Übungen zur Vorlesung
Einführung in die Algorithmik WS 15/16

Moritz Eissenhauer

Hasso-Plattner-Institut
Potsdam
11. November 2015

Lösung zu Übungsblatt 4

Lösung zu Aufgabe 1

Min(A, l, r)

Input: Array A, indezes l, r
Output: index eines lokalen Minimum
1: $i \leftarrow l + \lfloor \frac{r-l+1}{2} \rfloor$
2: $j \leftarrow i + 1$
3: **if** $j > r$ **then**
4:     **return** i
5: **else**
6:     **if** $A[i] < A[j]$ **then**
7:         **return** $Min(A, l, i)$
8:     **else**
9:         **return** $Min(A, j, r)$
10:     **end if**
11: **end if**

Zu begin wird Min(A, 0, n-1) aufgerufen, A ist 0-basiert. Auserdem nehme ich an, dass $n > 0$.

Korrektheit:

Zz. Korrektheit Min. Min(A, l, r) gibt Index eines lokalen Minimum für das Teilarray $A[l, ..., r]$ zurück.

Beweis. Über strukturelle Induktion:
IV: Zeige Korrektheit gilt für $r - l = 0$:
Wenn $r - l = 0$ dann wird i auf l gesetzt und j auf $l + 1$. Da $j > r$ gilt wird in Zeile 4 $i = l$ zurückgegeben. $A[l]$ hat keine Nachbarn in $A[l, ..., r]$, ist also ein lokales Minimum.
IA: Korrektheit gilt für $1 \leq r - l \leq k$
IS: Zeige Korrektheit gilt für $1 \leq r - l \leq k \Rightarrow$ Korrektheit gilt für $r - l \leq 2k$:
Sei $k < r - l \leq 2k$.
Zu begin wird i auf $\lfloor \frac{r-l+1}{2} \rfloor$ gesetzt und j auf $i + 1$. Da $r - l \leq 2k$ ist $i - l \leq k$ und $r - j \leq k$. Auserdem ist j auf jeden fall kleiner oder gleich r, weshalb wir in Zeile 5 in den else Block gehen. Es wird also entweder Min(A, l, i) zurückgegeben wenn $A[i] < A[j]$ (I) oder Min(A, j, r) wenn $A[i] > A[j]$ (II).
Fall I: $x = Min(A, l, i)$ ist wegen der Induktionsannahme der Index eines lokalen Minimum des Teilarrays $A[l, ..., i]$, da $i - l \leq k$. Falls nun $x \neq i$ ist, dann ist x offensichtlich auch ein lokales Minimum des Teilarrays $A[l, ..., r]$. Falls $x = i$, dann hat $A[x]$ keinen oder einen größeren linken Nachbar. Da $A[i] < A[j]$ ist auch der rechte Nachbar größer und i ist der Index eines lokalen Minimum in $A[l, ..., r]$.
Fall II verhält sich analog zu Fall I (da $r - j \leq k$) nur, dass der linke statt dem rechten Nachbarn betrachtet werden muss und andersherum.

Mit IV, IA und IS gilt also die Korrektheit für alle $r - l \geq 0$. Insbesonder ist der Algorithmus korrekt für $l = 0$ und $r = n - 1$ □

Laufzeit:

Für die Laufzeit gilt:
$T(1) = c_1$, da Zeile 1 bis 4 jeweils eine Konstante Laufzeit haben.
$T(n) = T(\lceil \frac{n}{2} \rceil) + c_2$, da das Teilarray jedes Mal halbiert wird.
Mit dem fall b) des Master-Theorem lässt sich die Laufzeit abschätzen als $T(n) \in \Theta(\log n)$

Lösung zu Aufgabe 2

Dom(A, n)

Input: Array A, länge des Arrays n
Output: index eines dominierenden Elements, falls vorhanden

```
1:  candidate ← 0
2:  initialisiere Array Deleted der länge n mit False
3:  for i:=0 to n-1 do
4:      if not eq(i, candidate) then
5:          Deleted[i] ← True
6:          Deleted[candidate] ← True
7:          while Deleted[candidate] ∧ candidate < n do
8:              candidate ← candidate + 1
9:          end while
10:     end if
11: end for
12: count ← 0
13: for i:=0 to n-1 do
14:     if eq(i, candidate) then
15:         count ← count + 1
16:     end if
17: end for
18: if count > n/2 then
19:     return candidate
20: end if
```

$eq(i, j)$ vergleicht A[i] mit A[j]. Zeilen 1 bis 11 nenne ich im Folgendem Teil 1, Zeilen 12 bis 20 nenne ich Teil 2.
Teil 1 findet den Index eines dominierenden Elementes falls es ein solches gibt, oder ein beliebigen anderen Index sonst. Der Kandidat wird in *candidate* gespeichert. Teil 2 prüft ob candidate tatsächlich ein dominierendes Element ist.
Korrektheit:
Ich werde zunächst die Korrektheit von Teil 1 zeigen.

Zz. Korrektheit Teil 1. Falls A ein dominierendes Element hat, dann ist am ende von Teil 1 der Index von einem der dominierenden Elemente in *candidate* gespeichert.

Beweis. Sei A ein beliebiges Array mit einem dominierenden Element d.
Behauptung: Sei $0 \leq i < j < n$ und A' das Array A ohne die elemente $A[i]$ und $A[j]$. Falls $eq(i, j) =$ False, dann ist d dominierendes Element in A'.

$$
\begin{aligned}
eq(i,j) = False &\Rightarrow (A[i] = d \wedge A[j] \neq d) \vee (A[j] = d \wedge A[i] \neq d) \vee (A[i] \neq d \wedge A[j] \neq d)\\
&\Rightarrow (|A'|_d \geq |A|_d - 1) \wedge (|A'| = |A| - 2)\\
&\text{Da } d \text{ dominierendes Element in A ist gilt } \tfrac{|A|_d}{|A|} > \tfrac{1}{2}\\
&\Rightarrow \frac{|A'|_d}{|A'|} \geq \frac{|A|_d - 1}{|A| - 2} \geq \frac{|A|_d}{|A|} > \frac{1}{2}\\
&\Rightarrow d \text{ ist dominierendes Element in} A'
\end{aligned} \tag{1}
$$

Das Array *Deleted* markiert die Indezes gelöschter Objekte. Zeile 7 bis 9 stellt sicher, dass nach Zeile 9 *candidate* auf dem kleinsten nicht markierten Index steht. Insbesondere gilt auch, dass nach Zeile 11, dass *candidate* auf dem ersten nicht markierten Index steht. Auserdem werden Indizees nur paarweise markiert (wenn Zeile 5 erreicht wird, dann wird auch Zeile 6 erreicht) und es werden nur Paare mit ungleichen werten markiert (Zeile 4).
Behauptung: Nach ablauf von Teil 1 gilt:

$$\neg\exists 0 \leq i < n : Deleted[i] \wedge A[i] \neq A[candidate]$$

Es muss $candidate < i$ gelten, da *candidate* immer der kleinste nicht markierte Index ist. Wenn es jetzt ein i wie oben gäbe, dann heißt das, dass beim Schleifendurchlauf i, $A[i] = A[candidate_i]$ galt, wobei $candidate_i$ der *candidate* beim Schleifendurchlauf i ist.

...unvollständig

□

Laufzeit:
Der **for** Block von Zeile 13 bis 17 läuft n mal und hat eine konstante laufzeit. Wodurch die laufzeit ein $\Theta(n)$ ist.
Der **for** Block in Zeile 3 bis 11 läuft n mal. Zeile 8 wird auch höchstens n mal ausgeführt, da dann $candidate \geq n$ ist und der **while** Loop nicht mehr betreten wird. Die Laufzeit des äuseren **for** Loops ein $\Theta(n)$.
Alle anderen Zeilen haben konstante Laufzeiten, also ist die Gesamtlaufzeit $T \in \Theta(n)$.

Lösung zu Aufgabe 3

a)
Sei n die Anzahl der Vergleiche.

Zz.. $(\forall A, l \leq m \leq r : n \leq r - l) \wedge (\exists A, l, m, r : n = r - l)$

Beweis: Zeige zuerst: $\forall A, l \leq m \leq r : n \leq r - l$
Sei A, l, m, r mit $l \leq m \leq r$ beliebig fest.
Initial ist $i = l$ und $j = m+1$. Auserdem gilt $l \leq m \leq r \Rightarrow (r-m)+(m-l) = (r-l)$. Nach jedem Vergleich wird i oder j inkrementiert. Wenn i $m - l + 1$ mal oder j $r - m$ mal inkrementiert wurde, dann bricht die while-Schleife in Zeile 3 ab (weil $l + (m - l + 1) = m + 1 > m$ und $m+1+(r-m) = r+1 > r$) und es finden keine Vergleiche mehr statt. Das Heißt im schlimmsten Fall wird i $m - l$ mal und j $r - m$ mal inkrementiert. Es gilt also $n \leq m - l + r - m = r - l$.
Zeige $\exists A, l, m, r : n = r - l$:
Wähle $l = 0, r = 1, m = 0, A = [1, 0]$. Dann wird $n = 1 = r - l$ Vergleich ausgeführt.
Da beide Teilaussagen gelten gilt auch die Konjunktion. □

c) Die Verschmelzung von X und Y benötigt bis zu $|X| + |Y|$ viele Vergleiche. Das lässt sich aus dem Beweis in a) ableiten. Nach der l-ten Verschmelzung gilt $|A'_l| = \sum_{i=1}^{l}(\frac{n}{k}) = l\frac{n}{k}$, wobei A'_l das Verschmolzene Array ist. Die l-te Versmelzung benötigt also bis zu $x_l = |A'_{l-1}| + \frac{n}{k} = (l-1)\frac{n}{k} + \frac{n}{k}$ Vergleiche. Für alle k Arrays macht das bis zu

$$\begin{aligned}
\sum_{l=1}^{k}(x_l) &= \sum_{l=1}^{k}((l-1)\frac{n}{k} + \frac{n}{k}) \\
&= \sum_{l=1}^{k}(l-1)\frac{n}{k} + \sum_{l=1}^{k}\frac{n}{k} \\
&= \frac{n}{k}\sum_{l=1}^{k} l - k + \sum_{l=1}^{k}\frac{n}{k} \\
&= \frac{nk(k+1)}{k2} + \sum_{l=1}^{k}\frac{n}{k} \\
&= \frac{n(k+1)}{2} + \sum_{l=1}^{k}\frac{n}{k} \\
&\in \mathcal{O}(nk + \log n) = \mathcal{O}(nk)
\end{aligned} \tag{2}$$

Vergleiche.
d) Idee:
Für alle geraden $i < k$ verschmelze jeweils A_i mit A_{i+1} zu $A^1_{\frac{i}{2}}$. Verschmelze dann für alle geraden $i \leq \frac{k}{2}$ A^1_i mit A^1_{i+1} zu $A^2_{\frac{i}{2}}$. Wiederhole den Vorgang bis alle Arrays verschmolzen sind. Ich gehe hier davon aus, dass die Indizes 0-basiert sind.
Die größe der Arrays im l-ten Durchgang ist $g = 2^l\frac{n}{k}$. Die Anzahl der Arrays ist $a = \frac{k}{2^l}$. Die Anzahl der Vergleiche im l-ten Durchgang ist somit $\sum_{i=1}^{a/2} 2g = ag = 2^l\frac{nk}{k2^l} = n$. Die Anzahl der Duchgänge ist $\log(k)$. Die Gesammtzahl der Vergleiche ist also ein $\mathcal{O}(n \log k)$.
e)