Lehrstuhl für Informatik I
Algorithmen und Komplexität
Universität Würzburg

Würzburg, den 19. Dezember 2024

Prof. Dr. Alexander Wolff
Samuel Wolf, M. Sc.

8. Übungsblatt zur Vorlesung Algorithmen und Datenstrukturen (Winter 2024/25)

Aufgabe 1 – Binär hochzählen

Die Datenstruktur D enthält eine einzige natürliche Zahl Z in Binärdarstellung (zu Beginn ist $Z = 0$) und stellt lediglich die Methode Increment zur Verfügung, die Z um Eins erhöht. Die Laufzeit der Methode entspricht dabei der Anzahl der Bits, die sich in Z durch die Erhöhung um Eins ändern.

Zeigen Sie, dass die *amortisierte* Laufzeit von Increment in $O(1)$ ist! **4 Punkte**

Aufgabe 2 – Rot-Schwarz-Baum augmentieren

Ein Rot-Schwarz-Baum zur Verwaltung einer dynamischen Menge verschiedener ganzer Zahlen soll so augmentiert werden, dass man zu jeder Zeit bestimmen kann, für welche zwei Zahlen i, j der Menge mit $i < j$ die Differenz $j - i$ am kleinsten ist.

a) Geben Sie die Methode MinGap in Pseudocode an, die das gesuchte Zahlenpaar in konstanter Zeit liefern soll.

   Benennen Sie die Extrainformation, die dafür zu speichern ist, und geben Sie an, wie Sie diese in den Methoden Insert, Delete und Search aufrechterhalten können, ohne deren asymptotische Worst-Case-Laufzeiten zu verschlechtern. **4 Punkte**

b) Können Sie das Problem auch mit konstantem Speicher für Extrainformation lösen, wenn Sie auf die Methode Delete verzichten (also nur eine *halbdynamische* Menge verwalten)? **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, 9. Januar 2025, 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.