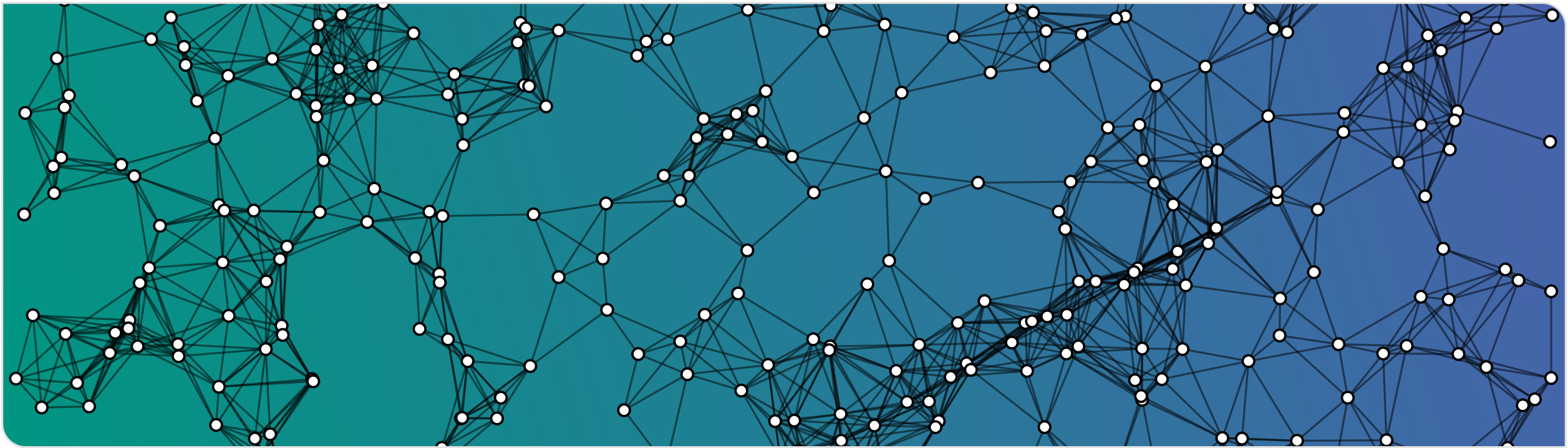
Karlsruher Institut für Technologie

Algorithmen 1

Übung 2
Aufgabe B01.A1, amortisierte Analyse



Organisatorisches

- Blatt 2, Aufgabe 1, erster Algo
  - schlechte Aufgabenstellung, zu schwierig
  - $\Rightarrow$ volle Punktzahl + Bonuspunkte (siehe Vorlesung 4)
  - Gleich: Vorstellung der Lösung
- Zusätzliche Tutorien zum Nachholen / Klausurvorbereitung im August
  - mehr Infos demnächst
- Update der Pseudocode Richtlinien

Übung 2: Aufgabe 1: algOne

Gegeben:

```
algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-1) + algOne(x, y-1)
```

Gesucht:

- „Problemgröße“ n
- Rekurrenz in Abh. von n
- Laufzeit in Abh. von n

Übung 2: Aufgabe 1: ALGONE

Gegeben:

```
algOne(x: ℕ, y: ℕ) // Ann
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-
```

Gesucht:

- „Problemgröße“ n
- Rekurrenz in Abh. von n
- Laufzeit in Abh. von n

```
algTwo(x, y) // Annahme: initial x ≤ y
    if x ≥ y then
        return x
    total := algTwo(x, ⌊(y+3x)/4⌋)
    total += algTwo(⌊(y+3x)/4⌋, ⌊(y+x)/2⌋)
    total += algTwo(⌊(y+x)/2⌋, ⌊(3y+x)/4⌋)
    return total
```

- „Problemgröße“ $n := \max\{0, y - x\}$
  - $n' = 1/4 \cdot n$
- $T(n) = 3T(n/4) + c_1, T(0) = c_2$
- $T(n) \in \Theta(n^{\log_4(3)})$

Übung 2: Aufgabe 1: algOne

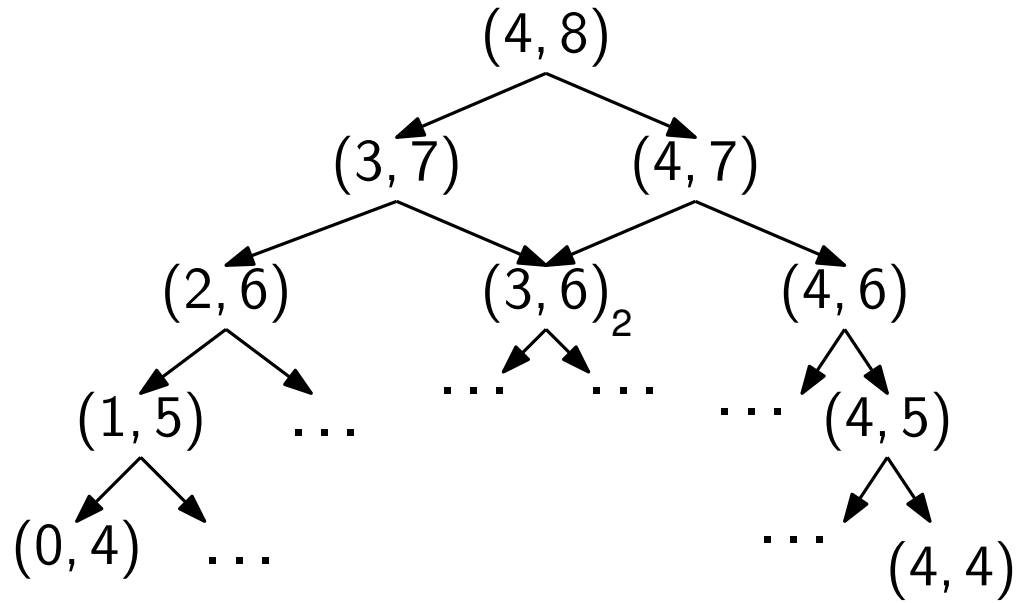
Gegeben:

```
algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-1) + algOne(x, y-1)
```

Gesucht:

- „Problemgröße“ n
- Rekurrenz in Abh. von n
- Laufzeit in Abh. von n

Und bei algOne?



Das schaut kompliziert aus :(

Plan

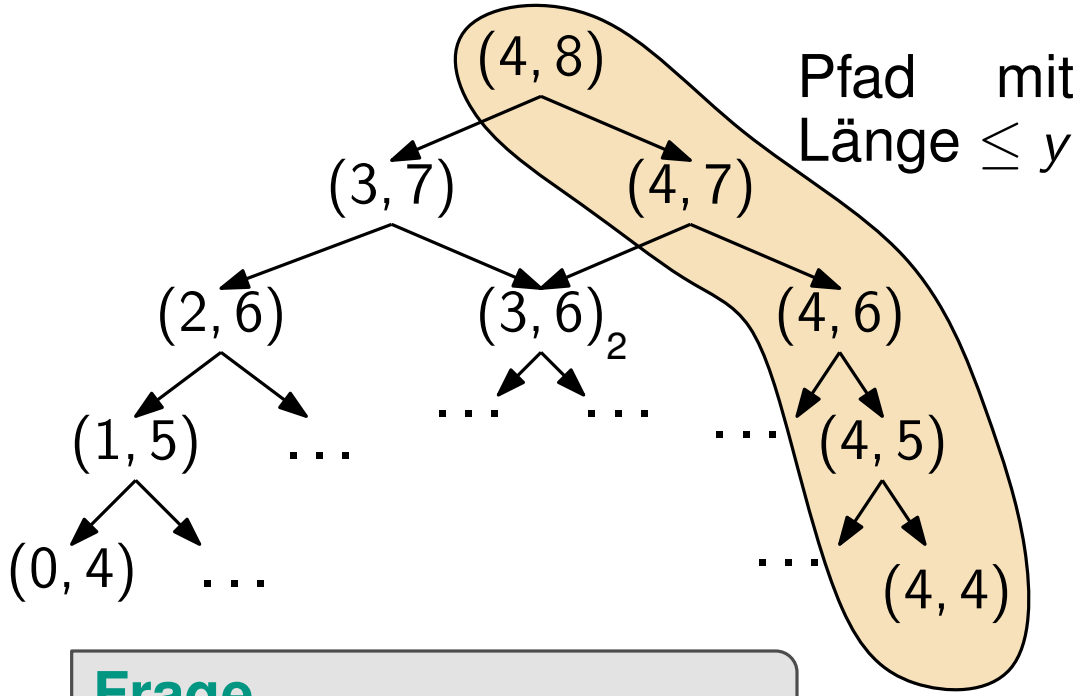
- (grobe) obere Schranke in y
- im Anschluss: genaue Analyse

Ungenaue obere Schranke ALGONE

```
algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-1) + algOne(x, y-1)
```

Beobachtung

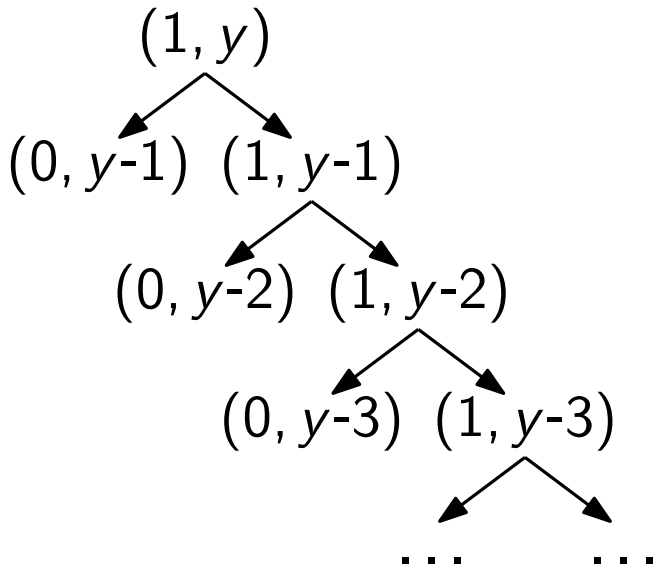
- y sinkt in jedem Aufruf um 1
- Länge der Pfade im Rekursionsbaum maximal y
- Pro Aufruf: zwei Rekursive Aufrufe
- #Knoten in Rekursionsbaum $\leq 2 \cdot 2^y$
- Für „Problemgröße“ $n := y$: Laufzeit in $O(2^n)$



Frage
Ist diese Analyse gut?

Ungenaue obere Schranke ALGONE

```
algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-1) + algOne(x, y-1)
```



Beobachtung

- y sinkt in jedem Aufruf um 1
- Länge der Pfade im Rekursionsbaum maximal y
- Pro Aufruf: zwei Rekursive Aufrufe
- #Knoten in Rekursionsbaum $\leq 2 \cdot 2^y$
- Für „Problemgröße“ $n := y$: Laufzeit in $O(2^n)$

Frage
Ist diese Analyse gut?

Nicht für kleine x, große x

Genaue Analyse ALGONE

```
algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-1) + algOne(x, y-1)
```

Schritt 1

- Kosten ausrechnen...

Genaue Analyse ALGONE

algOne$(x\colon \mathbb{N}, y\colon \mathbb{N})$ // Annahme: initial $x \leq y$

So

- K

x,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2		1	3	6	10	15	21	28	36	45	55	66	78	91	105
3			1	4	10	20	35	56	84	120	165	220	286	364	455
4				1	5	15	35	70	126	210	330	495	715	1001	1365
5					1	6	21	56	126	252	462	792	1287	2002	3003
6						1	7	28	84	210	462	924	1716	3003	5005
7							1	8	36	120	330	792	1716	3432	6435
8								1	9	45	165	495	1287	3003	6435
9									1	10	55	220	715	2002	5005
10										1	11	66	286	1001	3003
11											1	12	78	364	1365
12												1	13	91	455
13													1	14	105
14														1	15
15															1

Genaue Analyse ALGONE

```
algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-1) + algOne(x, y-1)
```

Schritt 1

- Kosten ausrechnen...
- Zahlen anstarren

Genaue Analyse ALGONE

algOne$(x\colon \mathbb{N}, y\colon \mathbb{N})$ // Annahme: initial $x \leq y$

So

- K
- Z

x,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2		1	3	6	10	15	21	28	36	45	55	66	78	91	105
3			1	4	10	20	35	56	84	120	165	220	286	364	455
4				1	5	15	35	70	126	210	330	495	715	1001	1365
5					1	6	21	56	126	252	462	792	1287	2002	3003
6						1	7	28	84	210	462	924	1716	3003	5005
7							1	8	36	120	330	792	1716	3432	6435
8								1	9	45	165	495	1287	3003	6435
9									1	10	55	220	715	2002	5005
10										1	11	66	286	1001	3003
11											1	12	78	364	1365
12												1	13	91	455
13													1	14	105
14														1	15
15															1

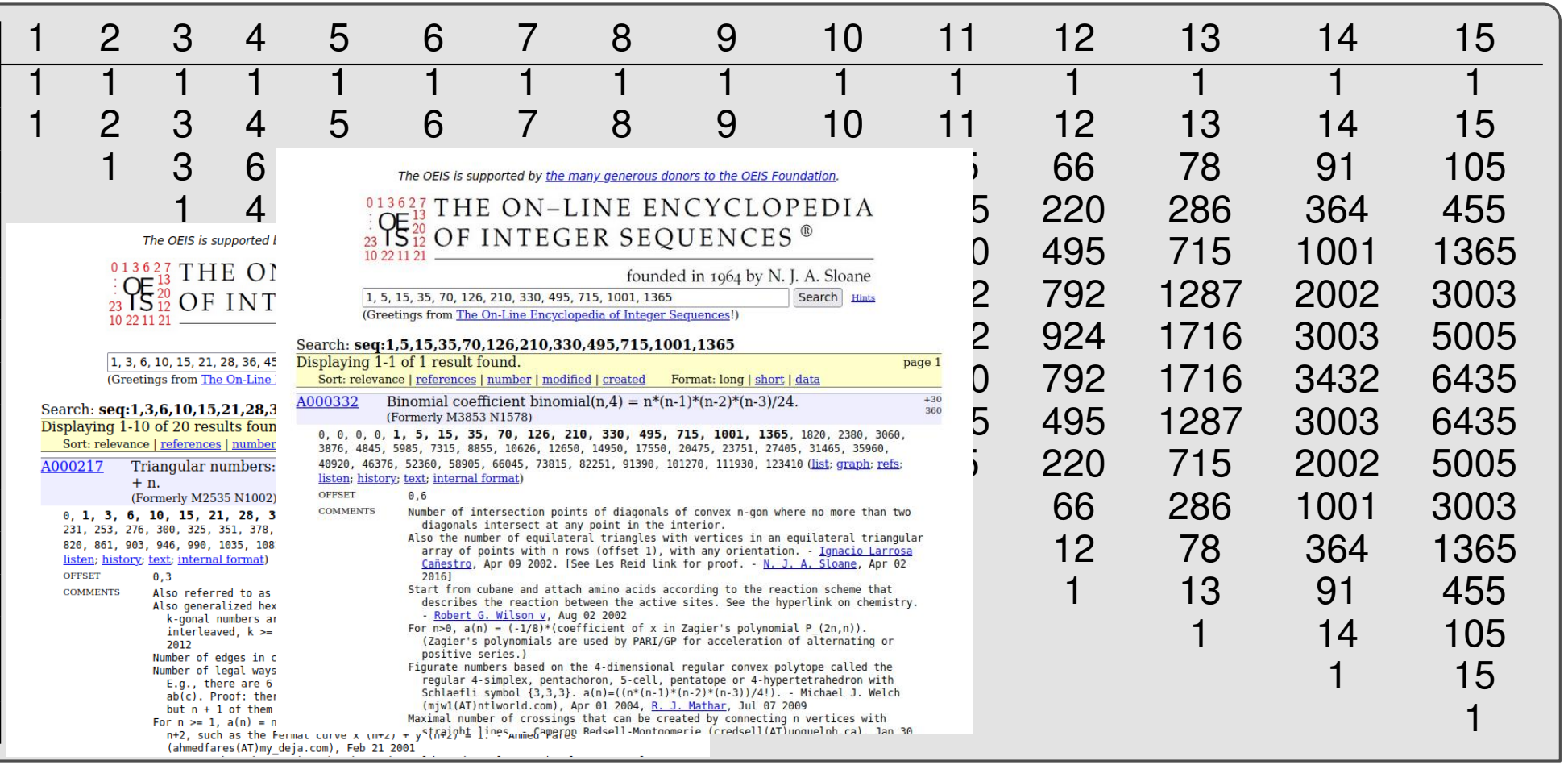
Genaue Analyse ALGONE

algOne(x: $\mathbb{N}$, y: $\mathbb{N}$) // Annahme: initial $x \leq y$

So

- K
- Z

x,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2		1	3	6								66	78	91	105
3			1	4								220	286	364	455
4				1								495	715	1001	1365
5												792	1287	2002	3003
6												924	1716	3003	5005
7												792	1716	3432	6435
8												495	1287	3003	6435
9												220	715	2002	5005
10												66	286	1001	3003
11												12	78	364	1365
12												1	13	91	455
13													1	14	105
14														1	15
15															1



Genaue Analyse ALGONE

```
algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-1) + algOne(x, y-1)
```

Schritt 1

- Kosten ausrechnen...
- Zahlen anstarren

Schritt 2

- Vermutung bilden:
  - $T(x, y) \leq 2\binom{y}{x} - 1$
- Vermutung beweisen

Genaue Analyse ALGONE

```
algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-1) + algOne(x, y-1)
```

Schritt 1

- Kosten ausrechnen...
- Zahlen anstarren

Schritt 2

- Vermutung bilden:
  - $T(x,y) \leq 2\binom{y}{x} - 1$
- Vermutung beweisen

Beweis per Induktion

- für $i \geq 0$: $T(i,i) = 2\binom{i}{i} - 1 = 2 - 1 = 1$
- Ang.: Vermutung gilt für alle kleineren Werte von x und y

$$\begin{aligned} T(x,y) &= T(x-1,y-1) + T(x,y-1) + 1 \\ &= 2\binom{y-1}{x-1} - 1 + 2\binom{y-1}{x} - 1 + 1 \\ &= 2\left(\binom{y-1}{x-1} + \binom{y-1}{x}\right) - 1 \\ &= 2\binom{y}{x} - 1 \end{aligned}$$

Genaue Analyse ALGONE

algOne$(x\colon \mathbb{N}, y\colon \mathbb{N})$ // Annahme: initial $x \leq y$

if $x = 0 \vee x = y$ **then**

Eigenschaften [Bearbeiten | Quelltext bearbeiten]

Wird außer k auch n auf nichtnegative ganze Zahlen eingeschränkt, so gilt:

- $\binom{n}{k}$ ist stets eine nichtnegative ganze Zahl. Ist $k \leq n$, so ist $\binom{n}{k} \geq 1$, anderenfalls ist $\binom{n}{k} = 0$.
- $\binom{n}{0} = 1 = \binom{n}{n}$
- $\binom{n}{1} = n = \binom{n}{n-1}$
- $\binom{n}{k} = \frac{n-k+1}{k}\binom{n}{k-1}$
- $\binom{n}{k} = \frac{n}{k} \cdot \binom{n-1}{k-1} \Leftrightarrow k \cdot \binom{n}{k} = n \cdot \binom{n-1}{k-1}$
- $\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$. Für $n = k$ ist der rechte Summand $\binom{n}{k+1} = 0$.

S

-

-

S

-

  - $T(x, y) \leq 2\binom{y}{x} - 1$

- Vermutung beweisen

Beweis per Induktion

- für $i \geq 0$: $T(i, i) = 2\binom{i}{i} - 1 = 2 - 1 = 1$

g.: Vermutung gilt für alle kleineren
erte von x und y

$$\begin{aligned} y) &= T(x-1, y-1) + T(x, y-1) + 1 \\ &= 2\binom{y-1}{x-1} - 1 + 2\binom{y-1}{x} - 1 + 1 \\ &= 2\left(\binom{y-1}{x-1} + \binom{y-1}{x}\right) - 1 \\ &= 2\binom{y}{x} - 1 \end{aligned}$$

Genaue Analyse ALGONE

```
algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-1) + algOne(x, y-1)
```

Schritt 1

- Kosten ausrechnen...
- Zahlen anstarren

Schritt 2

- Vermutung bilden:
  - $T(x, y) \leq 2\binom{y}{x} - 1$
- Vermutung beweisen

Beweis per Induktion

- für $i \geq 0$: $T(i, i) = 2\binom{i}{i} - 1 = 2 - 1 = 1$
- Ang.: Vermutung gilt für alle kleineren Werte von x und y

$$T(x, y) = T(x-1, y-1) + T(x, y-1) + 1$$
$$= 2\binom{y}{x} - 1$$

Genaue Analyse ALGONE

```
algOne(x: ℕ, y: ℕ) // Annahme: initial x ≤ y
    if x = 0 ∨ x = y then
        return 1
    return algOne(x-1, y-1) + algOne(x, y-1)
```

Schritt 1

- Kosten ausrechnen...
- Zahlen anstarren

Schritt 2

- Vermutung bilden:
  - $T(x,y) \leq 2\binom{y}{x} - 1$
- Vermutung beweisen

Beweis per Induktion

- für $i \geq 0$: $T(i,i) = 2\binom{i}{i} - 1 = 2 - 1 = 1$
- Ang.: Vermutung gilt für alle kleineren Werte von x und y

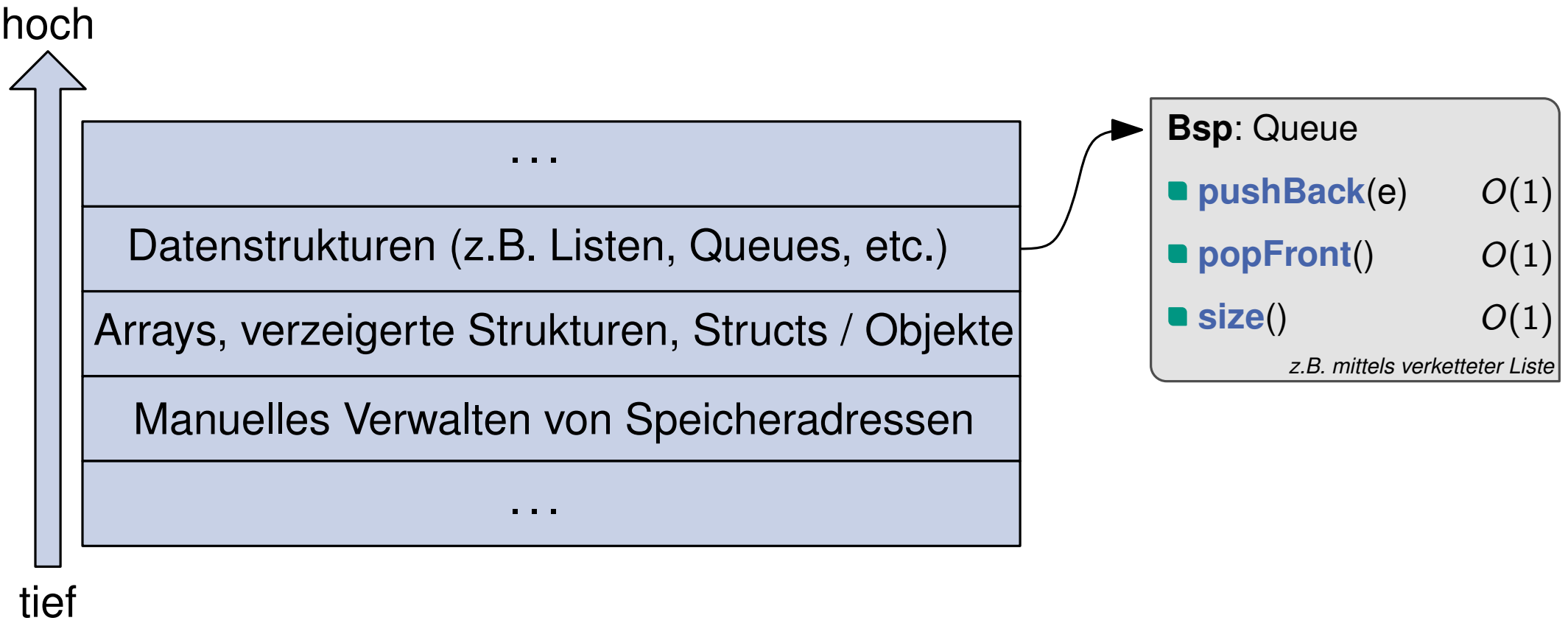
$$T(x,y) = T(x-1,y-1) + T(x,y-1) + 1$$
$$= 2\binom{y}{x} - 1$$

- Moment mal...
  - $\binom{y}{x} = \binom{y-1}{x-1} + \binom{y-1}{x}$...?
  - Das ist doch genau **algOne**!
  - $T(x,y) \in \Theta(\binom{y}{x})$

:-)

Motivation: Datenstrukturen

Abstraktionsebenen auf dem Speicher



Motivation: Amortisierte Analyse

- Laufzeit von Operationen auf DS nicht immer gleich
  - z.B.: Operation meist günstig, seltener teurer

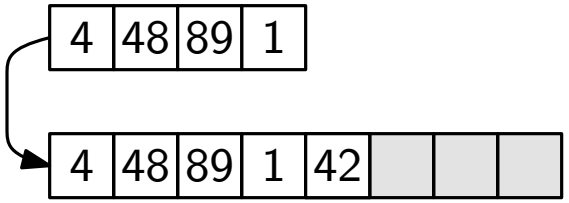
Idee

- **amortisierte Kosten**: durchschnittliche Kosten pro Operation in einer Folge von Operationen

Verschiedene Techniken zur Analyse

- Aggregation
- Charging
- Konto
- Potential

Beispiel: unbeschränktes Array



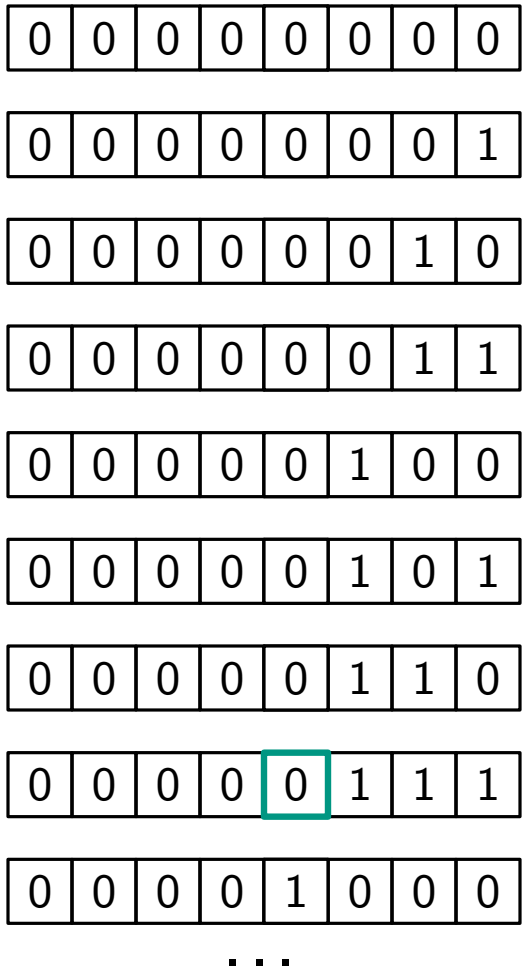
Beispiel: Binärzähler

Idee

- Array A verwaltet Bits
- Funktion **increment()** erhöht Zähler

Algorithmus

- suche Stelle i mit rechtester 0
- setze Stelle i auf 1
- setze Stellen rechts von i auf 0



Beispiel: Binärzähler

Idee

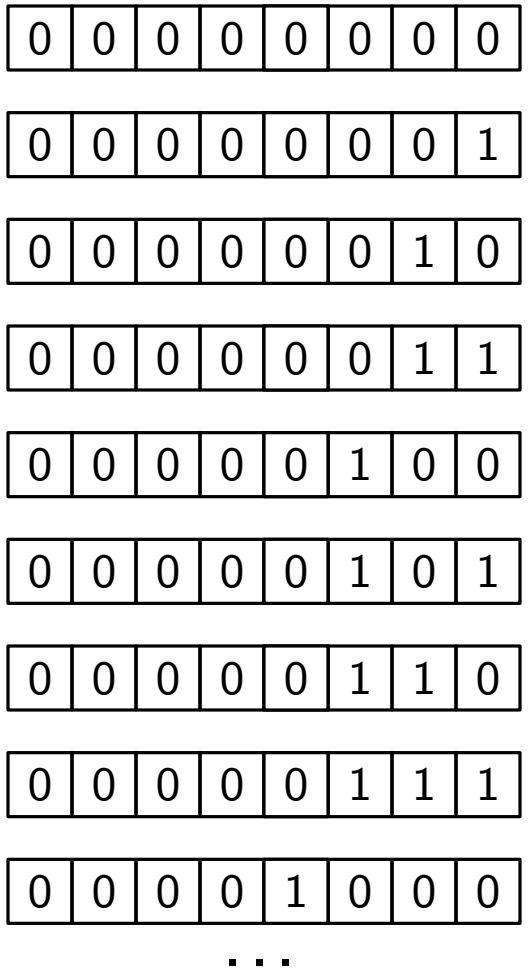
- Array A verwaltet Bits
- Funktion **increment()** erhöht Zähler

Algorithmus in Pseudocode

increment()

```
i := 0
while A[i] = 1 do
    A[i] := 0
    i := i+1
A[i] := 1
```

Laufzeit?



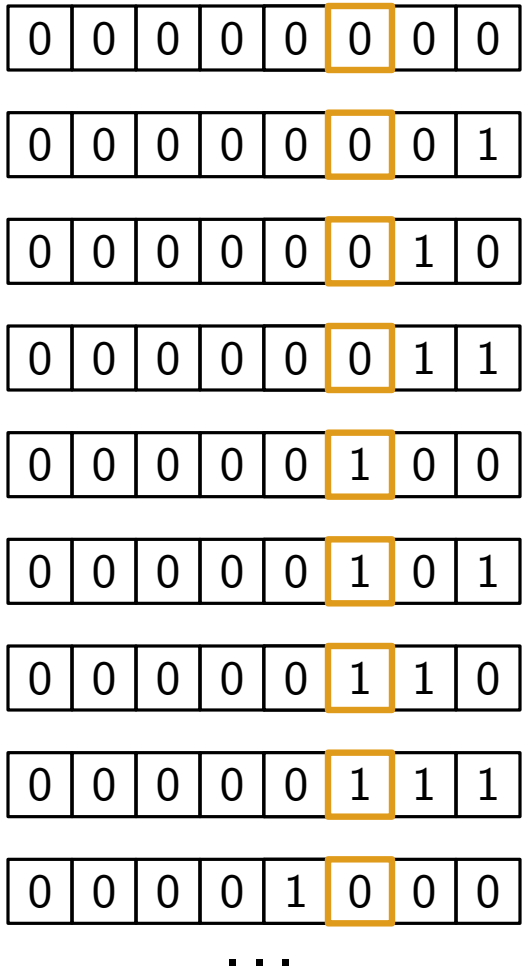
Analyse: Aggregatmethode

Idee

- berechne Gesamtkosten
- teile durch Anzahl von Operationen

Beobachtung

- nicht jedes Bit flipt bei jeder Operation
- $A[0]$ flipt jedes mal, $A[1]$ jedes zweite, etc.
- $A[i]$ flipt bei jedem 2^i-ten Inkrement
- Bei n Aufrufen: $A[i]$ flipt $\lfloor \frac{n}{2^i} \rfloor$ mal



Analyse: Aggregatmethode

Beobachtung

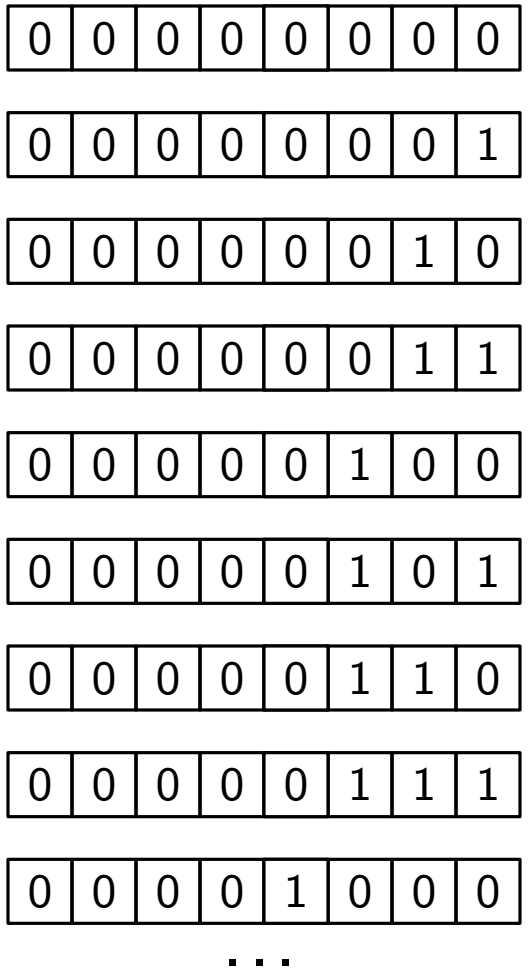
- nicht jedes Bit flipt bei jeder Operation
- $A[0]$ flipt jedes mal, $A[1]$ jedes zweite, etc.
- $A[i]$ flipt bei jedem 2^i-ten Inkrement
- Bei n Aufrufen: $A[i]$ flipt $\lfloor \frac{n}{2^i} \rfloor$ mal

In Summe ergibt sich für k bits:

$$\sum_{i=0}^{k-1} \lfloor \frac{n}{2^i} \rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n$$

D.h.: pro Operation nur 2 Bit-Flips

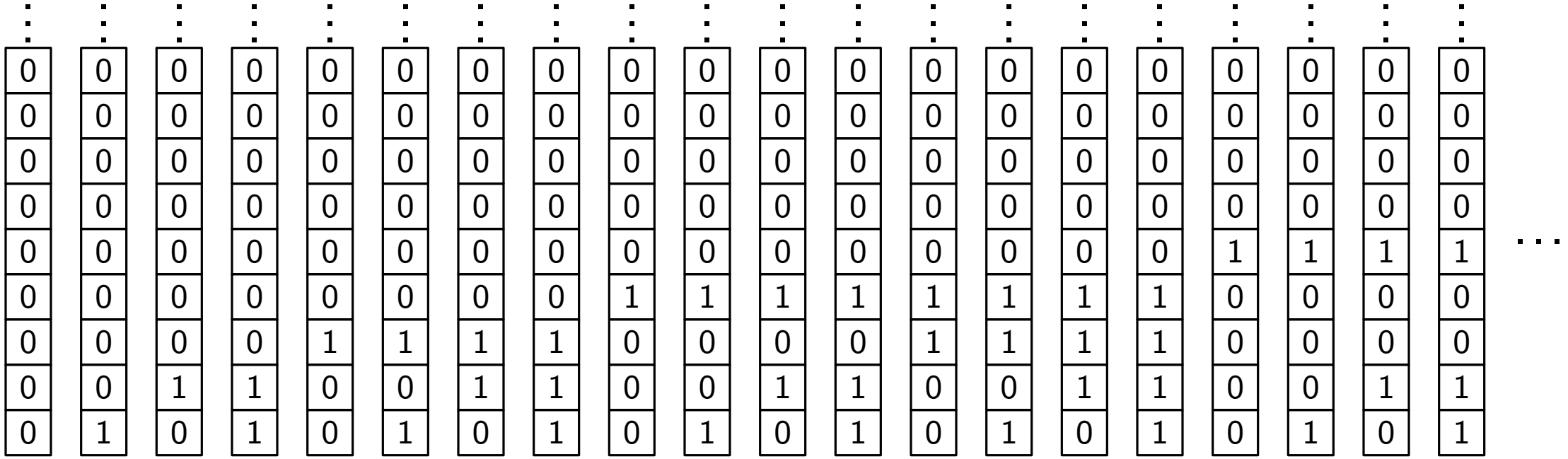
Somit: **amortisierte Kosten** in $O(1)$



Analyse mittels Charging

Idee

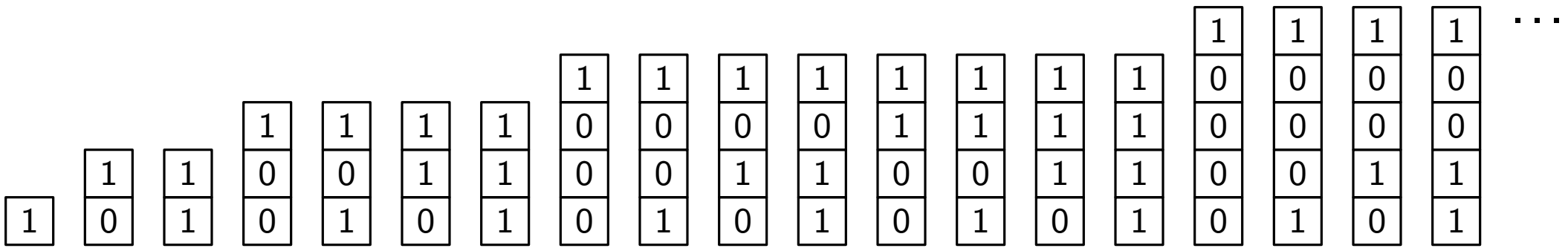
- teure Operationen legen Kosten um auf günstige Operationen



Analyse mittels Charging

Idee

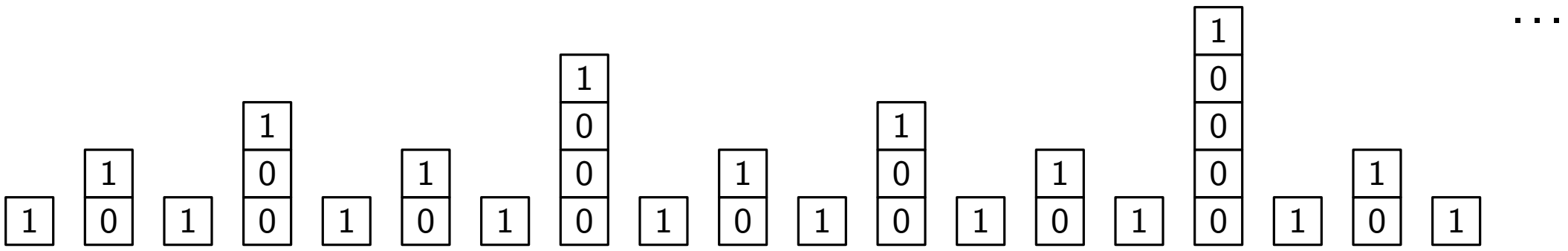
- teure Operationen legen Kosten um auf günstige Operationen



Analyse mittels Charging

Idee

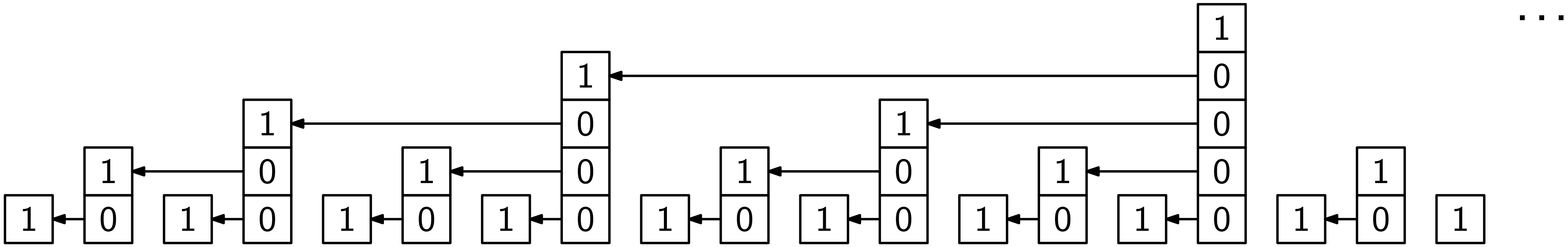
- teure Operationen legen Kosten um auf günstige Operationen



Analyse mittels Charging

Idee

- teure Operationen legen Kosten um auf günstige Operationen



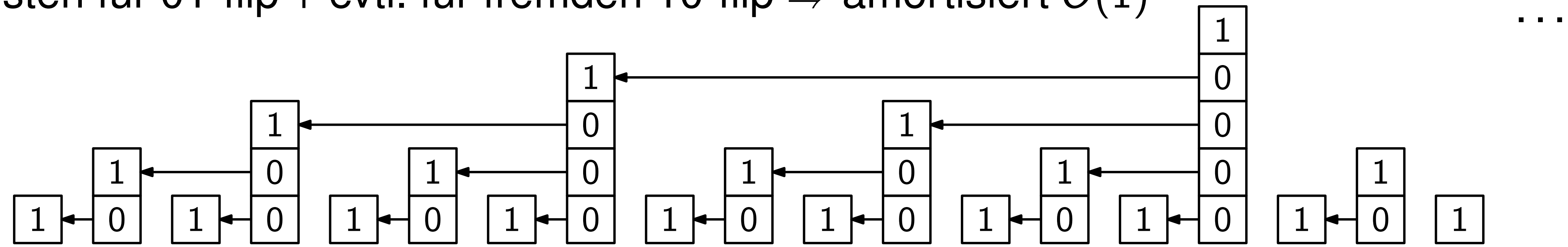
Analyse mittels Charging

Idee

- teure Operationen legen Kosten um auf günstige Operationen

Vorgehen hier:

- lege Kosten von 10-flip auf letzte Operation welche zuvor 01-flip ausgeführt hat
  - das geht für jeden 10-flip
- Pro Operation übrig:
  - Kosten für 01-flip + evtl. für fremden 10-flip $\Rightarrow$ amortisiert $O(1)$



Analyse: Potentialmethode

Idee

- definiere Potentialfunktion $\Phi(D_i)$
  - Maß für Unaufgeräumtheit von D zum Zeitpunkt i
- definiere amortisierte Kosten $\hat{c}_i = c_i + \underbrace{\Phi(D_i) - \Phi(D_{i-1})}_{\text{Anstieg des Potenzials}}$

tatsächliche Kosten

Analyse: Potentialmethode

Idee

- definiere Potentialfunktion $\Phi(D_i)$
  - Maß für Unaufgeräumtheit von D zum Zeitpunkt i
- definiere amortisierte Kosten $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$

Definition sinnvoll?

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \sum_{i=1}^{n} (c_i) + \sum_{i=1}^{n} (\Phi(D_i) - \Phi(D_{i-1}))$$

$$= \sum_{i=1}^{n} (c_i) + \Phi(D_n) - \Phi(D_0)$$

Falls $\Phi(D_n) \geq \Phi(D_0)$: amortisierte Kosten obere Schranke für tatsächliche Kosten

$\Rightarrow$ fordere $\Phi(D_i) \geq \Phi(D_0)$ f.a. $i > 0$ oder: $\Phi(D_0) = 0$ und $\Phi(D_i) \geq 0$ f.a. $i > 0$

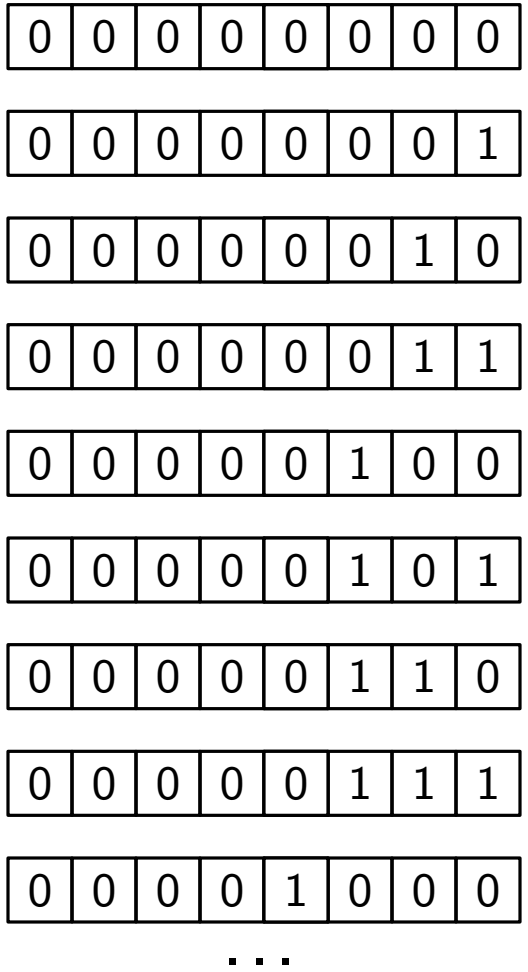
Analyse: Potentialmethode

Potentialfunktion für Binärzähler:

- def. $\Phi(D_i) = \sum_{j=0}^{k-1} A[j]$ (Anzahl 1-bits zum Zeitpunkt i)

Dann ergibt sich:

- $\Phi(D_0) = 0$, $\Phi(D_i) \geq 0$ f.a. $i > 0$ *gültige Pot.fun.* :)
- tätsächliche Kosten: $c_i = \#01\text{-flips} + \#10\text{-flips}$
- Potentialänderung: $\#01\text{-flips} - \#10\text{-flips}$
- amort. Kosten: $\hat{c}_i = 2 \cdot \#01\text{-flips} \leq 2$



Definition: amortisierte Kosten $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$

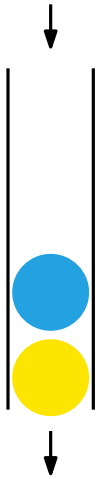
Beispiel 2: Stacks und Queues

Wiederholung

Queue

- **push**(e) $O(1)$
- **pop**() $O(1)$
- **size**() $O(1)$

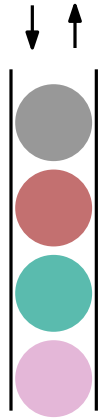
z.B. mittels verketteter Liste



Stack

- **push**(e) $O(1)$
- **pop**() $O(1)$
- **size**() $O(1)$

z.B. mittels verketteter Liste



Frage

Angenommen wir können Stacks (als Blackbox) verwenden.

(Wie) können wir daraus eine Queue bauen?

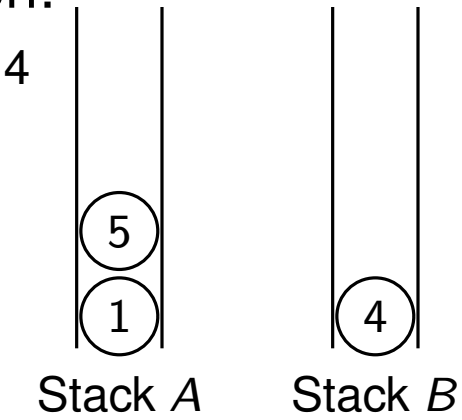
Eine Queue aus Stacks

Algorithmische Umsetzung

- **push**: auf A pushen $O(1)$
- **pop**: $O(|A|)$
  - falls B voll: von B poppen $O(1)$
  - falls B leer: alles von A nach B verschieben $O(|A|)$

Operationen:

- **push**: 3, 1, 4
- **pop**()
- **pop**()
- **push**: 1,5



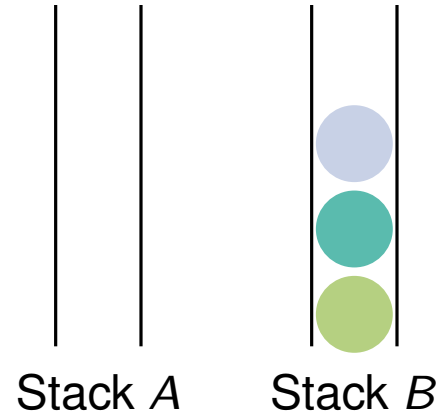
Analyse: Kontomethode

Idee

- günstige Operationen bauen Guthaben auf
- teure Operationen nutzen Guthaben

Umsetzung hier

- Gesamtkosten linear in **push** und **pop** Aufrufen
- Bei **push**: zahle 3
- Bei **pop**: zahle 1
  - Falls B leer: nutze 2 Guthaben für A.**pop()** und B.**push()** (für alle Elemente in A)
  - Falls B voll: B.**pop**() mit Kosten 1
- Konto wird nie negativ
- $3, 1 \in \Theta(1) \Rightarrow$ konstante amortisierte Kosten



Analyse: Potentialmethode

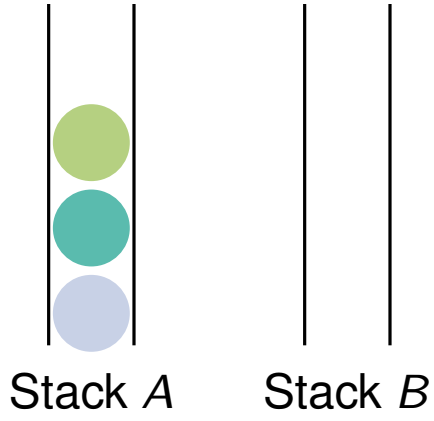
Potentialfunktion

- Definiere $\Phi(D_i) = 2 \cdot |A_i|$ (Anzahl Elemente auf A zum Zeitpunkt i)

Analyse

- $\Phi(D_0) = 0$, $\Phi(D_i) \geq 0$ f.a. $i > 0$ (d.h. gültige Potentialfunktion)
- amortisierte Kosten **push**:
  - $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 3$
- amortisierte Kosten **pop**:
  - $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1$
  - $c_i = 2 \cdot |A_{i-1}| + 1$
  - $\Phi(D_i) - \Phi(D_{i-1}) = -2|A_{i-1}|$

} amortisierte Kosten in $O(1)$



Definition: amortisierte Kosten $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$