

5. Übungsblatt

zur Vorlesung

Grundzüge der Informatik I

Abgabe über Ilias bis zum 10.5. 14:00 Uhr.
Besprechung in Kalenderwoche 20.

Aufgabe 1 Mastertheorem (2 + 2 + 2 + 2 Punkte)

Verwenden Sie das Master-Theorem, um für die folgenden Rekursionsgleichungen asymptotische Schranken anzugeben:

$$\text{a) } T(n) = \begin{cases} 12 & n = 1 \\ 2 \cdot T\left(\frac{n}{2}\right) + 12 & \text{sonst} \end{cases}$$

$$\text{b) } T(n) = \begin{cases} 1 & n = 1 \\ 1 \cdot T\left(\frac{n}{3}\right) + n^3 & \text{sonst} \end{cases}$$

$$\text{c) } T(n) = \begin{cases} 1 & n = 1 \\ 1 \cdot T\left(\frac{n}{2}\right) + n & \text{sonst} \end{cases}$$

$$\text{d) } T(n) = \begin{cases} 1 & n = 1 \\ 2 \cdot T\left(\frac{n}{4}\right) + \sqrt{n} & \text{sonst} \end{cases}$$

Geben Sie in allen Teilaufgaben an, welchen Teil des Theorems Sie nutzen, und überprüfen Sie stets, ob die nötige Voraussetzung erfüllt ist.

1.)

$$a) T(n) = \begin{cases} 12 \\ 2 \cdot T\left(\frac{n}{2}\right) + 12 \end{cases}$$

$a=2$

$b=2$

$f(n)=12$

$\text{Wähle } \gamma = \frac{1}{2}$

$f(n) \leq \gamma \cdot 2 \cdot f\left(\frac{n}{2}\right)$

$12 \leq \gamma \cdot 2 \cdot 12$

$12 \leq \frac{1}{2} \cdot 2 \cdot 12$

$12 \leq 12$

 $\hookrightarrow \text{Fall 3}$

$$\hookrightarrow T(n) \in O(a^{\log_2 n}) = O(2^{\log_2 n})$$

$$= \underline{\underline{O(n)}}$$

b)

$$T(n) = \begin{cases} 1 \\ 1 \cdot T\left(\frac{n}{3}\right) + n^3 \end{cases}$$

$a=1$

$b=3$

$f(n)=n^3$

$\text{Wähle } \gamma=27$

$f(n) \geq \gamma \cdot 1 \cdot f\left(\frac{n}{3}\right)$

$n^3 \geq \gamma \cdot 1 \cdot \left(\frac{n}{3}\right)^3$

$n^3 \geq 27 \cdot \underline{\underline{n^3}}$

$n^3 \geq n^3$

 $\hookrightarrow \text{Fall 2}$

$\hookrightarrow T(n) \in O(f(n)) = \underline{\underline{O(n^3)}}$

c)

$$T(n) = \begin{cases} 1 \\ 1 \cdot T\left(\frac{n}{2}\right) + n \end{cases}$$

$$a=1 \quad f(n) \geq \gamma \cdot 1 \cdot f\left(\frac{n}{2}\right)$$

$$b=2 \quad n \geq \gamma \cdot 1 \cdot \frac{n}{2}$$

$$f(n)=n \quad n \geq 2 \cdot \frac{n}{2}$$

$$\text{Wähle } \gamma=2 \quad n \geq n$$

↳ Fall 2

$$\hookrightarrow T(n) \in O(f(n))$$

$$= \underline{\underline{O(n)}}$$

d)

$$T(n) = \begin{cases} 1 \\ 2 \cdot T\left(\frac{n}{4}\right) + \sqrt{n} \end{cases}$$

$$a=2 \quad \sqrt{n} = 2 \cdot \sqrt{\frac{n}{4}}$$

$$b=4 \quad \sqrt{n} = 2 \cdot \sqrt{\frac{n}{4}}$$

$$f(n)=\sqrt{n} \quad \sqrt{n} = \frac{2}{2} \cdot \sqrt{n}$$

$$\sqrt{n} = \sqrt{n}$$

↳ Fall 4

$$\hookrightarrow T(n) \in O(f(n) \log n)$$

$$= \underline{\underline{O(\sqrt{n} \log n)}}$$

Aufgabe 2 Teile und Herrsche (2 + 4 + 3 + 3 Punkte)

Wir betrachten den Aktienkurs der Firma *Informatik AG*. Der Wert einer Aktie über n Zeiteinheiten wird dabei in dem Array A gespeichert. Wir möchten Wissen zu welchen Zeiten wir die Aktie kaufen und verkaufen sollten, um den maximalen Gewinn zu erzielen. Kauf und Verkauf dürfen dabei aber nur einmal passieren.

Formal ist also ein Array $A[1..n]$ von positiven ganzen Zahlen gegeben und wir wollen die größte Wertdifferenz $A[i] - A[j]$ für $j \leq i$ bestimmen.

- a) Betrachten Sie zwei benachbarte Teilarrays $A[i..j]$ und $A[j + 1..\ell]$ mit $1 \leq i \leq j < \ell \leq n$. Welche Informationen über die Teilarrays werden benötigt, um die maximale Differenz zweier Elemente in $A[i..\ell]$ in konstanter Zeit zu berechnen? Erklären Sie auch wie diese Berechnung funktioniert.
- b) Entwickeln Sie einen Teile-und-Herrsche Algorithmus, der Ihre Überlegungen aus Aufgabenteil a) verwendet, um die größte Wertdifferenz zu berechnen. Geben Sie eine Implementierung Ihres Algorithmus in Pseudocode an und kommentieren Sie diesen. Für die volle Punktzahl wird ein Algorithmus erwartet, dessen Worst-Case-Laufzeit durch $\mathcal{O}(n)$ beschränkt ist.
- c) Analysieren Sie die Laufzeit Ihres Algorithmus. Stellen Sie hierzu eine Rekursionsgleichung für die Laufzeit Ihres Algorithmus auf und lösen Sie diese. Sie dürfen das Mastertheorem verwenden.
- d) Zeigen Sie die Korrektheit Ihres Algorithmus.

a)



Gesucht: $\max \text{Diff}$ in $A[i..l]$

Also gesucht $(m, n) \in \{i..l\} \times \{i..l\}$ mit $m \leq n$

und maximaler Diff $A[n] - A[m]$

Es gilt entweder $i \leq m \leq n \leq j$,

$j+1 \leq m \leq n \leq \ell$

$i \leq m \leq j < j+1 \leq n \leq \ell$

↳ Also benötigen wir die maximale Differenz aus $A[i..j]$,

die maximale Differenz aus $A[j+1..l]$,

und $\min\{A[i..j]\}$, sowie $\max\{A[j+1..l]\}$.

Da ein Teilarray von beiden Seiten kombiniert werden kann, hat

unsere Funktion die Rückgabe ($\max\text{Diff}$ in $A[i..j]$, $\min\{A[i..j]\}$, $\max\{A[i..j]\}$),

wenn sie auf dem Teilfeld $A[i..j]$ aufgerufen wird.

Für die Kombination passiert, indem die drei Fälle berechnet und der kleinste Wert verwendet wird.

b)

Stock(A, n)

$$(\text{res}, \min, \max) = \text{FindDiff}(A, \ell, n)$$

return res

FindDiff(A, ℓ , r)

- 1 if ($\ell = r$)
- 2 return (0, $A[\ell], A[r]$)
- 3 $m = \lfloor \frac{\ell+r}{2} \rfloor$
- 4 ($\text{diff}_e, \min_e, \max_e$) = FindDiff(A, ℓ, m)
- 5 ($\text{diff}_r, \min_r, \max_r$) = FindDiff(A, $m+1, r$)
- 6 $\min_{\text{all}} = \min \{\min_e, \min_r\}$
- 7 $\max_{\text{all}} = \max \{\max_e, \max_r\}$
- 8 $\text{diff}_{\text{all}} = \max \{\text{diff}_e, \text{diff}_r, \max_r - \min_e\}$
- 9 return ($\text{diff}_{\text{all}}, \min_{\text{all}}, \max_{\text{all}}$)

c)

$$T(n) \quad \text{mit } n = r - \ell + 1$$

$$1 \quad 1$$

$$1$$

$$1$$

$$T\left(\frac{n}{2}\right)$$

$$T\left(\frac{n}{2}\right)$$

$$2$$

$$2$$

$$3$$

$$1$$

$$T(n) = \begin{cases} 2 \\ 2T\left(\frac{n}{2}\right) + 10 \end{cases}$$

$$a = 2$$

$$b = 2$$

$$f(n) = 10$$

$$\text{Wähle } \delta = \frac{1}{2}$$

$$f(n) \leq \delta \cdot 2 \cdot f\left(\frac{n}{2}\right)$$

$$10 \leq \delta \cdot 2 \cdot 10$$

$$10 \leq \frac{1}{2} \cdot 2 \cdot 10$$

$$10 \leq 10$$

↳ Fall 3

$$\hookrightarrow T(n) \in O(a^{\log_2 n})$$

$$= O(2^{\log_2 n}) = \underline{\underline{O(n)}}$$

d) Beh FindDiff(A, l, r) gibt das Tripel (d, m, x) aus, wobei

- d die maximale Differenz $A[i] - A[j]$ zwischen zwei Werten aus dem Teilarray $A[l..r]$ mit $j \neq i$,
- m der minimale Wert im Teilarray $A[l..r]$ und
- x der maximale Wert im Teilarray $A[l..r]$ ist.

Bew durch Induktion über $n = r - l + 1$

I A $n=1$

Da $l=r$ ist wird die Bedingung in Z1 erfüllt und in Z2 $(0, A[l], A[r])$ zurückgegeben.

Da das Teilarray nur ein Element hat, ist die einzige mögliche Differenz die zu dem selben Element, also 0.

Da das Teilarray nur ein Element hat, ist dieses Element sowohl das größte, als auch das kleinste. Also $\min = A[l] = A[r] = \max$.

Die Rückgabe ist also korrekt

IV FindDiff(A, l', r') gibt das Tripel (d, m, x) für das Teilarray $A[l'..r']$ aus, wobei das Tripel wie oben für $A[l'..r']$ definiert ist, und $l' \leq r' - l' + 1 < n$

IS $n > 1$

Da $n > 1$ ist, gilt $r \neq l$ und die Bedingung in Z1 wird nicht erfüllt. Also wird in Z3 die Mitte des Teillarrays gefunden. In Z4 & Z5 wird der Algorithmus rekursiv aufgerufen. da wir IV gilt:

diffe ist maximale Differenz im Teilarray $A[l..m]$
min ist minimales Element im Teilarray $A[l..m]$
max ist maximales Element im Teilarray $A[l..m]$

diffr ist maximale Differenz im Teilarray $A[m+1..r]$
minr ist minimales Element im Teilarray $A[m+1..r]$
maxr ist maximales Element im Teilarray $A[m+1..r]$

In Z 6 wird \min_{all} berechnet und als m ausgegeben. Es gilt $\min_{\text{all}} = \min \{\min_e, \min_r\}$

Da die Teilarrays $A[\ell..m]$ & $A[m+1..r]$ das gesamte Teilarray $A[\ell..r]$ abdecken, muss das kleinste Element aus $A[\ell..r]$ entweder \min_e oder \min_r sein.

Also ist das Minimum der beiden die korrekte Ausgabe.

In Z 7 wird \max_{all} berechnet und als X ausgegeben. Es gilt $\max_{\text{all}} = \max \{\max_e, \max_r\}$

Da die Teilarrays $A[\ell..m]$ & $A[m+1..r]$ das gesamte Teilarray $A[\ell..r]$ abdecken, muss das größte Element aus $A[\ell..r]$ entweder \max_e oder \max_r sein.

Also ist das Maximum der beiden die korrekte Ausgabe.

In Z 8 wird diff_{all} berechnet und als d ausgegeben. Es gilt $\text{diff}_{\text{all}} = \max \{ \text{diffe}, \text{diff}, \max_r - \min_e \}$

Das Elementpaar $(A[i], A[j])$ mit $j \leq i$ und der größten Differenz $A[j] - A[i]$ im Teilarray $A[\ell..r]$ ist entweder komplett in der ersten Hälfte $A[\ell..m]$, also gilt $\ell \leq i \leq j \leq m$, und in diesem Fall ist die Differenz diffe ,

oder $(A[i], A[j])$ ist komplett in der zweiten Hälfte $A[m+1..r]$, also gilt $m+1 \leq i \leq j \leq r$, und in diesem Fall ist die Differenz diff ,

oder $A[i]$ ist in der ersten Hälfte & $A[j]$ ist in der zweiten Hälfte, also gilt $\ell \leq i \leq m < m+1 \leq j \leq r$. Im letzten Fall muss $A[i]$ das kleinste Element in $A[\ell..m]$, also \min_e , & $A[j]$ das größte Element in $A[m+1..r]$, also \max_r , sein. Sonst ist die Differenz nicht maximal. Also ist $\max_r - \min_e$ in diesem Fall korrekt.

Andere Fälle gibt es nicht, da $i \leq j$ gelten muss.

Insgesamt ist also das Maximum der drei möglichen Fälle die korrekte Ausgabe.