Algorithmentechnik — Übung 1

http://i11www.ira.uka.de/teaching/WS_0607/algotech

Robert Görke (rgoerke@ira.uka.de)

WS 0607

Übungsblätter, Termine, Klausurbonus, etc.

- Algorithmentechnik (Hauptklausur): 01.03.07
- Algorithmentechnik (Wiederholerklausur): 12.04.07

- Übungsblätter erscheinen Dienstags (14-tg.)
- Abgabe Mittwochs bis 15:30 im 3. Stock Fasaneng. Ostflügel
- *erfolgreich* = mind. 50% Punkte = 0.5 Klausurpunkte (v. 60)
- maximal ein Teilnotenschritt
- alle Blätter *erfolgreich* $\Rightarrow$ ein Teilnotenschritt
- Zweiergruppen ausdrücklich erwünscht

- Inhalt der Übungen: Lösungen der Blätter + Diverses

- $O(g(n)) =$ asymp. obere Schranke $=$
  $\{f(n) : \exists c \in \mathbb{R}_+ : 0 \leq f(n) \leq cg(n), \forall n \geq n_0\}$

- $O(g(n)) =$ asymp. obere Schranke $=$
  $\{f(n) : \exists c \in \mathbb{R}_+ : 0 \leq f(n) \leq cg(n), \forall n \geq n_0\}$
- $\Omega(g(n)) =$ asymp. untere Schranke $=$
  $\{f(n) : \exists c \in \mathbb{R}_+ : 0 \leq cg(n) \leq f(n), \forall n \geq n_0\}$

- $O(g(n)) =$ asymp. obere Schranke $=$
  $\{f(n) : \exists c \in \mathbb{R}_+ : 0 \leq f(n) \leq cg(n), \forall n \geq n_0\}$
- $\Omega(g(n)) =$ asymp. untere Schranke $=$
  $\{f(n) : \exists c \in \mathbb{R}_+ : 0 \leq cg(n) \leq f(n), \forall n \geq n_0\}$
- $\Theta(g(n)) =$ asymp. scharfe Schranke $=$
  $\{f(n) : \exists c_1, c_2 \in \mathbb{R}_+ : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}$

- $O(g(n)) =$ asymp. obere Schranke $=$
  $\{f(n) : \exists c \in \mathbb{R}_+ : 0 \leq f(n) \leq cg(n), \forall n \geq n_0\}$
- $\Omega(g(n)) =$ asymp. untere Schranke $=$
  $\{f(n) : \exists c \in \mathbb{R}_+ : 0 \leq cg(n) \leq f(n), \forall n \geq n_0\}$
- $\Theta(g(n)) =$ asymp. scharfe Schranke $=$
  $\{f(n) : \exists c_1, c_2 \in \mathbb{R}_+ : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}$
- $o(g(n)) =$ nicht asymp. scharfe obere Schranke $=$
  $f(n) : \forall c \in \mathbb{R}_+ \exists n_0 : 0 \leq f(n) < cg(n), \forall n \geq n_0$

- $O(g(n)) =$ asymp. obere Schranke $=$
  $\{f(n) : \exists c \in \mathbb{R}_+ : 0 \leq f(n) \leq cg(n), \forall n \geq n_0\}$
- $\Omega(g(n)) =$ asymp. untere Schranke $=$
  $\{f(n) : \exists c \in \mathbb{R}_+ : 0 \leq cg(n) \leq f(n), \forall n \geq n_0\}$
- $\Theta(g(n)) =$ asymp. scharfe Schranke $=$
  $\{f(n) : \exists c_1, c_2 \in \mathbb{R}_+ : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}$
- $o(g(n)) =$ nicht asymp. scharfe obere Schranke $=$
  $f(n) : \forall c \in \mathbb{R}_+ \exists n_0 : 0 \leq f(n) < cg(n), \forall n \geq n_0$
- $\omega(g(n)) =$ nicht asymp. scharf untere Schranke $=$
  $f(n) : \forall c \in \mathbb{R}_+ \exists n_0 : 0 \leq cg(n) < f(n), \forall n \geq n_0$

Ein naiver Sortieralgorithmus

Eingabe: Unsortiertes Array von n verschiedenen Zahlen $A = (A[1], \ldots, A[n])$

Ausgabe: Sortiertes Array B der Zahlen aus A

1. $B \leftarrow A$
2. Solange B unsortiert
3. $\quad\quad B \leftarrow$ zufällige Permutation von A

Ein naiver Sortieralgorithmus

Eingabe: Unsortiertes Array von n verschiedenen Zahlen $A = (A[1], \ldots, A[n])$
Ausgabe: Sortiertes Array B der Zahlen aus A

1. $B \leftarrow A$
2. Solange B unsortiert
3. $\quad\quad B \leftarrow$ zufällige Permutation von A

Worst-case Laufzeit:

Ein naiver Sortieralgorithmus

Eingabe: Unsortiertes Array von n verschiedenen Zahlen $A = (A[1], \ldots, A[n])$
Ausgabe: Sortiertes Array B der Zahlen aus A

1. $B \leftarrow A$
2. Solange B unsortiert
3. $\quad\quad B \leftarrow$ zufällige Permutation von A

Worst-case Laufzeit: unbeschränkt

Ein naiver Sortieralgorithmus

Eingabe: Unsortiertes Array von n verschiedenen Zahlen $A = (A[1], \ldots, A[n])$
Ausgabe: Sortiertes Array B der Zahlen aus A

1. $B \leftarrow A$
2. Solange B unsortiert
3. $\quad\quad B \leftarrow$ zufällige Permutation von A

Worst-case Laufzeit: unbeschränkt
Average-case Laufzeit:

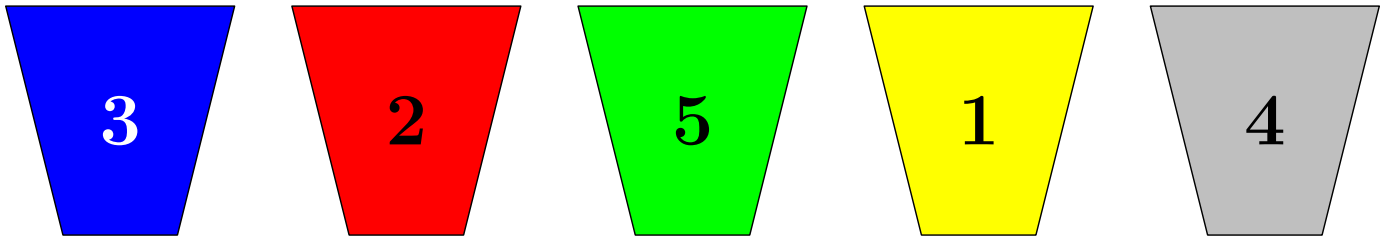
Ein naiver Sortieralgorithmus

Eingabe: Unsortiertes Array von n verschiedenen Zahlen $A = (A[1], \ldots, A[n])$
Ausgabe: Sortiertes Array B der Zahlen aus A

1. $B \leftarrow A$
2. Solange B unsortiert
3. $\quad B \leftarrow$ zufällige Permutation von A

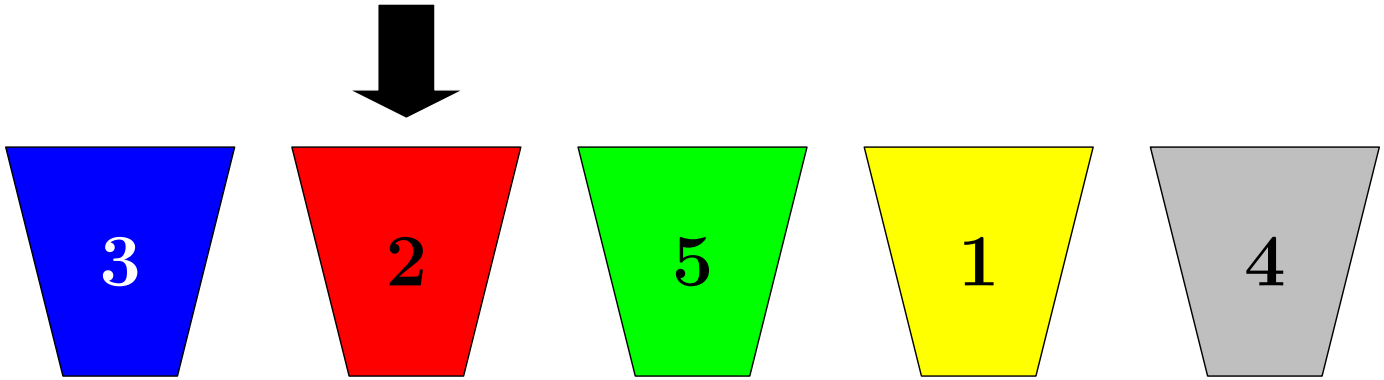
Worst-case Laufzeit: unbeschränkt
Average-case Laufzeit: $\in \Omega(n!)$

Beispiellauf



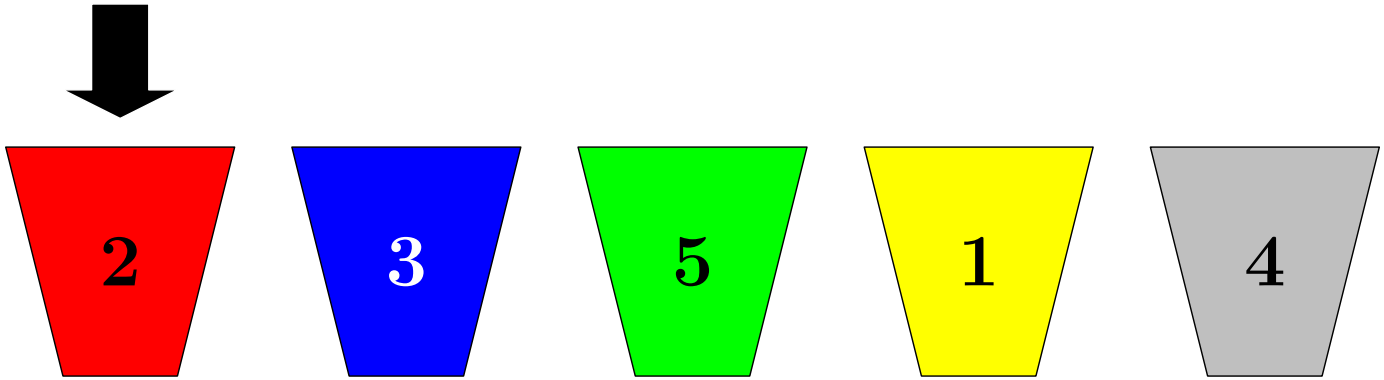
Wie sortiert ein holländischer Gartenzwerg Blumentöpfe?

Beispiellauf



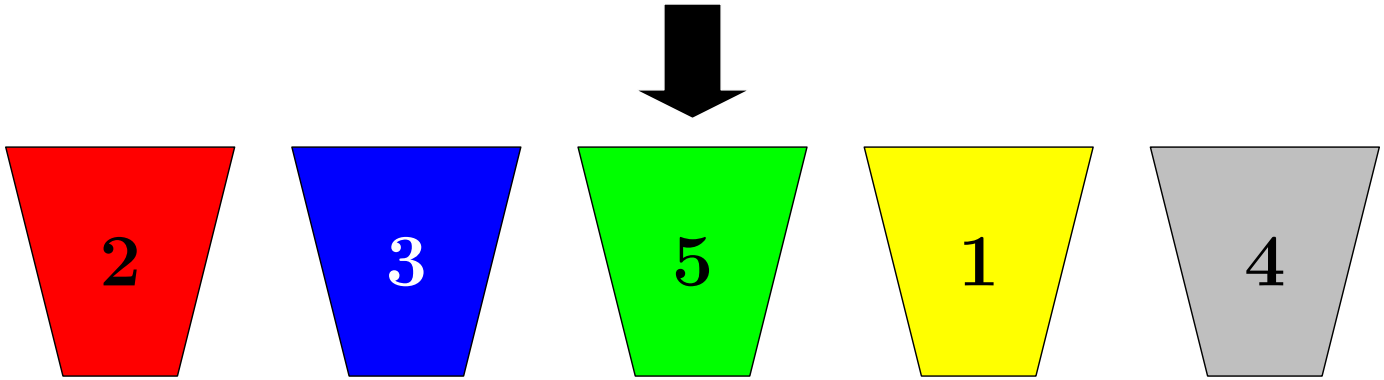
Vergleiche den zweiten Topf mit dem ersten . . .

Beispiellauf



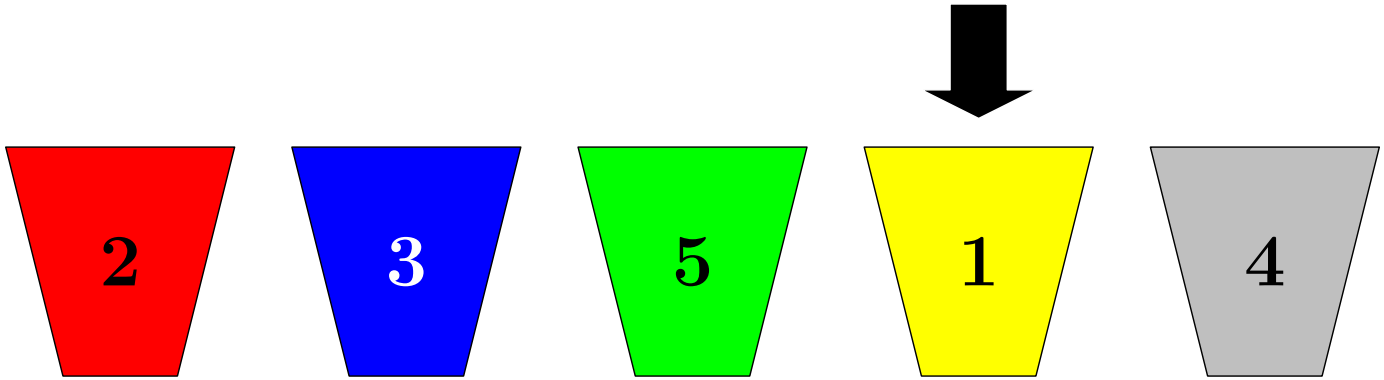
Tauschen!

Beispiellauf



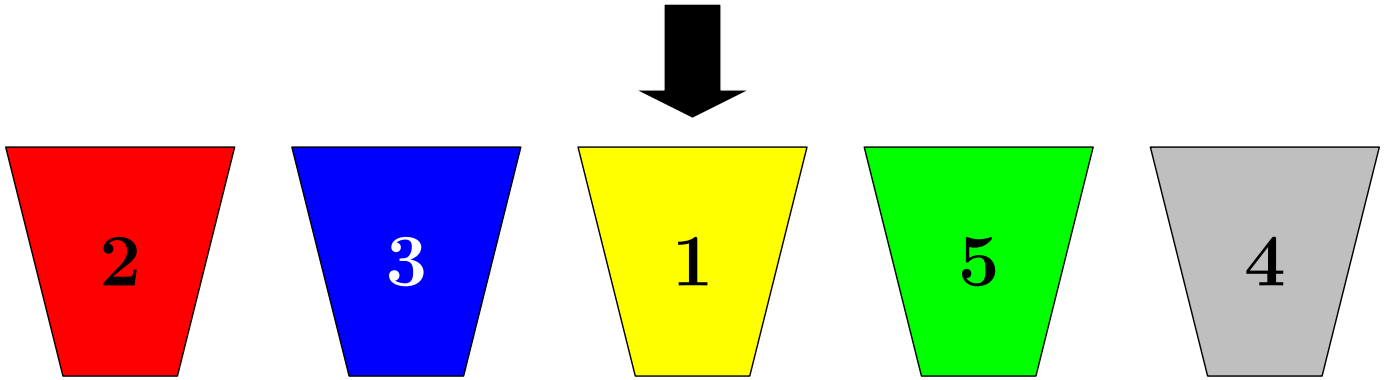
Betrachte nun den dritten . . .

Beispiellauf



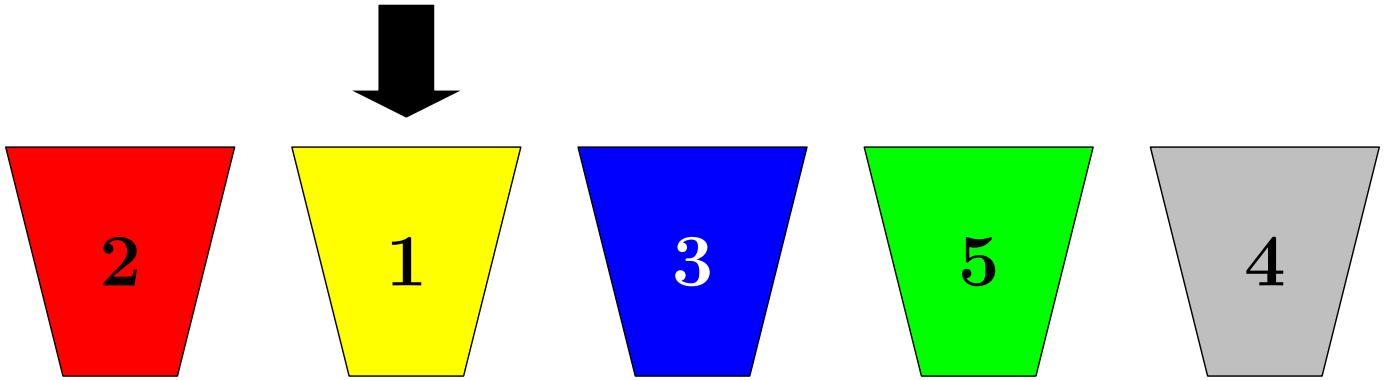
. . . und den vierten . . .

Beispiellauf



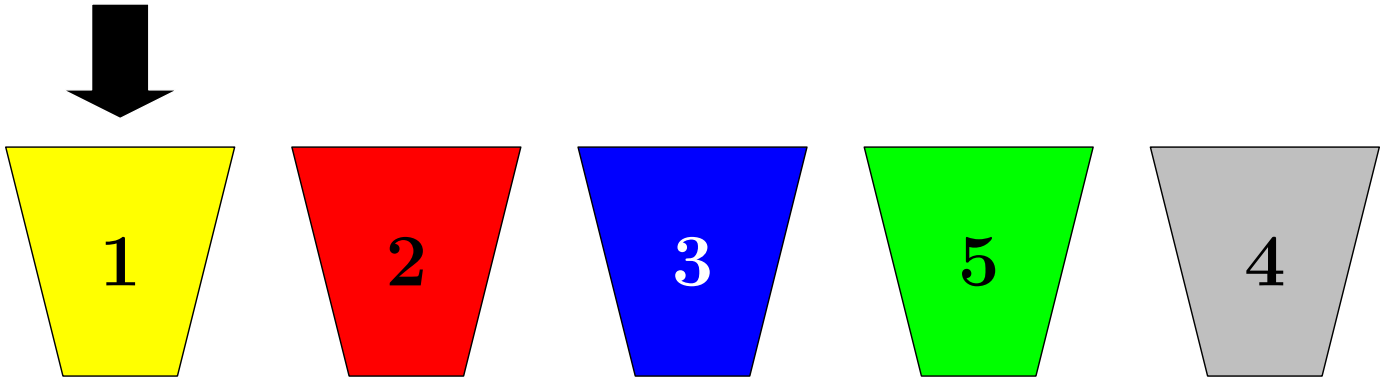
. . . und den vierten . . .

Beispiellauf



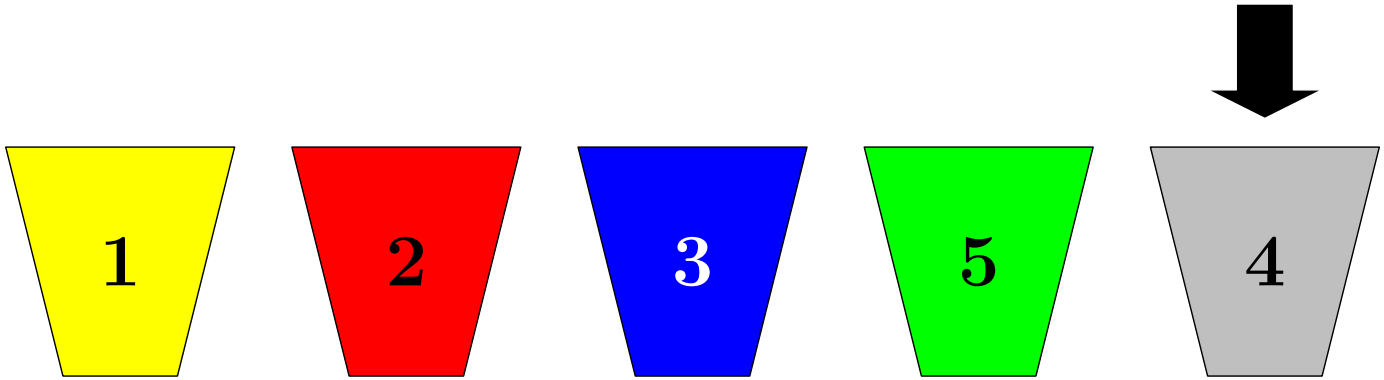
. . . und den vierten . . .

Beispiellauf



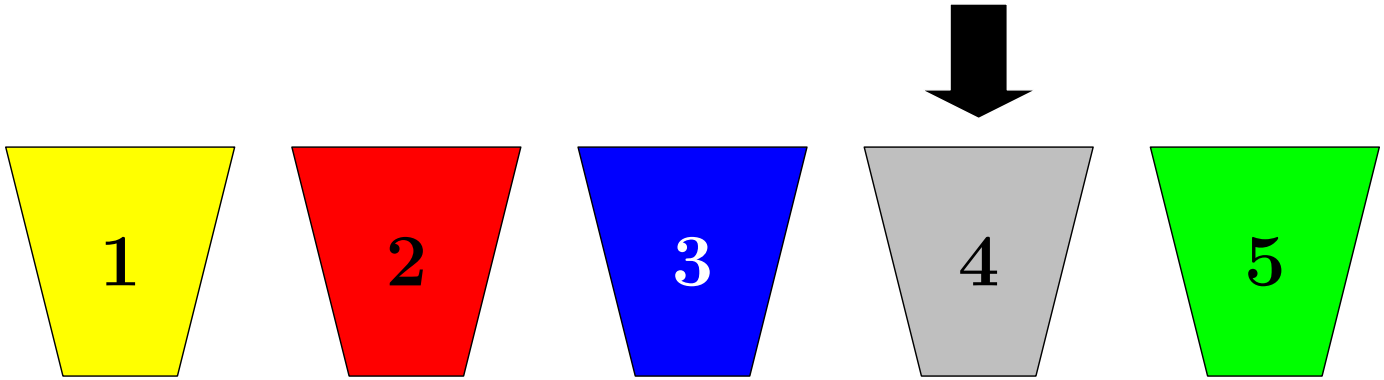
. . . und den vierten . . .

Beispiellauf



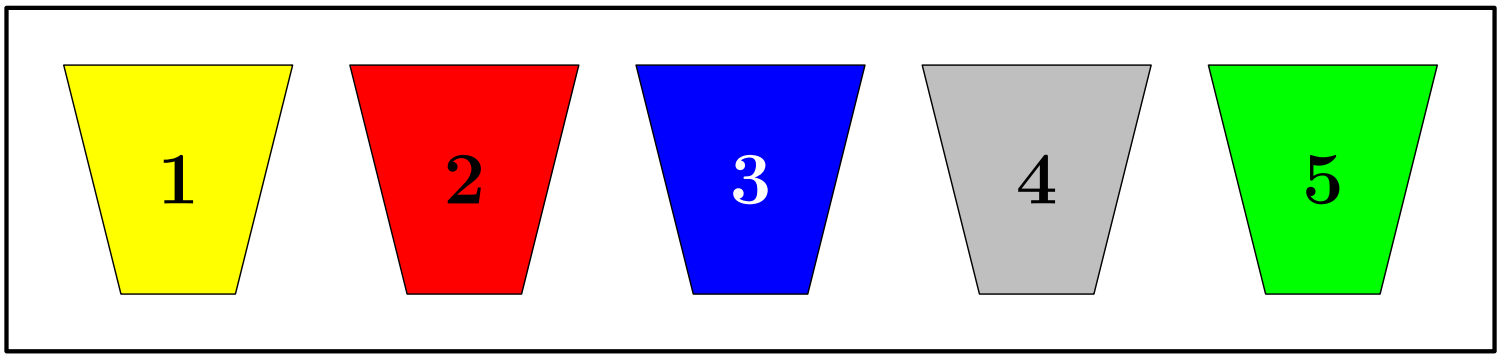
. . . und den letzten.

Beispiellauf



. . . und den letzten.

Beispiellauf



Fertig!

Pseudocode

Eingabe: Unsortiertes Array von n verschiedenen Zahlen $A = (A[1], \ldots, A[n])$
Ausgabe: Sortiertes Array A

1. $i \leftarrow 2$
2. **Solange** $i \leq n$
3. **Wenn** $A[i-1] \leq A[i]$
4. $i \leftarrow i + 1$
5. **Sonst**
6. Tausche $A[i-1]$ und $A[i]$
7. $i \leftarrow i - 1$
8. **Wenn** $i = 1$
9. $i \leftarrow 2$

Ganzheitlicher Ansatz

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

Scharfe Schranke für den amortisierten Aufwand?

Ganzheitlicher Ansatz

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

Scharfe Schranke für den amortisierten Aufwand?

Ganzheitlicher Ansatz

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

Scharfe Schranke für den amortisierten Aufwand?

Wie oft kippt das erste Bit?

Ganzheitlicher Ansatz

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

Scharfe Schranke für den amortisierten Aufwand?

Wie oft kippt das erste Bit? Jedes Mal (jedes 2^0. mal.)

Ganzheitlicher Ansatz

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

Scharfe Schranke für den amortisierten Aufwand?

Wie oft kippt das erste Bit? Jedes Mal (jedes 2^0. mal.)
Wie oft kippt das zweite Bit?

Ganzheitlicher Ansatz

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

Scharfe Schranke für den amortisierten Aufwand?

Wie oft kippt das erste Bit? Jedes Mal (jedes 2^0. mal.)
Wie oft kippt das zweite Bit? Jedes 2^1. mal.

Ganzheitlicher Ansatz

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

Scharfe Schranke für den amortisierten Aufwand?

Wie oft kippt das erste Bit? Jedes Mal (jedes 2^0. mal.)
Wie oft kippt das zweite Bit? Jedes 2^1. mal.
Wie oft kippt das dritte Bit?

Ganzheitlicher Ansatz

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

Scharfe Schranke für den amortisierten Aufwand?

Wie oft kippt das erste Bit? Jedes Mal (jedes 2^0. mal.)
Wie oft kippt das zweite Bit? Jedes 2^1. mal.
Wie oft kippt das dritte Bit? Jedes 2^2. mal.

Ganzheitlicher Ansatz

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

Scharfe Schranke für den amortisierten Aufwand?

Wie oft kippt das erste Bit? Jedes Mal (jedes 2^0. mal.)
Wie oft kippt das zweite Bit? Jedes 2^1. mal.
Wie oft kippt das dritte Bit? Jedes 2^2. mal.
$\vdots$

Ganzheitlicher Ansatz

Es sei ein Zähler gegeben, der mit Hilfe von Bits binär von 0 bis n zählt. Eine Operation sei das Kippen eines einzelnen Bits.

Scharfe Schranke für den amortisierten Aufwand?

Wie oft kippt das erste Bit? Jedes Mal (jedes 2^0. mal.)
Wie oft kippt das zweite Bit? Jedes 2^1. mal.
Wie oft kippt das dritte Bit? Jedes 2^2. mal.
$\vdots$

$$\Rightarrow T(n) = \sum_{i=1}^{\lceil \log_2 n \rceil} \left\lfloor \frac{n}{2^{i-1}} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n \in O(n)$$

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	0\$	0\$	0\$	0\$
Zähler	0	0	0	0

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	0\$	0\$	0\$	1\$
Zähler	0	0	0	1

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	0\$	0\$	1\$	0\$
Zähler	0	0	1	0

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	0\$	0\$	1\$	1\$
Zähler	0	0	1	1

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	0\$	1\$	0\$	0\$
Zähler	0	1	0	0

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	0\$	1\$	0\$	1\$
Zähler	0	1	0	1

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	0\$	1\$	1\$	0\$
Zähler	0	1	1	0

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	0\$	1\$	1\$	1\$
Zähler	0	1	1	1

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	1\$	0\$	0\$	0\$
Zähler	1	0	0	0

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	1\$	0\$	0\$	1\$
Zähler	1	0	0	1

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	1\$	0\$	1\$	0\$
Zähler	1	0	1	0

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	1\$	0\$	1\$	1\$
Zähler	1	0	1	1

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	1\$	1\$	0\$	0\$
Zähler	1	1	0	0

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	1\$	1\$	0\$	1\$
Zähler	1	1	0	1

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	1\$	1\$	1\$	0\$
Zähler	1	1	1	0

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	1\$	1\$	1\$	1\$
Zähler	1	1	1	1

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	1\$	1\$	1\$	1\$
Zähler	1	1	1	1

Jede entstehende 1 bekommt Guthaben 1\$, und finanziert damit selbstständig seinen Wechsel zur 0

Buchungsmethode

Beobachtung: Es wird bei jedem Inkrement genau *eine* 0 zur 1.
Definiere: Gebühren pro Inkrement = 2\$, Überschuss bekommt die entstandene 1

Guthaben	1\$	1\$	1\$	1\$
Zähler	1	1	1	1

Jede entstehende 1 bekommt Guthaben 1\$, und finanziert damit selbstständig seinen Wechsel zur 0
Amortisierter Aufwand: $2 = O(1)$ pro Operation $\Rightarrow T(n) \in O(n)$

Potentialmethode (Theorie)

- Für $i = 1, 2, \ldots, n$ seien:
  c_i: die tatsächlichen Kosten der i-ten Operation
  D_i: die Datenstruktur nach der i-ten Operation (angewandt auf Datenstruktur D_{i-1})

Potentialmethode (Theorie)

- Für $i = 1, 2, \ldots, n$ seien:
  c_i: die tatsächlichen Kosten der i-ten Operation
  D_i: die Datenstruktur nach der i-ten Operation (angewandt auf Datenstruktur D_{i-1})
- Definiere *Potentialfunktion* $\mathcal{C} : \{D_i\} \to \mathbb{R}$ mit $\mathcal{C} : D_i \mapsto \mathcal{C}(D_i)$, sie bildet Zustand auf Potential ab

Potentialmethode (Theorie)

- Für $i = 1, 2, \ldots, n$ seien:
  c_i: die tatsächlichen Kosten der i-ten Operation
  D_i: die Datenstruktur nach der i-ten Operation (angewandt auf Datenstruktur D_{i-1})
- Definiere *Potentialfunktion* $\mathcal{C} : \{D_i\} \to \mathbb{R}$ mit $\mathcal{C} : D_i \mapsto \mathcal{C}(D_i)$, sie bildet Zustand auf Potential ab
- Die amortisierten Kosten der i-ten Operation bzgl. $\mathcal{C}$:
  $\widehat{c_i} := c_i + \mathcal{C}(D_i) - \mathcal{C}(D_{i-1})$

Potentialmethode (Theorie)

- Für $i = 1, 2, \ldots, n$ seien:
  c_i: die tatsächlichen Kosten der i-ten Operation
  D_i: die Datenstruktur nach der i-ten Operation (angewandt auf Datenstruktur D_{i-1})
- Definiere *Potentialfunktion* $\mathcal{C} : \{D_i\} \to \mathbb{R}$ mit $\mathcal{C} : D_i \mapsto \mathcal{C}(D_i)$, sie bildet Zustand auf Potential ab
- Die amortisierten Kosten der i-ten Operation bzgl. $\mathcal{C}$:
  $\widehat{c_i} := c_i + \mathcal{C}(D_i) - \mathcal{C}(D_{i-1})$
- Gefordert: $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$, i.d.R. $\mathcal{C}(D_i) \geq \mathcal{C}(D_0)$

Potentialmethode (Theorie)

- Für $i = 1, 2, \ldots, n$ seien:
  c_i: die tatsächlichen Kosten der i-ten Operation
  D_i: die Datenstruktur nach der i-ten Operation (angewandt auf Datenstruktur D_{i-1})
- Definiere *Potentialfunktion* $\mathcal{C} : \{D_i\} \to \mathbb{R}$ mit $\mathcal{C} : D_i \mapsto \mathcal{C}(D_i)$, sie bildet Zustand auf Potential ab
- Die amortisierten Kosten der i-ten Operation bzgl. $\mathcal{C}$:
  $\widehat{c_i} := c_i + \mathcal{C}(D_i) - \mathcal{C}(D_{i-1})$
- Gefordert: $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$, i.d.R. $\mathcal{C}(D_i) \geq \mathcal{C}(D_0)$
- Gesucht: $\sum_{i=1}^{n} \widehat{c_i}$, (oft über maximales nötiges $\widehat{c_i}$)

Potentialmethode bei Binärzähler

- tatsächlichen Kosten: $1, 2, 3, 1, 2, 4, 1, \cdots \Rightarrow$ verschieden ;)

Potentialmethode bei Binärzähler

- tatsächlichen Kosten: $1, 2, 3, 1, 2, 4, 1, \cdots \Rightarrow$ verschieden ;)
- *Potentialfunktion* $\mathcal{C} : \{D_i\} \longrightarrow \mathbb{N}$ mit $\mathcal{C} : D_i \mapsto \sharp$ Einsen

Potentialmethode bei Binärzähler

- tatsächlichen Kosten: $1, 2, 3, 1, 2, 4, 1, \cdots \Rightarrow$ verschieden ;)
- *Potentialfunktion* $\mathcal{C} : \{D_i\} \longrightarrow \mathbb{N}$ mit $\mathcal{C} : D_i \mapsto \sharp$ Einsen
- Erfüllt: $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$, sogar ordentlich: $\mathcal{C}(D_i) \geq \mathcal{C}(D_0)$

Potentialmethode bei Binärzähler

- tatsächlichen Kosten: $1, 2, 3, 1, 2, 4, 1, \cdots \Rightarrow$ verschieden ;)
- *Potentialfunktion* $\mathcal{C} : \{D_i\} \longrightarrow \mathbb{N}$ mit $\mathcal{C} : D_i \mapsto \sharp$ Einsen
- Erfüllt: $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$, sogar ordentlich: $\mathcal{C}(D_i) \geq \mathcal{C}(D_0)$
- Die amortisierten Kosten der i-ten Operation bzgl. $\mathcal{C}$:
  $\widehat{c_i} := c_i + \sharp$ Einsen in $D_i - \sharp$ Einsen in D_{i-1}

Potentialmethode bei Binärzähler

- tatsächlichen Kosten: $1, 2, 3, 1, 2, 4, 1, \cdots \Rightarrow$ verschieden ;)
- *Potentialfunktion* $\mathcal{C} : \{D_i\} \longrightarrow \mathbb{N}$ mit $\mathcal{C} : D_i \mapsto \sharp$ Einsen
- Erfüllt: $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$, sogar ordentlich: $\mathcal{C}(D_i) \geq \mathcal{C}(D_0)$
- Die amortisierten Kosten der i-ten Operation bzgl. $\mathcal{C}$:
  $\widehat{c_i} := c_i + \sharp$ Einsen in $D_i - \sharp$ Einsen in D_{i-1}
  $= x + y - (y - 1 + (x - 1))$

Potentialmethode bei Binärzähler

- tatsächlichen Kosten: $1, 2, 3, 1, 2, 4, 1, \cdots \Rightarrow$ verschieden ;)
- *Potentialfunktion* $\mathcal{C} : \{D_i\} \longrightarrow \mathbb{N}$ mit $\mathcal{C} : D_i \mapsto \sharp$ Einsen
- Erfüllt: $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$, sogar ordentlich: $\mathcal{C}(D_i) \geq \mathcal{C}(D_0)$
- Die amortisierten Kosten der i-ten Operation bzgl. $\mathcal{C}$:
  $\widehat{c_i} := c_i + \sharp$ Einsen in $D_i - \sharp$ Einsen in D_{i-1}
  $= x + y - (y - 1 + (x - 1))$
  $= 2$

Potentialmethode bei Binärzähler

- tatsächlichen Kosten: $1, 2, 3, 1, 2, 4, 1, \cdots \Rightarrow$ verschieden ;)
- *Potentialfunktion* $\mathcal{C} : \{D_i\} \rightarrow \mathbb{N}$ mit $\mathcal{C} : D_i \mapsto \sharp$ Einsen
- Erfüllt: $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$, sogar ordentlich: $\mathcal{C}(D_i) \geq \mathcal{C}(D_0)$
- Die amortisierten Kosten der i-ten Operation bzgl. $\mathcal{C}$:
  $\widehat{c_i} := c_i + \sharp$ Einsen in $D_i - \sharp$ Einsen in D_{i-1}
  $= x + y - (y - 1 + (x - 1))$
  $= 2$
- Gesucht: $\sum_{i=1}^{n} \widehat{c_i}$, wobei $\widehat{c_i} \equiv 2$

Potentialmethode bei Binärzähler

- tatsächlichen Kosten: $1, 2, 3, 1, 2, 4, 1, \cdots \Rightarrow$ verschieden ;)
- *Potentialfunktion* $\mathcal{C} : \{D_i\} \rightarrow \mathbb{N}$ mit $\mathcal{C} : D_i \mapsto \sharp$ Einsen
- Erfüllt: $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$, sogar ordentlich: $\mathcal{C}(D_i) \geq \mathcal{C}(D_0)$
- Die amortisierten Kosten der i-ten Operation bzgl. $\mathcal{C}$:
  $\widehat{c_i} := c_i + \sharp$ Einsen in $D_i - \sharp$ Einsen in D_{i-1}
  $= x + y - (y - 1 + (x - 1))$
  $= 2$
- Gesucht: $\sum_{i=1}^{n} \widehat{c_i}$, wobei $\widehat{c_i} \equiv 2$
- $\Rightarrow T(n) \leq 2n \Rightarrow T(n) \in O(n)$

Potentialmethode bei Binärzähler

- tatsächlichen Kosten: $1, 2, 3, 1, 2, 4, 1, \cdots \Rightarrow$ verschieden ;)
- *Potentialfunktion* $\mathcal{C} : \{D_i\} \rightarrow \mathbb{N}$ mit $\mathcal{C} : D_i \mapsto \sharp$ Einsen
- Erfüllt: $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$, sogar ordentlich: $\mathcal{C}(D_i) \geq \mathcal{C}(D_0)$
- Die amortisierten Kosten der i-ten Operation bzgl. $\mathcal{C}$:
  $\widehat{c_i} := c_i + \sharp$ Einsen in $D_i - \sharp$ Einsen in D_{i-1}
  $= x + y - (y - 1 + (x - 1))$
  $= 2$
- Gesucht: $\sum_{i=1}^{n} \widehat{c_i}$, wobei $\widehat{c_i} \equiv 2$
- $\Rightarrow T(n) \leq 2n \Rightarrow T(n) \in O(n)$
- $T(n) \in O(n) \wedge T(n) \in \Omega(n) \Rightarrow T(n) \in \Theta(n)$

Termine

- Nächste Vorlesung: Dienstag, 7. November, 15:45 Uhr
- Übungsblatt Abgabe: Mittwoch, 8. November 15:30 Uhr
- Nächste Übung: Donnerstag, 16. November, 15.45 Uhr