Datenstrukturen und effiziente Algorithmen
Blatt 4

Markus Vieth, David Klopp, Christian Stricker

18. November 2015

Nr.1

Erklärung:

Eingabe: Integer k für Anzahl an Farben, Integer l für Anzahl an Muster. Ein 2x2n Array. In der ersten Zeile stehen die Farben in Integer codiert, in der zweiten Zeile analog für die Muster(bei 5 verschiedenen Muster/Farben gibt es die Integer 0-4).
Man iterriert über das Eingabearray und speichert die Anzahl der jeweiligen Farben in ein Array. Dann wird nochmal über das Array iterriert, wenn an der i-ten Stelle, die Socke mit falsche Farbe n steht, wird diese mit einer anderen Socke im n Bereich vertauscht.
Danach wird für jeden Farbbereich analog zum Farbensortieren nach Muster sortiert(Anzahl der Muster bestimmen, an die richtige Stelle tauschen).
Damit kommt eine Laufzeit von O(n+l+k)= O(n) zustande.
O(4*n), da man 2 mal über das ganze Array iteririert, um die Anzahl (an Farben/Muster) zu bestimmen
O(8*n), da höchstens n-mal vertauscht wird, 2n Zugriffe zum Zwischenspeichern, 4n Zugriffe zum Überschreiben und 2n Zugriffe für die Zwischenspeicherungen wieder zu speichern.
Analog beim Mustertauschen.

Code:

```java
public class Suche {

        static int[][] schrank = {{2,2,1,2,1,0,2,0,1,2,2,1,0,0,0,0},
                {2,1,0,2,0,0,1,0,0,1,0,2,0,2,0,2}};
        static int l = 3;         //Anzahl der Muster
        static int k = 3;         //Anzahl der Farben
        static int n = 8;         //Anzahl der Sockenpaare

        public static void main(String[] args){
                //Erstelle 3xk Array, erste Zeile sind die Farben, zweite Zeile
                   kommen die Anzahl der einzelnen Farben rein, dritte Zeile der
                   Index vom Pointer
                int[][] socke = new int[k][3];

                //Die Anzahl der Farben werden ermittelt
                for(int i = 0; i < 2*n; i++)
                        socke[1][schrank[0][i]]++;

                //Die Anfangsindezes der Farben für das Hauptarray werden
                   berechnet und abgespeichert
                for(int i = 1; i < k; i++)
                        socke[2][i] = socke[1][i-1]+socke[2][i-1];

                int i = 0;
                int farbe = 0;   //Die Farbe 0 wird ausgewählt
                int index = socke[2][farbe+1];  //Der Anfangsindex für die
                   nächste Farbe/Endindex für die erste Farbe wird ausgewählt
```

```
//Die Farben werden von 0-k in der richtigen Reihenfolge
   sortiert.
while(i < n){

        //Wenn die Socken für die eine FArbe schon eingeordnet
           sind, geht es wieter zur nächsten Farbe
        if(i >= index){
                farbe++;
                index = index + socke[1][farbe];
        }

        //Wenn die Farben schon an der richtigen stelle stehen,
           wird der Laufindex erhöht
        while(i < n && schrank[0][i] == farbe)
                i++;

        //Solange der Laufindex nicht bei der nächsten Farbe
           ankommt, wird getauscht. Die Farben, die nicht zur
           (wenn farbe = 0 ausgewählt ist) 0-ten Farbe gehören,
           werden in den richtige Farbenteil getauscht
        if(i < index){

                int farbeindex = schrank[0][i];
                                                   //
                swap(i,socke[2][farbeindex],schrank);
                      //Tauschen
                socke[2][farbeindex]++;
                                                            //Pointer wird
                   erhöht
        }
}

//Anzeige des Arrays
System.out.println("Nur␣nach␣Farben␣sortiert:␣");
for(int b = 0; b < 4*n; b++){
        if(b<2*n)System.out.print(schrank[0][b]+"␣");
        if(b == 2*n) System.out.println("");
        if(b>=2*n)System.out.print(schrank[1][b % (2*n)
           ]+"␣");
}
System.out.println("");

//Die Anfangsindezes der Farben für das Hauptarray werden
   berechnet und abgespeichert
for( i = 1; i < k; i++)
        socke[2][i] = socke[1][i-1]+socke[2][i-1];

//in dem Array wird in der ersten Spalte die Anzahl der
   jeweiligen Muster in einer Farbe gespeichert, in der zweiten
```

```
   Spalte die Anfangspointer im Gesamtarray
int[][] musteranzahl = new int[2][l];
int j = 0;       //äußerer Schleifenzähler

while(j < k){    //Wie viele Farben es gibt, so oft wird die
   Schleife durchlaufen

         int muster = 0;

         for(int k = 0; k < l; k++)
                 musteranzahl[0][k] = 0; //Die Werte für
                    die Anzahl der jeweiligen Muster werden für
                    die nächste Farbe zurückgesetzt

         //Bestimme die Anzahl der jeweiligen Muster
         for(int k = socke[2][j]; k < socke[2][j] +
            socke[1][j]; k++)
                 musteranzahl[0][schrank[1][k]]++;

         //Ende des Teilarrays des (ersten) Musters in der
            (ersten) Farbe
         index = musteranzahl[0][0] + socke[2][j];

         //Bestimme die Laufinizes der Einzelnen Muster in dem
            Teilarray von schrank
         for(i = 0; i < l-1; i ++ ){
                 musteranzahl[1][i+1] = musteranzahl[0][
                    i] + socke[2][j];

         }

         i = socke[2][j];          //Es wird einmal durch den
            Bereich des Arrays einer Farbe durchiteriert
         while(i < socke[2][j] + socke[1][j]){

                 if(i >= index){
                          muster++;
                          index = index + musteranzahl
                             [0][muster];
                 }

                 //Wenn das Muster schon an der richtigen
                    stelle stehen, wird der Laufindex erhöht
                 while(i < socke[2][j] + socke[1][j] &&
                    schrank[1][i] == muster)
                          i++;

                 //Solange der Laufindex nicht bei dem nächsten
                    Muster ankommt, wird getauscht. Das Muster,
```

```
                                        das nicht zur (wenn muster = 0 ausgewählt
                                        ist) 0-ten Muster gehören, werden in das
                                        richtige Musterteil getauscht
                                    if(i < index){

                                            int musterindex = schrank[1][i
                                               ];
                                            swap(i,musteranzahl[1][
                                               musterindex],schrank);
                                            musteranzahl[1][musterindex]++;
                                    }
                            }
                            j++;
                    }

                    //Anzeige des Arrays
                    System.out.println("");
                    System.out.println("nach Farben und Muster sortiert: ")
                       ;
                    for(int b = 0; b < 4*n; b++){
                            if(b<2*n)System.out.print(schrank[0][b]+" ");
                            if(b == 2*n) System.out.println("");
                            if(b>=2*n)System.out.print(schrank[1][b % (2*n)
                               ]+" ");
                    }
            }

            //Vertauschung der Elemente vom Indize i mit j
            private static void swap(int i, int j, int[][] schrank2) {
                    int a = schrank2[0][i]; //Werte zwischenspeichern
                    int b = schrank2[1][i];
                    schrank2[0][i] = schrank2[0][j];
                    schrank2[1][i] = schrank2[1][j];

                    schrank2[0][j] = a;
                    schrank2[1][j] = b;
            }
}
```

Nr.2

a)

Überlegung analog zur Vorlesung:

m=3

$$\exists\ \frac{2n}{6} \text{ Elemente } \leq p \Rightarrow \text{Es existieren maximal } \frac{4n}{6} \text{ Elemente } \geq p$$

$$\exists\ \frac{2n}{6} \text{ Elemente } \geq p \Rightarrow \text{Es existieren maximal } \frac{4n}{6} \text{ Elemente } \leq p$$

m=7

$$\exists\ \frac{4n}{14}\text{ Elemente }\ \leq p \Rightarrow \text{Es existieren maximal }\frac{10n}{14}\text{ Elemente }\ \geq p$$

$$\exists\ \frac{4n}{14}\text{ Elemente }\ \geq p \Rightarrow \text{Es existieren maximal }\frac{10n}{14}\text{ Elemente }\ \leq p$$

b)

Überlegung analog zur Vorlesung:

m=3

$$T(n) = T\left(\frac{n}{3}\right) + n + T\left(\frac{2n}{3}\right)$$

Akra-Bazzi:

$$g(n) = n,\ a_1 = a_2 = 1,\ b_1 = 3\ b_2 = \frac{3}{2}$$

$$1 = \left(\frac{1}{3}\right)^{\alpha} + \left(\frac{2}{3}\right)^{\alpha}$$

$$\Leftrightarrow \alpha = 1$$

$$T(n) = n\left(1 + \int_1^n \frac{x}{x^2}dx\right) = n\left(1 + \int_1^n \frac{1}{x}dx\right) = n\left(1 + [\ln(x)]_1^n\right) = n + n\ln(n) \in O(n\log(n)) \neq O(n)$$

Somit ist die Laufzeit für $m = 3$ nicht linear.

m=7

$$T(n) = T\left(\frac{n}{7}\right) + n + T\left(\frac{5n}{7}\right)$$

Zu zeigen:

$$\exists\ c > 0 : T(n) \leq c \cdot n$$

Beweis:

$$T(n) \leq c \cdot \frac{n}{7} + n + c \cdot \frac{5n}{7} \leq c \cdot n$$

$$c = 7 \Leftrightarrow n + n + 5n \leq 7n \Rightarrow \exists\ c > 0 : T(n) \leq c \cdot n \Rightarrow T(n) \in O(n)$$

Somit ist die Laufzeit für $m = 7$ linear.

Nr.3

a)

x	Ergebnisse:
0	18
1	15
2	14
3	13
4	14
5	15
6	16
7	17
8	18
9	19
10	20
11	21
12	22
13	23

Für $x = x_2 = 3$, also dem Median der Menge $\{x_1, x_2, x_3\}$, wird die Summe minimal.

b)

Beh:
$\sum_{i=1}^{n} |x_i - x|$ wird minimal für $x = x_{k+1}$, d.h x := Median $\{x_1, ..., x_n\}$

Bew:
Sei die Menge aller x_i sortiert. Teile die Summe in zwei gleich große Teilsummen, wobei Summe 1 kleiner als x_{k+1} und Summe 2 größer als x_{k+1} ist:

$$\sum_{i=1}^{2k+1} |x_i - x|$$

$$= \sum_{i=1}^{k} |x_i - x| + \sum_{i=k+2}^{2k+1} |x_i - x| + |x_{k+1} - x|$$

$$= kx - \sum_{i=1}^{k} x_i - kx + \sum_{i=k+2}^{2k+1} x_i + |x_{k+1} - x|$$

$$= \sum_{i=k+2}^{2k+1} x_i - \sum_{i=1}^{k} x_i + |x_{k+1} - x|$$

<u>Für $x = x_{k+1}$:</u>

$$= \sum_{i=k+2}^{2k+1} x_i - \sum_{i=1}^{k} x_i$$

<u>Für $x \neq x_{k+1}$:</u>
Der Term $|x_{k+1} - x|$ wird immer > 0 und somit größer als der Fall $x = x_k$, d.h $\sum_{i=1}^{n} |x_i - x|$ ist genau dann minimal, wenn $x = x_{k+1}$ ist.

c)

Beh:
$\sum_{i=1}^{n} (x_i - x)^2$ wird minimal für $x = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$, d.h x := arithmetisches Mittel von $\{x_1, ..., x_n\}$

Bew:

$$f(x) = \sum_{i=1}^{n} (x_i - x)^2$$

$$f'(x) = -\sum_{i=1}^{n} 2(x_i - x)$$

$$f''(x) = \sum_{i=1}^{n} 2$$

$\underline{f'(x) = 0:}$

$$-\sum_{i=1}^{n} 2(x_i - x) = 0$$

$$\Leftrightarrow -2nx + \sum_{i=1}^{n} 2x_i = 0$$

$$\Leftrightarrow 2\sum_{i=1}^{n} x_i = 2nx$$

$$\Leftrightarrow \frac{1}{n} \cdot \sum_{i=1}^{n} x_i = x$$

Da $f''(x)$ immer größer als Null ist, handelt es sich bei $x = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$ um ein Minimum.
q.e.d