TH Brandenburg
Online Studiengang Medieninformatik
Fachbereich Informatik und Medien
Algorithmen und Datenstrukturen
Prof. Dr. rer. nat. Ulrich Baum

Einsendeaufgabe 1

Sommersemester 2021
Abgabetermin 25.04.2021

Maximilian Schulke
Matrikel-Nr. 20215853

1 Zweitkleinstes Element einer Folge von $n \geq 2$ Zahlen

a) Algorithmus in Pseudocode

```
def second_minimum(list):
    second = list[0]
    minimum = list[0]

    for n in list[1:]:
        if n > minimum:
            second = n
            break

    for n in list[1:]:
        if n < minimum:
            second = minimum
            minimum = n

    return second
```

b) Laufzeit-Analyse

Der Algorithmus braucht im **Best-Case n** Vergleiche, liegt also dementsprechend in $\Omega(n)$. Der Best-Case tritt ein, wenn direkt das zweite Element größer als das erste ist, da dann die erste Schleife nach dem ersten Schritt abgebrochen wird und die 2. Schleife immer genau $n-1$ vergleiche ausführt.

Er braucht im **Worst-Case 2(n - 1)** Vergleiche und liegt daher in $O(n)$. Der Worst-Case kommt zustande wenn wir z.B. eine List der Länge n betrachten, die n mal das gleiche Element enthält. Dann benötigen wir beim durchlaufen der ersten und der zweiten Schleife $n-1$ Vergleiche.

2 Asymptotische Notation

Gegeben sei die Funktion $f(n) = 2n^2 + 3n \log_2 n - 72$

a) Beweis von $f(n) \in O(n^2)$

$$\begin{aligned} f(n) &= 2n^2 + 3n \log_2 n - 72 \\ &\leq 2n^2 + 3n \log_2 n \\ &\leq 2n^2 + 3n^2 \\ &= 5n^2 \end{aligned}$$

Somit können wir sagen, dass mit $c \geq 5$ und $n_0 = 1$ die Behauptung $f(n) \in O(n^2)$ gilt

b) Beweis von $f(n) \in \Omega(n^2)$

$$\begin{aligned} f(n) &= 2n^2 + 3n \log_2 n - 72 \\ &\geq 2n^2 - 72 \\ &\geq n^2 \end{aligned}$$

Nun können wir n_0 als Schnittpunkt der beiden Funktionen $2n^2 - 72$ und n^2 berechnen.

$$\begin{aligned}
2n^2 - 72 &= n^2 \mid -2n^2 \\
-72 &= -n^2 \mid * - 1 \\
72 &= n^2 \\
n &= \sqrt{72}
\end{aligned}$$

Also, mit $c = 1$ und $n_0 = \lceil\sqrt{72}\rceil = 9$ gilt $f(n) \in \Omega(n^2)$

c) Gilt $f(n) \in \Theta(n^2)$?

$\Theta(g)$ ist im Skript mit der *Definition 2.5* als $\{\, f \mid f \in O(g) \land f \in \Omega(g) \,\}$ definiert.

Somit wissen wir, dass $f(n) \in \Theta(n^2)$, da wir in 2 a) und 2 b) gezeigt haben, dass $f \in O(g)$ und $f \in \Omega(g)$ gelten.

3 Average-Case-Aufwand der binären Suche

a) Durchschnittliche Anzahl der Vergleiche für einen Hit

Element	0	1	2	3	4	5	6	7	8	9
Vergleiche	3	2	3	4	1	3	4	2	3	4

Macht in Summe 29 Vergleiche und somit $\frac{29}{10} = 2.9$ Vergleiche im Durchschnitt.

b) Summenformel für Vergleiche bei $2^k - 1$ Elementen

Beispiel für $k = 3$

			3			
	1				5	
0		2		4		6

Beispiel für $k = 4$

							7							
			3								11			
	1				5				9				13	
0		2		4		6		8		10		12		14

Es gibt bei $2^k - 1$ immer einen perfekten, gleichmäßigen Baum und immer genau $\log_2 n$ bzw. k Ebenen. Auf (einer 0 indizierten) Ebene i haben wir den Baum i Mal geteilt und haben $(i+1)2^i$ Vergleiche auf dieser Ebene. Wenn wir nun alle Ebenen addieren möchten, um die gesamt Anzahl der Vergleiche zu bekommen, müssen wir lediglich alle Ebenen addieren. Also bei $k = 4$ wären wir bei $1 * 2^0 + 2 * 2^1 + 3 * 2^2 + 4 * 2^3$ Vergleichen. Dies lässt sich durch die gaußsche Summenformel eleganter (und allgemeingültiger) Zusammenfassen zu $\sum_{i=0}^{k-1}(i+1) * 2^i$. Um jetzt auf die durchschnittlichen Vergleiche zu kommen muss nun einfach die Gesamtanzahl durch die Anzahl der Elemente geteilt werden. Also entweder $\sum_{i=0}^{k-1} \frac{(i+1)*2^i}{2^k-1}$ oder alternativ $\frac{1}{2^k-1} \sum_{i=0}^{k-1}(i+1) * 2^i$

c) Laufzeit-Analyse für Average-Case und vergleich mit Worst-Case

$$\frac{1}{2^k - 1} \sum_{i=0}^{k-1}(i+1) * 2^i = \frac{1}{2^k - 1} \sum_{i=1}^{k} i * 2^{i-1} = k + \frac{k}{2^k - 1} - 1$$

Von `http://www.mcs.sdsmt.edu/ecorwin/cs251/binavg/binavg.html`

Der Worst-Case der binären Suche ist $O(\log_2 n)$. Wenn wir in der o.g. geschlossenen Formel k durch $\lceil \log_2 n \rceil$ ersetzen erhalten wir:

$$\lceil \log_2 n \rceil + \frac{\lceil \log_2 n \rceil}{n} - 1$$

Wenn wir diese Gleichung nun asymptotisch betrachten entfällt $\frac{\lceil \log_2 n \rceil}{n}$

$$\lim_{n \to \infty} \lceil \log_2 n \rceil + \frac{\lceil \log_2 n \rceil}{n} - 1 = \lceil \log_2 n \rceil + 0 - 1 = \lceil \log_2 n \rceil - 1$$

Somit können wir sagen, dass der Average-Case der binären Suche asymptotisch gesehen nur um 1 Schritt besser ist, als der Wort-Case $\log_2 n$

4 Analyse einer rekursiven Funktion

a) Für welche n terminiert die Rekursion, für welche nicht?

Die Funktion terminiert nur für gerade postive Zahlen und $n = 0$. Bei ungeraden positiven Zahlen, verfehlen wir den Basis-Fall immer um genau 1 und landen danach in einer Endlosschleife. Bei negativen Zahlen, sind wir schon initial “unter“ dem Basis-Fall.

b) Geben Sie F als geschlossene nicht-rekursive Formel an und beweisen Sie Ihre Formel durch Induktion.

Geschlossene Formel für F:

$$F(n) = \frac{n}{2} + \frac{n^2}{4}$$

Induktionsbeweis:

$$\begin{aligned}
Induktionsvoraussetzung &= \frac{n}{2} + \frac{n^2}{4} = F(n) \qquad\qquad n \in \{x \mid x \in N_0 \wedge x \; mod \; 2 = 0\} \\
Induktionsbehauptung &= \frac{n+2}{2} + \frac{(n+2)^2}{4} = F(n+2) \\
Induktionsanfang &= \frac{0}{2} + \frac{0^2}{4} = F(0) \Leftrightarrow 0 = 0 \\
Induktionsschritt &= \frac{n+2}{2} + \frac{(n+2)^2}{4} \\
&= \frac{n}{2} + \frac{n^2+4n+4}{4} + \frac{2}{2} \\
&= \frac{n}{2} + \frac{n^2}{4} + \frac{4n+4}{4} + \frac{2}{2} \\
&= F(n) + (n+2) = F(n+2)
\end{aligned}$$

c) Stellen Sie eine Rekursionsgleichung für $T(n)$ auf und geben Sie eine geschlossene Formel für $T(n)$ an.

Rekursionsgleichung:

$$\begin{aligned}
T(0) &= 0 \\
T(n) &= T(n-2) + 2
\end{aligned}$$

Geschlossene Formel für $T(n)$:

$$T(n) = n$$