Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Thomas Tschager
Antonis Thomas

9. März 2016

Datenstrukturen & Algorithmen Blatt 3 FS 16

Aufgabe 3.1 *Vergleich von Sortieralgorithmen.*

Gegeben seien ein Array $A[1..n]$ und die folgenden Java-Implementationen der Sortieralgorithmen *Bubblesort*, *Sortieren durch Einfügen*, *Sortieren durch Auswahl* und *Quicksort*. Diese werden mit den Parametern $l = 1$ und $r = n$ aufgerufen, um A in aufsteigender Reihenfolge zu sortieren.

```
public void bubbleSort(int[] A, int l, int r) {
    for (int i=r; i>l; i--)
        for (int j=l; j<i; j++)
            if (A[j]>A[j+1])
                swap(A, j, j+1);
}

public void selectionSort(int[] A, int l, int r) {
    for (int i=l; i<r; i++) {
        int minJ = i;
        for (int j=i+1; j<=r; j++)
            if (A[j]<A[minJ])
                minJ = j;
        if (minJ != i)
            swap(A, i, minJ);
    }
}
```

```
public void insertionSort(int[] A, int l, int r) {
    for (int i=l; i<=r; i++)
        for (int j=i-1; j>=l && A[j]>A[j+1]; j--)
            swap(A, j, j+1);
}

public void quicksort(int[] A, int l, int r) {
    if (l<r) {
        int i=l+1, j=r;
        do {
            while (i<j && A[i]<=A[l]) i++;
            while (i<=j && A[j]>=A[l]) j--;
            if (i<j) swap(A, i, j);
        } while (i<j);
        swap(A, l, j);
        quicksort(A, l, j-1);
        quicksort(A, j+1, r);
    }
}
```

Der Aufruf von `swap(A, i, j)` vertauscht die Elemente $A[i]$ und $A[j]$. Geben Sie für jeden der obigen Algorithmen asymptotisch an, wie viele Vertauschungen und Vergleiche von Elementen aus A jeweils mindestens und höchstens ausgeführt werden. Geben Sie auch Folgen aus den Zahlen $1, 2, \ldots, n$ an, bei denen diese Fälle jeweils eintreten. Die Folgen sollen möglichst so angegeben werden, dass n beliebig gewählt werden kann. Beispielsweise kann die absteigend sortierte Folge durch $n, n-1, \ldots, 1$ beschrieben werden.

Aufgabe 3.2 *Algorithmenentwurf: Zahlensummen.*

Gegeben sei ein Array $A[1..n]$ aus natürlichen Zahlen. Geben Sie für die folgenden Probleme einen möglichst effizienten Algorithmus an und bestimmen Sie dessen worst-case-Laufzeit.

a) Gegeben sei eine natürliche Zahl z. Gibt es im Array A zwei (nicht notwendigerweise verschiedene) Elemente a und b mit $a + b = z$?

b) Angenommen, A sei bereits aufsteigend sortiert. Wie effizient kann das Problem aus a) nun gelöst werden? *Hinweis:* Es ist möglich, in diesem Fall eine bessere Laufzeit als im vorherigen Aufgabenteil zu erreichen.

c) Gibt es in A drei verschiedene Elemente a, b, c mit $a + b = c$?

Bitte wenden.

Aufgabe 3.3 *Median nach Blum (**Programmieraufgabe**).*

In dieser Aufgabe soll der *Algorithmus von Blum* zur Medianberechnung implementiert werden. Sei $x_1, ..., x_n$ eine Folge mit $n > 5$ Elementen (Duplikate sind erlaubt). Der Algorithmus führt die folgenden Schritte aus, um das k-kleinste Element zu finden.

1) Teile die Elemente der Reihe nach in $\lfloor \frac{n}{5} \rfloor$ Fünfergruppen sowie maximal eine Gruppe mit den verbleibenden n mod 5 Elementen auf. So sind die ersten fünf Elemente in der ersten Gruppe, usw.

2) Für jede der obigen Gruppen, berechne den Median der Gruppe. Für eine Gruppe mit 2 Elementen ist der Median das kleinere Element, für eine Gruppe mit 4 Elementen ist der Median das zweitkleinste Element.

3) Berechne den Median m der Mediane aus dem vorigen Schritt rekursiv. Dieses Element wird *Median der Mediane* genannt.

4) Führe den Aufteilungsschritt von Quickselect durch, um das Element m an die korrekte Position p_m in der sortierten Folge zu bringen. Dann gibt es $p_m - 1$ Elemente links von m (mit Wert höchstens m) und $n - p_m$ Elemente rechts von m (mit Wert mindestens m).

5) Falls $k = p_m$, dann wissen wir, dass das Pivotelement sich auf der gesuchten Position befindet, und wir liefern m zurück. Ist dagegen $k < p_m$, dann befindet sich das k-kleinste Element links vom m, und wir suchen rekursiv nach dem k-kleinsten Element unter den $p_m - 1$ Elementen links von m. Ansonsten ist $k > p_m$, und wir suchen rekursiv nach dem $(k - p_m)$-kleinsten Element unter den $n - p_m$ Elementen rechts vom m.

Unser Ziel ist die Berechnung des Medians, d.h. des $\lceil n/2 \rceil$-kleinsten Elements. Für die Sequenz $3, 4, 2, 6, 4, 7, 1$ ist der Median 4.

Eingabe Die erste Zeile der Eingabe enthält lediglich die Anzahl t der Testinstanzen. Es folgt eine Zeile für jede Testinstanz mit den Zahlen $n, x_1, ..., x_n$. Dabei ist $n \in \mathbb{N}$, $1 \leq n \leq 1000$, die Anzahl der folgenden Ganzzahlen, und $x_i \in \mathbb{Z}$, $-10^8 \leq x_i \leq 10^8$, die i-te Zahl der Eingabesequenz.

Ausgabe Für jede Testinstanz soll lediglich eine Zeile ausgegeben werden. Sie enthält die Mediane der Gruppen im ersten Schritt, den Median der Mediane im ersten Schritt sowie den Median der gesamten Sequenz.

Beispiel

Input:

```
3
5 1 2 3 4 5
6 7 4 3 2 1 2
13 7 3 5 1 9 8 11 21 4 10 2 6 9
```

Output:

```
3 3 3
3 2 2 2
5 10 6 6 7
```

Abgabe: Am Mittwoch, den 16. März 2016 in Ihrer Übungsgruppe.