Lehrstuhl für Informatik I
Algorithmen und Komplexität
Universität Würzburg

Würzburg, den 23. November 2020

Prof. Dr. Alexander Wolff
Felix Klesen, M. Sc.

3. Übungsblatt zur Vorlesung Algorithmen und Datenstrukturen (Winter 2020/21)

Aufgabe 1 – Hirsch-Index

Professor Munroe ist sehr stolz auf seine Veröffentlichungen, die er aufsteigend nach Erscheinungsdatum nummeriert:

1. How much Force power can Yoda output?
2. How fast can you hit a speed bump while driving and live?
3. At what speed would you have to drive for rain to shatter your windshield?
4. Can you use a magnifying glass and moonlight to light a fire?

Ein Kollege zweifelt aber daran, dass die Veröffentlichungen von Prof. Munroe relevant sind, und führt daher eine Liste, in der er fortlaufend notiert, wenn eine Veröffentlichung von Prof. Munroe in einer anderen Veröffentlichung zitiert wird. Er vermerkt dabei nur die Nummer der Veröffentlichung aus Prof. Munroes Liste. Aktuell kommt er zu diesem Ergebnis: 1,3,4,4,1,4,2. Daraus berechnet er den *Hirsch-Index* von Prof. Munroe. Das ist die größte natürliche Zahl k, für die gilt, dass Prof. Munroe mindestens k Veröffentlichungen hat, die mindestens k-mal zitiert wurden. Der Hirsch-Index von Prof. Munroe ist momentan also 2, denn nur Veröffentlichungen 1 und 4 wurden mindestens zweimal zitiert. Die Anzahl aller Zitierungen bezeichnen wir mit n.

a) Geben Sie textuell und in Pseudocode einen Algorithmus an, der den Hirsch-Index eines Wissenschaftlers berechnet. Der Algorithmus soll als Eingabe nur eine Folge von Zahlen erfordern, welche (wie im obigen Beispiel) die Zitierungen der Veröffentlichungen des Wissenschaftlers angibt.

   Die asymptotische Worst-Case-Laufzeit des Algorithmus soll $O(n \log n)$ sein. Sie dürfen Algorithmen aus der Vorlesung als Unterroutinen verwenden. **4 Punkte**

PABS b) Implementieren Sie in der Klasse `HirschIndex` im Paket `hIndex` die Methode

   - `public static int hirschIndex(int[] citations)`

   in Java. Diese soll den Hirsch-Index aus einer Folge von Zitierungen berechnen.

   Die asymptotische Worst-Case-Laufzeit Ihrer Implementierung soll $O(n^2)$ sein. Zum Sortieren eines Feldes `int[] a` dürfen Sie `java.util.Arrays.sort(a);` verwenden.

   4 Punkte

Aufgabe 2 – Heaps

a) Wo kann sich das kleinste Element in einem MaxHeap befinden, wenn alle Elemente verschieden sind? **1 Punkt**

b) Ist ein Array, das absteigend sortiert ist, ein MaxHeap? **1 Punkt**

c) Was bewirkt $\texttt{MaxHeapify}(A, i)$, wenn $A[i]$ größer als seine Kinder ist? **1 Punkt**

d) Zeigen Sie, dass die maximale Laufzeit von MaxHeapify über alle Felder der Länge n in $\Omega(\log n)$ ist. Dazu genügt es (für beliebig große $n \in \mathbb{N}$) ein konkretes Feld A_n der Größe n und ein Element $1 \leq i \leq n$ anzugeben, so dass $\texttt{MaxHeapify}(A_n, i)$ tatsächlich $\Omega(\log n)$ Zeit benötigt. **1 Punkt**

Aufgabe 3 – Rekursionsgleichung aufstellen

SomeAlgo(int[ ] A, int $\ell = 1$, int $r = A.length$)
if $\ell == r$ **then return** $A[\ell]$
$g \leftarrow r - \ell + 1$
let $B[1..3]$ be new array of int
$B[1] \leftarrow$ SomeAlgo($A,\ \ell,\ \ell + \lceil g/2 \rceil - 1$)
$B[2] \leftarrow$ SomeAlgo($A,\ \ell + \lfloor g/4 \rfloor,\ \ell + \lfloor g/4 \rfloor + \lceil g/2 \rceil - 1$)
$B[3] \leftarrow$ SomeAlgo($A,\ \ell + \lfloor g/2 \rfloor,\ r$)
InsertionSort(B)
return $B[1]$

a) Was gibt SomeAlgo zurück? **1 Punkt**

b) Stellen Sie eine Rekursionsgleichung für die asymptotische Laufzeit $T(n)$ von SomeAlgo(A) auf (mit $n := A.length$). Denken Sie an den Basisfall. **2 Punkte**

Aufgabe 4 – Rekursionsgleichungen lösen

Sei $T\colon \mathbb{N} \to \mathbb{R}$ eine Funktion, so dass $T(n)$ für $n = 1$ einen konstanten Wert annimmt und für alle anderen $n \in \mathbb{N}$ durch eine der folgenden Rekursionsgleichungen definiert ist. Geben Sie für jedes der folgenden T eine Funktion g an, so dass $T \in \Theta(g)$. Eine Begründung Ihrer Lösung ist zwingend erforderlich.

Drei Teilaufgaben sind mit der Meistermethode lösbar (geben Sie den jeweiligen Fall an), eine jedoch nicht (diese wird mit 2 Punkten bewertet). Nutzen Sie für diese die Rekursionsbaummethode. Nehmen Sie an, dass $n = 2^m$, für ein $m \in \mathbb{N}$. **5 Punkte**

a) $T(n) = 3T(\lfloor n/9 \rfloor) + \sqrt{n}$
b) $T(n) = 4T(\lfloor n/2 \rfloor) + n^3$
c) $T(n) = 4T(\lfloor n/2 \rfloor) + n^2 \log_2 n$
d) $T(n) = 5T(\lfloor n/2 \rfloor) + n^2$

Bitte geben Sie Ihre Lösungen bis **Montag, 30. November 2020, 16:00 Uhr** einmal pro Gruppe über Wuecampus als pdf-Datei ab. Vermerken Sie dabei stets die Namen und Übungsgruppen aller BearbeiterInnen auf der Abgabe.

Grundsätzlich sind stets alle Ihrer Aussagen zu begründen und Ihr Pseudocode ist stets zu kommentieren.

Die Lösungen zu den mit PABS gekennzeichneten Aufgaben, geben Sie bitte nur über das PABS-System ab. Vermerken Sie auf Ihrem Übungsblatt, in welchem Repository (*sXXXXXX*-Nummer) die Abgabe zu finden ist. Geben Sie Ihre Namen hier als Kommentare in den Quelltextdateien an.