Algorithmische Mathematik I

Wintersemester 2021
Prof. Dr. Jürgen Dölz
David Ebert

UNIVERSITÄT BONN

Blatt 6

Abgabe **Montag, 29.11.21, 10:00**.

Aufgabe 1. (Master-Theorem)

Bei der Laufzeitanalyse von rekursiven Algorithmen ist der Hauptsatz der Laufzeitfunktionen (Master-Theorem) hilfreich. Seien $a \geq 1$ und $b > 1$ Konstanten und bezeichne $T(n)$ die Laufzeit oder Kosten eines rekursiven Algorithmus für ein Problem der Größe n, sodass

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n) \tag{1}$$

mit einer von T unabhängigen Funktion $f(n) \geq 0$. Die Anzahl der rekursiven Aufrufe ist also durch a gegeben, der Bruchteil der einzelnen Teilprobleme durch $1/b$, sowie die Kosten für das Aufteilen und Zusammenfügen der Teillösungen durch $f(n)$. Dann besagt das Master-Theorem:

- Falls $f(n) \in \mathcal{O}(n^{\log_b a - \epsilon})$ für ein $\epsilon > 0$ gilt, dann folgt $T(n) \in \Theta(n^{\log_b a})$.
- Falls $f(n) \in \Theta(n^{\log_b a})$, dann folgt $T(n) \in \Theta(n^{\log_b a} \log n)$.
- Falls $f(n) \in \Omega(n^{\log_b a + \epsilon})$ für ein $\epsilon > 0$ gilt und ein c mit $0 < c < 1$ existiert, sodass $af(\frac{n}{b}) \leq cf(n)$ für hinreichend große n, dann folgt $T(n) \in \Theta(f(n))$.

Dies zeigt also systematisch den Einfluss der Nebenkosten auf die Gesamtlaufzeit auf.

a) Beweisen Sie die drei Aussagen des Master-Theorems. Nehmen Sie dazu der Einfachheit halber an, dass $n = b^k$ für ein $k \in \mathbb{N}$, sowie dass $T(1) = 1$ gilt. Ferner dürfen Sie ohne Beweis benutzen:

$$n^m + n^{m-\epsilon} \in \Theta(n^m) \text{ für } m \in \mathbb{N}, \epsilon > 0,$$
$$f_1 \in \Theta(g_1), f_2 \in \Theta(g_2) \Rightarrow f_1 f_2 \in \Theta(g_1 g_2).$$

b) Stellen Sie eine Formel gemäß (1) für die Laufzeit von Mergesort auf (siehe Vorlesung), und geben Sie die Laufzeit mithilfe des Master-Theorems an.

(3+1 Punkte)

Aufgabe 2. (Radixsort)

Ein weiterer populärer Sortieralgorithmus, der nicht in der Vorlesung besprochen wurde, ist Radixsort. Er kommt ganz ohne Vergleiche der zu sortierenden Objekte aus! Allerdings ist er nur anwendbar, wenn die Schlüssel, nach denen sortiert werden soll, Zeichenketten aus einem endlichen, totalgeordneten Alphabet der Mächtigkeit b sind, wobei die maximal vorkommende Kettenlänge l im Voraus bekannt sein muss. Zum Beispiel fallen Zahlen mit Darstellung im b-adischen System und fester Stellenanzahl, also auch Datentypen wie `unsigned int` darunter. Wir wollen die rekursive Version dieses Algorithmus betrachten:
Jedes Objekt wird nacheinander in b verschiedene Bins, einen für jedes Zeichen, abgelegt, wobei die erste Ziffer des Schlüssels den Bin bestimmt. Jeder Bin wird nun wieder mit

Radixsort sortiert, wobei diesmal nach der darauffolgenden Ziffer kategorisiert wird. Bins mit weniger als zwei Elementen und solche, die bereits durch die letzte Ziffer bestimmt wurden, werden ohne weitere Sortierung zurückgegeben. Danach werden die Listen der einzelnen Bins, in aufsteigender Reihenfolge, konkateniert.

a) Sortieren Sie folgende Listen von Schlüsseln per Hand mit Radixsort. Leere Bins müssen nicht notiert werden.

   - Binärzahlen $(111, 000, 110, 011, 100, 101, 010, 001)$ mit $b = 2$ und $l = 3$.
   - Dezimalzahlen $(0284, 0005, 0532, 0281, 0239, 1512, 0095, 0065, 0742, 9027)$ mit $b = 10$ und $l = 4$.
   - Vornamen (Otto,Lisa,Karl,Anna,Bela,Alex,Elia,Cloe,Hugo,Yael,Liam,Sina) mit $b = 26$ und $l = 4$.

   Wann ist Radixsort im Allgemeinen besonders effektiv?

b) Nehmen Sie an, die zu sortierende Liste enthalte alle $n = b^l$ paarweise verschiedene Schlüssel, für ein beliebiges l. Berechnen Sie eine Abschätzung für die Laufzeit $T(n)$ (in Landau-Notation), wobei angenommen werden soll, dass jedes Einfügen in einen Bin sowie die Konkatenation der Bins jeweils einen Arbeitsschritt kostet.

(3+1 Punkte)

Aufgabe 3. (Graphen)

Zeigen Sie die folgenden Eigenschaften von ungerichteten Graphen $G = (V, E)$.

a) Die Anzahl von Knoten mit ungeradem Grad ist gerade.

b) Falls G genau zwei Knoten v und w mit ungeradem Grad hat, so gibt es einen v-w-Weg.

(2 + 2 Punkte)

Programmieraufgabe 1. (Fibonacci-Folge)

Eine der bekanntesten Zahlenfolgen überhaupt ist die Fibonacci-Folge, welche durch folgende Rekursionsgleichung gegeben ist:

$$f_0 = 1, \quad f_1 = 1, \quad f_n = f_{n-1} + f_{n-2} \quad \text{für } n \geq 2.$$

a) Schreiben Sie eine Funktion, die f_n zu gegebenem n *rekursiv* berechnet. Berechnen Sie damit f_{30} und lassen Sie das Ergebnis ausgeben.

b) Schreiben Sie eine weitere Funktion, die f_n zu gegebenem n *iterativ* berechnet. Lassen Sie auch hiermit f_{30} ausgeben. Was fällt Ihnen auf?

c) Messen Sie die Laufzeit für beide Implementierungen für $1 \leq n \leq 35$ und stellen Sie die Ergebnisse jeweils als semilogy-Plot dar. Geben Sie zusätzlich zur Überprüfung die ersten 20 Ergebnisse beider Implementierungen aus.

(4 Punkte)

Veranstaltungshinweis der Gleichstellungs-AG:
'Ally Day', eine Veranstaltung zum Thema 'Inklusion und Diversität' am 27. November, 15-18 Uhr. Eingeladen sind alle Studierenden! Nähere Infos und Anmeldung: `https://www.hcm.uni-bonn.de/ally-day`