Übungen zur Algorithmischen Bioinformatik I

Blatt 6

Xiheng He

Mai 2021

4. Aufgabe: Algorithmenanalyse (10 Punkte)

Folgender Algorithmus in Pseudocode erhält einen Vektor $F \in \mathbb{R}^n$ und eine Zahl $p \in \mathbb{R}$, $0 < p < 1$ als Eingabe.

Was berechnet der Algorithmus? Analysieren Sie die Laufzeit des Algorithmus z.B. durch Aufstellen und Lösen der entsprechenden Rekurrenzgleichung.

Dieser Algorithmus berechnet das größte Element im Vektor F, indem er die Größe des Arrays durch $/p$ reduziert und die Elemente absteigend sortiert. Schließlich wird die Größe auf 1 reduziert und das größte Element im Vektor F ausgegeben.

Die entsprechende Rekurrenzgleichung sind (falls argmax(F) auch in $O(|F|)$):

$$T(n) = T(\lfloor np \rfloor) + 2n\lfloor pn \rfloor \tag{1}$$

Aus (1) gilt: $f(n) = 2n\lfloor pn \rfloor \leq 2pn^2 \Longrightarrow f(n) \in O(n^2), a = 1, b = \frac{1}{p}, \log_b a = \log_{\frac{1}{p}} 1 = 0$

Dann $\forall e > 0 \not\Longrightarrow f(n) = O(n^{\log_b(a)-e}) \Longrightarrow$ Master-Theorem nicht anwendbar.

Aber wir können auch die Laufzeit in worst-case betrachten. Da p nicht explizit gegeben wurde, kann es sein, dass die Größe des Arrays in jeder Rekursion genau um 1 verringert, d.h. $f(n)$ muss genau n mal durchlaufen. Deswegen gilt:

$$\begin{aligned} T(n) &\leq T(\lfloor np \rfloor) + 2n\lfloor pn \rfloor = 2pn^2 + 2p(n-1)^2 + 2p(n-2)^2 + \cdots + 2p^2 \\ &= 2p \cdot \sum_{i=1}^{n} n^2 \\ &= 2p \cdot \frac{n(n+1)(2n+1)}{6} \\ &\overset{0<p<1}{\Longrightarrow} T(n) \in O(n^3) \end{aligned}$$

Daraun folgt: $T(n) \in O(n^3)$