl für Informatik I
Effiziente Algorithmen und wissensbasierte Systeme
Universität Würzburg

Würzburg, den 8. Februar 2016

Prof. Dr. Alexander Wolff
M.Sc. Krzysztof Fleszar

Vorlesung Algorithmen und Datenstrukturen WS 2015/16

Klausur - Gruppe A

Vorname: _____ Name: _____

Matrikelnummer: _____

Aufgabe	1	2	3	4	5	6	7	8	**Gesamt**
mögliche Punkte	3	4	4	5	3	4	4	6	33
erreichte Punkte									

Zusatzaufgabe: Aufgabe 8(d)–(e) mit insgesamt 8 Zusatzpunkten.

Bearbeitungszeit: 90 Minuten. Mit 16 der 33 (+8) Punkte haben Sie bestanden.

Aufgabe 1

/ 3 Punkte

Zeigen Sie mittels vollständiger Induktion, dass für alle $k \in \mathbb{N}_{\geq 0}$ gilt:

$$2 \sum_{i=0}^{k} 3^i = 3^{k+1} - 1$$

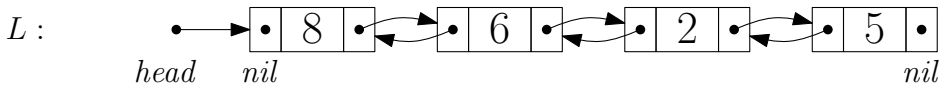
Aufgabe 2

/ 4 Punkte

Gegeben sei folgende Methode:

```
SomeMethod(List L, int k)
    x = L.head
    while x ≠ nil and k > 0 do
        x = x.next
        if x ≠ nil then
            x.prev = L.head.prev
        L.head = x
        k = k − 1
```

SomeMethod wird auf die unten abgebildete Liste L mit dem Parameter $k = 2$ angewandt. Zeichnen Sie die Liste nach jedem Schleifendurchlauf nochmal hin. (Wir nehmen an, dass die Methode Zugriff auf die Attribute der Listen und der Listenelemente hat.)



Was macht die Methode mit L in Abhängigkeit von k?

Aufgabe 3

/ 4 Punkte

Gegeben sei ein Feld $A = \langle s, s, s, \ldots, s, w, w, w, \ldots, w \rangle$ der Länge n. Geben Sie in gut kommentiertem Pseudocode einen Algorithmus an, der die Länge der s-Sequenz in Zeit $O(\log n)$ bestimmt.

int SLength(int[] A)

Aufgabe 4

/ 5 Punkte

Sie möchten eine Wüste möglichst schnell überqueren. Sie haben eine anfangs volle Wasserflasche dabei, die für k Kilometer ausreicht. Auf Ihrem Weg passieren Sie n Oasen, wo Sie Ihre Flasche wieder auffüllen können. Da das Auffüllen jedoch lange dauert (Sie müssen erst das Wasser aus einem tiefen Brunnen holen), möchten Sie möglichst wenige Zwischenstopps vornehmen.

Geben Sie in gut kommentiertem Pseudocode einen Algorithmus CountMinStops an, der die minimale Anzahl der Zwischenstopps zurückgibt. Als Eingabe erhält der Algorithmus den Wert k sowie ein Feld A der Länge n, wobei $A[i]$ für jedes $i = 1, \ldots, n$ den Abstand (in km) der i-ten Oase zum Startpunkt enthält. Die n-te Oase ist das Ziel. Falls das Ziel nicht erreicht werden kann, geben Sie ∞ zurück.

int CountMinStops(int[] A, int k)

Um was für einen Typ von Algorithmus handelt es sich?

Analysieren Sie die Worst-Case-Laufzeit Ihres Algorithmus in Abhängigkeit von der Anzahl n der Oasen.

Beweisen Sie die Korrektheit Ihres Algorithmus: Warum liefert er die kleinstmögliche Anzahl von Zwischenstopps?

Aufgabe 5

/ 3 Punkte

Beim doppelten Hashing ist es wichtig, dass die Hashfunktionen zueinander passen. Argumentieren Sie bei den folgenden Hashfunktionen, warum sie als Funktion $h_1(k)$ für das doppelte Hashing in einer Tabelle $T[0..31]$ mit

$$h(k) = (h_0(k) + ih_1(k)) \bmod 32$$

und

$$h_0(k) = (3k + 5) \bmod 32$$

nicht geeignet sind.

Geben Sie hierzu jeweils auch ein k an, das die Problematik aufzeigt.

(a) $h_1(k) = 30 - 2 \cdot (k \bmod 15)$

(b) $h_1(k) = (k + 3) \bmod 10$

(c) $h_1(k) = (29 - 2k) \bmod 19$

Aufgabe 6

/ 4 Punkte

Wählen Sie aus der folgenden Menge von Funktionen eine maximale Teilmenge aus, deren Summe in $O(n^{3/2} \log n)$ liegt. Kreisen Sie die gewählten Funktionen ein!

$\{$ $7n^2$, $\frac{1}{2}n^3$, $n^2 - n^{1/2}$, $n^{1.5} \log_2(100n)$,

$(n+1)^{3/2} \log_2 n$, $n^{1.1}$, 2^n, $100000^{\log_2 n}$,

$1000^{100^{10}} n \cdot (\log_2 n)^{100^{10}}$, $n\sqrt{n}$, $n^{\log_2 n}$, $2^{\log_2 n}$,

$(\sqrt{8})^{\log_2 n}$, $2^{3 \log_2 \sqrt{n}} \cdot \sqrt{n}$, $(\sqrt{2})^{\log_2(n^3)} \log(n^3)$, $3^{3/2 \cdot \log_2 n}$ $\}$

Aufgabe 7

/ 4 Punkte

Gesucht ist ein Algorithmus, der ermittelt, wieviele Züge man mindestens benötigt, um auf einem Schachbrett mit $n \times n$ Feldern einen Springer von Feld (x_1, y_1) zu Feld (x_2, y_2) zu bewegen.

Die Schachregeln legen für jedes Feld (x, y) eine Menge $Z_{(x,y)}$ von bis zu acht Feldern fest. In einem Zug muss der Springer von seinem Feld (x, y) auf eines der Felder in $Z_{(x,y)}$ bewegt werden.

Geben Sie *in Worten* einen effizienten Algorithmus an, der die erforderliche Anzahl an Zügen in $O(n^2)$ Zeit berechnet. Ist das Zielfeld nicht erreichbar, so soll ∞ zurückgegeben werden.

Sie dürfen Methoden aus der Vorlesung verwenden.

Aufgabe 8

Sei A ein Feld der Länge n. Eine *Inversion* ist ein Paar (i, j) mit der Eigenschaft, dass $1 \leq i < j \leq n$ und $A[i] > A[j]$.

(a) Geben Sie alle Inversionen im Feld $\langle 2, 3, 8, 6, 1 \rangle$ an. / 2 P.

(b) Geben Sie ein Feld der Länge n mit Wertebereich $\{1, \ldots, n\}$ an, bei dem die Anzahl der Inversionen maximal ist (über alle Felder der Länge n mit gleichem Wertebereich). / 2 P.

Wie viele Inversionen sind es?

(c) Erklären Sie, wie die Laufzeit $T(A)$ von InsertionSort von der Anzahl $I(A)$ der Inversionen im Feld A abhängt, wobei $n = A.length$. / 2 P.

$T(A) =$

(d) / 4 Zusatzpunkte

Die Datenstruktur Rot-Schwarz-Baum soll um eine Methode CountSmaller(k) augmentiert werden, die die *Anzahl* der Elemente im Baum bestimmt, die kleiner als die Eingabe k sind. Die Laufzeit von CountSmaller soll in $O(\log m)$ sein, wobei m die aktuelle Anzahl der Elemente im Baum ist.

Hierzu wird für jeden Knoten v das Attribut $v.size$ eingeführt, das die Anzahl der Elemente im Teilbaum mit Wurzel v speichert.

Begründen Sie, warum der Wert von $size$ in allen Knoten aufrechterhalten werden kann, ohne die asymptotische Laufzeit der übrigen Rot-Schwarz-Baum-Methoden zu ändern:

Vervollständigen Sie die angegebene rekursive Methode CountSmaller.

```
int  CountSmaller(int k, Node x = root)
    if x == nil then
        return _____
    if k < x.key then
        return CountSmaller(k, _____)
    else if k > x.key then
        return x._____ + 1 + CountSmaller(k, _____)
    else
        if x.left == nil then
            return _____
        else return x._____
```

(e) / 4 Zusatzpunkte

Geben Sie in gut kommentiertem Pseudocode eine Methode CountInversions an, die die Anzahl der Inversionen eines Feldes A der Länge n in Zeit $O(n \log n)$ bestimmt. Verwenden Sie hierzu CountSmaller. Langsamere Lösungen geben hier weniger Punkte.

```
int  CountInversions(int[] A)
```