Lehrstuhl für Informatik I
Algorithmen und Komplexität
Universität Würzburg

Würzburg, den 23. Dezember 2021

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

8. Übungsblatt zur Vorlesung Algorithmen und Datenstrukturen (Winter 2021/22)

Aufgabe 1 – Spezialsuche

Gegeben sei ein Feld $A[1..k]$ mit ganzen Zahlen, für die $A[1] < A[2] < \cdots < A[k]$ gilt. Geben Sie in Worten und im Pseudocode einen Algorithmus an, der ermittelt, ob es eine Zahl $j \in \{1, \ldots, k\}$ mit $A[j] = j$ gibt. Die Worst-Case Laufzeit Ihres Algorithmus soll $\Theta(\log k)$ sein. **5 Punkte**

Hinweis: Finden Sie ein geeignetes, leicht zu berechnendes Kriterium, um damit den gesuchten Index zu suchen. Bedenken Sie auch die besondere Struktur von A.

Aufgabe 2 – Schlange aus zwei Stapeln analysieren

Zwei Stapel S_1, S_2 können wie folgt eine Schlange Q simulieren:

- Enqueue(x): führe S_1.Push(x) aus – lege also x auf den ersten Stapel.
- Dequeue(): Ist S_2 leer, führe solange S_2.Push(S_1.Pop()) aus, bis S_1 leer ist – verschiebe die Elemente der Reihe nach von S_1 auf S_2. Danach führe S_2.Pop() aus.
- Empty(): gibt true zurück, falls S_1 und S_2 leer sind, sonst false.

Betrachten Sie nun eine zufällige Folge der Länge n bestehend aus Enqueue-, Dequeue- und Empty-Operationen auf Q.

a) Argumentieren Sie, warum die *Worst-Case*-Laufzeit einer einzelnen Dequeue-Operation auf Q in $\Theta(n)$ liegt. **3 Punkte**

b) Zeigen Sie mit amortisierter Analyse: die Gesamtlaufzeit für jede Folge liegt ebenfalls in $\Theta(n)$. Eine einzelne Dequeue-Operation benötigt amortisiert also nur konstante Laufzeit. Wieso ist dies kein Widerspruch zu Teilaufgabe a)? **3 Punkte**

Aufgabe 3 – MultiPop

Gegeben sei die folgende Funktion, welche auf einem Stapel S arbeitet:

```
MultiPoP(k)
while S nicht leer and k > 0 do
    S.Pop()
    k = k − 1
```

Wir betrachten nun eine Sequenz von n Stapel-Operationen (Push, Pop und MultiPop).

a) Zeigen Sie, dass die *Worst-Case*-Laufzeit einer einzelnen MultiPop-Operation auf S in $\Theta(n)$ liegt. **2 Punkte**

b) Zeigen Sie nun mit Hilfe von amortisierter Analyse, dass die Gesamtlaufzeit einer solchen Sequenz ebenfalls in $\Theta(n)$ liegt. Wieso ist dies kein Widerspruch zu Teilaufgabe a)? **2 Punkte**

Aufgabe 4 – Listen-Augmentierung

Die Datenstruktur *doppelt verkettete Liste* soll um eine Methode Invert erweitert werden, nach deren Ausführung sich die Methoden der Liste so verhalten, als ob die Listenelemente in umgekehrter Reihenfolge in der Liste stehen würden, beispielsweise durchsucht Search die Liste nun von hinten nach vorne. Nach erneutem Aufruf von Invert soll die Reihenfolge wieder umgekehrt werden.

Beispiel: Gegeben sei die Liste $A = \langle 1, 2, 3\rangle$. Führt man die Operation A.Insert(4) aus, so ergibt sich die Liste $\langle 4, 1, 2, 3\rangle$. Nachdem man nun nacheinander die Operationen

A.Invert, A.Insert(5), A.Invert

ausgeführt hat, ergibt sich die Liste $\langle 4, 1, 2, 3, 5\rangle$.

Skizzieren Sie in Worten, wie man die aus der Vorlesung bekannte Liste augmentieren kann, so dass Invert nur $O(1)$ Zeit benötigt und sich die asymptotische Worst-Case-Laufzeiten von Search, Insert und Delete nicht ändern. Erklären Sie, wie Sie Invert implementieren und wie Sie die genannten Operationen der Liste anpassen um die Anforderungen zu erfüllen. **5 Punkte**

Bitte geben Sie Ihre Lösungen bis **Donnerstag, 13. Januar 2022, 14:00 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.