Westfälische Wilhelms-Universität Münster
Übungen zur Vorlesung „Datenstrukturen und Algorithmen“ im SoSe 2017

Prof. Dr. Klaus Hinrichs
Aaron Scherzinger

Blatt 7

Abgabe im Learnweb bis zum 22.06.2017 um 10 Uhr

Aufgabe 22: (4+4+4+4+4=20 Punkte) Betrachten Sie die in einem Array `int[] a` gespeicherte Folge `a=[18,19,33,10,24,23,41,36,35]`. Sortieren Sie „von Hand“ diese Folge aufsteigend mit den folgenden Verfahren:

(a) *insertion sort*: Geben Sie das Array zu Beginn sowie nach jedem Einfügen an.

(b) *bubble sort*: Geben Sie das Array zu Beginn sowie nach jedem vollständigen Durchlauf der inneren Schleife an.

(c) *selection sort*: Geben Sie das Array zu Beginn sowie nach jedem Einfügen an.

(d) *merge sort*: Geben Sie bei jedem Rekursionsschritt die zu verschmelzenden Teilfolgen sowie das Resultat der Verschmelzung an.

(e) *quick sort*: Geben Sie bei jedem Rekursionsschritt die zu sortierenden Teilfolgen sowie das Pivot-Element an. (Das Pivot-Element soll wie in der Beispiel-Implementation aus der Vorlesung bestimmt werden.)

Hinweis: Verwenden Sie keine Implementierung der Sortieralgorithmen zur Erstellung Ihrer Lösung.

Aufgabe 23: (8 Punkte) Jedes Jahr werden die Erstsemester von Hogwarts, der Schule für Hexerei und Zauberei, auf die vier Häuser Gryffindor, Hufflepuff, Ravenclaw und Slytherin verteilt. Diese Zuteilung nimmt der Sprechende Hut (engl.: *sorting hat*) vor: Jeder Schüler setzt den Sprechenden Hut auf, und dieser verkündet dann das Haus, in dem der Schüler wohnen wird.

In diesem Jahr haben jedoch die Brüder Fred und George Weasley durch ein fehlgeschlagenes Experiment zur Erzeugung von „Bohnen aller Geschmacksrichtungen“ die gesamte große Halle, in der die Verteilung üblicherweise stattfindet, mit klebrigem Schmalz überzogen. Aus diesem Grund findet die Verteilung in den Gängen vor der Halle statt. Diese Gänge sind jedoch leider so eng, dass die Schüler in einer langen Reihe stehen müssen und dass sich nicht mehr als ein Schüler gleichzeitig bewegen kann.

Nachdem jeder Schüler erfahren hat, in welchem Haus er wohnen wird, versucht der Sprechende Hut, die Schüler nach Häusern zu gruppieren. Der Sprechende Hut hieße nicht „*sorting hat*“, wenn er nicht die Vorlesung „Informatik II“ gehört hätte, und daher ist ihm sofort klar, wie er die Schüler möglichst schnell gruppieren kann.

Welches Verfahren benutzt der Sprechende Hut? Bestimmen Sie die asymptotische Laufzeitkomplexität dieses Verfahrens.

(Aufgabenstellung für Muggels: Geben Sie einen möglichst effizienten Algorithmus an, der eine Menge von n Elementen nach ihren Markierungen gruppiert, wobei die Anzahl der unterschiedlichen Markierungen konstant ist. Ihr Algorithmus darf nur $\mathcal{O}(1)$ zusätzlichen Speicher benötigen. Bestimmen Sie die asymptotische Laufzeitkomplexität dieses Verfahrens.)

Diese Aufgabenstellung geht zurück auf Jeff Erickson, University of Illinois at Urbana-Champaign, USA.

Aufgabe 24: (5 Punkte) Betrachten Sie zwei Arrays `int[] a, int[] b`, die jeweils eine ungeordnete Folge von n bzw. m `int`-Werten enthalten. Ein *Duplikat* ist ein Wert, der sowohl im Array `a` als auch im Array `b` gespeichert ist. Geben Sie einen möglichst effizienten Algorithmus an, der bestimmt, wie viele Duplikate in `a` und `b` enthalten sind. Dabei darf nicht ausgenutzt werden, dass der Wertebereich von `int` Zahlen endlich ist. Eine Implementation Ihres Verfahrens ist nicht erforderlich.

Beispiel: Die Arrays `a=[6,4,2,2]` und `b=[2,1,2,1,4]` enthalten zwei Duplikate, nämlich die Werte 2 und 4.

Aufgabe 25: (20 Punkte) Implementieren Sie den in der Vorlesung besprochenen Quicksort Algorithmus in `JAVA` iterativ **ohne Verwendung von Rekursion**. Verwenden Sie einen Stack für die notwendige Buchhaltung, d.h. zum Abspeichern der Grenzen der noch zu sortierenden Teilarrays. Überlegen Sie sich eine Strategie, um die Größe des Stacks minimal zu halten. Geben Sie eine möglichst minimale obere Schranke für die Größe Ihres Stacks in Anhängigkeit von der Anzahl n der zu sortierenden Elemente an. Ihr Implementierung soll für beliebige Datenobjekte verwendbar sein, welche das in der Vorlesung vorgestellte Interface `Comparable` implementieren.

Testen Sie ihre Implementierung mit mindestens drei Arrays, von denen eines aufsteigend, eines absteigend und eines in zufälliger Reihenfolge vorliegt.