Übungen zur Algorithmischen Bioinformatik I

Blatt 6

Xiheng He

Mai 2021

2.Aufgabe: (Amortisierte) Algorithmenanalyse (10 Punkte)

Betrachten Sie folgendes einfache Problem (BinaryCounter):

Zählen Sie von 0 bis n mittels einer binären String-Darstellung des Zählers, z.B. i = 0 . . . 0010101011

Ein Beispiel für das Zählen:

0 -> 1 -> 10 -> 11 -> 100 -> 101 -> 110 -> 111 -> ... 0111 1111 -> 1000 0000 -> 1000 0001

Die Kosten einer Transition von i -> i+1 ist definiert als die Anzahl der 'carries', die bei der Transition auftreten.

Notiz: 'carries'sind die Überträge, die beim (schriftlichen) Addieren (hier '+1') auftreten (können), in Hardware-Addierwerken von CPUs sind sie wichtig für die notwendigen Zyklen der Operation.

Analysieren Sie die Laufzeit $T(n)$ des Algorithmus in Abhängigkeit von n.

(a) Überlegen Sie wie teuer eine Operation (= 'transition') sein kann!

Angenommen, i hat k Bits:

In worst-case kann meistens k Positionen 'carries' auftreten wenn Position i -> i+1 übergeht. Damit beträgt die Kosten $O(nk)$. In best-case wird aber genau nur 1 Position 'carries' auftreten somit ist die Kosten in $O(1)$. Dann kann es abgeleitet werden, dass $\sum kosten$ in Interval $(1, nk)$ liegt. Weil 'carries' nicht bei jeder 'transition' genau k mal vorkommt und die Kosten von n abhängig ist, betrachten wir die $\sum \frac{kosten}{n}$ als eine unendliche Reihe, dann sollte diese Reihe konvergent sein.

Vermutung: Die Operation (= 'transition') besitzt $O(n)$ Laufzeit-komplexität.

(b) Beweisen oder widerlegen Sie $T(n) = O(n)$!

Amortisierte Algorithmenanalyse:

Sei i $= n$ eine binäre String-Darstellung des Zählers und i hat k Bits sodass bestrachten wir i als ein A Array und $|A| = k$.

$n =$	A[2]	A[1]	A[0]	'carries'
0	0	0	0	0
1	0	0	1	+1
10	0	1	0	+2
11	0	1	1	+1
100	1	0	0	+3

Durch amortisierte Analyse folgt, dass 'carries'

in A[0] in jeder 'transition' vorkommt;

in A[1] in jeder zwei 'transition' vorkommt;

in A[2] in jeder vier 'transition' vorkommt;

in A[3] in jeder acht 'transition' vorkommt;

in A[n] in jeder 2^{k-1} 'transition' vorkommt;

Damit gilt $T(n)$:

$$\begin{aligned} T(n) &= n + \lfloor n/2 \rfloor + \lfloor n/4 \rfloor + \lfloor n/8 \rfloor + \cdots + \lfloor n/2^{k-1} \rfloor \\ &\leq \sum_{i=0}^{k-1} \frac{n}{2^i} \qquad \text{(geometrische Reihe: } \sum_{n=0}^{\infty} \frac{1}{2^n}) \\ &\leq 2n \\ &\Longrightarrow T(n) \in O(n) \end{aligned}$$

(c) Was ist $T(I)$, wenn I wie üblich die Instanzgrösse des BinaryCounter Problems ist?

Angenommen, n sei die entsprechende binären Darstellung zu I und $|n| = k$. Jede Bit der n hat genau zwei Möglichkeiten: 0 und 1. Somit gilt: $2^k \geq I$ und $2^{k-1} \leq I$.

$$2^k \geq I \Longrightarrow k \geq \log I \Longrightarrow \tag{1}$$

$$2^{k-1} \leq I \Longrightarrow k - 1 \leq \log I \Longrightarrow k \leq \log I + 1 \tag{2}$$

Aus (1)und (2) folgt, dass $\log I \leq k \leq \log I + 1$. Somit gilt $T(I) \in O(\log I)$.