Technische Universität Braunschweig

Algorithmen und Datenstrukturen – Übung #10

Fragestunde

Matthias Konitzny, Arne Schmidt

17.02.2022

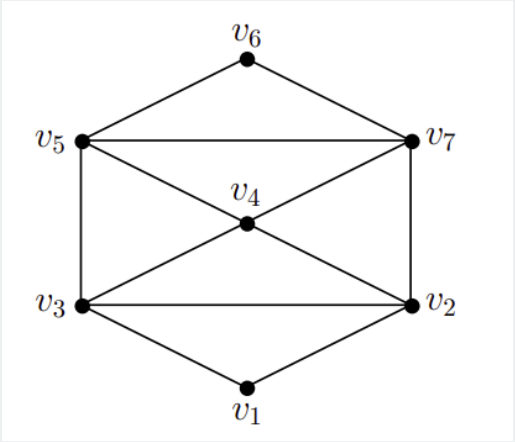
Quiz!

Prüfung

Testlauf

1 Graphenscan - Anwenden

Betrachte folgenden Graphen.



Führe Breitensuche auf diesem Graphen aus. Starte bei Knoten v_1. Kommen in einem Schritt des Algorithmus mehrere Knoten in Frage, wähle denjenigen mit dem kleinsten Index.

1.1	Welche der folgenden Kanten werden in den Breitensuchbaum aufgenommen? (Teil 1)	☐ $\{v_1,v_2\}$ ☐ $\{v_2,v_3\}$	☐ $\{v_1,v_3\}$
1.2	Welche der folgenden Kanten werden in den Breitensuchbaum aufgenommen? (Teil 2)	☐ $\{v_2,v_4\}$ ☐ $\{v_4,v_7\}$	☐ $\{v_2,v_7\}$
1.3	Welche der folgenden Kanten werden in den Breitensuchbaum aufgenommen? (Teil 3)	☐ $\{v_3,v_4\}$ ☐ $\{v_4,v_5\}$	☐ $\{v_3,v_5\}$
1.4	Welche der folgenden Kanten werden in den Breitensuchbaum aufgenommen? (Teil 4)	☐ $\{v_5,v_6\}$ ☐ $\{v_6,v_7\}$	☐ $\{v_5,v_7\}$

Testlauf

2 Master-Theorem

Betrachte folgende Rekursionsgleichung.

$$T(n) := 3 \cdot T\left(\frac{n}{3}\right) - 3 + 2n$$

2.1	Gib alle im Master-Theorem auftretenden Parameter an.	m = [ ] , k = [ ] , α1 = ... = α3 = [ ] / [ ]
2.2	Die Summe der α_i hoch k ist...	○ < 1 ○ = 1 ○ > 1
2.3	Mit dem Master-Theorem ergibt sich somit welche Laufzeit?	○ Θ(n) ○ Θ(n log n) ○ $\Theta(n^2)$

3 Algorithmenentwurf

3.1	Beschreibe kurz wie aus einem gegebenen sortierten Array ein AVL-Baum in O(n) Zeit konstruiert werden kann.	

Einladungsmail an die tubs-Adresse spätestens morgen an alle, die für die Prüfung angemeldet sind.

Fragerunde

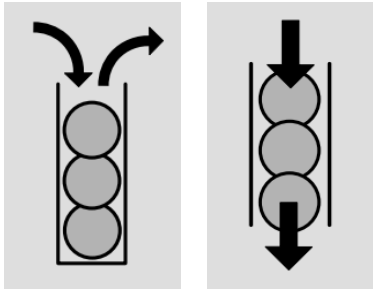
Themen

- Laufzeiten dynamische Datenstrukturen
- Wachstum von Funktionen
  - Vergleichen von Klassen
  - Bestimmen der Klassen
  - Beweise
- Mediane
- Algorithmenverständnis

Laufzeiten dynamische Datenstrukturen

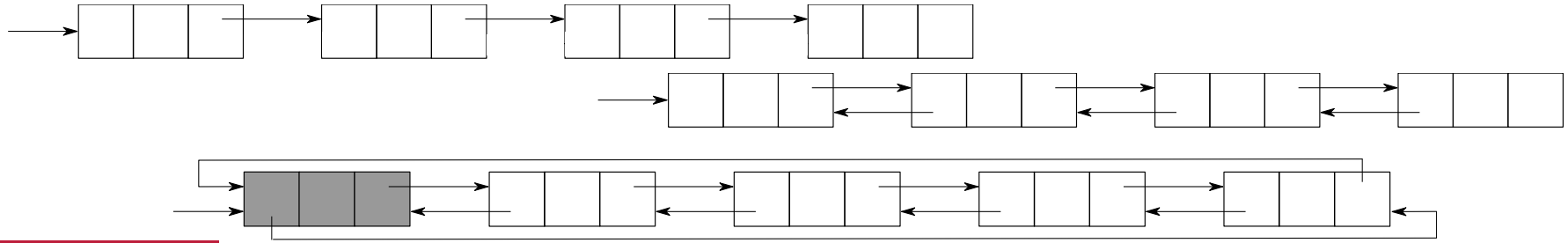
Laufzeiten – dynamische Datenstrukturen (ohne Sortierung)

Datentyp	Stack	Queue
Einfügen	$O(1)$	$O(1)$
Nächstes Element	$O(1)$	$O(1)$
Löschen	$O(1)$	$O(1)$

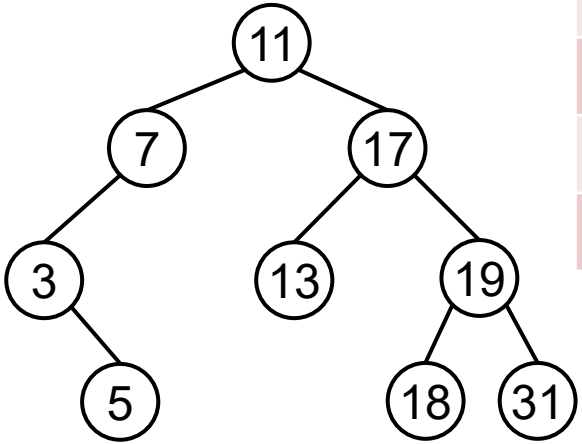


Laufzeiten – dynamische Datenstrukturen (ohne Sortierung)

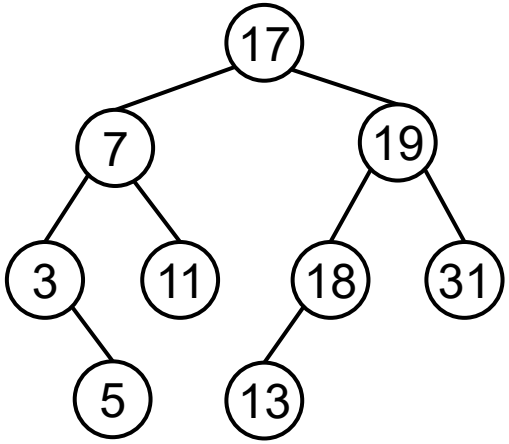
Datentyp	Listen		
Subtyp	Einfach	Doppelt	Zyklisch
Suchen	$O(n)$	$O(n)$	$O(n)$
Einfügen	$O(1)$	$O(1)$	$O(1)$
Löschen	$O(n)$	$O(1)$	$O(1)$
Traversierung	$O(n)$	$O(n)$	$O(n)$



Laufzeiten – dynamische Datenstrukturen (mit Sortierung)



Datentyp	Suchbäume	
Subtyp	-	AVL
Suchen	$O(h)$	$O(\log n)$
Einfügen	$O(h)$	$O(\log n)$
Löschen	$O(h)$	$O(\log n)$
Traversierung	$O(n)$	$O(n)$



Frage: Wie schnell kann ein bel. bin. Suchbaum in einen AVL-Baum umstrukturiert werden?

Laufzeiten – dynamische Datenstrukturen (partielle Sortierung)

Datentyp	(Max)Heaps	Fibonacci-Heaps
Einfügen	$O(\log n)$	$O(1)$
Löschen	$O(\log n)$	$O(\log n)^*$
Minimum/Maximum	$O(1)$	$O(1)$
Extrahiere Min/Max	$O(\log n)$	$O(\log n)^*$

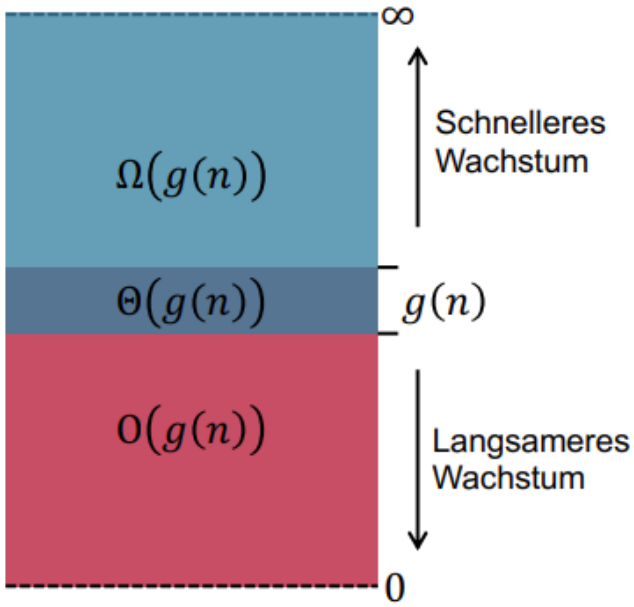
*: Amortisierte Laufzeit, d.h. Durchschnitt über viele solche Operationen

Wachstum von Funktionen

Wachstum von Funktionen (Relationen von Klassen)

Wachstum von Funktionen

Vergleichen von Klassen



Hierarchie-Ausschnitt:

$$O(1) \subset O(\log^a n) \subset O\left(n^b\right) \subset O(c^n) \subset O(n!) \subset O(n^n)$$

Bei Ω dreht sich das Inklusionszeichen um!

Wo passt dort nun $O(n \log n)$ rein?

Wie steht das zu $O(n \log \log n)$?

Wir wissen:

$$O(\log \log n) \subset O(\log n)$$

Also muss gelten:

$$O(n \log \log n) \subset O(n \log n)$$

Wachstum von Funktionen (Bestimmen der Klasse)

Wachstum von Funktionen

Bestimmen von Klassen

Laufzeiten Merkzettel

1 Definitionen

O-Notation Gibt eine obere Schranke für Funktionen. Gilt $f(n) \in O(g(n))$, so wächst $f(n)$ (asymptotisch) nicht schneller als $g(n)$ denn:

Es existieren zwei Konstanten $c \in \mathbb{R}^+$ und $n_0 \in \mathbb{N}$, sodass für alle $n \geq n_0$ die Ungleichung $0 \leq f(n) \leq c \cdot g(n)$ gilt.

Ω-Notation Gibt eine untere Schranke für Funktionen. Gilt $f(n) \in \Omega(g(n))$, so wächst $f(n)$ (asymptotisch) nicht langsamer als $g(n)$ denn:

Es existieren zwei Konstanten $c \in \mathbb{R}^+$ und $n_0 \in \mathbb{N}$, sodass für alle $n \geq n_0$ die Ungleichung

2 O-Notation

Tipps zum Abschätzen von Funktionen bei der O-Notation:

1. Bei Polynomen können Subtrahenden einfach ignoriert werden. Das Weglassen macht die Funktion nur größer.
2. Bei Polynomen können alle Exponenten von (positiven) Summanden auf den Grad des Polynoms hochgestuft werden. Das macht die Funktion größer.
3. Bei Funktionen die kein Polynom sind, können andere Methoden zum Abschätzen vorteilhaft sein, z.B. das Benutzen von monoton-wachsenden Funktionen (Logarithmieren, Potenzieren[1], Wurzelziehen, etc.).

3 Ω-Notation

Tipps zum Abschätzen von Funktionen bei der Ω-Notation:

1. Bei Polynomen können (positive) Summanden einfach ignoriert werden. Das Weglassen macht die Funktion nur kleiner.
2. Bei Polynomen können alle Exponenten von Subtrahenden **nicht** auf den Grad des Polynoms hochgestuft werden. Das würde die Funktion zwar kleiner machen, aber unter Umständen wird dadurch die Funktion negativ.
3. Bei Funktionen die kein Polynom sind, können andere Methoden zum Abschätzen vorteilhaft sein, z.B. das Benutzen von monoton-wachsenden Funktionen (Logarithmieren, Potenzieren, Wurzelziehen, etc.).

Wachstum von Funktionen

Bestimmen von Klassen

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n \in \Theta(n)$$

Beweis (O-Notation)

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n$$

Wachstum von Funktionen

Bestimmen von Klassen

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n \in \Theta(n)$$

Beweis (O-Notation)

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n \leq \frac{3n^2 + 23n}{3n\log n - 6n} \cdot \log n$$

Wachstum von Funktionen

Bestimmen von Klassen

$$\frac{3n^2 - 5n\log n + 23n\ - 40}{3n\log n - 6n} \cdot \log n \in \Theta(n)$$

Beweis (O-Notation)

$$\frac{3n^2 - 5n\log n + 23n\ - 40}{3n\log n - 6n} \cdot \log n \leq \frac{3n^2 + 23n}{3n\log n - 6n} \cdot \log n \leq \frac{26n^2}{3n\log n - 6n} \cdot \log n$$

Wachstum von Funktionen

Bestimmen von Klassen

$$\frac{3n^2 - 5n\log n + 23n\ - 40}{3n\log n - 6n} \cdot \log n \in \Theta(n)$$

Beweis (O-Notation)

$$\frac{3n^2 - 5n\log n + 23n\ - 40}{3n\log n - 6n} \cdot \log n \leq \frac{3n^2 + 23n}{3n\log n - 6n} \cdot \log n \leq \frac{26n^2}{3n\log n - 6n} \cdot \log n$$

$$\frac{26n^2}{3n\log n - 6n} \cdot \log n \leq \frac{26n^2}{3n\log n - n\log n} \cdot \log n$$

Ab $n_0 \geq 2^6 = 32$, da $6 \leq \log n$ gelten muss!

Wachstum von Funktionen

Bestimmen von Klassen

$$\frac{3n^2 - 5n\log n + 23n\ - 40}{3n\log n - 6n} \cdot \log n \in \Theta(n)$$

Also $n_0 \geq 32$ und $c_1 = 13$.

Beweis (O-Notation)

$$\frac{3n^2 - 5n\log n + 23n\ - 40}{3n\log n - 6n} \cdot \log n \leq \frac{3n^2 + 23n}{3n\log n - 6n} \cdot \log n \leq \frac{26n^2}{3n\log n - 6n} \cdot \log n$$

$$\frac{26n^2}{3n\log n - 6n} \cdot \log n \leq \frac{26n^2}{3n\log n - n\log n} \cdot \log n = 13n$$

Ab $n_0 \geq 2^6 = 32$, da $6 \leq \log n$ gelten muss!

Wachstum von Funktionen

Bestimmen von Klassen

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n \in \Theta(n)$$

Also $n_0 \geq 32$ und $c_1 = 13$.

Beweis (Ω-Notation)

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n$$

Wachstum von Funktionen

Bestimmen von Klassen

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n \in \Theta(n)$$

Also $n_0 \geq 32$ und $c_1 = 13$.

Beweis (Ω-Notation)

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n \geq \frac{3n^2 - 5n\log n + 23n - 40}{3n\log n} \cdot \log n$$

Wachstum von Funktionen

Bestimmen von Klassen

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n \in \Theta(n)$$

Also $n_0 \geq 32$ und $c_1 = 13$.

und $c_2 = \frac{2}{3}$.

Beweis (Ω-Notation)

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n \geq \frac{3n^2 - 5n\log n + 23n - 40}{3n\log n} \cdot \log n$$

$$\geq \frac{1}{3n}(3n^2 - 5n\log n + 23n - 40)$$

Ab $n_0 \geq 2$, da $40 \leq 20n$ gelten muss! →

$$\geq \frac{1}{3n}(3n^2 - 5n\log n + 23n - 20n) \geq \frac{1}{3n}(3n^2 - 5n\log n)$$

Ab $n_0 \geq 23$, da $5\log n \leq n$ gelten muss! →

$$\geq \frac{1}{3n}(3n^2 - n^2) \geq \frac{2}{3}n$$

Tipps für Wachstumsanalyse

Mit „Abuse of Notation“:

$$O\big(f(n)\big) + O\big(g(n)\big) = O\big(\max\big(f(n), g(n)\big)\big)$$

$$O\big(f(n)\big) \cdot O\big(g(n)\big) = O\big(f(n) \cdot g(n)\big)$$

$$\frac{O\big(f(n)\big)}{O\big(g(n)\big)} = O\left(\frac{f(n)}{g(n)}\right)$$

$$\Omega\big(f(n)\big) + \Omega\big(g(n)\big) = \Omega\big(\max\big(f(n), g(n)\big)\big)$$

$$\Omega\big(f(n)\big) \cdot \Omega\big(g(n)\big) = \Omega\big(f(n) \cdot g(n)\big)$$

$$\frac{\Omega\big(f(n)\big)}{\Omega\big(g(n)\big)} = \Omega\left(\frac{f(n)}{g(n)}\right)$$

Tipps für Wachstumsanalyse

$O(f(n)) - O(g(n)) = O(\max(f(n), g(n))$?

Achtung: Subtrahieren funktioniert so nur, wenn $f(n) \notin O(g(n))$!

Damit also:

$$\frac{3n^2 - 5n\log n + 23n - 40}{3n\log n - 6n} \cdot \log n$$

Wie erlangt man die Konstanten?

$$\frac{\Theta(n^2) - \Theta(n\log n) + \Theta(n) - \Theta(1)}{\Theta(n\log n) - \Theta(n)} \cdot \Theta(\log n) = \frac{\Theta(n^2)}{\Theta(n\log n)} \cdot \Theta(\log n)$$

$$= \Theta\left(\frac{n^2}{n\log n} \cdot \log n\right) = \Theta(n)$$

Wachstum von Funktionen (Beweise)

Satz 3.12

Seien $f, g: \mathbb{N} \to \mathbb{R}$, dann gilt $f(n) \in O\big(g(n)\big) \Leftrightarrow g(n) \in \Omega\big(f(n)\big)$

$f(n) \in O\big(g(n)\big)$

$\Leftrightarrow \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N}: \forall n \geq n_0: \quad 0 \leq f(n) \leq c \cdot g(n)$

$\Leftrightarrow \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N}: \forall n \geq n_0: \quad 0 \leq \frac{1}{c} \cdot f(n) \leq g(n)$

$\Leftrightarrow \exists c' \in \mathbb{R}^+, n_0 \in \mathbb{N}: \forall n \geq n_0: \quad 0 \leq c' \cdot f(n) \leq g(n)$ (nämlich $c' = \frac{1}{c}$)

$\Leftrightarrow g(n) \in \Omega\big(f(n)\big)$

Mediane

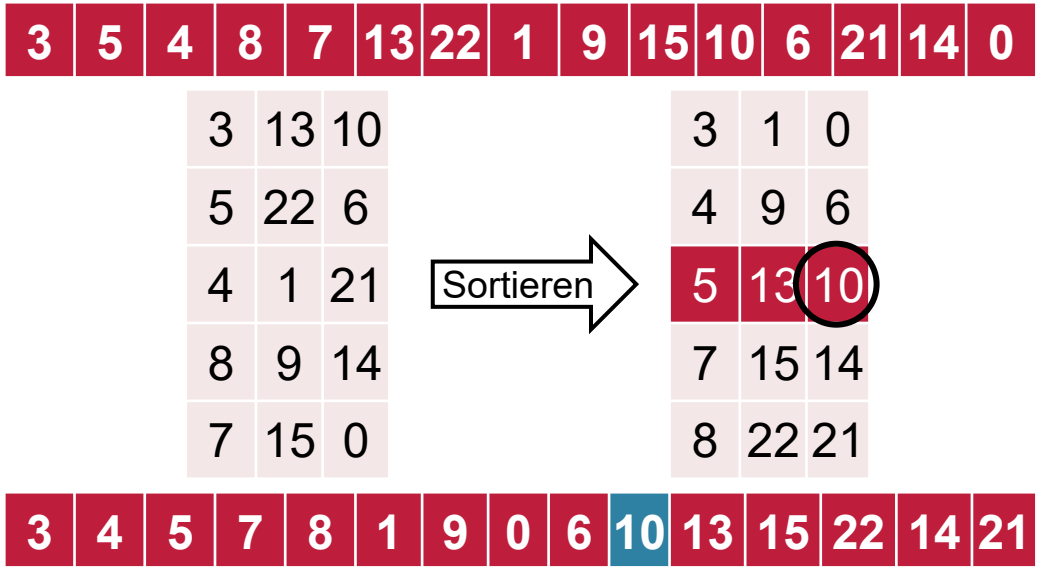
Mediane – Algorithmus (III)

2. Ist das schneller als $\Theta(n \log n)$?

Wie bei Quicksort kann größeres Teilarray $n - 1$ Elemente enthalten. Dadurch Laufzeit $\Omega(n^2)$

Idee: Wähle besseres Pivotelement

1. Teile Zahlen in 5er Gruppen
2. Bestimme Median in jeder Gruppe
3. Bestimme Median der Mediane m
4. Benutze m als Pivotelement



Mediane - Laufzeit

Wir haben also als Laufzeit:

$$T(n) = \underbrace{T\left(\frac{1}{5}n\right)}_{\text{Bestimmung Median der Mediane}} + \underbrace{T\left(\frac{7}{10}n\right)}_{\text{Rekursion in linke/rechte Teilhälfte}} + \underbrace{\Theta(n)}_{\text{Aufteilen in Gruppen/ Pivotisieren/ …}}$$

Mediane – Analyse

Wie viele Zahlen gibt es, die größer/kleiner als m sind?

Sortiere die t 5er Gruppen **gedanklich** nach deren Median

$\leq m$	$\leq m$	$\leq m$	$\leq m$	m	$\geq m$	$\geq m$	$\geq m$	$\geq m$	$\geq m$

Mediane – Analyse

Wie viele Zahlen gibt es, die größer/kleiner als m sind?

Sortiere die t 5er Gruppen **gedanklich** nach deren Median

$\leq m$	$\leq m$	$\leq m$	$\leq m$	$\leq m$					
$\leq m$	$\leq m$	$\leq m$	$\leq m$	$\leq m$					
$\leq m$	$\leq m$	$\leq m$	$\leq m$	m	$\geq m$	$\geq m$	$\geq m$	$\geq m$	$\geq m$

Der rote Bereich enthält nur Elemente, die höchstens m sind. Wie viele sind das?

Mediane – Analyse

$\leq m$	$\leq m$	$\leq m$	$\leq m$	$\leq m$					
$\leq m$	$\leq m$	$\leq m$	$\leq m$	$\leq m$					
$\leq m$	$\leq m$	$\leq m$	$\leq m$	m	$\geq m$	$\geq m$	$\geq m$	$\geq m$	$\geq m$

Der rote Bereich enthält nur Elemente, die höchstens m sind. Wie viele sind das?

Nehmen wir an wir haben t Gruppen, so ist der Median m in der $\left\lceil \frac{t}{2} \right\rceil$-ten Gruppe.

Bei 5er Gruppen, sind pro Gruppe mindestens 3 Elemente $\leq m$.

Entsprechend gibt es mindestens $3 \cdot \left\lceil \frac{t}{2} \right\rceil$ viele Elemente $\leq m$.

Damit gibt es maximal $n - 3 \cdot \left\lceil \frac{t}{2} \right\rceil \leq n - \frac{3}{2}t \leq n - \frac{3}{2} \cdot \frac{n}{5} = \frac{7}{10}n$ Elemente größer als m.

Analog: Maximal $\frac{7}{10}n$ Elemente kleiner als m.

Algorithmenverständnis

Algorithmenverständnis

Gegeben zwei Stacks S_0, S_1 (und eine Objekt-Variablen)
Annahme: Stack S_1 ist leer.

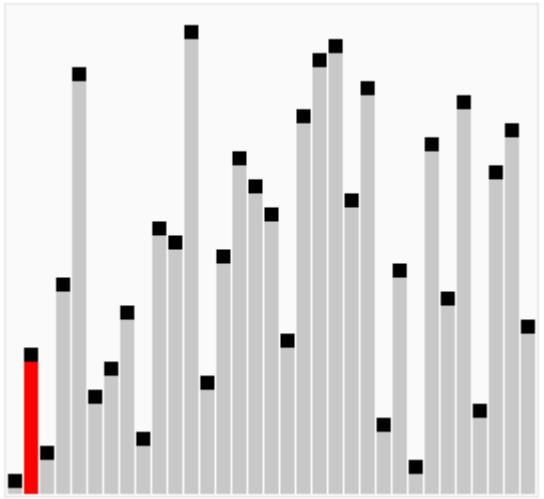
Was tut dieser Algorithmus?

1. Function COCKTAILSHAKER(S_0, S_1)
2. For $j = 1$ to $\frac{n}{2}$ do
3. $item :=$ POP(S_0)
4. While !(IS_EMPTY(S_0))
5. if (TOP(S_0) $>$ $item$)
6. PUSH(S_1, POP(S_0)
7. else
8. PUSH(S_1, $item$)
9. $item :=$ POP(S_0)
10. Wiederhole 4. bis 9. und tausche dabei sowohl S_0 und S_1 als auch $>$ und $<$

Algorithmenverständnis

Gegeben zwei Stacks S_0, S_1 (und eine Objekt-Variablen)
Annahme: Stack S_1 ist leer.

1. Function COCKTAILSHAKERSORT(S_0, S_1)
2. For $j = 1$ to $\frac{n}{2}$ do
3. $item :=$ POP(S_0)
4. While !(IS_EMPTY(S_0))
5. if (TOP(S_0) $>$ $item$)
6. PUSH(S_1, POP(S_0)
7. else
8. PUSH(S_1, $item$)
9. $item :=$ POP(S_0)
10. Wiederhole 4. bis 9. und tausche dabei sowohl S_0 und S_1 als auch $>$ und $<$



Algorithmenverständnis – Laufzeit

Gegeben zwei Stacks S_0, S_1 (und eine Objekt-Variablen)
Annahme: Stack S_1 ist leer.

1. Function CocktailShakerSort(S_0, S_1)
2. For $j = 1$ to $\frac{n}{2}$ do
3. $item :=$ POP(S_0)
4. While !(IS_EMPTY(S_0))
5. if (TOP(S_0) $> item$)
6. PUSH(S_1, POP(S_0)
7. else
8. PUSH(S_1, $item$)
9. $item :=$ POP(S_0)
10. Wiederhole 4. bis 9. und tausche dabei sowohl S_0 und S_1 als auch $>$ und $<$.

Laufzeit:
- Schleife Zeile 2: $\sim n$ Iterationen
- Schleife Zeile 4: $\sim n$ Iterationen
- Zeile 10: Laufzeit von 4.-9.
- Zeilen 3, 5-9: O(1)

Insgesamt also:
$n \cdot \left(1 + 2 \cdot n \cdot O(1)\right) = O(n^2)$

Fragen?

Bei Fragen per Mails:
Immer an mich (aschmidt@ibr.cs.tu-bs.de) **und** Matthias (konitzny@ibr.cs.tu-bs.de).

Dabei beachten:
Fragen wie „Ich habe Thema X nicht verstanden. Kannst du das noch mal erklären?“ helfen weder euch, noch uns! – Warum?

Viel Erfolg
und
frohes Schaffen!