Karlsruher Institut für Technologie

Institut für Theoretische Informatik
T.T.-Prof. Dr. Thomas Bläsius
https://scale.iti.kit.edu

Übungsblatt 2

Algorithmen 1 - Sommersemester 2023

Abgabe im ILIAS bis 5.05.2023, 14:00 Uhr
Bitte beschrifte deine Abgabe gut sichtbar mit deinem Namen und deiner Matrikelnummer. Achte insbesondere bei handschriftlichen Abgaben auf Lesbarkeit und genügend Platz für Korrektur-Anmerkungen. Die Abgabe erfolgt über das Übungsmodul in der Gruppe deines Tutoriums im ILIAS. Gib deine Ausarbeitungen in *einer* Datei im PDF-Format ab.
Achte darauf, effiziente Algorithmen zu formulieren, also solche mit möglichst geringer asymptotischer Laufzeit! Wenn du die Korrekheit deines Algorithmus begründen oder dessen Laufzeit analysieren sollst, tue dies getrennt von der Beschreibung deines Algorithmus.

Gesamtpunktzahl: 20

Aufgabe 1 - Rekurrenzen lösen (6 Punkte)

In dieser Aufgabe wollen wir die Laufzeit rekursiver Funktionen mittels ihres Rekursionsbaums herausfinden. Beantworte dazu für die folgenden rekursiven Funktionen jeweils:

- Wie viele Lagen hat der Rekursionsbaum?
- Wie viele Knoten sind auf Lage i?
- Wie groß ist ein Knoten auf Lage i?
- Wie viel Zeit kostet ein Knoten auf Lage i?

Finde dann die Laufzeit der rekursiven Funktion heraus.

a) (1,5 Punkte)

$$T_1(n) = \begin{cases} 1 & \mid n = 1 \\ 4 \cdot T_1\left(\frac{n}{2}\right) + n^2 & \mid \text{sonst} \end{cases}$$

b) (1,5 Punkte)

$$T_2(n) = \begin{cases} 1 & \mid n = 1 \\ 8 \cdot T_2\left(\frac{n}{4}\right) + \sqrt{n} & \mid \text{sonst} \end{cases}$$

c) (1,5 Punkte)

$$T_3(n) = \begin{cases} 1 & \mid n = 1 \\ T_3\left(\frac{n}{2}\right) + 3^{\log_2(n)} & \mid \text{sonst} \end{cases}$$

d) (1,5 Punkte)

$$T_4(n) = \begin{cases} 1 & \mid n = 1 \\ 9 \cdot T_4\left(\frac{n}{3}\right) + n^2 \log(n) & \mid \text{sonst} \end{cases}$$

Aufgabe 2 - Holz Kaufen und die Welt Erobern (7 Punkte)

Nach deiner unglaublich effizienten Holzlegestrategie auf Blatt 1, hast du nun den Auftrag von Dr. Meta bekommen, diese Strategie in die Tat umzusetzen! Für einen großen Damm braucht man natürlich auch eine große Menge Holzstämme. Daher machst du dich auf zu deinem Holzgroßhändler des Vertrauens!
Der Händler hat alle Stämme hintereinander aufgereiht, wobei jeder Stamm mit einer ganzen (möglicherweise negativen) Zahl, der *Stablitätszahl*, versehen wurde. Als Großabnehmer darfst

du nur einen konsekutiven Teil der Reihe kaufen. Das heißt, du darfst einen Startindex und Endindex angeben und kaufst das ganze Holz dazwischen, ohne einzelne Stämme auszulassen. Um einen möglichst stabilen Damm zu bauen, darfst du natürlich kein schlechtes Holz kaufen. Du möchtest also die Sektion an Holz kaufen, sodass die Summe der Stabilitätszahlen möglichst groß ist.

Formal bekommen wir also ein Array $A = [a_1, a_2, ..., a_n]$ mit ganzen Zahlen $a_i \in \mathbb{Z}$ und suchen nun Indizes $1 \leq i \leq j \leq n$, sodass die Dammstabilität von A im Interval $[i, j]$

$$D(A[i,j]) \coloneqq a_i + a_{i+1} + ... + a_j = \sum_{k=i}^{j} a_k$$

maximal ist.
Wir wollen nun einen Divide-and-Conquer Algorithmus für das Problem entwickeln.

a) Mal angenommen, wir kennen schon den Index i und suchen nur noch das dafür optimale j. Beschreibe einen Algorithmus der in Zeit $O(n)$ den Index $j \in [i, n]$ findet, sodass $D(A[i, j])$ maximal ist. (2 Punkte)

b) Nehmen wir nun an, wir haben zwei Arrays $B = [b_1, ..., b_\ell]$ und $C = [c_1, ..., c_r]$ und wir haben schon folgende Informationen:

- die Indizes i_b, j_b für die $D\big(B\big[i_b, j_b\big]\big)$ maximal ist,
- die Indizes i_c, j_c für die $D\big(C\big[i_c, j_c\big]\big)$ maximal ist,
- den Index i_ℓ für den $D(B[i_\ell, \ell])$ maximal ist und
- den Index j_r für den $D\big(C\big[1, j_r\big]\big)$ maximal ist.

(Tipp: Es kann helfen, sich die vorhanden Informationen einmal aufzumalen.)
Nun konkatenieren wir die beiden Arrays und bekommen $\mathrm{BC} = [b_1, ..., b_\ell, c_1, ..., c_r]$. Wie können wir nun Indizes $i_{\mathrm{bc}} \leq j_{\mathrm{bc}} \in [1, \ell + r]$ bestimmen, sodass $D\big(\mathrm{BC}\big[i_{\mathrm{bc}}, j_{\mathrm{bc}}\big]\big)$ maximal ist? (1 Punkt)

c) Benutze nun die Erkenntnisse der letzten beiden Teilaufgaben, um einen Divide-and-Conquer Algorithmus zu beschreiben, der als Eingabe ein Array $A = [a_1, ..., a_n]$ bekommt, und Indizes $1 \leq i \leq j \leq n$ ausgibt für die $D(A[i, j])$ maximal ist. Begründe warum dein Algorithmus eine Laufzeit in $O(n \log(n))$ hat. (4 Punkte)

Aufgabe 3 - Magischer Hut (7 Punkte)

Wir haben eine Menge blauer Kugeln. Per Zauberhand kann die Farbe solcher Kugeln von blau nach rot geändert werden. Wie sich das für richtige Magier und Magierinnen gehört, wollen wir geheim halten, wie der Trick funktioniert. Darum kann die Farbe von Kugeln nur geändert werden, wenn sie sich im *magischen Hut* befinden! Zu jedem Zeitpunkt bezeichnen wir mit n_b und n_r jeweils die Anzahl blauer und roter Kugeln im Hut. Der Hut unterstützt nun folgende Operationen:

- HIDE: Legt eine neue blaue Kugel in den Hut, mit Kosten $\Theta(1)$.
- MAGIC: Färbt die aufgerundete Hälfte der blauen Kugeln im Hut rot, mit Kosten $\Theta(n_b)$.
- SHOW: Holt alle roten Kugeln aus dem Hut, mit Kosten $\Theta(n_r)$.

a) Gib für die folgenden beiden Folgen von Operationen die Gesamtkosten an:

(1 Punkt)	(1 Punkt)
$n \cdot$ HIDE	$2n \cdot$ HIDE
MAGIC	MAGIC
SHOW	$2n \cdot$ HIDE
	MAGIC
	SHOW

b) Zeige mit Hilfe der Kontomethode, dass in jeder beliebigen Abfolge von Operationen jede Operation amortisiert konstante Kosten hat. (2 Punkte)

c) Zeige mit Hilfe der Potentialmethode, dass in jeder beliebigen Abfolge von Operationen jede Operation amortisiert konstante Kosten hat. (3 Punkte)