3.4.17

Effiziente Algorithmen

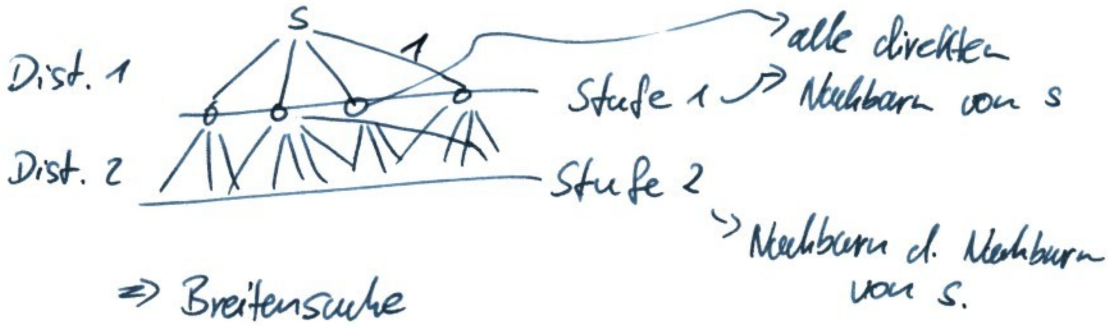
0. Einführung / Wiederholung

* Heap
  - Struktur
  - Operationen (Einfügen, Löschen, Update)
  - Implementierung im Array
  - Zusammenhang max. Tiefe eines Elements und (max.) Anzahl Elemente im Heap
  - Index für Namen (d. Elemente)
    ↳ als Array mit direkten Adressen
    ↳ oder: Suchbaum für die Namen, zusätzlich "Rückzeiger" aus Heap in Suchbaum hinein
    (Sonst muß bei den Heapoperationen ev. log n mal im Suchbaum gesucht (und aktualisiert) werden
    ⇒ gib dann insgesamt $\mathcal{O}((\log n)^2)$

3.4.17

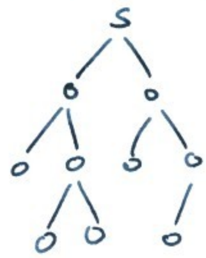
* Anwendung Heap im Dijkstra Algorithmus

- Wiederholung d. Algorithmus
- Bsp.: · ganz einfach, alle Gewichte = 1



⇒ Breitensuche

· oder noch einfacher: Baum!

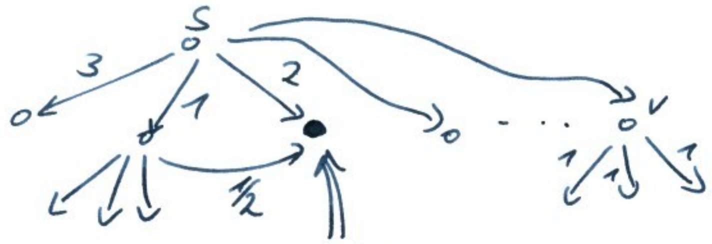


- Erster Schritt d. Algorithmus:

  ⇒ erster kürzester Weg von s aus ist kleinste von s ausgehende Kante!

- Bei Gewichten < 0 geht das nicht mehr!!!

Beispiel Dijkstra



Verbesserung d. akt. KW in der 2. Runde

Laufzeiten: - ohne Mitführen d. aktuell kürzesten Wege (jedesmal Neuberechnung d. Minimums aus der Menge der abgearbeiteten Knoten ~~ra~~ heraus.)

n Runden mit $O(n^2)$ bzw. besser gezählt $O(|E|)$ [jedesmal alle Kanten durchgehen]

$\Rightarrow O(n^3)$ bzw. $O(n \cdot |E|)$

- aktuell KW. Mitführen (im Array) (nur Min. suchen und KW aktualisieren, jede Kante wird nur einmal betrachtet.)

$\Rightarrow O(n^2 + |E|)$ (n Runden mit $O(n)$ f. neues Minimum suchen + $O(1)$ für jede Kante (global))

- aktuelle Suchfront im Heap
  (sonst wie oben)

  ⇝ neues Min. suchen $O(\log n)$

  ⇝ für jede Kante ~~HO~~ ev. KW aktualisieren,
  jedes mal $O(\log n)$

$\Rightarrow$ insg. $O(\underbrace{n \cdot \log n}_{\text{für } n \text{ Runden.}} + |E| \cdot \log n) = O(|E| \cdot \log n)$

~~Runde~~

Ziel: Verbesserung d. Laufzeit durch bessere Datenstruktur!

Untere Schranken: $\Omega(|E|)$, $\Omega(n \cdot \log n)$

$\Rightarrow$ bestmöglich ist $O(n \cdot \log n + |E|)$

$\Omega(\text{Max}\{n \cdot \log n, |E|\})$ ist $\Omega(n \cdot \log n + |E|)$

$O(\text{Max}\{n \cdot \log n, |E|\})$ ist $O(n \cdot \log n + |E|)$

Denn $n \cdot \log n + |E| \leq 2 \cdot \text{Max}\{n \cdot \log n + |E|\}$.

$f(n) \geq \Omega(\text{Max} \ldots)$

$\Leftrightarrow \exists$ Konst. $c \geq 0$

$$f(n) \geq c \cdot \text{Max}\{\ldots\}$$

$$\geq \tfrac{1}{2} c \cdot (n \cdot \log n + |E|)$$

also ist $f(n) = \Omega(n \cdot \log n + |E|)$ //

1. Binomialer Heap

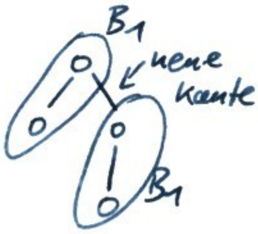
10.4.17

binomialer Baum B_n, $n \geq 0$

B_0 = O (Wurzel)

B_1 = B_0 / B_0 =

B_2 = B_1 / B_1 =



B_3 = B_2 / B_2 =

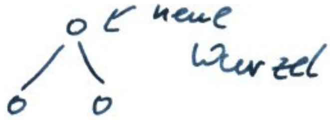
usw.

B_n = B_{n-1} \ B_{n-1}

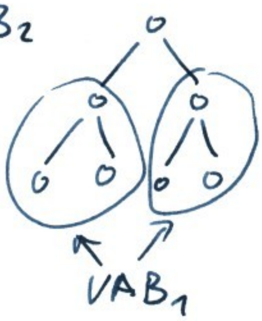
vollst. ausgeglichener bin. Baum VAB_n

VAB_0 = O

VAB_1

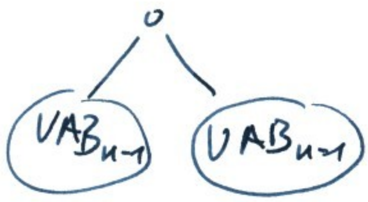


VAB_2



⋮

VAB_n



* Eigenschaften von B_n:

1. #Knoten (Elemente) $\Rightarrow 2^n$
2. #Kinder der Wurzel (Grad) von B_n $\Rightarrow n$ (dir. Nachfolger)
3. Maximale Tiefe von B_n $= n$

10.4.17

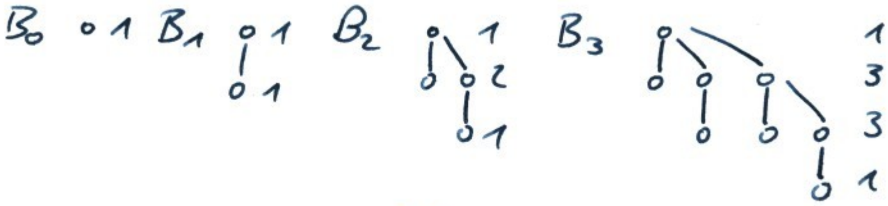
* Eig. von VAB_n:

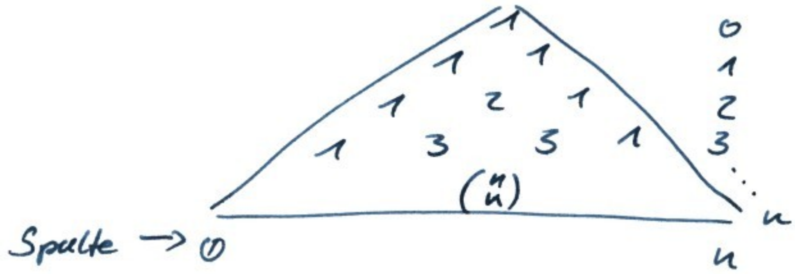
1. #Knoten $= 2^{n+1} - 1$ (geom. Reihe)
2. # Knoten in Tiefe $k = 2^k$ $(k \leq n)$
3. Max. Tiefe $= n$

13.4.17

Binomialer Baum: · # Knoten in Tiefe n ist 1.

· # Knoten von B_n in Tiefe $k = \binom{n}{k}$





(Pascalsches Dreieck)

An Zeile n, Spalte k steht Zeile $n-1$, $k-1$ + Zeile $n-1$, k (~~n-1, k~~ $n-1 \geq k \geq 1$)

Koeffizient in Zeile n, Spalte $k = \binom{n}{k}$

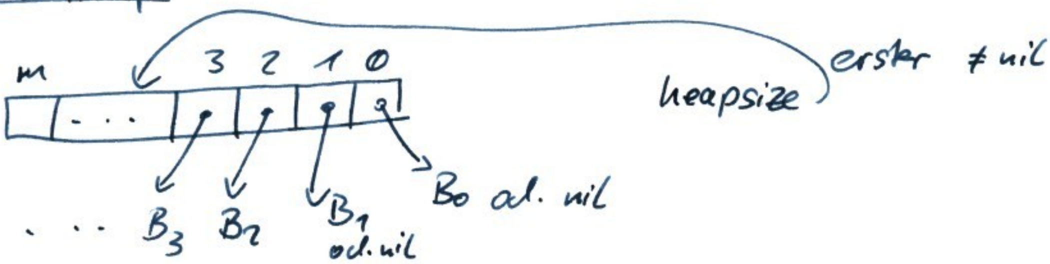
Also gilt: $\binom{n}{0} = \binom{n}{n} = 1$, f. $1 \leq k \leq n-1$ ist $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$

$\binom{n}{k}$ = # der genau k-elementigen Teilmengen aus $\{1, \dots, n\}$

– Herleitung $\binom{n}{k} = \dfrac{n!}{k!\,(n-k)!}$.

13.4.17

Binomialer Heap



$$n = \sum_{i=0}^{l} 2^i \cdot b_i \qquad b_i \in \{0,1\}, \; l = \lceil \log_2 n \rceil$$

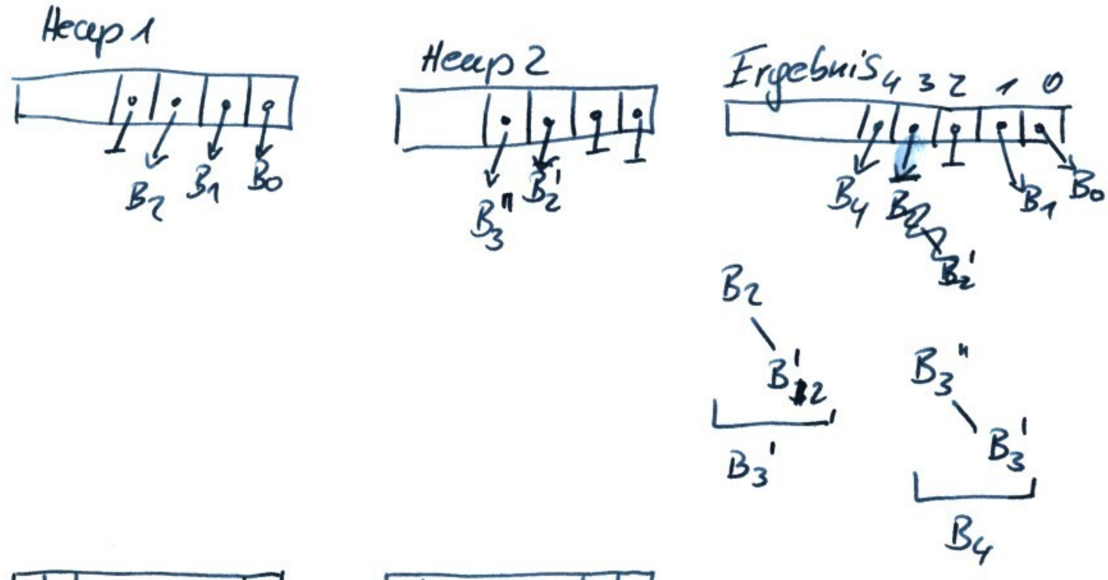
$$2 \cdot 2^{heapsize} - 1 \geq \#Elemente \geq 2^{heapsize}$$

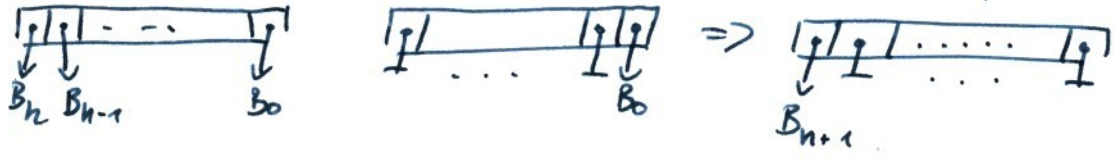
$$\Rightarrow heapsize + 1 \geq \log_2(\#Elemente) \geq heapsize$$

Operationen auf dem binomialen Heaps

- Verschmelzen: - Eingabe: 2 Heaps
  - Ausgabe: 1 neuer Heap, der die beiden Eingabeheaps zusammenfügt.

⇝ geht wie binäre Addition



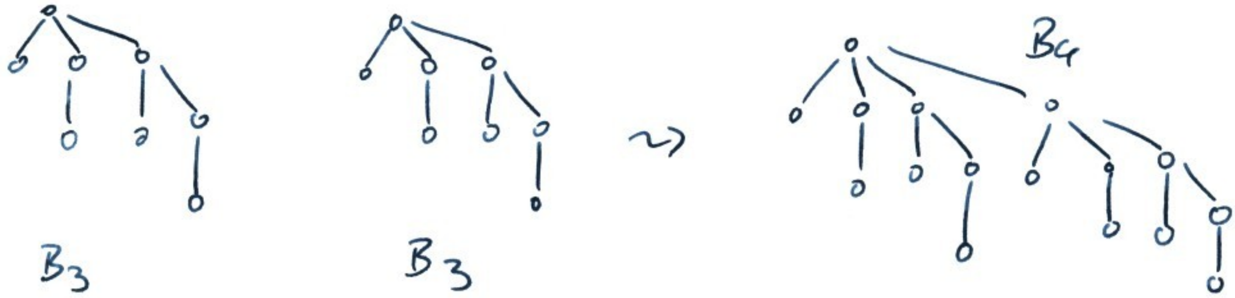


- Zeit: $O(\log_2$ (# Elemente der beiden Heaps zusammen))

  wenn die B_n geeignet implementiert sind,

  so dass ein Zusammenfügen zweier Bäume in $O(1)$ geht.

· Implementierung von B_n / Zusammenfügen:



$O(1)$ setzen eines Pointers

Zusatzsituation: Heapeigenschaft in jedem Baum für sich.
(beachten beim Verschmelzen!)

- Vereinigen von 2 Heaps in $O(\log_2 n)$
  n = # Elemente insgesamt.

Die anderen Operationen vom Heap:

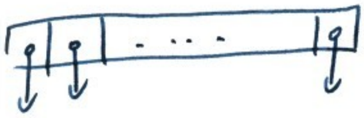
- Minimum ~~Finden~~ finden: $O(\log_2 n)$

- Minimum löschen: s. u.

- Einfügen: Verschmelzen von B_0 mit Heap $O(\log_2 n)$

13.4.17

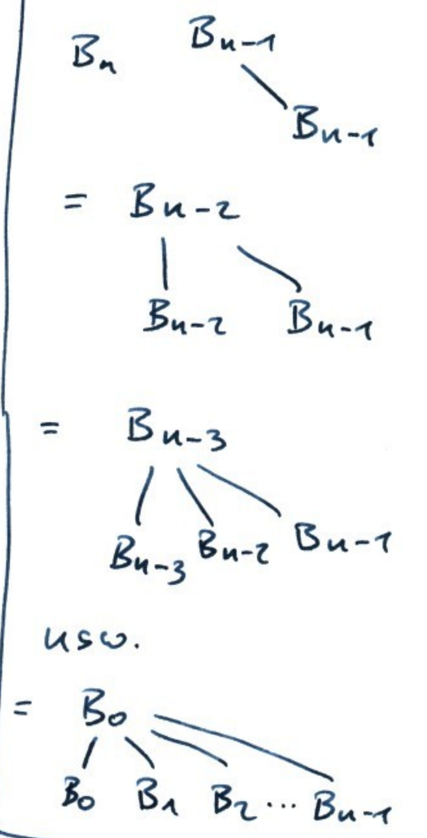
Minimum löschen:



1. Minimum suchen
2. Nehme Kinder des Minimums. Diese sind Wurzeln von $B_0, B_1, \dots, B_{m-1}$ (Min. Wurzel von B_m)
3. Mache Heap aus $B_0, \dots, B_{m-1}$

4. Verschmelzen mit dem Rest.

Struktur eines B_n

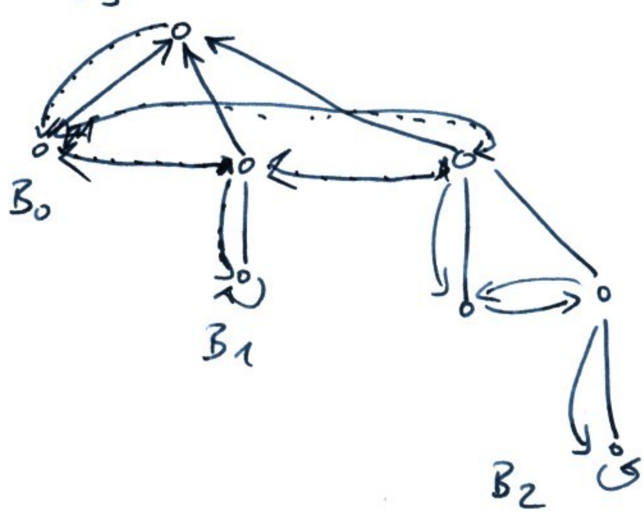


---

1. $O(\log_2 n)$
2. m ist $O(\log_2 n)$ Imp. folgt, so dass in $O(\log_2 n)$ geht.
3. $O(\log_2 n)$

- Pointerstruktur im Baum

$B_3 =$



Zusammenbauen von 2 Bäumen

1. Schritt

⇒ Überlegen: welche Pointer genau benötigt?!

⇒ Verhält sich wie normaler Heap

- (Min. finden in $O(\log n)$, geht besser)
- Zusätzlich: Heap verschmelzen $O(\log n)$

  (Vorher $O(n \cdot \log n)$ bzw. $O(n)$)

Wichtige Beobachtung: n Elemente in den leeren ~~Heaps~~ binomialen Heap einfügen.

Zeit: $O(n \cdot \log_2 n)$

Es gilt aber tatsächlich besser $O(n)$!

Wie sehen die Heaps aus?

⇓

13.4.17

1. B_0 *

2. B_1

3. $B_1\ B_0$ *

4. B_2

5. $B_2\ B_0$ *

6. $B_2\ B_1$

7. $B_2\ B_1\ B_0$ *

8. B_3

Zeit: ~~$\frac{1}{2} n \cdot c$~~ ~~ungerade~~

- Bei jeden Schritt ein B_0 $c \cdot n$
- Bei jeden 2. Schritt ein B_1 $\frac{1}{2} d \cdot n$
- 4. mal ein B_2 $\frac{1}{4} d \cdot n$

8. mal ein B_3 $\frac{1}{8} d \cdot n$

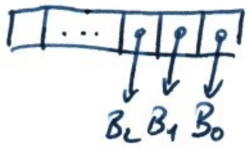
$$\sum_{i=0}^{\infty} \frac{1}{2^i}$$

12

24.4.17

- n Elemente einfügen in binomialen Heap.

$O(n \cdot \log n)$ ist klar, tatsächlich aber in $O(n)$



Die Einfügungen in $O(\log n)$ sind „selten". Wie selten?

- B_0 bilden: n-mal
- B_1 bilden, genau dann wenn nach der Einfügung # Elemente gerade ist.
- B_2 bilden, gdw. # Elemente durch 4 teilbar.
- usw.

$\Rightarrow$ $n = 2^k$ 2er Potenz:

B_0 n-mal

B_1 $\frac{n}{2}$-mal

$\vdots$

$$= 1 \cdot n + \frac{1}{2} n + \ldots + \frac{1}{2^{\log n}} \cdot n \leq \underline{\underline{2n}}$$

jetzt allgemeinere Methode: <u>Amortisierte Analyse</u>

24. 4. 17

- Dasselbe mit amortisierter Analyse:

t_i = Zeit der i-ten Einfügung

$$\boxed{\sum_{i=1}^{n} t_i \leq 2n}$$ Ziel!

Definiere Potential: $\Phi(h)$ = Potential des Heaps := # nicht-leerer Bäume im Heap.

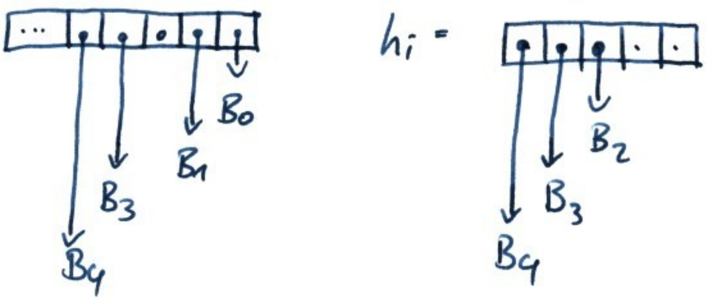
(Vorstellung: An jedem Baum ist eine (Konstante z.B. 2,3) Zeiteinheit gespeichert.)

Amortisierte Zeit der i-ten Einfügung:

$$a_i := \underbrace{t_i}_{\text{reale Zeit}} + \underbrace{\Phi(h_i) - \Phi(h_{i-1})}_{\text{Potentialdifferenz}}$$

h_i := Heap nach der i-ten Einfügung für $i \geq 1$, h_0 = leerer Heap, $\Phi(h_0) = 0$

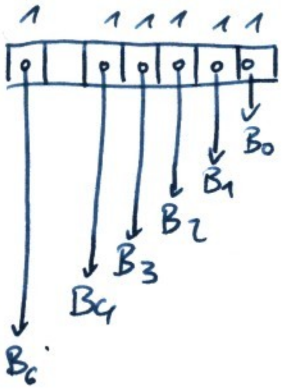
Bsp: $h_{i-1} =$ ... $h_i =$



24.4.17

Bsp.

h_{i-1}



Potential = 6

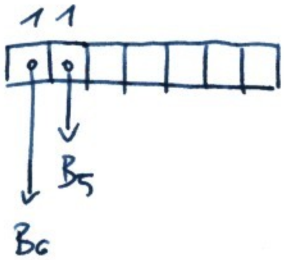
$t_i = 5 + 1 + 1$

$\Phi_i = \Phi(h_i)$

$\Phi_{i-1} = 6$

Einfügen

h_i



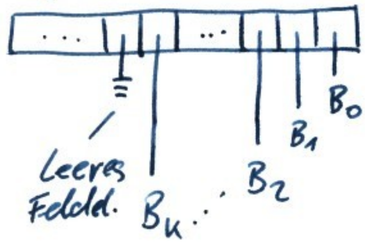
$a_i = 7 + \underbrace{2 - 6}_{= -4 < 0} = 3$

$\Phi_i = 2$

Behauptung: $a_i \leq 3$ für $i = 1, \dots, n$

allgemein:

h_{i-1}:



Einfügen ↓

$t_i = (k+1) + 2$

$(k+1)$ → Überträge

2 → B_0 erzeugen & abschließendes Eintragen

$a_i = t_i + \Phi_i - \Phi_{i-1} = t_i + \underbrace{1 - (k+1)}_{} = \underline{\underline{t_i - k}}$

1 neuer Baum

k+1 Bäume rechts verschwinden

$= \underbrace{(k+1) + 2}_{= t_i} - k = \underline{\underline{3}}$

24.4.17

$$\left(\text{bei } k=0: \quad a_i = 2+1 = 3\right)$$

Was heißt das für $\sum_{i=1}^{n} t_i$?

$$\underbrace{\sum_{i=1}^{n} a_i}_{=3n} = \sum_{i=1}^{n}\left(t_i + (\phi_i - \phi_{i-1})\right)$$

Amortisierte Analyse

$$= \left(\sum_{i=1}^{n} t_i\right) + (\phi_n - \phi_{n-1}) + (\phi_{n-1} - \phi_{n-2}) + \ldots + (\phi_1 - \phi_0)$$

$$= \sum_{i=1}^{n} t_i + \underbrace{\Phi_n}_{\geq 0} - \underbrace{\Phi_0}_{=0}$$

$$\geq \sum_{i=1}^{n} t_i$$

$$\Downarrow$$

$$3n \geq \sum_{i=1}^{n} t_i$$

$\Rightarrow$ Die mittlere Zeit pro Einfügung ist $O(1)$.

$$\frac{t_1 + \ldots + t_n}{n} = 3$$

$a_i \geq$ Mittlere Zeit einer Einfügung

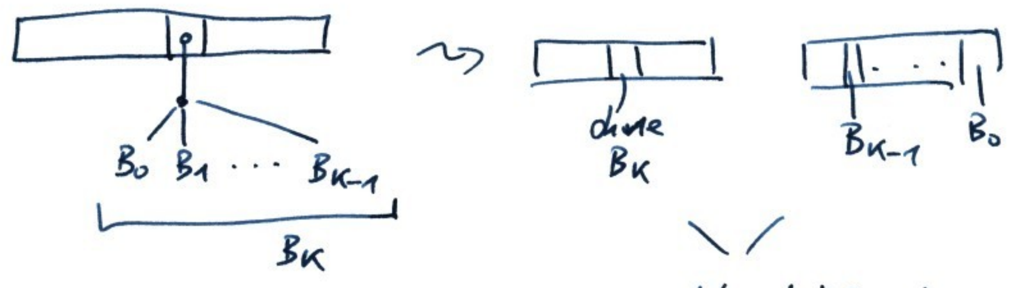
24. 4. 17

Laufzeiten: · Einfügen $O(\log n)$

amort. $O(1)$

· Vereinigen $O(\log n)$

· Min. Löschen



Vereinigen in $O(\log n)$

· Schlüsselwert verbessern $(O(\log n))$
(Decrease Key)

Amortisierte Analyse (gilt für jedes n !)

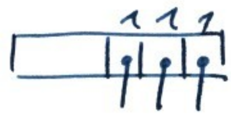
· Einfügen $O(1)$

· Min Löschen $O(\log n)$

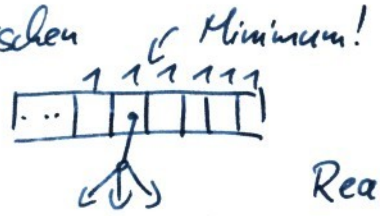
· Min Lesen ~~$O(\log n)$ / $O(\log n)$~~ $O(1)$ mitführen!

$\phi_0 = \emptyset$, leerer Heap am Anfang.

Idee: Einfügen



Minimum Löschen



Reale Zeit $O(\log n)$

$\Phi_i \leq \log n$

$a_i \leq c \cdot \log n + \underbrace{\log n}_{\Phi_i}$

$= O(\log n)$

24.4.17.

- Dijkstra mit binomialen Heap:

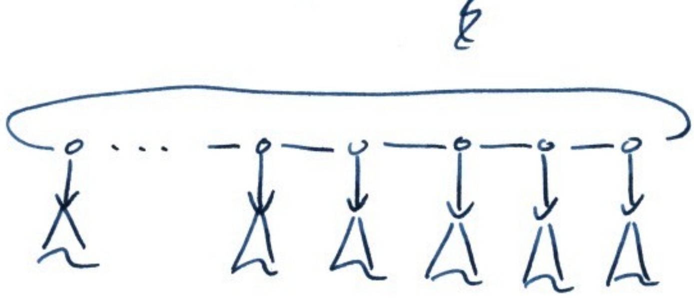
$$O(\underbrace{n \cdot \log n}_{\text{Min. Lösche}} + \underbrace{|E| \cdot \log n}_{\text{Decrease Key}})$$

$\Rightarrow$ Haben noch nichts gewonnen bisher?

2. Fibonacci Heap

Ziel: Decrease Key amortisiert in $O(1)$.

- Datenstruktur: Ringliste (statt Array) mit Bäumen



- Grad eines Baumes = # Kinder der Wurzel.

  (Grad $(B_i) = i$ z.B.)

- Operationen darauf:
  - Decrease Key mit Rausschneiden.
  - Knoten des Baumes können noch markiert sein (mit *)
  - Cut (x) : 1. Falls x Wurzel, dann Schluß.
    2. Schneide Teilbaum mit Wurzel x raus, d.h. Teilbaum ist neu in der Ringliste,
    falls x markiert: ~~ma~~ Markierung lösche.

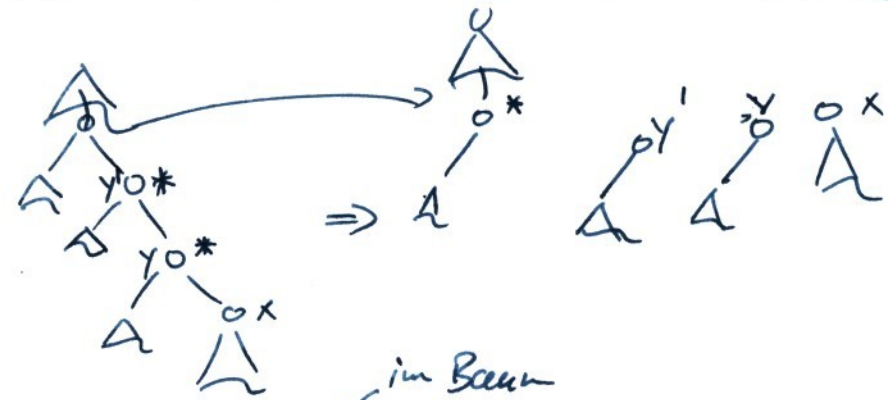
  x: Name eines Knotens

· Cut (x):
⋮

3. y = Vater (x)
4. ~~While y markiert do~~
   if y nicht markiert und nicht Wurzel
   then y markieren, Schluß
5. else cut (y).

---

z.B.



An einem Knoten x im Baum, der nicht Wurzel ist, kann nur einmal ein Kind weggeschnitten werden. (Sonst wird der Knoten x selbst zur Wurzel.)

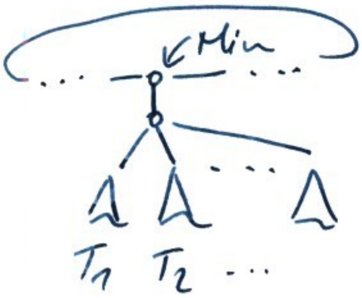
(Cut im worst-case $O(n)$?!)

· DecreaseKey (x, k):
  · Falls erforderlich Cut (x).

· Einfügen: · Einfach an Ringliste hängen, $O(1)$

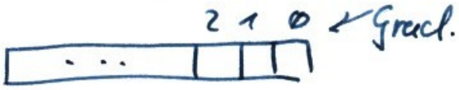
· Minimum Löschen: · haben Minimum

24. 4. 17



Fibonacci-Heap

1. Eintragen aller Bäume, die man noch hat, in ein Array



wobei der Grad des Baumes bestimmt, wo der Baum hinkommt.

Gleicher Grad → Prinzip der binären Addition

2. Neues Minimum suchen.
3. Dann wieder Ringliste.

Satz: n Einfügungen
m Min. Löschen
p Decrease Key
q Min. Finden

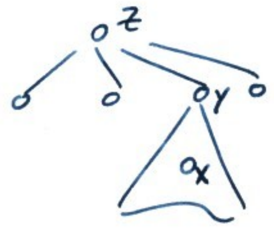
auf anfänglich leerer Struktur haben die Zeit

$$O(n) + O(m \cdot \log n) + O(p) + O(q) \;!$$

Dijkstra jetzt in $O(n \cdot \log n + E)$.

27.4.17

<u>Nachtrag</u>: ∘ Beliebiges Element im binomialen Heap löschen geht in $O(\log n)$



- x mit z Tauschen
- Baum auflösen, gibt max. $\log n$ Kinder von x, x löschen
- im Baum mit z das z steigen lassen, steigt bis maximal y
- die Bäume mit dem restlichen Heap verschmelzen

<u>Beweis</u>: Φ(Fibonacci Heap) = # Bäume + (# markierter Knoten) $\cdot 2$

Fh_0 = leerer Heap

$Fh_1, Fh_2, \ldots, Fh_N$ mit realen Zeiten $t_1, t_2, \ldots, t_N$

Reale Zeiten

Fh_i: Baum nach i-ter Operation

$N = n+m+p+q$

$$\phi_0 = \Phi(Fh_0) = 0 \xrightarrow{a_1} \phi_1 = \Phi(Fh_1) \xrightarrow{a_2} \phi_2 \xrightarrow{a_3} \cdots \xrightarrow{a_N} \phi_N$$

$$a_i = t_i + \underbrace{(\phi_i - \phi_{i-1})}_{\text{Potentialdifferenz}}$$

27.4.17

Zeigen jetzt: $a_i = O(\log n)$ wenn i-te Operation Minimum löschen

$= O(1)$ wenn i-te Operation Decrease Key, Min. finden, Einfügen

---

$$\sum_{i=1}^{N} a_i = \sum_{i=1}^{N} \left(t_i + (\phi_i - \phi_{i-1})\right)$$

$$= \left(\sum_{i=1}^{N} t_i\right) + (\phi_1 - \phi_0) + (\phi_2 - \phi_1) + \ldots + (\phi_N - \phi_{N-1})$$

$$= \sum_{i=1}^{N} t_i + \underbrace{\phi_N}_{\geq 0} - \underbrace{\phi_0}_{=0} \quad \text{nach Def.}$$

d.h. $\boxed{\sum a_i \geq \sum t_i}$

(Können wir obiges $a_i = O(\log n)$ usw. zeigen, folgt die Behauptung!)

Für die einzelnen Operationen, sei die i-te Op...:

- Min. Finden: $a_i = O(1) + \underbrace{\phi_i - \phi_{i-1}}_{=0} = O(1)$

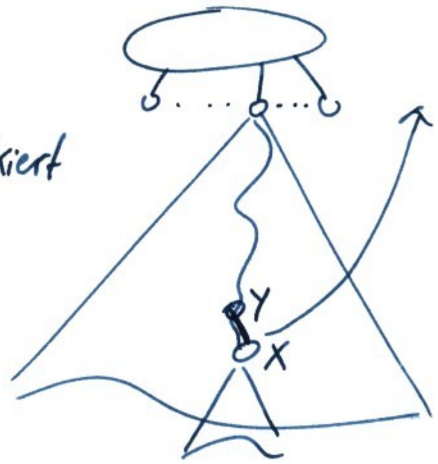
- Einfügen: $a_i = t_i + \underbrace{\phi_i - \phi_{i-1}}_{=1}$ → ein Baum kommt dazu!

$= O(1)$ da $t_i = O(1)$

27.4.17

- **Decrease Key:** $a_i = t_i + \phi_i - \phi_{i-1}$

  - einfacher Fall:

    - Vater von x nicht markiert

      $a_i \le O(1) + 2$

      y markiert (+1)
      x wird neuer Baum
      (+1 falls x noch nicht markiert, sonst 0)

  - allg. Fall:

    - Vater von x markiert und ev. weitere aufeinanderfolgende Vorgänger auch markiert.

      $y_1, y_2, \dots, y_k$ $\quad$ y_k Vater von y_{k-1} … y_1 Vater von x, alle markiert.

    - $a_i \le k \cdot O(1) + k - 2(k-1) + 1$

      $= k \cdot O(1) - k + 3 = \cancel{O(1) + 3 = O(1)}$

      $= O(1)$

      mit passender Konst. für Rausschneiden.

27.4.17

∘ **Minimum Löschen:**

$$t_i = \cancel{\Theta}\, O(\#\text{Kinder d. Minimums}) + \text{Zeit zum Heap aufbauen.}$$

Was ist die # Kinder des Minimums?

Lemma: Im Fibonacci Heap ~~ist~~ gilt folgende Beziehung:

$$\#\text{Elemente in einem Baum } T = 2^{\Omega(\text{Grad}(T))} \quad \cancel{\Omega(2^{\text{Grad}(T)})}$$

$$\Leftrightarrow \text{Grad}(T) = O(\log(\#\text{ Elemente im Baum}))$$

Bsp: Grad = 1

Grad = 2

$2 \leq O(\log_2 3)$

geht nicht

$n \not\leq O(\log_2 n)$

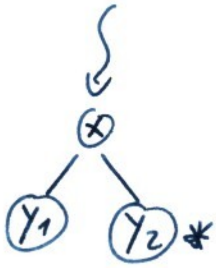
Beweis: Sei x Knoten irgendwo im Baum.

Dann gilt:

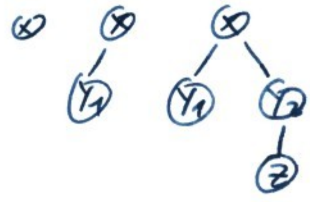
Hat x ein Kind

27.4.17

x hat 2 Kinder:

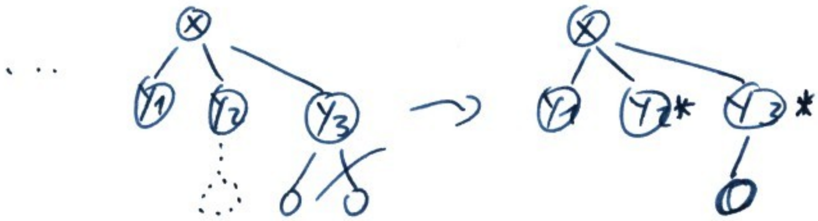


entsteht so:

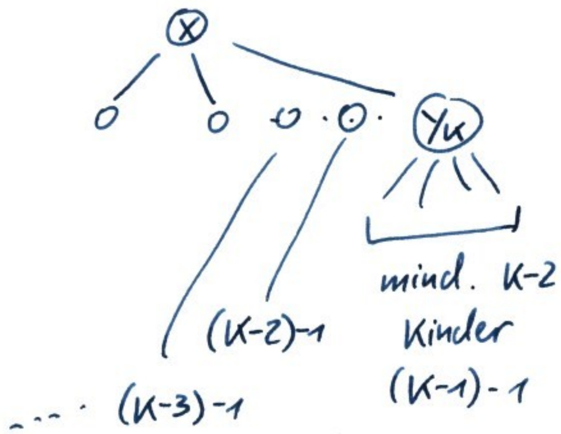


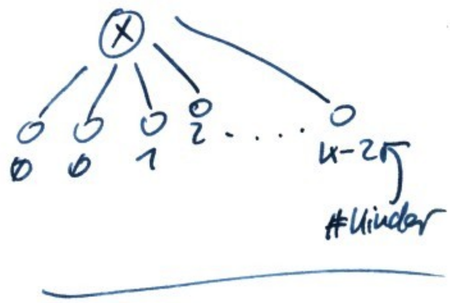
muß hier da sein!
kann dennoch ~~gelöscht~~
gelöscht werden.

x hat 3 Kinder:



allgemein: x hat k Kinder





27.4.17

F_K = minimale #Elemente eines Baumes vom Grad K

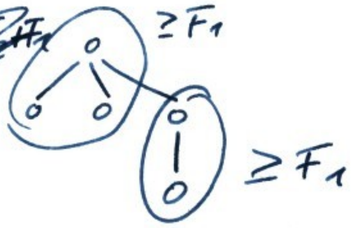
Ziel: $F_K = \cancel{\Omega(2^k)}\ 2^{\Omega(k)}$

$F_0 = 1$

$F_1 = 2$

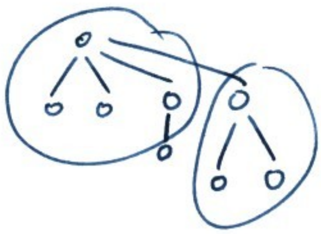
$F_2 = 3$

$\cancel{F_3 \geq F_2 + F_1}$

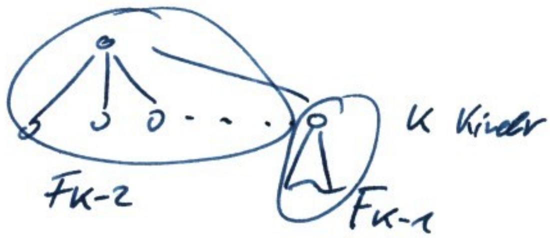


$F_3 = F_2 + F_1$

$F_4 = F_3 + F_2$



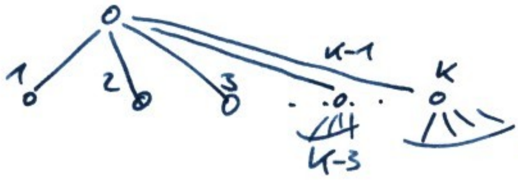
$F_K = F_{K-1} + F_{K-2}$



8.5.17

Für einen beliebigen Knoten im Baum gilt:

Für irgendwelche K Kinder dieses Knotens haben wir die Situation:



Kind i hat $\geq i-2$ Kinder, für $i \geq 2$.

Kind 1 hat ≥ 0 Kinder.

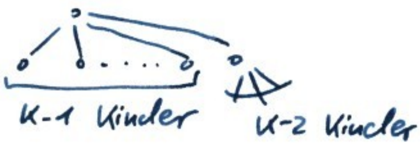
Beziehung zwischen #Elemente im Teilbaum und # Kinder der Wurzel des Teilbaums

$F_K :=$ minimale # Elemente eines irgendwo vorkommenden Teilbaums vom Grad K.

$F_0 = 1$ $\quad F_1 = 2$ $\quad F_2 = 3$ $\quad F_3 = 5 = F_2 + F_1$

(nur Wurzel)

$F_K = F_{K-1} + F_{K-2}$, $K \geq 2$



8.5.17

Ziel: $F_K = 2^{\Omega(K)}$

Ansatz: $F_K = a \cdot c^K$ → Ziel $c \geq 1$

↳ Konstante $\neq 0$

Das bedeutet: $c^K = c^{K-1} + c^{K-2}$ $\quad c \neq 0$

$\Leftrightarrow$

$c^2 = c + 1$

$\Leftrightarrow$

$\left(c - \frac{1}{2}\right)^2 - \frac{1}{4} - 1 = 0$

$\Leftrightarrow c = \frac{1}{2} \pm \sqrt{\frac{5}{4}} = \frac{1}{2}\left(1 \pm \sqrt{5}\right)$

$(c = 1{,}618?$ bzw. $0{,}6\ldots ?)$

Ziel: $F_K \geq a \cdot \left(\frac{1+\sqrt{5}}{2}\right)^K$

$F_0 = 1$ ✓

$F_1 = 2$ ✓

$F_2 = 3$ ✓

Induktionsschluß mit $a = 1$ ✓

Damit haben wir gezeigt:

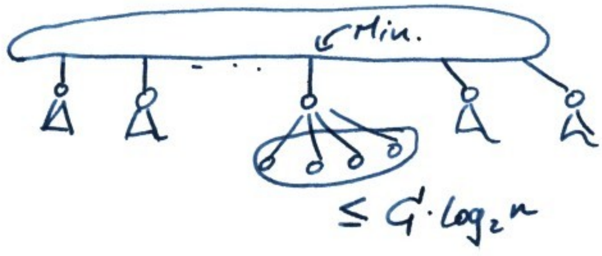
\# Knoten im Baum vom Grad K

$\geq 2^{K \cdot \log\left(\frac{1+\sqrt{5}}{2}\right)}$

■

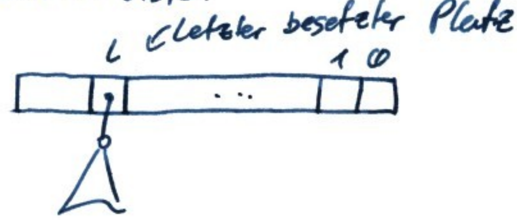
8.5.17

Zurück zur amortisierten Analyse von Minimum Löschen:



Reale Zeit = Zeit die Bäume wobei vom Minimum die Kinder in Array einzutragen und dann in Liste.

Array:



$l \le O(\log n)$

Neues Potential $\le l \cdot d + \#$ markierte Knoten $\cdot c$

Altes Potential $= \#$Bäume $\cdot d + \#$markierte $\cdot c$

$$\left(\begin{array}{l} \#\text{mark. neu} - \#\text{mark. alt} \\ \le - \#\text{Kinder d. Min} \cancel{= O(\log n)} \le 0 \end{array} \right)$$

~~max~~ Markierungen der Kinder des Min verschwinden

$$\text{Reale Zeit} = O(\log n) + \underbrace{\#\text{ alte Bäume}}_{\text{kann } n \text{ werden.}}$$

Amortisierte Zeit:

$$\le O(\log n) + \#\text{alte Bäume} + \text{neues Pot.} - \text{altes Pot.}$$

$$\le O(\log n) + \#\text{alte Bäume} + O(\log n) \cdot d - \#\text{alte Bäume}$$

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

8.5.17

Implementierung (Verpointerung)

Grad mitführen

Min Löschen

- Zeiger auf 1. Kind
- Zeiger auf Vater
- Doppelte Ringliste der Kinder.
- Grad

3. Selbstorganisierende Listen

8.5.17

Dictionary Datenstruktur:

- Operationen: Einfügen, Löschen, Finden
- Jedes Element hat Namen (Schlüssel)
- Auf Listen: $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n$

> Beim Einfügen: Vorher testen, ob Element dabei.
> **Damit:** alle Operationen im worst-case $\Omega(n)$
> n: maximale Elementanzahl

Bsp: Find(x), Find(x), ..., Find(x), dann günstig

$x \rightarrow \ldots$
↑ x vorne

(Liste bei Operationen umorganisieren!)

Ein genauer Rahmen zur Umorganisation:

- Find(x):
  1. Suchen vom Anfang aus bis zum x.
  2. Transpositionen

     ↳ Eine Transposition = Vertauschung von zwei benachbarten Elementen.

Bsp. Finde (x_n)

$x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n$

$\Rightarrow x_n \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots$

$n-1$ Transpositionen.

<u>Übung</u>: Wert von F_K genau.

mit $F_K = c_1^K + c_2^K$ oder so ?!

<u>Übung</u> Aufstellung der Zeiten

worst-case / amortisiert.

8.5.17

- Einfügen (x)
  1. Find (x)
  2. Falls x nicht dabei, x ans Ende hängen.
  3. Transpositionen (nach belieben)

- Löschen (x)
  1. Finde (x)
  2. Falls gefunden, dann Löschen
  3. Transpositionen

Möglichkeiten für Transpositionen

1. ~~Nichts~~ Nichts (Blank, B)
2. MF (move-to-front)

   Find (x): x an die Spitze tauschen.

   Einfügen (x): nach Einfügen, x vom Ende an die Spitze tauschen

   Löschen (x): Keine Transpositionen

3. TR (transpose)

   Find (x), Einfügen (x): x eins nach links tauschen.

   Löschen (x): keine Transposition

4. FC (frequency counter)

   In Zähler (x) merken, wie oft Find (x) aufgerufe

   Nach Einfüge (x): Zähler (x) = 1

   ⇒ Liste nach Zähler (x) ordnen, vorne ~~groß~~ groß / hinten klein.

11.5.17

Beispiele

Nichts:

$\overbrace{\text{Einf.}(1);\ \text{Einf}(2);\ \ldots;\ \text{Einf}(n)\qquad \text{Finde}(n);\ \ldots;\ \text{Finde}(n)}$ m-mal

() 1 1,2 1,…,n

Zeit bei Heuristik Nichts:

$$1+2+\ldots+n+m\cdot n = \frac{n(n+1)}{2}+m\cdot n$$

Move-to-front:

() $\to 1$ $\to 2, \to 1$ $\to 3 \to 2 \to 1$, …, $n \to n-1 \to \ldots \to 2 \to 1$,

bleibt so.

Zeit:

ans Ende gehen

$$1+(2+1)+(3+2)+\ldots+(n+(n-1))+m\cdot 1$$

nach vorne tauschen

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

Mit $m=n^2$ hat Nichts $O(n^3)$ und MF $O(n^2)$

Frequency Counter:

() $\to 1_1$ $\to 1_1 \to 2_1$ … $\to 1_1 \to 2_1 \to \ldots \to n_1$ nach d. Einfügen.

Zähler

1. Finde (n)

$\to n_2 \to 1_1 \to 2_1 \to \ldots \to (n-1)_1$

dann bleibt, erhöht nur Zähler

$$\frac{n(n+1)}{2}+n+(n-1)+(m-1)\cdot 1 = O(n^2)+m$$

Tausche

Transpose (TR)

$() \quad 1 \quad 2 \to 1 \quad 2 \to 3 \to 1 \quad \ldots \quad 2 \to 3 \to \ldots \to n \to 1$

Find (n)

$2 \to 3 \to \ldots \to n \to (n-1) \to 1$

nach insg. $n-2$ Find (n)

$n \to 2 \to 3 \to \ldots \to (n-1) \to 1$

Zeit: $1 + (2+1) + (3+1) + \ldots + (n+1)$

$$\frac{n(n+1)}{2} + n - 1 + (n-1) + 1 + (n-2) + 1 + \ldots + (n-(n-1)+1)$$
$$+ (m-(n-1)+1)$$

$$O(n^2) + m.$$

Transpositionen nach links zählen beim Kosten mit!

Zu FC vs. MF. betrachte folgende Operationen:

$\cancel{E}$ $\text{Einf}(1), \underbrace{\text{Finde}(1), \ldots, \text{Finde}(1)}_{n-1 \text{ mal}}, \text{Einf}(1), \underbrace{\text{Finde}(2), \ldots, \text{Finde}(2)}_{n-2 \text{ mal}},$

$\ldots, \text{Einf.}(n-1), \text{Finde}(n-1), \text{Einf.}(n)$

FC: $\to 1_1 \quad \ldots \quad \to 1_n \quad 1_n \to 2_1 \quad \ldots \quad 1_n \to 2_{n-1} \quad \ldots \quad 1_n \to 2_{n-1} \to \ldots$
$\to (n-1)_2 \to n_1$

Zeit: $\underset{E(1) + n \cdot F(1)}{1 \cdot n} + \underset{\substack{\text{Einf}(2) + (n-2) \cdot F(2) \\ 2 \qquad 2(n-2) \\ \cancel{2n-4+2}}}{2(n-1)} + 3(n-2) + 4(n-3)$
$+ \ldots + (n-(n-2))(n-1)$
$+ (n-(n-1))\, n$

11.5.17

$$\cancel{= \sum_{i=1}^{n} i + \sum_{i=1}^{n} in - \sum_{i=1}^{n} i^2}$$

$$= \sum_{i=1}^{n} i \cdot n - \left(\sum_{i=1}^{n-1} i^2 + \sum_{i=1}^{n-1} i \right)$$

$$= n \frac{(n+1)n}{2} - \left(\frac{1}{6} \left((n-1)(n)(n+1) \right) + \frac{n(n \cancel{+1} {-1})}{2} \right.$$

$$= \underline{\underline{\Omega(n^3)}}$$

<u>**MF**</u>

( ) 1 . . . 1 → 2 → 1 . . . . 3 → 2 → 1 . . . . n → (n−1) → . . . → 1

<u>Zeit</u> $1 \cdot n + 2+1+(n-2) + 3+2+(n-3) + 4+3+(n-4)$

$+ (n-1)+(n-2)+n-(n-1) + n + n-1$

$= n + 1+n + 2+n + 3+n + 4+n + \ldots +$

$(n-2)+n + (n-1)+n$

$$= n^2 + \frac{n(n+1)}{2} = \mathcal{O}(n^2)$$

Satz: Es gibt eine Konstante c, so dass
Für jede Folge von Operationen σ
beginnend auf der leeren Liste gilt:

$\underbrace{\text{Zeit}(\sigma)}_{\text{unter MF}} \leq c \cdot \text{Zeit von } A$, wobei A irgendeine Heuristik in unserem Rahmen
(A kann sogar von σ abhängen!)

Beachte: Gilt ~~nicht~~ für FC statt MF nicht!

Beweis: $\sigma = \sigma_1, \ldots, \sigma_n$

		a_1	a_2	a_n
		t_1	t_2	t_n ← Zeit MF
MF	$()=S_0$	S_1	S_2	S_n
		σ_1	a_2	σ_n ← Operation
A	$()=S_0'$	S_1'	S_2'	S_n'
		t_1'	t_2'	t_n' ← Zeit A
	ϕ_0	ϕ_1	ϕ_2	ϕ_n

S_i' = Vertauschung von S_i

Potentialfunktion:

$\phi(S_i, S_i')$ = # Inversionen von S_i, S_i'

Eine Zweiermenge $\underset{x \neq y}{\{x, y\}}$ ist eine Inversion von S_i, S_i'
gdw.
$S_i = \ldots x \to \ldots \to y \ldots\ldots$ „x vor y"
$S_i' = \ldots y \to \ldots \to x \ldots\ldots$ „y vor x"

z.B.:
$S_i \quad 1 \to 2 \to 3$
$S_i' \quad 3 \to 2 \to 1$
$\binom{3}{2} = 3$ Inversionen

$0 \leq \phi(S_i, S_i') \leq \binom{m}{2}$, m = # Elemente von S_i

Z.B.: 2 Listen mit nur 1 ~~F~~ Inversion

$1 \to 2 \to 3$ nur 2 benachbarte
$2 \to 1 \to 3$ vertauscht.

$\phi_i = \phi(s_i, s_i')$

- amortisierte Zeit über Zeit von MF.

$$\boxed{a_i = t_i + \phi_i - \phi_{i-1}}$$

- Es gilt $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} t_i + \phi_n - \phi_0$, also $\sum t_i \leq \sum a_i$

Ziel: $a_i \leq c \cdot t_i'$ für alle i.

Zeit von A

$t_i + \phi_i - \phi_{i-1}$

Bsp: Konstruieren Fall: t_i groß, t_i' klein

$\sigma_i = \text{Find}(a)$

$t_i = n + n - 1$

$S_{i-1} = \underbrace{\ldots \text{Rest} \ldots \to a}_{n \text{ Elemente}}$ $\qquad S_i = a \to \underbrace{\ldots \text{Rest gleich} \ldots}$

$\underbrace{S_{i-1}' = a \to \underbrace{\ldots \text{Rest}' \ldots}}_{\phi_{i-1}}$ $\qquad \underbrace{S_i' = a \to \underbrace{\text{Rest}'}}_{t_i' = 1 \quad \phi_i}$

~~$t_i =$~~

11.5.17

$\phi_i - \phi_{i-1} = ?$ Im Rest, Rest' keine Änderung

$\phi_{i-1} = \#\text{Inv. in } (Rest, Rest') + n-1$

$\phi_i = \#\text{Inv. in } (Rest, Rest') + 0$

↳ Keine Inv. mit a.

$\phi_i - \phi_{i-1} = -(n-1)$

$a_i = n + (n-1) - (n-1)$

$= n$ Stimmt noch nicht.

neue Potentialfunktion.

$\phi(S_i, S_i') = D \cdot (\#\text{Inversionen von } S_i, S_i')$

wobei D eine Konstante ≥ 2 ist.

jetzt:

$\phi_{i-1} = D \cdot (\#\text{Inv. in } (Rest, Rest') + n-1)$

$\phi_i = D \cdot (\#\text{Inv. in } (Rest, Rest') + 0)$

$\phi_i - \phi_{-1} = -D(n-1)$

$a_i = n + (n-1) - D(n-1) \leq 1$ für $D \geq 2$ damit $a_i \leq t_i'$

Ziel: $a_i \leq c \cdot t_i'$

$$\boxed{\begin{array}{ccc} & a_i & \\ & t_i & \\ S_{i-1} & \rightsquigarrow & S_i \\ & \sigma_i & \\ S'_{i-1} & \rightsquigarrow & S'_i \\ & t'_i & \\ \phi_{i-1} & & \phi_i \end{array}}$$

Fallunterscheidung nach σ_i:

$\sigma_i = \text{Finde}(x)$

Teilen die Schritte auf:

$$\begin{array}{lcccc} MF & S_{i-1} & \xrightarrow{t_i} & S_i & S_i \\ A & S'_{i-1} & & S'_{i-1} \xrightarrow[t'_i]{} & S'_i \end{array}$$

$a_{i,1} = t_i + \phi(S_i, S'_{i-1}) - \cancel{\phi(} \phi_{i-1}$

$a_{i,2} = 0 + \phi_i - \phi\cancel{(S_i, S'_i)}\, \cancel{S_{i-1}}\, \phi(S_i, S'_{i-1})$

$a_i = a_{i,1} + a_{i,2}$

15.5.17

<u>Satz:</u> Folge $\sigma = \sigma_1, \dots, \sigma_m$ auf Listen.

Zeit von σ auf MF beginnend bei () (leere Liste)

$\leq 4 \cdot$ Zeit von σ bei A, für beliebiges A.

<u>Beweis:</u>

$$\text{MF:}\quad () = S_0 \overset{\sigma_1}{\rightsquigarrow} S_1 \rightsquigarrow \quad \dots \quad \overset{\sigma_m}{\rightsquigarrow} S_m$$

$$t_1 \qquad\qquad t_m$$

$$a_1 \qquad\qquad a_m$$

$$\text{A:}\quad () = S_0' \rightsquigarrow S_1' \rightsquigarrow \quad \dots \quad \rightsquigarrow S_m'$$

$$t_1' \qquad\qquad t_m'$$

↑ beliebig

S_i, S_i' sind als Mengen gleich (gleiche Elemente, nur Permutationen voneinander)

Amortisierte Zeit: $a_i = t_i + \Phi_i + \Phi_{i-1}$

Potentialfunktion: $\phi_i = 2 \cdot (\#\text{Inversionen in } (S_i, S_i'))$

<u>Bsp.:</u> S_i: $x \to \dots \to y$

S_i': $y \to \dots \to x$

↑ gleich

$\Rightarrow$ (x,y) $(x,\cdot)$ $(y,\cdot)$

$1 + (n-2) + (n-2)$ Inversionen

$\phi(S_i, S_i') = 2 \cdot (\dots)$

Es gilt $\sum t_i \leq \sum a_i$. Zeigen jetzt, dass gilt: für alle i ist $a_i \leq 4 \cdot t_i'$!

Fallunterscheidung nach σ_i:

⇓

15.5.17.

<u>O_i = Einfügen(x)</u>, x nicht dabei

MF: $S_{i-1} \rightsquigarrow \overset{x \to S_{i-1}}{\cancel{S_{i-1} \to x}} \rightsquigarrow S_i = x \to S_{i-1}$

A: $S'_{i-1} \rightsquigarrow S'_{i-1} \to x \rightsquigarrow S'_i = \dots \to x \to \dots$

$$a_i = \overset{n+1}{\cancel{t_i}} + \underbrace{\phi(\overset{x \to S_{i-1}}{\cancel{S_{i-1} \to x}}, S'_{i-1} \to x)}_{=\phi'} - \phi(S_{i-1}, S'_{i-1})$$

Zeit von MF

$$+ \overset{n}{\cancel{\phi}} + \phi(S_i, S'_i) - \underbrace{\phi(\overset{x \to S_{i-1}}{\cancel{S_{i-1} \to x}}, S'_{i-1} \to x)}_{=\phi'}$$

$(t_i = n+1+n)$

n = # Elemente in S_{i-1} bzw. S'_{i-1}

$

- $\phi' - \phi(S_{i-1}, S'_{i-1}) \le 2n$ (jedes Paar mit x trägt eine Inversion bei)

~~$\phi_i - \phi'_i =$~~

- $\phi(S_i, S'_i) - \phi' \le 2 \cdot (t'_i - (n+1))$

A hat Zeit t'_i. Wieviel Zeit für Transpositionen hat A noch übrig? Nur $t'_i - (n+1)$!

Jede Transposition erzeugt maximal eine neue Inversion!

Fassen wir zusammen:

$$a_i \le n+1+n + 2n + 2(t'_i - (n+1))$$
$$\le 2t'_i + 2n \le \underline{4t'_i} \quad (\text{mit } t'_i \ge n)$$

15.5.17

$\sigma_i = \text{Finde}(x)$, x dabei

MF: $S_{i-1} \overset{t_i}{\leadsto} x \to S_{i-1} \leadsto x \to S_{i-1} = S_i$

A: $S'_{i-1} \leadsto S'_{i-1} \underset{t'_i}{\leadsto} S'_i$

$\Phi' = \Phi(x \to S_{i-1}, S'_{i-1})$

S_{i-1} sieht so aus: $C \to x \to D$

S'_{i-1} : $C' \to x \to D'$

Elemente in $C =: c$ usw.

$C' =: c'$

$a_i = t_i + (\Phi' - \Phi_{i-1}) + 0 + (\Phi_i - \Phi')$

$t_i = (c+1) + c$ (Finden von x + Transpositionen)

$= \underline{\underline{2c+1}}$

$S_{i-1}: \overbrace{\underbrace{\dots y \dots}_{l \quad c-l}}^{C} \to x \to D \leadsto x \to \dots \to y \to \dots \to D$

$S'_{i-1}: C' \to x \to D' \leadsto C'' \to x \to D'$

y trägt Inv. bei, sofern es in C' vorkommt!

l := # Elemente ~~die~~ in C, die auch in C' stehen.

$(\Phi' - \Phi_{i-1}) = 2(l - (c-l)) = 2(2l - c) = \underline{\underline{4l - 2c}}$

15.5.17

Zeit von A für Transpositionen:

$$t_i' - (c'+1)$$

$$\phi_i - \phi' \leq 2 \cdot \left(t_i' - (c'+1) \right)$$

$$a_i \leq 2c + 1 + 4\ell - 2c + 2t_i' - 2c' - 2$$

$$= \cancel{4\ell + c}\ 4\ell + 2t_i' - 2c' - 1$$

$$\leq 4\ell + 2(t_i' - c') \qquad (\ell \leq c' \text{ nach } \cancel{\text{def}} \text{ Def. } \ell)$$

$$\leq 2\ell + 2t_i' \qquad (\ell \leq t_i')$$

$$\leq \underline{\underline{4t_i'}}$$

$\underline{\underline{\sigma_i = \text{Lösche}(x)\ ,\ x \text{ dabei}}}$

$$MF: \quad S_{i-1} = C \to x \to D \leadsto C \to D \leadsto C \to D = S_i$$

$$A: \quad S_{i-1}' = \underbrace{C' \to x \to D'}_{\phi_{i-1}} \leadsto \underbrace{C' \to D'}_{\phi'} \leadsto \underbrace{S_i'}_{\phi_i}$$

$c :=$ # Elemente in C, usw.

$$a_i = t_i + (\phi' - \phi_{i-1}) + 0 + (\phi_i - \phi')$$

$$\underline{\underline{t_i = c+1}}$$

$$\phi' - \phi_{i-1} = 2(-\ell - \ell') = -2(\ell + \ell')$$

($\ell :=$ # vor x in C, nach x in D'

$\ell' :=$ # vor x in C', nach x in D)

15.5.17

$$(\phi_i - \phi') \leq 2(t_i' - (c'+1))$$

$$a_i \leq c+1 - 2(l+l') + 2(t_i' - (c'+1))$$

$$\left(\begin{array}{l} c - l = c' - l' \\ \Leftrightarrow c = c' - l' + l \end{array} \right.$$

$$= c' - l' + l - 2(l+l') + 2(t_i' - (c'+1))$$

$< 4 t_i'$ (da rest < 0).

<u>Sonderfälle</u>: Einfügen mit x dabei, usw.
eventuell Übungsaufgabe!

<u>Bsp</u>: Satz gilt <u>nicht</u>, wenn Liste am Anfang nicht leer.

$A: \; x \to \overbrace{\ldots\ldots}^{n \text{ Elemente}}$ $\qquad t' \quad \overset{\text{alles bleibt.}}{x \ldots \ldots}$

$MF: \; \to \underbrace{\ldots\ldots}_{\text{Dieselben } n} \to x$ $\qquad t \quad x \to \ldots\ldots$

$t = n + 1 + n$

$t' = 1$, keine Transp.

Wo geht der Beweis nicht?

$a = t - 2n$

$a = 1 \leq 4 \cdot 1$
$\qquad\qquad\quad = t'$

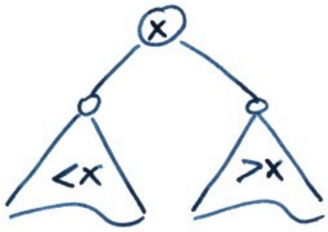
$a = t + \overbrace{\phi_1 - \phi_0}^{= -2n} \leftarrow > 0$

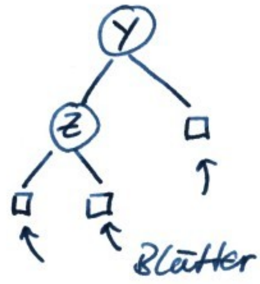
$t = a + \underbrace{\phi_0 - \phi_1}_{\neq 2n}$ $\qquad$ nicht $\underline{\underline{t \leq a}}$

22.5.17

4. Splay-Bäume

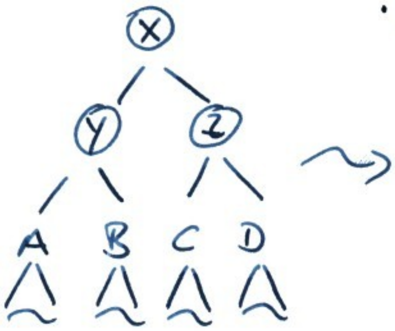
(MF für Suchbäume)





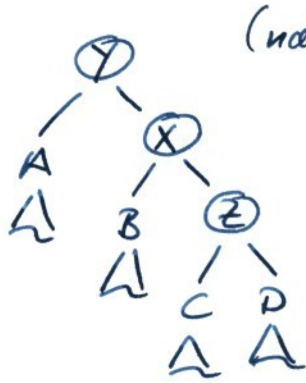
Immer 2 od. 0 Kinder.

Rotationen



· Rotation um x nach rechts:

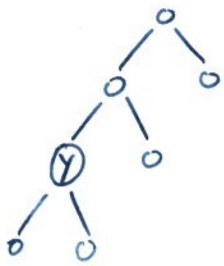
(nach links geht analog)



· Operationen Einfügen, Finden, Löschen analog zu Listen von vorher

· Rotation entspricht einer Transposition

· Idee: „MF" auf Suchbäumen



Finde (y)

1. Von Wurzel zu y gehen
2. So rotieren, dass y an Wurzel kommt

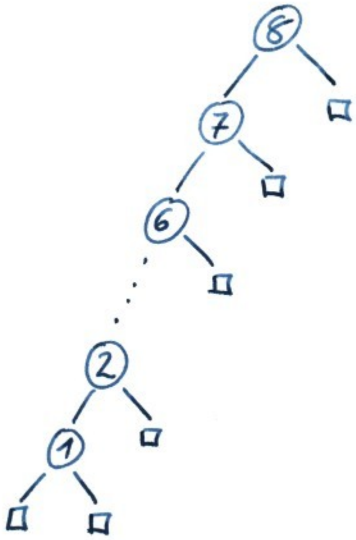
Dazu iteriert: Rotation um Vater von y

22.5.17

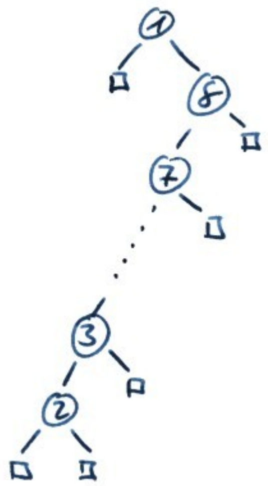
(y linker Sohn, dann nach rechts; sonst nach links) [Zeit: O(Tiefe von y)]

<u>Beispiele:</u> · ein Gegenbeispiel

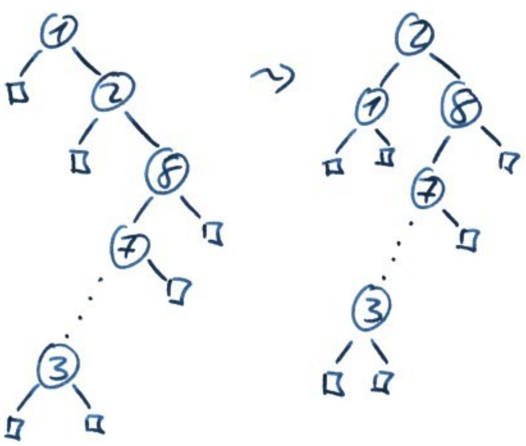
· Angenommen, wir haben zu Liste entarteten Baum.



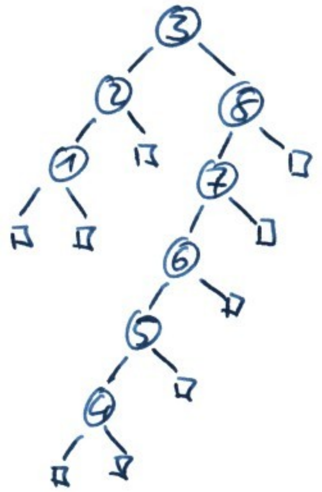
· <u>Finde(1) führt zu:</u>



· <u>Finde(2)</u>



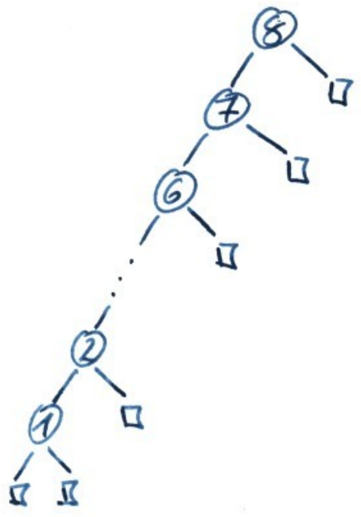
· <u>Finde(3)</u>



usw. Finde(4), …, Finde(8)

22.5.17

nach Finde(8):



n Elemente am Anfang als Liste. (absteigend)
Machen darauf Find(1), ..., Find(n).
Dann Zeit:

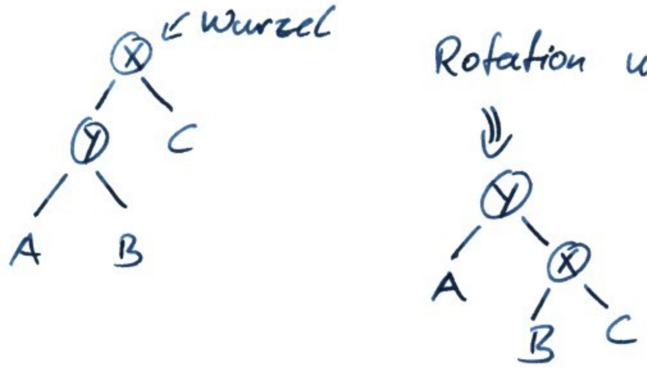
$$n + (n-1) + \ldots + 1 = \frac{n(n+1)}{2}$$

Machen m Runden, m·n Operationen
⇝ im mittel $\frac{n+1}{2}$ f. eine Operation
(Linear!)

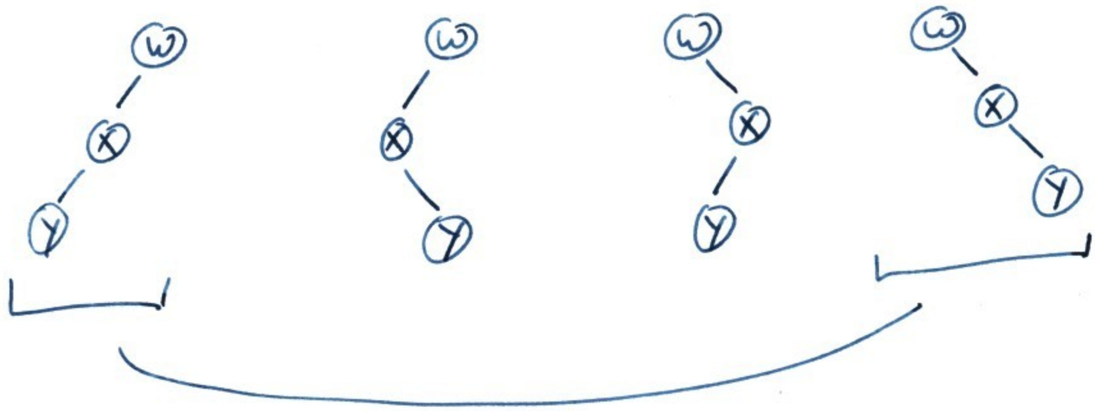
Das motiviert ein etwas geschickteres Rotieren!

Splaying(y)

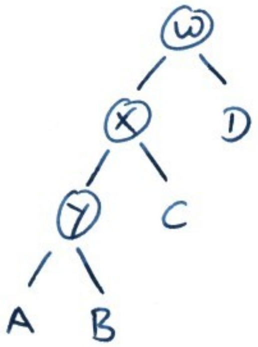
1. y ist Wurzel, dann nichts
2. Rotation um x



3. y hat Vater x, ~~Großfater~~ ~~Groso~~ Großvater w

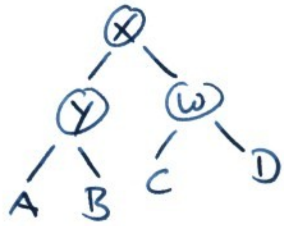


besondere Rotation

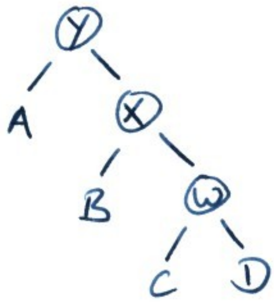


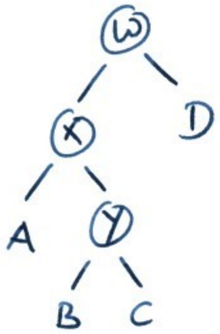
Splaying (y):

1. Rotation um w nach rechts



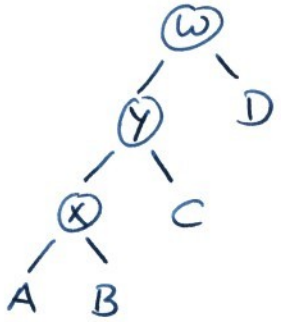
2. Rotation um x nach rechts



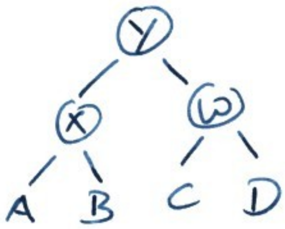


Splaying (y):

1. Rotation um x nach links



2. Rotation um w nach rechts

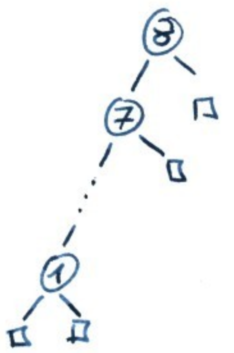


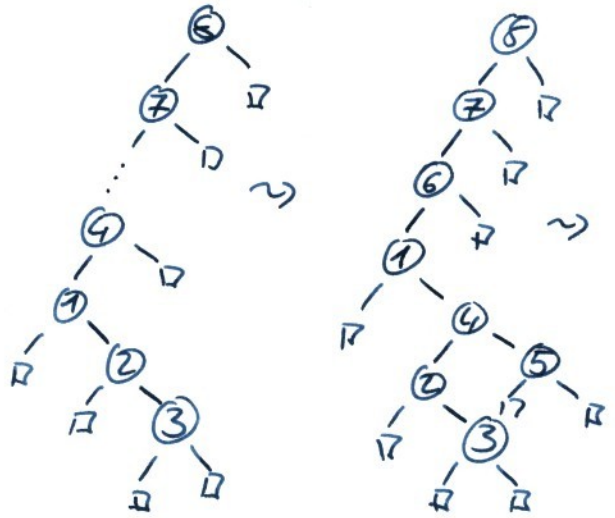
- Symmetrische Fälle gehen dann analog.

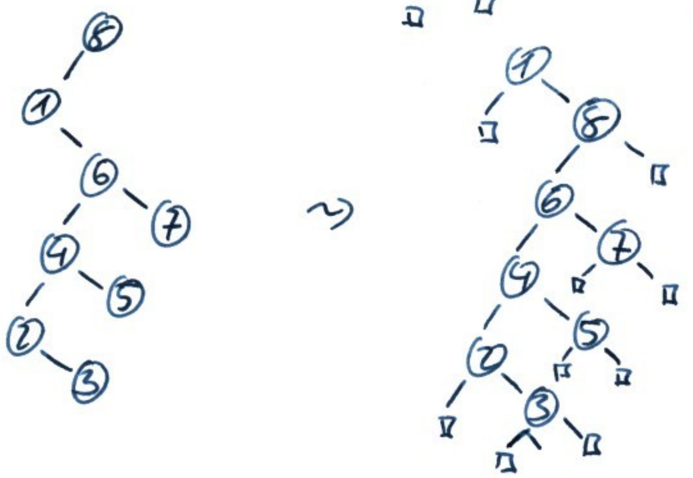
22.5.17

Das Eingangsbeispiel mit Splaying-Operation.

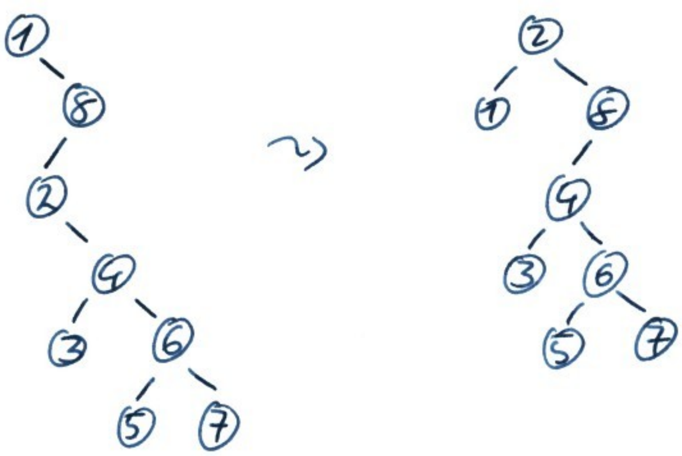
Finde (1):



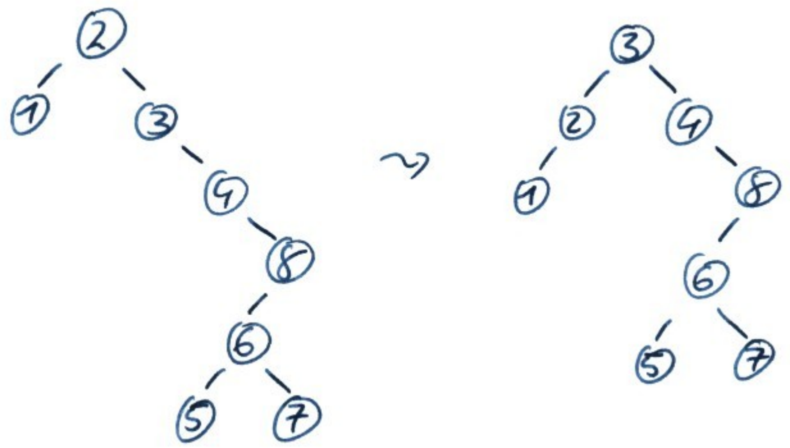




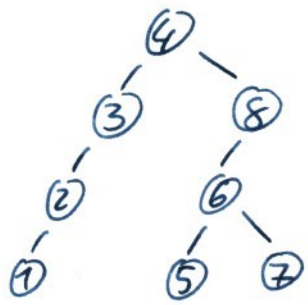
Finde (2):



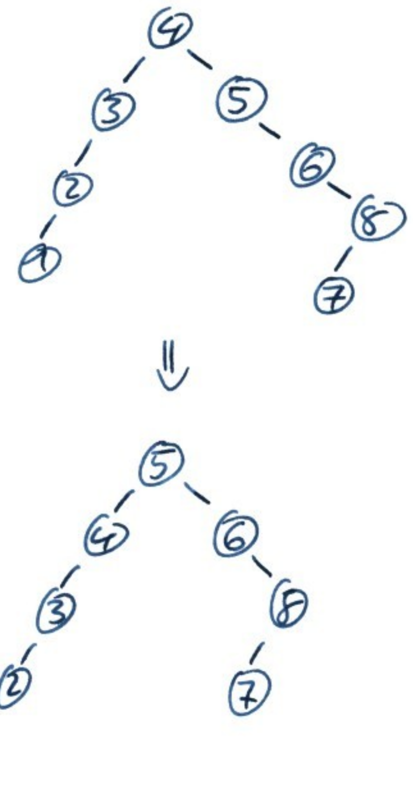
Finde (3)



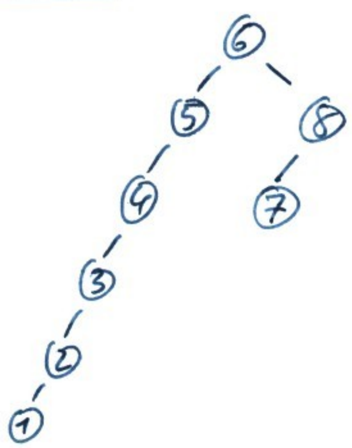
Finde (4):



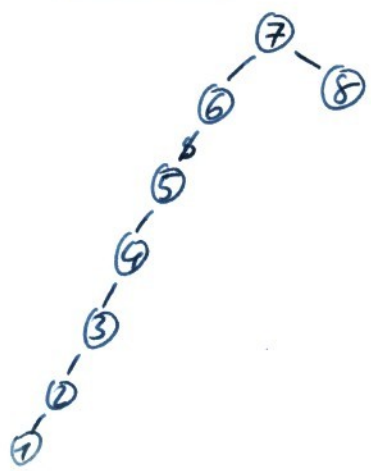
Finde (5):



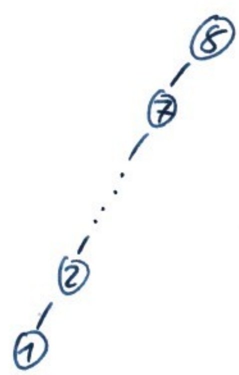
Finde (6):



Finde (7):



Finde (8):



22.5.17

Zeit: Finde 1: 8
2: 5
3: 4
4: 2
5: 4
6: 2
7: 3
8: 2

30

Am ~~Ein~~ Ende wieder entarteter Baum, aber Zeit eventuell besser!

Operationen auf dem Splay-Baum

Finde(y):
1. gehe zu y
2. if y gefunden
   while y ≠ Wurzel
      Splaying(y)
3. if y nicht gefunden
   z = Vater des Blattes, wo Suche endet
   while z ≠ Wurzel
      Splaying(z)

Ergebnis der Suche von y.

Verhältnis von y zu z.:

$y < z$

es gibt kein a mit $y < a < z$!

(z ist kleinstes Element > y) | z ist dir. Nachfolger von y

Bezogen auf El. im Baum

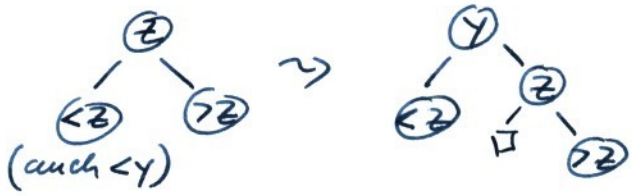
22.5.17

Einfüge(y): 1. Finde(y)

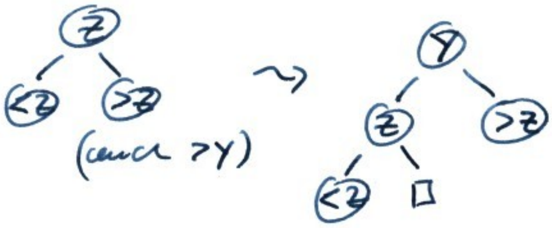
2. if Wurzel ≠ y

z = Wurzel

if $y < z$ then



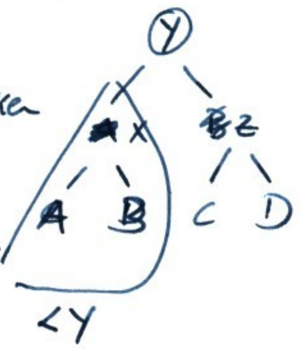
if $y > z$ then



Lösche(y): 1. Finde(y)

2. if y Wurzel

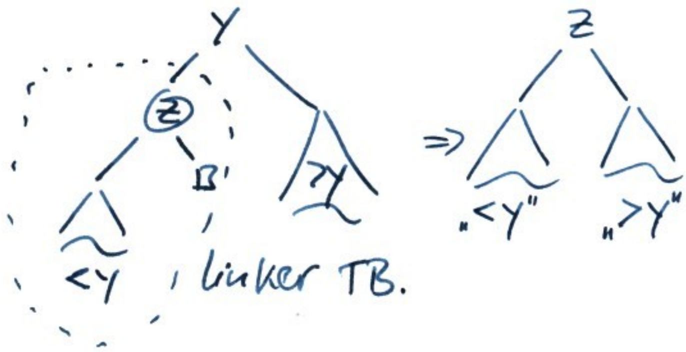
- Find (y) im linken Teilbaum
- findet dir. Vorgänger von y



(größtes El. in Linken TB, das $< y$ ist)

- sei z ~~dieses~~ der Vater des Blattes, wo die Suche nach y endet.

⇓ Baum dann so:

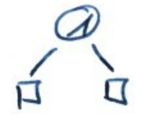


3. z an die Wurzel.

29.5.17

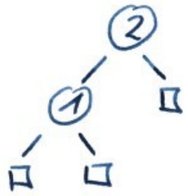
Bsp: Entstehen einer Schlange aus leerem Baum.

- Einfüge(1)

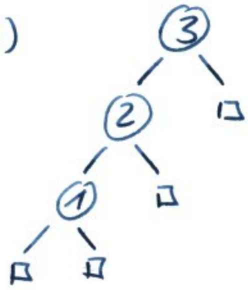


⇒ worst-case für Finden kann $O(n)$ sein.
(hier: dafür aber n mal Einfügen in $O(1)$)

- Einfüge(2)



- Einfüge(3)



- usw.

Ziel: · Operationen $\sigma_1, \sigma_2, \dots, \sigma_m$ auf leerem Baum.

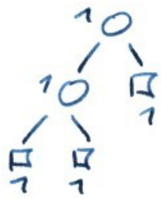
· Maximalanzahl von Elementen im Baum ist n.

· dann Zeit von $\sigma_1, \dots, \sigma_m = \underbrace{O(m \cdot \log n)}_{\text{Konstante hängt von } n \text{ ab.}}$, d.h.

arith. Mittel $O(\log n)$

Definition: gegeben binärer Suchbaum.

a) Einzelgewicht eines jeden Knotens ist $= 1$



(Blätter zählen mit)

b) Totales Gewicht von x, einem Element, das gespeichert ist. (Baum heißt S')

$$Tw_{S}(x) = \#\text{Knoten im Teilbaum mit Wurzel } x$$

(Totales Gew. eines Blattes $= 1$)

c) Rang von x in S'

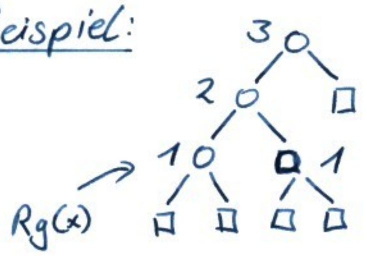
$$Rg_{S'}(x) = \lfloor \log_2 (Tw_{S'}(x)) \rfloor$$

(Der Rang eines Blattes ist 0)

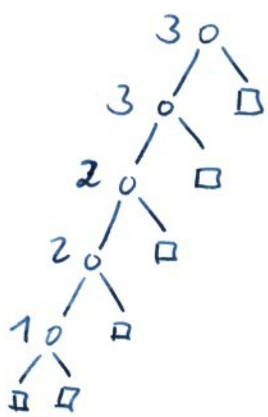
d) Potential Φ

$$\Phi(S') = \sum_{x \in S'} Rg_{S'}(x)$$

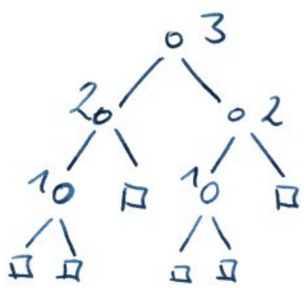
Beispiel:



$\phi = 7$



$\phi = 11$



$\phi = \cancel{7}\,9$

29.5.17

Definition:

Operation σ:

- $\underbrace{\text{Zeit von } \sigma}_{Zeit_{S'}(\sigma)}$ = # Aufrufe von Splaying (≤ 2 Rotationen)

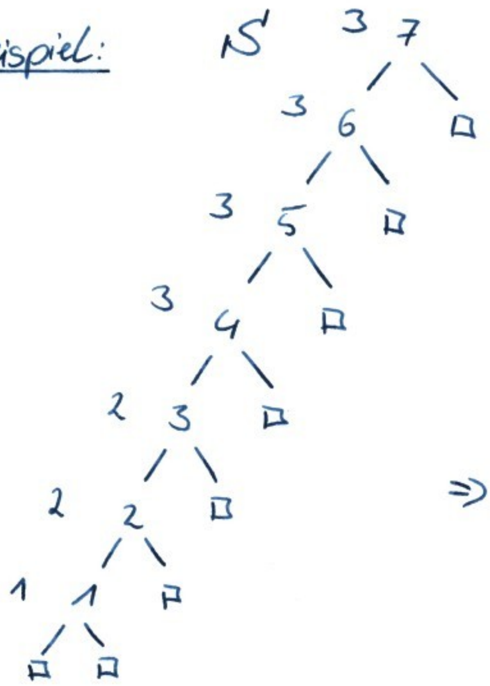
- Amortisierte Zeit von σ auf S'

$$a_{S'}(\sigma) \quad \cancel{= Zeit_{S'}(\sigma)} = Zeit_{S'}(\sigma) + \underbrace{\Phi(S')}_{\text{neuer Baum}} - \Phi(S)$$

$$S \overset{\sigma}{\rightsquigarrow} S'$$

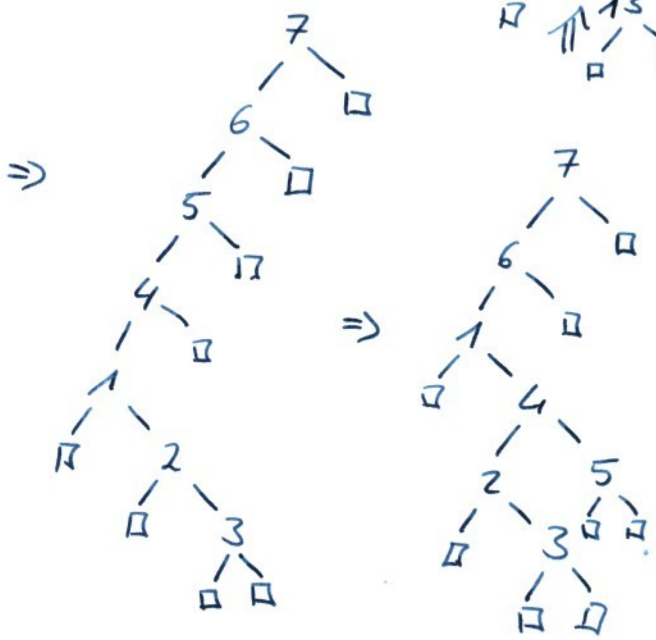
Ziel: $a_{S'}(\sigma) \leq O(\log n)$, n = # Element in S'.

Beispiel:

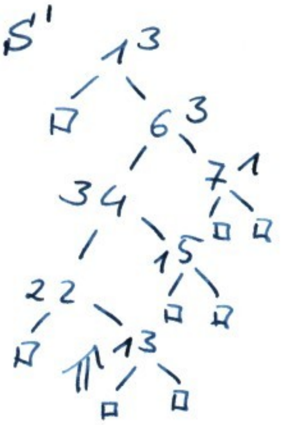


$\boxed{\phi(S) = 17}$

Finde (1)



$\boxed{\phi(S') = 14}$



29.5.17

$$a_{S'}\left(\text{Finde}(1)\right) = \underbrace{3}_{\#\text{ splayings}} + \underbrace{14}_{\phi(S')} - \underbrace{17}_{\phi(S)} = \underline{\underline{0}}$$

Der entscheidende Satz:

· haben binären Suchbaum S

$$a_S\left(\text{Finde}(x)\right) \leq \cancel{11 + 3 \cdot Rg(S)}$$
$$1 + 3 \cdot \left(Rg_S(\overset{w}{\cancel{z}}) - Rg_S(x') \right)$$

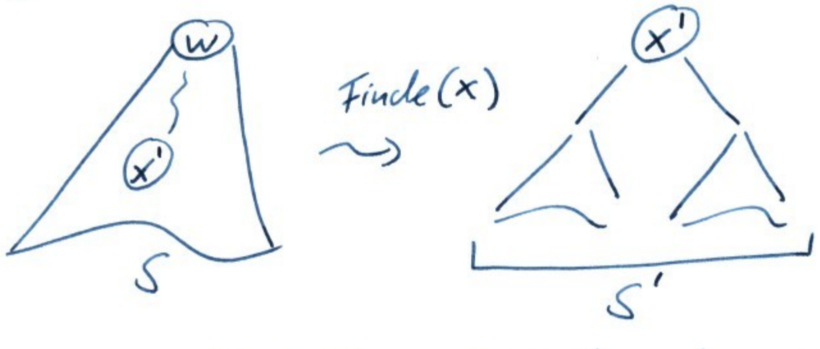
wobei w = Wurzel von S und

x' = Knoten in S, der an die Wurzel kommt.

(Für x dabei, dann $x = x'$)

$$\left(Rg_S(w) - Rg_S(x') \right) \geq 0$$

Sachlage:



Folgerung: $a_S\left(\text{Finde}(x)\right) \leq 1 + 3 \cdot (\log n) = O(\log n)$

29.5.17

Beweis:

· Einfachster Fall: $w = x'$

$$a_S(\text{Finde}(x)) = \cancel{\ldots + \phi(S') - \phi(S)}$$

$$= 1 + \underbrace{\phi(S') - \phi(S)}_{=0},$$

$$= \underline{\underline{1}} = 1 + 3(\underbrace{Rg_S(w) - Rg_S(x')}_{=\underline{\underline{0}}})$$

· ab jetzt $w \neq x'$!

· Finde(x) besteht aus einer Folge von Splayings.

$m = $ #Aufrufe von Splaying bei Finde(x) auf S

$= \text{Zeit}_S(\text{Finde}(x))$ nach Def.

$$S = S_0 \rightsquigarrow S_1 \rightsquigarrow S_2 \rightsquigarrow \ldots \rightsquigarrow S_m = S'$$

1. Splaying, 2. Splaying, …, m. Splaying

$a_i = $ amortisierte Zeit des i-ten Splayings

$$:= 1 + \Phi(S_i) - \Phi(S_{i-1})$$

Dann: $a_S(\text{Finde}(x))$

$= a_1 + \ldots + a_m$

$(= m + \phi(S') - \phi(S))$

Zeigen jetzt:

$$a_m \leq 1 + 3\left(Rg_{S_m}(x') - Rg_{S_{m-1}}(x')\right)$$

$$a_i \leq 3 \cdot \left(Rg_{S_i}(x') - Rg_{S_{i-1}}(x')\right) \qquad (i \neq m)$$

29.5.17

$$Rg_{S_m}(x') = Rg_{S'}(x') = Rg_S(w)$$

$\Rightarrow$ Behauptung folgt direkt. ~~mit~~

$$Rg_{S_m}(x') = Rg_{S}(w)$$

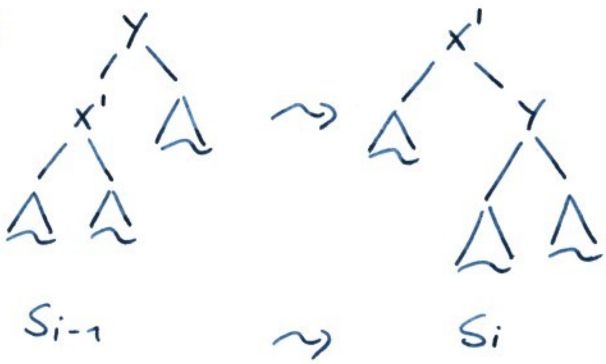
$$Rg_{S_0}(x') = Rg_S(x')$$

Anmerkung:

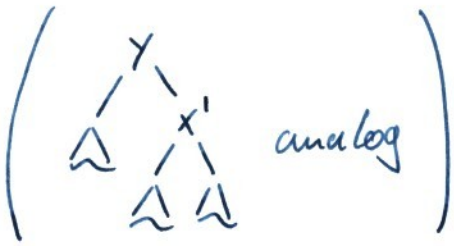
- # Blätter bleibt gleich
- # Blätter = # Nicht-Blätter + 1

Betrachten i-ten Aufruf von Splaying:

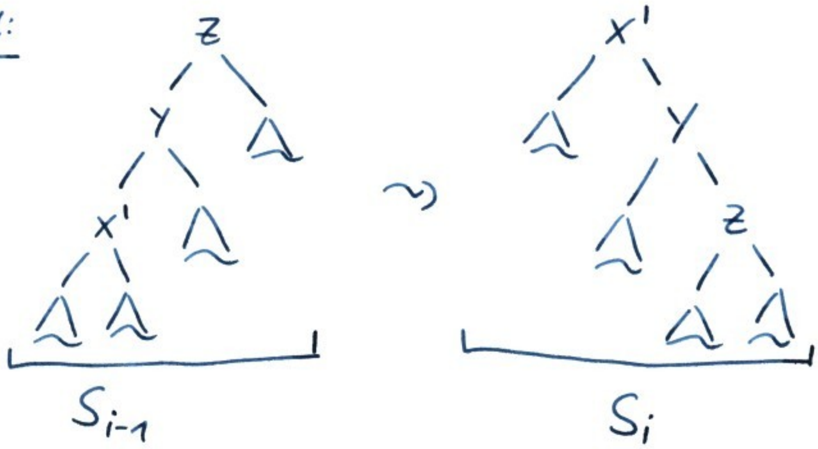
1. Fall:



bei $i=m$

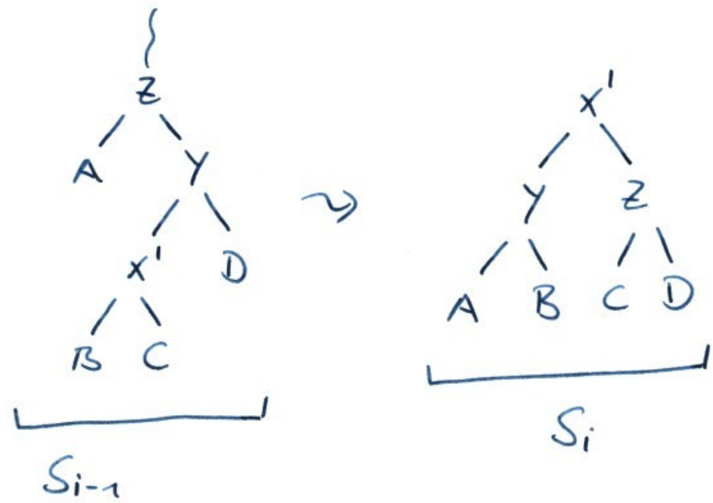


2. Fall:



29.5.17

3. Fall:



$$R_{x'} = Rg_{S_{i-1}}(x') \qquad \hat{R}_{x'} = Rg_{S_i}(x_i)$$

$$R_y = Rg_{S_{i-1}}(y) \qquad \hat{R}_y = Rg_{S_i}(y)$$

$$\underbrace{R_z = Rg_{S_{i-1}}(z)}_{\text{Ränge in alten Baum}} \qquad \underbrace{\hat{R}_z = Rg_{S_i}(z)}_{\text{Ränge in neuen Baum}}$$

zum 1. Fall: $a_i = a_m = 1 + \Phi(S_m) - \Phi(S_{m-1})$

$$= 1 + \hat{R}_{x'} + \hat{R}_y - R_{x'} - R_y$$

$$= 1 + \hat{R}_y - R_{x'}$$

(Ränge d. Teilbäume gehen weg)

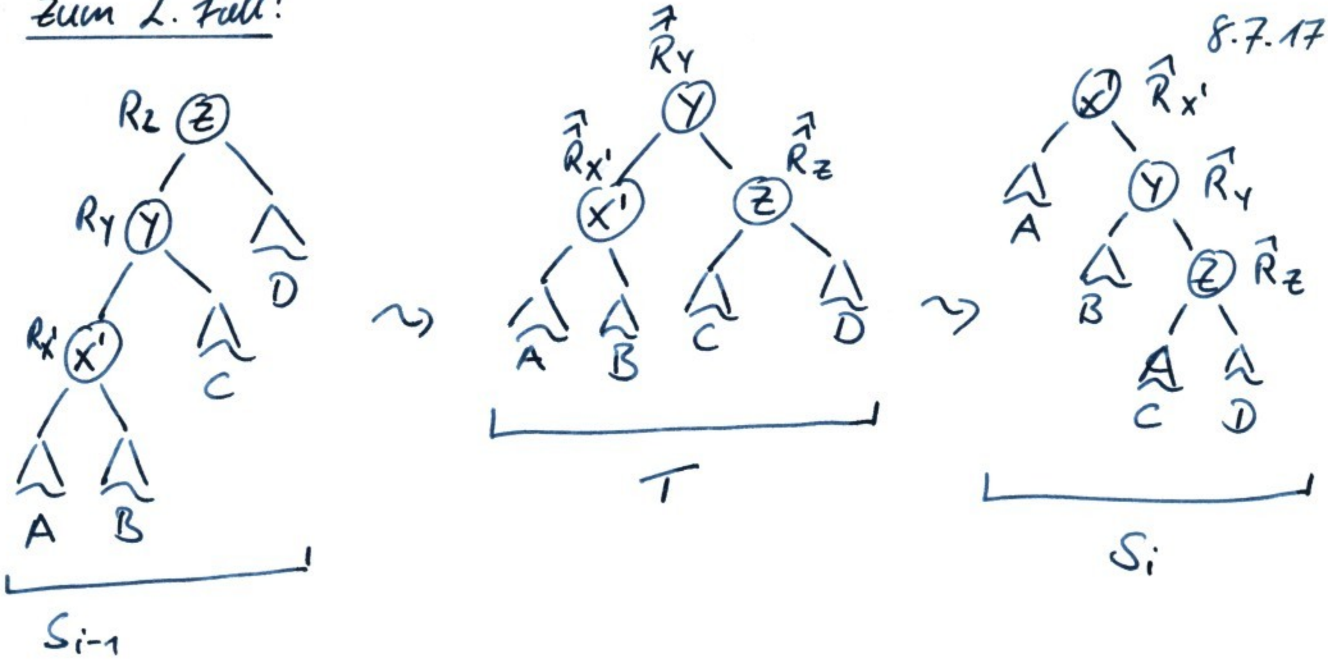
(da $\hat{R}_{x'} = R_y$ „Ränge d. Wurzeln gleich)

$$\le 1 + \underbrace{\hat{R}_{x'} - R_{x'}}_{\ge 0} \le 1 + 3(\hat{R}_{x'} - R_{x'})$$

$$(\hat{R}_{x'} = R_y \ge R_{x'})$$

8.7.17

zum 2. Fall:



Es gilt: $R_Z = \hat R_{X'}$, Also:

$$a_i = 1 + \phi(S_i) - \phi(S_{i-1})$$

$$= 1 + \hat R_Y + \hat R_Z - R_Y - R_{X'} + \underbrace{(\hat R_{X'} - \hat R_Z)}_{=0}$$

Fall 2a): $R_Z \gneq R_{X'}$

$\Downarrow$

$= \hat R_{X'}$

$= \vec R_Y$

$\vec R_Y = R_Z = \hat R_{X'}$

$\vec R_Z = \hat R_Z$

$\vec R_{X'} = R_{X'}$

$$a_i \leq 1 + \hat R_{X'} + \hat R_{X'} - R_{X'} - R_{X'}$$

$$= 1 + 2\underbrace{(\hat R_{X'} - R_{X'})}_{\geq 1} \leq 3(\hat R_{X'} - R_{X'})$$

wegen Annahme oben (2a).

8.7.17

<u>Fall 2b)</u>: $R_z = R_{x'}$. Also

$$R_{x'} = R_y = R_z = \hat{R}_{x'}$$

Es gilt: $\hat{R}_z \lneqq (R_{x'} = R_y = R_z = \hat{R}_{x'})$

Denn wäre $\hat{R}_z = R_{x'} = R_y = R_z = \hat{R}_{x'}$ dann wäre

$$\underbrace{R_z}_{= \hat{R}_y} \geq 1 + R_{x'}$$ (wegen Teilbäumen in Baum T)

Übung: geht das auch direkt, also ohne den Baum T?

$$a_i = 1 + \cancel{\hat{R}_{x'}} + \hat{R}_y + \hat{R}_z - \cancel{R_z} - R_y - R_{x'}$$

$$= 1 + \underbrace{(\hat{R}_y - \underbrace{R_y}_{= \hat{R}_{x'}})}_{\leq 0} + (\hat{R}_z - R_{x'})$$

$$\leq 1 + \underbrace{(\hat{R}_z - R_{x'})}_{\leq -1}$$

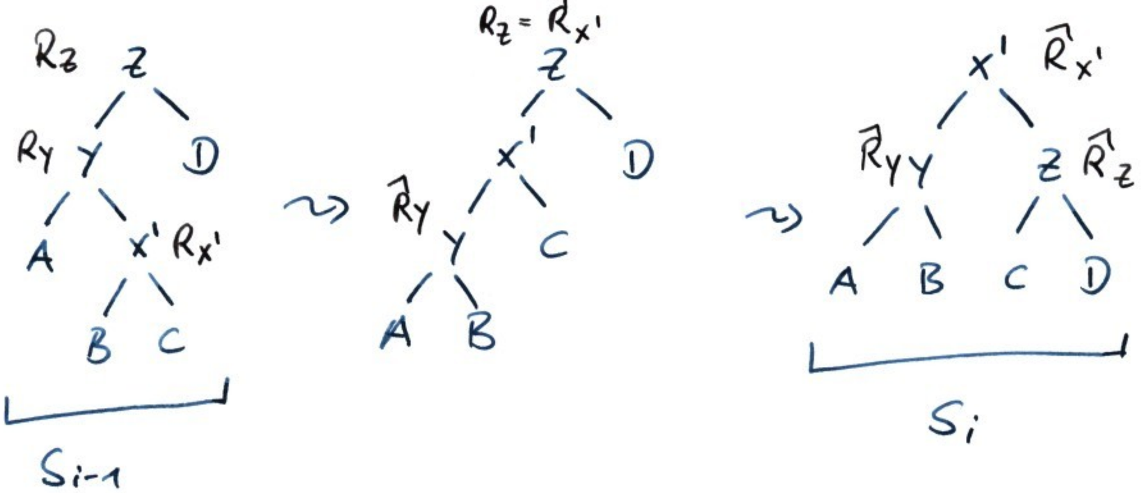
wegen $\hat{R}_z \lneqq R_{x'}$ wie gezeigt.

also

$$\underline{a_i \leq 0} \leq 3 \cdot \underbrace{(\hat{R}_{x'} - R_{x'})}_{= 0}$$

8.7.17

3. Fall:



Fall 3a): $\hat{R}_{x'} = R_z \gneq R_{x'}$

$$
\begin{aligned}
a_i &= 1 + \hat{R}_y + \hat{R}_z - R_y - R_{x'} \\
&\leq 1 + \hat{R}_{x'} + \hat{R}_{x'} - R_{x'} - R_{x'} \\
&= 1 + 2(\underbrace{R_z - R_{x'}}_{\geq 1}) \leq 3(R_z - R_{x'}) \\
&= 3(\hat{R}_{x'} - R_{x'})
\end{aligned}
$$

Fall 3b): $R_z = R_{x'} = R_y = \hat{R}_{x'}$

Dann gilt: $\hat{R}_y \lneq R_y$ oder $\hat{R}_z \lneq R_z$

Es gilt: $\hat{R}_y \leq R_y$ und auch $\hat{R}_z \leq R_z = \hat{R}_{x'}$

Sind $\hat{R}_y = R_y$ und $\hat{R}_z = R_z = \hat{R}_{x'}$. Dann ist $\hat{R}_y = \hat{R}_z = \hat{R}_{x'}$. Das kann nicht sein wegen Fall 3b.

8.7.17

$$a_i = 1 + \hat{R}_y + \hat{R}_z - R_y - R_{x'}$$

$$= 1 + \underbrace{(\hat{R}_y - R_y)}_{\leq 0} + \underbrace{(\hat{R}_z - \underbrace{R_{x'}}_{= R_z})}_{\leq 0} \quad \text{od.}$$

$$\Rightarrow \leq 0 \leq 3(\hat{R}_{x'} - R_{x'})$$

12.6.17

Folgerung: S mit n Knoten

$$a(\text{Finde}_S(x)) \leq O(\log n)$$

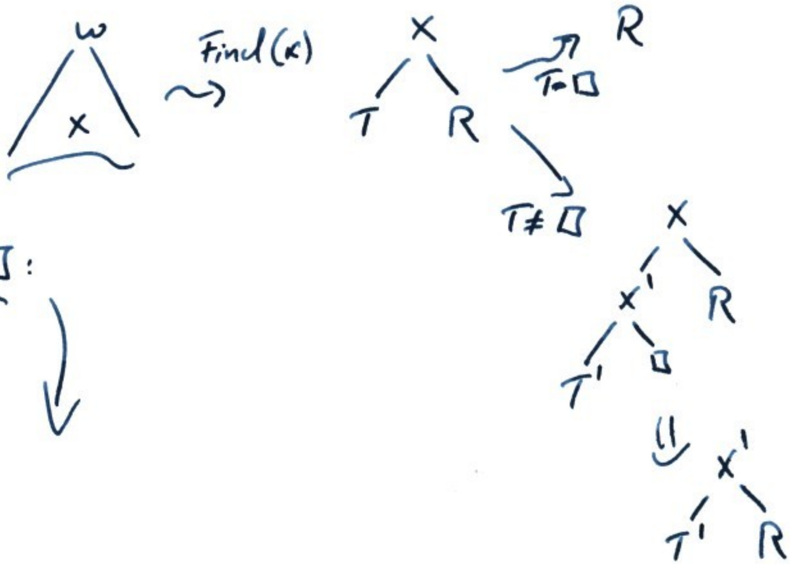
$$a(\text{Lösche}_S(x)) \leq O(\log n)$$

$$a(\text{Einfüge}_S(x)) \leq O(\log n)$$

Beweis: ① $a(\text{Find}_S(x)) \leq 1 + 3(R_S(w) - R_S(x'))$ ← eben gezeigt

$$= O(\log n)$$

Löschen:



Fall $T \neq \square$:

12.6.17

$$a(\text{Lösche}_S(x)) = \underbrace{a(\text{Find}_S(x))}_{\substack{= \#\text{Splayings} \\ + \text{Pot.-diff.}}} + \underbrace{a(\text{Find}_T(x)) + O(1)}_{?}$$

$$\leq 1 + 3\big(R_S(w) - R_S(x)\big) \qquad x = x'$$

$T =$ 

$$+ 1 + 3\big(R_T(v) - R_T(x')\big)$$

<u>Müssen zeigen:</u>

(2) $a(\text{Lösche}_S(x)) = \#\text{Splayings} + O(1) + \phi(S') - \phi(S)$

$$= O(\log n)$$

<u>nochmal neu:</u>

- $a(\text{Find}_S(x)) \leq 1 + 3\big(R_S(w) - R_S(x)\big)$

$$\downarrow$$

$$= \#\text{Splayings} + \Phi\left(\begin{smallmatrix} & \otimes & \\ T & & R \end{smallmatrix}\right) - \Phi(S)$$

- $a(\text{Find}_T(x)) = \#\text{Splayings} + \Phi\left(\begin{smallmatrix} & & \otimes & \\ & x' & & R \\ T' & & \text{II} & \end{smallmatrix}\right) - \Phi\left(\begin{smallmatrix} & \otimes & \\ T & & R \end{smallmatrix}\right)$

$$\leq 1 + 3\big(R_T(v) - R_T(x')\big)$$

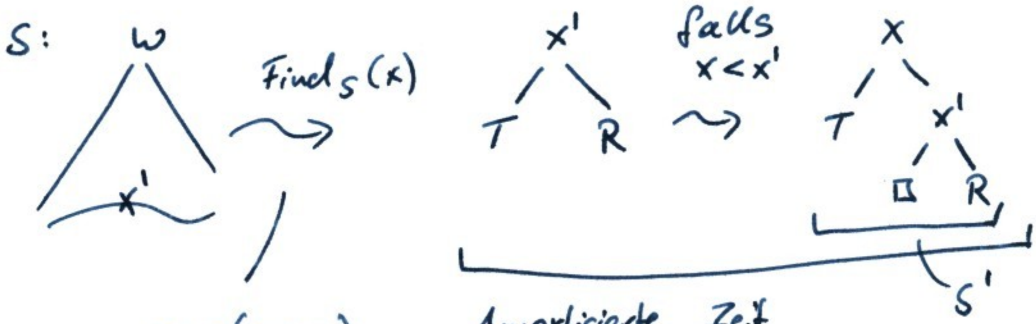
- ~~Letz~~ Letzter Schritt

$$a(\text{Direktes Löschen}) = 1 + \underbrace{\Phi\left(\begin{smallmatrix} & x' & \\ T' & & R \end{smallmatrix}\right) - \Phi\left(\begin{smallmatrix} & & \otimes & \\ & x' & & R \\ T' & & \square & \end{smallmatrix}\right)}_{\leq 0}$$

Also: $a(\text{Lösche}_S(x)) \leq 2 + 6 \cdot \log n$

$= O(\log n)$

③ $\text{Einfügen}_S(x)$



$\leq O(\log n)$ Amortisierte Zeit

$\leq 1 + R_{S'}(x) = O(\log n)$

Folgerung: m Operationen wobei

- Maximalzahl Elemente $\leq n$
- anfangs leerer Baum,

dann Zeit: $\underline{\underline{O(m \cdot \log n)}}$

Folgerung: · S Baum mit n Elementen.

- Nur Finde-Operationen, (Menge d. Elemente jedes x kommt vor bleibt gleich)
- σ = Folge der Finde-Operationen

S_0 ist weiterer Suchbaum mit denselben Elementen.

12.6.17

$\text{Kosten}_{S_\otimes}(\sigma)$ wie im Splaybaum

$= O(\text{Kosten}_S(\sigma) + h \cdot n)$

S unverändert ; h = Tiefe von $S_\otimes$

Dabei $\text{Kosten}_S(\sigma)$ S unverändert

= Summe der Tiefen+1 der Eelemente,
die gefunden werden sollen

<u>Sinn:</u> · S optimal für σ

Falls σ lang genug (etwa $\geq n^2$), dann

$\text{Kosten}_{S_\otimes}(\sigma)$ im Splaybaum
$= O(\text{Kosten}_S(\sigma);\ S \cdot \text{unverändert})$

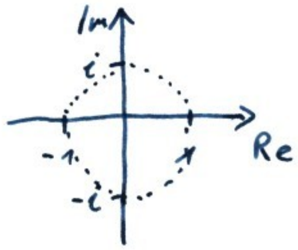
(ohne Beweis)

12.6.17

5. Diskrete Fourier Transformation

Komplexe Zahlen

$$z = \underbrace{x + iy}_{\in \mathbb{R}} \qquad i \cdot i = -1$$

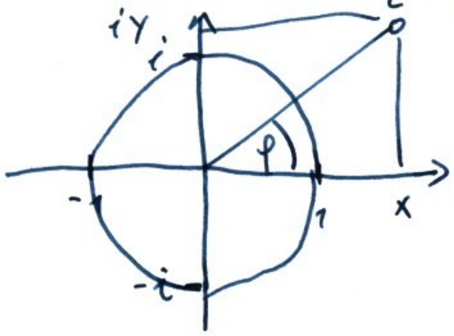


$$z \cdot z' = x \cdot x' + i(yx' + y'x) - yy'$$

$$z + z' = x + x' + i(y + y')$$

Polarkoordinaten

$z = x + iy$



$$|z| = \sqrt{x^2 + y^2}$$

$$z = \sqrt{x^2+y^2} \cdot (\cos\varphi + i \sin\varphi)$$

Eindeutigkeit: $0 \leq \varphi \lneq 2\pi$

φ im Bogenmaß

$$-1 = \cos(\pi)$$

$$i = \sin\left(\frac{\pi}{2}\right)$$

Nun gilt: $e^{i\varphi} = \cos(\varphi) + i\sin(\varphi) \quad , \varphi \in \mathbb{R}$

$$e^{i\varphi} \cdot e^{i\psi} = \cos(\varphi + \psi) + i \sin(\varphi + \psi)$$

$\boxed{\psi = \text{psi}}$

12.6.17

Satz von Taylor: $f: \mathbb{R} \to \mathbb{R}$

$$f(x) = f(a) + f'(a)\cdot(x-a) + \frac{f''(a)}{2}\cdot(x-a)^2 + \frac{f'''(a)}{3!}\cdot(x-a)^3 + \dots \qquad \left[\frac{f^{(n)}(a)}{n!}\cdot(x-a)^n\right]$$

[a fest
$f(x)$ gesucht,
Konvergenz

mit $a=0$ Anwendung auf $\sin(x)$

$$\sin(x) = \underbrace{0}_{\sin(0)} + \underbrace{1}_{\cos(0)}\cdot x + 0 - 1\cdot\frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \dots$$

$$= \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n+1}}{(2n+1)!} \qquad \text{für alle } x.$$

für $\cos(x)$

$$\cos(x) = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \frac{x^6}{6!} + - \dots$$

$$= \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n}}{(2n)!}$$

für e^x

$$\exp(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \dots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Analog: $x \in \mathbb{R}$ 12.6.17

$$e^{ix} = 1 + \frac{ix}{1} + \frac{(ix)^2}{2} + \frac{(ix)^3}{3!} + \dots$$

$$= \sum_{n=0}^{\infty} \frac{(ix)^n}{n!}$$

$$= \underline{1} + ix \underline{- \frac{x^2}{2}} - \frac{ix^3}{3} \underline{+ \frac{x^4}{4!}} + \frac{ix^5}{5!} \underline{- \frac{x^6}{6!}} - \frac{ix^7}{7!} + + - - \dots$$

$$\boxed{e^{ix} = \cos(x) + i \sin(x)} \quad \text{für alle } x \in \mathbb{R}.$$

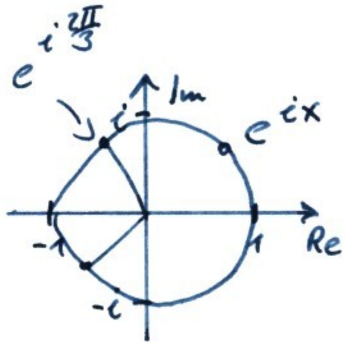
19.6.17

Def.: Einheitswurzel, z mit $z^n = 1$

$z \in \mathbb{Z}$: $-1; 1$ sind Einheitswurzeln

in $\mathbb{R}$: $z^n = 1 \iff z$ ist n-te Einheitswurzel wenn $n \in \mathbb{N}^{>0}$

in $\mathbb{C}$



$z = r \cdot e^{i\varphi}$, $r \neq 1$
$r \in \mathbb{R}^{\geq 0}$
$\varphi \in \mathbb{R}$
dann z keine Einheitswurzel.

$z^n = r^n \cdot e^{i\varphi \cdot n}$
$r^n \neq 1$

$n = 3$, dann die Einheitswurzeln

1, $e^{i\varphi}$ $\quad \varphi = \frac{2\pi}{3}$

$\varphi = \frac{4\pi}{3} = -\frac{2\pi}{3}$

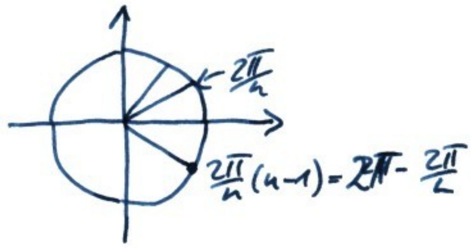
Das sind alle 3-ten Einheitswurzeln.
($x^3 - 1 = 0$ 3 Nullstellen über $\mathbb{C}$.)

19.6.17

Satz: Es gibt genau n n'te Einheitswurzeln.

$$1,\ e^{i\frac{2\pi}{n}},\ e^{i\frac{2\pi}{n}\cdot 2},\ \dots,\ e^{i\frac{2\pi}{n}(n-1)}$$

Beweis: $x^n - 1 = 0$ hat nur $\leq n$ Lösungen.



Def: $e^{i\frac{2\pi}{n}}$ heißt Primitive n-te Einheitswurzel.

Rechenregeln:

$$\left(e^{i\frac{2\pi}{n}\cdot k}\right)^m = e^{i\frac{2\pi}{n}\cdot k\cdot m} = e^{i\frac{2\pi}{n}\underbrace{(k\cdot m \bmod n)}_{\text{Rest von } k\cdot m \text{ Modulo } n}}$$

ist wieder Einheitswurzel

$$e^{i\cdot\frac{2\pi}{n}\cdot k}\cdot e^{i\frac{2\pi}{n}\cdot l} = e^{i\frac{2\pi}{n}(k+l)} = e^{i\frac{2\pi}{n}(k+l \bmod n)}$$

n-te Einheitswurzeln mit Multiplikation ist wie $\mathbb{Z}_n$ mit Addition. $e^{i\frac{2\pi}{n}\cdot k} \mapsto k \quad 0 \leq k \leq n-1$

Bsp: $n = 2 \quad 1 \in \mathbb{C} \mapsto 0$

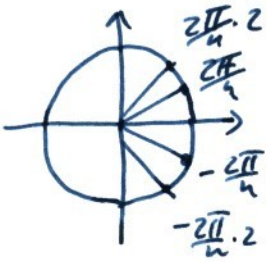
$-1 \in \mathbb{C} \mapsto 1$

Satz:

Summe der n-ten Einheitswurzeln ist 0.

$$\left(\sum_{k=0}^{n-1} e^{i\frac{2\pi}{n}k}\right) = 0$$

Beweis: Wieso heben sich die Imaginärteile weg?



$$\sin\varphi = -\sin(-\varphi)$$

$$\sum_{k=0}^{n-1} e^{i\frac{2\pi}{n}\cdot k} = \frac{e^{i\frac{2\pi}{n}\cdot n}-1}{\underbrace{e^{i\frac{2\pi}{n}}-1}_{\neq 0 \text{ für } n\geq 2}} = 0$$

Beachte: Auch für festes m

$$\sum_{k=0}^{n-1}\left(e^{i\frac{2\pi}{n}\cdot m}\right)^k$$

sofern m kein Vielfaches von n.

Satz: n gerade $\left(e^{i\frac{2\pi}{n}}\right)^{\frac{n}{2}} = -1$

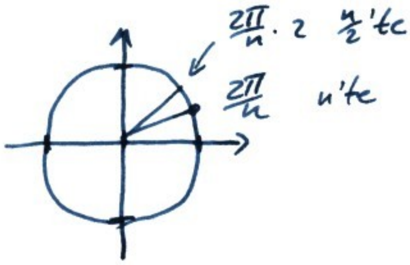
Satz: (Kürzungslemma)

$$\left(e^{\frac{i2\pi}{n}}\right)^k = \left(e^{i\frac{2\pi}{d\cdot n}}\right)^{d\cdot k} \qquad d\geq 1$$

Satz: n gerade, dann betrachten wir die $\frac{n}{2}$'ten Einheitswurzeln. Diese sind die Quadrate der n'ten Einheitswurzeln. (jede $\frac{n}{2}$'te kommt 2 mal vor.)

19.6.17

__Beweis__ $n=2: \quad -1, 1 \quad \rightsquigarrow \quad n=1: \quad 1$



$$\left(e^{i\frac{2\pi}{n}\cdot k}\right)^2 = e^{i\cdot\frac{2\pi}{n/2}\cdot k}$$

$$\cancel{\left(e^{i\frac{2\pi}{n}\cdot k}\right)^2}$$

$$\left(e^{i\frac{2\pi}{n}\left(k+\frac{n}{2}\right)}\right)^2 = e^{i\frac{2\pi}{n/2}\cdot k}$$

Diskrete Fouriertransformation

$$DFT_n : \mathbb{C}^n \longrightarrow \mathbb{C}^n$$

n-dim. Vektorraum.

Basis $\begin{pmatrix}1\\|\end{pmatrix}\begin{pmatrix}0\\1\\|\end{pmatrix} \dots \begin{pmatrix}0\\|\\1\end{pmatrix}$

$$\begin{array}{c} \\ 0 \\ 1 \\ \\ k \\ \\ n-1 \end{array}
\begin{array}{c}
\begin{array}{cccccccc} 0 & 1 & & & & l & & n-1 \end{array} \\
\left(\begin{array}{cccccccc}
1 & 1 & 1 & 1 & \dots & \dots & \dots & 1 \\
1 & & & & & | & & \\
1 & & & & & | & & \\
1 & — & — & — & — & e^{i\frac{2\pi}{n}\cdot k\cdot l} & & \\
\vdots & & & & & & & \\
1 & & & & & & &
\end{array}\right)
\end{array} \cdot \frac{1}{\sqrt{n}}$$

$\uparrow$

Bild von $\begin{pmatrix}1\\0\\|\\0\end{pmatrix}$

19.6.17

Fouriermatrix:

$n=2$: $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

$n=4$: $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & 1 \\ 1 & -i & -1 & i \end{pmatrix}$ $\left(e^{i \frac{2\pi}{n} \cdot k \cdot l} = e^{i \frac{2\pi}{n} \cdot l \cdot k} \right.$ Symmetrisch $\left.\right)$

⇝ Überlegung: n ~~zu~~ aus $\frac{n}{2}$ zusammensetzen.

- obere Hälfte Spalte $0, 2, \ldots, n-2$

  $\frac{n}{2}$ viele Spalten

  ist $DFT_{n/2}$

- untere Hälfte genauso

- Was ist mit dem Rest?

  → obere Hälfte, ungerade Spalten

  $\cancel{\left(e^{i\frac{2\pi}{n}}\right)} \cdot DFT_{n/2}$

  außer Zeile 0

  → untere Hälfte, ungerade Spalten

  $-1 \cdot \cancel{\left(e^{i\frac{2\pi}{n}}\right)} \cdot DFT_{n/2}$

  so mod.

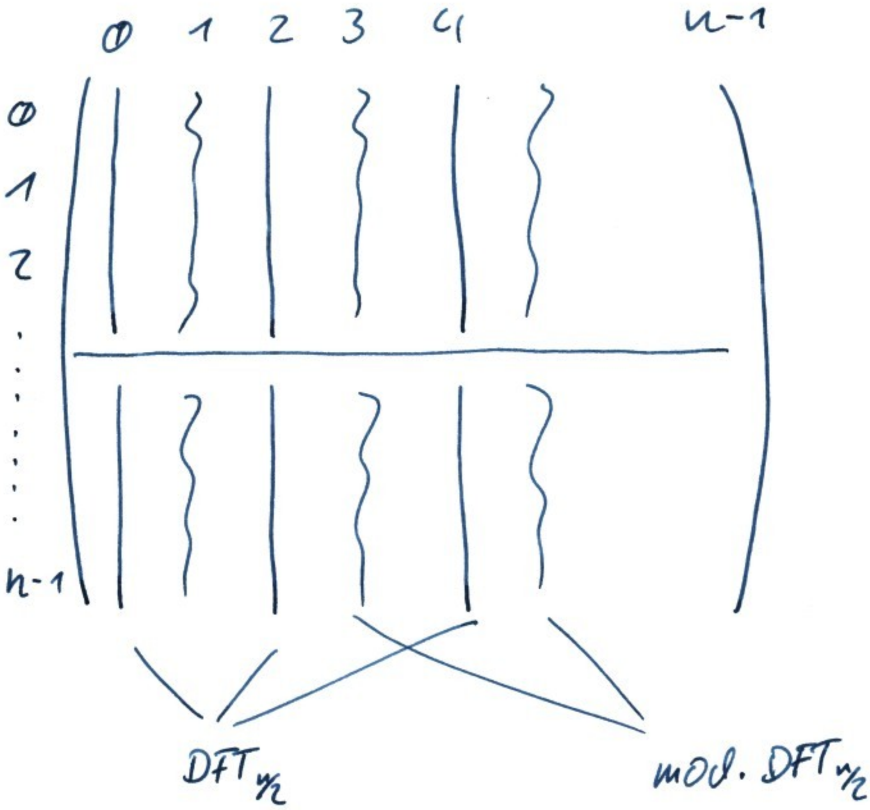
  $Z0: \cdot 1$

  $Z1: \cdot e^{i\frac{2\pi}{n}}$

  $Z2: \cdot e^{i\frac{2\pi}{n}\cdot 2}$

  $\vdots$

19.6.17



<u>Wollen berechnen:</u>

$$DFT_n \cdot \begin{pmatrix} z_0 \\ \vdots \\ z_{n-1} \end{pmatrix} \qquad \text{Zeit } O(n^2)$$

<u>Mit divide & conquer</u>

· Eintrag in <u>oberer</u> Hälfte des Ergebnisses.

Teilen auf $\begin{pmatrix} z_0 \\ z_2 \\ \vdots \\ z_{n-2} \end{pmatrix}$ und $\begin{pmatrix} z_1 \\ z_3 \\ \vdots \\ z_{n-1} \end{pmatrix}$

1. $DFT_{\frac{n}{2}} \cdot \begin{pmatrix} z_0 \\ | \\ z_{n-2} \end{pmatrix} = \begin{pmatrix} a_0 \\ | \\ a_{\frac{n}{2}-1} \end{pmatrix}$

2. $DFT_{\frac{n}{2}} \cdot \begin{pmatrix} z_1 \\ | \\ z_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 \\ | \\ b_{\frac{n}{2}-1} \end{pmatrix}$

19.6.17.

Obere Hälfte des Ergebnisses:

$$\begin{pmatrix} c_0 \\ | \\ c_{\frac{n}{2}-1} \end{pmatrix} \quad \text{durch} \quad c_k = a_k + e^{\frac{i 2\pi}{n} \cdot k} \cdot b_k$$

untere Hälfte:

$$\begin{pmatrix} d_0 \\ | \\ d_{\frac{n}{2}-1} \end{pmatrix} \quad d_k = a_k - e^{i \frac{2\pi}{n} k} \cdot b_k$$

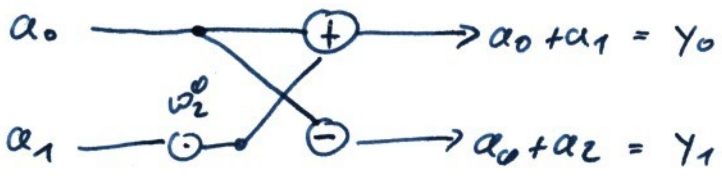
oder auch

$$\begin{pmatrix} c_{n/2} \\ | \\ c_{n-1} \end{pmatrix} \quad c_{\frac{n}{2}+k} = a_k - e^{i \frac{2\pi}{n} \cdot k} \cdot b_k$$

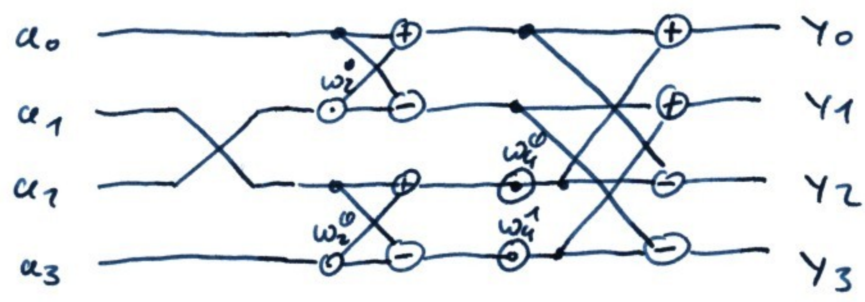
22.6.17

$$\omega_n = e^{\frac{2\pi i}{n}}$$

$$DFT_2 : \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} : \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \mapsto \begin{pmatrix} a_0 + a_1 \\ a_0 - a_1 \end{pmatrix}$$



DFT_4:



$$y_0 = a_0 + a_1 + a_2 + a_3$$

$$y_1 = a_0 - a_2 + i(a_1 - a_3)$$

$$y_2 = a_0 + a_2 - (a_1 + a_3)$$

$$y_3 = (a_0 - a_2) - i(a_1 - a_3)$$

$$DFT_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

22.6.17

<u>Invertierbar?</u>

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \cancel{\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & e^{\frac{2\pi i}{3}} & e^{\frac{2\pi i}{3}\cdot 2} \\ 1 & e^{\frac{2\pi i}{3}\cdot 2} & e^{\frac{2\pi i}{3}\cdot 4} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix}$$

$$\cdot \begin{pmatrix} 1 \\ e^{\frac{2\pi i}{3}\cdot -1} \\ e^{\frac{2\pi i}{3}\cdot -2} \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix}$$

$$DFT_n^{-1} = \begin{pmatrix} & & \\ & \exp\left(-\frac{2\pi i}{n} k \cdot l\right) & \\ & & \end{pmatrix}$$

$$DFT_n \cdot DFT_n^{-1} = \begin{array}{c c} l & \\ \begin{pmatrix} n & & 0 \\ & \ddots & \\ 0 & & n \end{pmatrix} & k \end{array}$$

22.6.17

Zeile k, Spalte l

$$\sum_{j=0}^{n-1} \exp\left(\frac{2\pi i}{n} k \cdot j\right) \cdot \exp\left(-\frac{2\pi i}{n} l \cdot j\right)$$

$$= \sum_{j=0}^{n-1} \exp\left(\frac{2\pi i}{n}(k-l) j\right)$$

$$= \begin{cases} 0 & (k-l) \bmod n \neq 0 \\ n & (k-l) \bmod n = 0 \end{cases}$$

$0 \leq k, l \leq n-1$

$-(n-1) \leq (k-l) \leq (n-1)$, also $k-l = 0 \bmod \Leftrightarrow k = l$

• Inverse Matrix ist Konjugierte Matrix!

22.6.17

Anwendung der DFT – Polynommultiplikation

$$A(x) = a_0 x^0 + a_1 x^1 + \dots + a_m x^m \qquad a_m \neq 0,\ a \in \mathbb{C}$$

$$B(x) = b_0 x^0 + b_1 x^1 + \dots + b_l x^l \qquad b_l \neq 0,\ b \in \mathbb{C}$$

$x_0 \in \mathbb{R}$

$$\underbrace{A(x_0)}_{\in \mathbb{R}} = a_0 + x_0(a_1 + x_0(a_2 + x_0(a_3 + x_0 a_4)))) \qquad O(m)$$

$A(x) + B(x)$ in $O(m+l)$

$$A(x) \cdot B(x) = a_0 b_0 + (a_0 b_1 + a_1 b_0) x^1$$

$$+ a_0 b_2 + a_1 b_1 + a_2 b_0) x^2$$

$$+ \dots$$

$$\dots$$

$$+ a_m \cdot b_l \cdot x^{m+l}$$

$$\sum_{j=0}^{k} (a_j b_{k-j}) x^k$$

$$A(x) \cdot B(x) = C(x) = c_0 + c_1 x_1 + \dots + c_{m+l} \cdot x^{m+l}$$

$$c_k = \sum_{j=0}^{k} a_j \cdot b_{k-j}$$

$$\begin{pmatrix} c_0 \\ \vdots \\ c_{m+l} \end{pmatrix} = \text{Faltung von } \begin{pmatrix} a_0 \\ \vdots \\ a_m \end{pmatrix}, \begin{pmatrix} b_0 \\ \vdots \\ b_l \end{pmatrix}$$

Mit DFT_n $\quad n \geq m, l \quad n = 2$'er Potenz

1. $$DFT_n \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_m \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ \\ \\ \vdots \\ y_{n-1} \end{pmatrix}$$ dann ist

$$y_0 = A(1) = A(x)_{|x=1}$$

$$y_1 = A(\omega_n) = A(x)_{|x=\omega_n}$$

$$\omega_n = \exp\left(\frac{2\pi i}{n}\right)$$

$$y_2 = A\left(\exp\left(\frac{2\pi i}{n} \cdot 2\right)\right)$$

$$DFT_n \cdot \begin{pmatrix} b_0 \\ \vdots \\ b_l \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} z_0 \\ \vdots \\ \\ \\ \vdots \\ z_{n-1} \end{pmatrix} \qquad z_k = B\left(\exp \frac{2\pi i}{n} \cdot k\right)$$

2. $$\begin{pmatrix} x_0 \\ \vdots \\ x_{n-1} \end{pmatrix}$$ mit $x_k = y_k \cdot z_k$

22.6.17

3. $DFT_n^{-1} \cdot \begin{pmatrix} x_0 \\ | \\ x_{n-1} \end{pmatrix} = \underbrace{\begin{pmatrix} u_0 \\ | \\ u_{n-1} \end{pmatrix}}$

Faltung von

$\begin{pmatrix} a_0 \\ | \\ a_{m-1} \\ 0 \\ | \\ 0 \end{pmatrix}, \begin{pmatrix} b_0 \\ | \\ b_l \\ 0 \\ | \\ 0 \end{pmatrix}$

$u_0 = x_0 + \dots + x_{n-1}$

$= \cancel{a_1 b_1 +}\; A(1) \cdot B(1) + A\left(e^{\frac{2\pi i}{n}}\right) \cdot B\left(e^{\frac{2\pi i}{n}}\right)$

$+ \dots + A\left(e^{\frac{2\pi i}{n} \cdot (n-1)}\right) B\left(e^{\frac{2\pi i}{n}(n-1)}\right)$

$\overset{?}{=} 0$

Koeffiziente von Produkt.

Es ist $DFT_n \cdot \begin{pmatrix} u_0 \\ | \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} x_0 \\ | \\ x_{n-1} \end{pmatrix}$ $\quad x_k = A(\omega_n^k) \cdot B(\omega_n^k)$

Polynom von Grad n ist durch n+1 Argument-Wert-Paare bestimmt.

26.6.17

Wdh. DFT liefert Konvolution (Faltung)

$$A(x) = a_0 + a_1 x + \dots + a_m x^m$$
$$B(x) = b_0 + b_1 x + \dots + b_l x^l$$

$$C(x) = A(x) \cdot B(x)$$
$$= a_0 b_0 + (a_0 b_1 + a_1 b_0) x + (a_0 b_2 + a_1 b_1 + a_2 b_0) x^2$$
$$+ \dots + \left(\sum_{j=0}^{k} a_j b_{k-j} \right) x^k + \dots + a_m b_l x^{m+l}$$

$n \geq m+l$

$$DFT_n \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_m \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} A(1) \\ A(\omega_n) \\ \vdots \\ A(\omega_n^m) \\ \vdots \\ A\left(e^{i \frac{2\pi}{n} \cdot (n-1)}\right) \end{pmatrix}$$

$$DFT_n \cdot \begin{pmatrix} b_0 \\ \vdots \\ b_l \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} B(1) \\ B(\omega_n) \\ B(\omega_n^2) \\ \vdots \\ B(\omega_n^{n-1}) \end{pmatrix}$$

Koordinatenweise Multiplikation:

$$\begin{pmatrix} M(0) \\ M(1) \\ \vdots \\ M(n-1) \end{pmatrix} = \begin{pmatrix} A(1) \cdot B(1) \\ A(\omega_n) \cdot B(\omega_n) \\ \vdots \\ A(\omega_n^{n-1}) \cdot B(\omega_n^{n-1}) \end{pmatrix}$$

26.6.17

$$DFT_n^{-1} \circ \begin{pmatrix} M(0) \\ M(1) \\ | \\ M(n-1) \end{pmatrix} = \begin{pmatrix} C(0) \\ C(1) \\ | \\ C(n-1) \end{pmatrix}$$

Es gilt: $C(k) = \sum_{j=0}^{k} a_j b_{k-j} = \sum_{j=0}^{k} a_{k-j} \cdot b_j$

das ist zu zeigen. Rechne das anhand DFT_n^{-1} aus.

$$C_0 = \underbrace{A(1) \cdot B(1)}_{a_0 b_0} + \overbrace{A\left(e^{\frac{2\pi i}{n}}\right) \cdot B\left(e^{\frac{2\pi i}{n}}\right)}^{a_0 b_0 \text{ dabei}} + \dots + A\left(e^{\frac{2\pi i}{n} \cdot (n-1)}\right) \cdot B\left(e^{\frac{2\pi i}{n} (n-1)}\right)$$

$$= n \cdot a_0 b_0 + a_1 b_1 + \underbrace{a_1 b_1 e^{\frac{2\pi i}{n} \cdot 2} + a_1 b_1 e^{\frac{2\pi i}{n} \cdot 4} + \dots + a_1 b_1 e^{\frac{2\pi i}{n} (\cancel{n+}(n-1)+(n-1))}}_{a_1 b_1 \underbrace{\left(\sum_{j=0}^{n-1} e^{\frac{2\pi i}{n} \cdot 2 \cdot j} \right)}_{=0}}$$

$= 0$ für 2 teilt nicht n, da $n \geq k+l$

$$= \underline{n \cdot a_0 b_0}$$

26.6.17

$$C(k) = \sum_{l=0}^{n-1} \left(\exp\left(-\frac{2\pi i}{n} \cdot k \cdot l\right) \cdot A\left(\exp\left(\frac{2\pi i}{n} \cdot l\right)\right) \cdot B\left(\exp\left(\frac{2\pi i}{n} \cdot l\right)\right) \right)$$

- Koeffizient von $a_j b_{k-j}$, $0 \leq j \leq k$ (aus Konvolution)

j fest

$$a_j \cdot b_{k-j} \cdot \underbrace{\exp\left(\frac{2\pi i}{n} \cdot l \cdot j\right)}_{\text{Aus } A()} \cdot \underbrace{\exp\left(\frac{2\pi i}{n} l \cdot (k-j)\right)}_{\text{Aus } B()} \cdot \underbrace{\exp\left(-\frac{2\pi i}{n} \cdot k \cdot l\right)}_{\text{aus der Summe}}$$

$$\sum_{l=0}^{n-1} a_j \cdot b_{k-j} = n \cdot a_j b_{k-j}$$

- für $a_{j'} \cdot b_{k-j}$ $\quad j' \neq j$ $\quad \Rightarrow$ fallen alle weg!

$$\sum_{l=0}^{n-1} \exp\left(-\frac{2\pi i}{n} \cdot k \cdot l\right) \cdot \exp\left(\frac{2\pi i}{n} \cdot l \cdot j'\right) \cdot \exp\left(\frac{2\pi i}{n} \cdot l \cdot (k-j)\right)$$

$$= \sum_{l=0}^{n-1} \exp\left(\frac{2\pi i}{n} l \cdot \underbrace{(j'-j)}_{\neq 0}\right)$$

$$= 0$$

26.6.17

Was bedeutet

$$DFT^{-1} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

Bzgl. welcher Basis ist das die Darst. von $(a_0, \dots, a_{n-1})^T$?

$$a_0 \cdot \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \; a_1 \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \dots, \; a_{n-1} \cdot \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

$$(y_0 \cdot B_0 + y_1 \cdot B_1 + \dots + y_{n-1} \cdot B_{n-1}) = n \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

$$\text{Basis} \quad B_j = \underbrace{\begin{pmatrix} 1 \\ \exp\left(\frac{2\pi i}{n} \cdot j\right) \\ \exp\left(\frac{2\pi i}{n} \cdot 2 \cdot j\right) \\ \vdots \\ \exp\left(\frac{2\pi i}{n} \cdot (n-1) j\right) \end{pmatrix}}_{j\text{'te Spalte der } DFT_n.} \cdot \frac{1}{n}$$

$$DFT_n \left(DFT_n^{-1} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix} \right) = n \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

$$\begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \quad DFT_n^{-1} \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} n \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Darstellung in Fourier Basis.

6. Sortieren im Mittel

~~1.~~ Nur mit Vergleichen: Eingabe $(a_1, \dots, a_n)$.
Sortieren auf der Basis von
$a_i \lneq a_j$, $a_i \leq a_j$.

[$a_i = 0$ nicht erlaubt

- Worst case $\Omega(n \cdot \log n)$

  # Vergleiche gezählt.

~~Satz:~~ Wahrscheinlichkeitsraum: $Prob[(a_1, \dots, a_n)] = \frac{1}{n!}$,
Annahme alle a_i verschieden.

$$\Omega = \{(a_1, \dots, a_n) \mid (a_1, \dots, a_n) \text{ Permutation}\}$$

- Sortieralgorithmus ist fest.

$$X : \Omega \to \mathbb{N}$$

$$X((a_1, \dots, a_n)) = \#\text{Vergleiche}$$

Erwartungswert: $$E[X] = \sum_{\Omega} X(a_1, \dots, a_n) \cdot \frac{1}{n!}$$

$$= \frac{1}{n!} \cdot (\text{Summe aller Laufzeiten})$$

$E[X] \leq$ Worst-case-Laufzeit

best-case-Laufzeit $\leq E[X]$

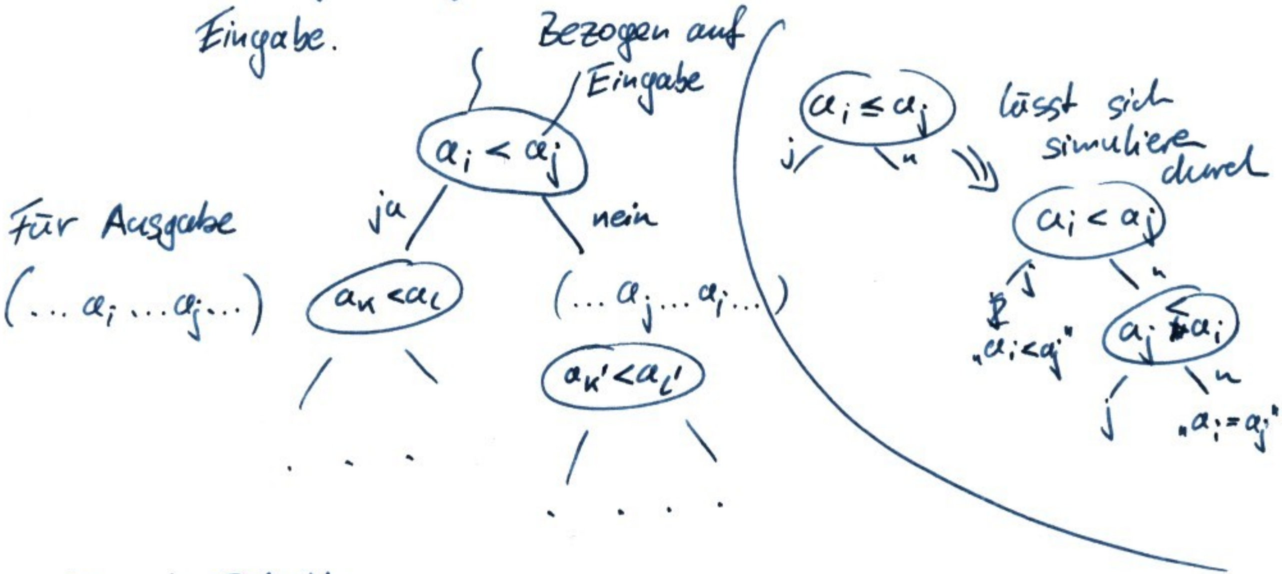
$$\Omega(n) \leq E[X] \leq \Omega(n \cdot \log n)$$

26.6.17

Satz: $E[X] = \Omega(n \cdot \log n)$

(Im Rahmen wie oben definiert.)

Beweis: Beliebiger Algorithmus hat $(a_1, \dots, a_n)$ als Eingabe.

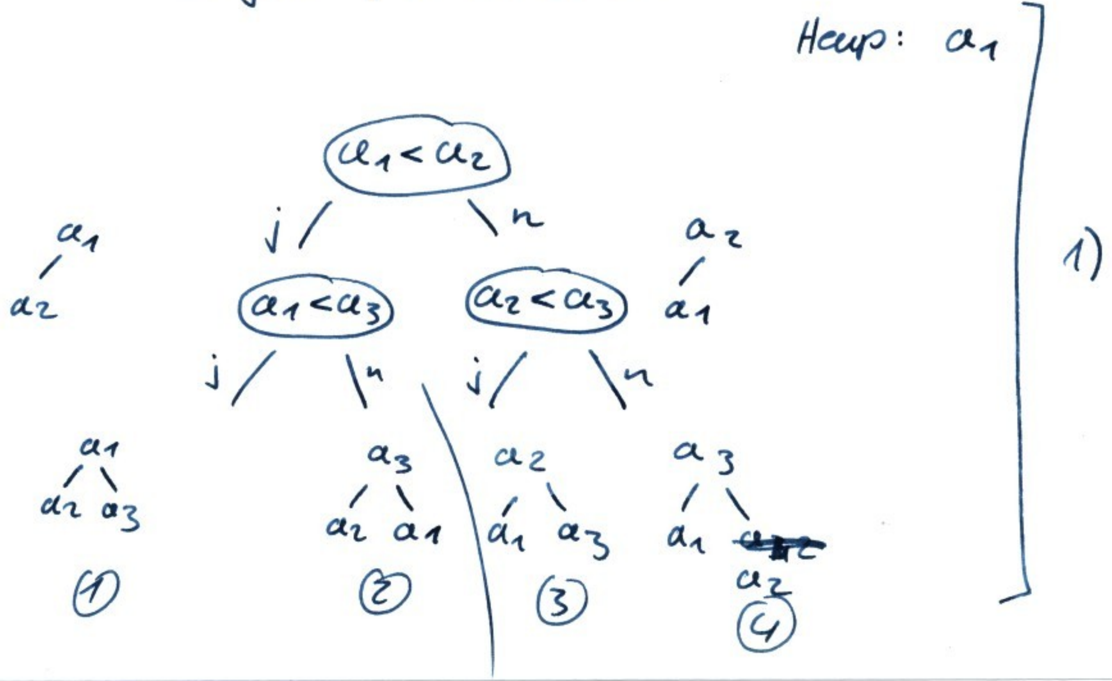


$\geq n!$ Blätter.

Bsp. $n = 3$, Heapsort

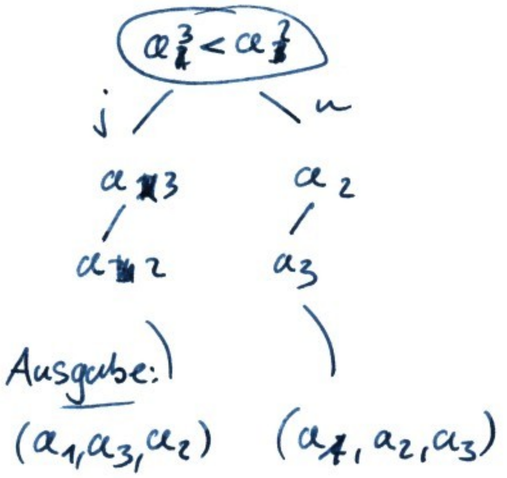
- Eingabe (a_1, a_2, a_3)
- 1.) a_1, a_2, a_3 in den Heap
- 2.) Elemente raustun.

Eingabe (a_1, a_2, a_3) bleibt



1

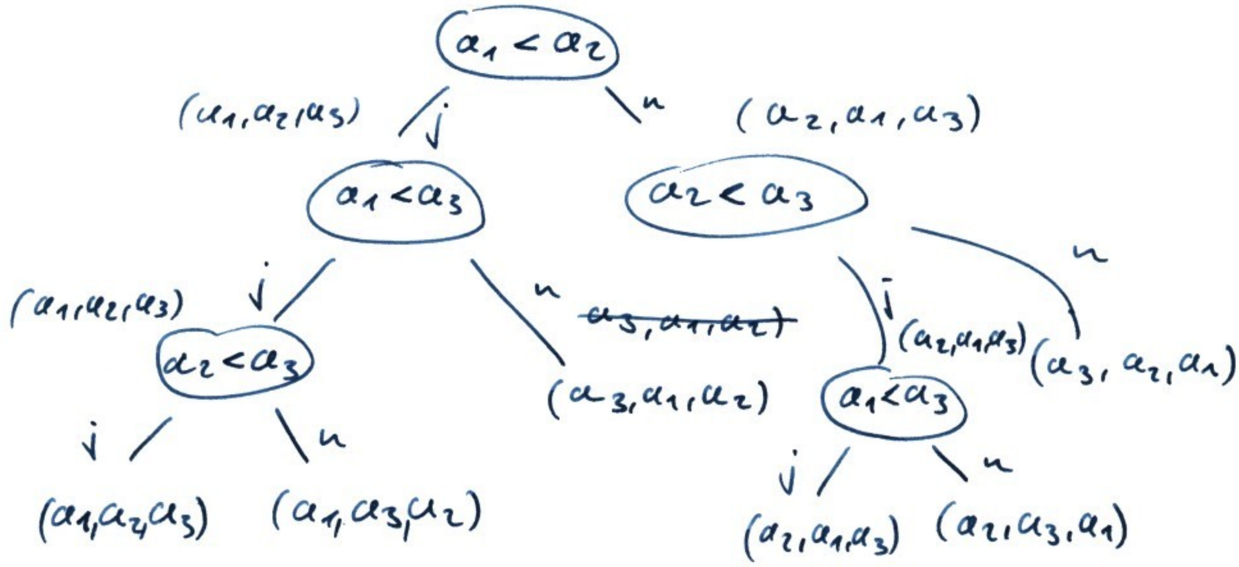
Ausgabe $(a_1, \ldots)$



3.7.17

<u>Mergesort</u>

(a_1, a_2, a_3) Eingabe



(jede Permutation nur 1x,
Keine überflüssigen Vergleiche!)

3.7.17

Worst-case-Laufzeit (Sortieren mit Vergleichen)

- Es gibt Weg (der vorkommt) der $\Omega(n \cdot \log n)$ lang ist.

- Entscheidungsbaum hat (mind.) $n!$ Blätter, die erreicht werden.
- Tiefe t, dann $\leq 2^t$ Blätter.

$$2^t \geq n! \quad , \quad t \geq \log_2 (n!)$$

führt zu $\Omega(n \cdot \log n)$

$$n! \geq \underbrace{\frac{n}{2} \cdot \ldots \cdot \frac{n}{2}}_{n/2 \text{ Fakt.}} = \left(\frac{n}{2}\right)^{n/2}$$

~~$\log(n!) =$~~

$$\log\left(\left(\frac{n}{2}\right)^{n/2}\right) = \frac{n}{2}\left((\log n) - 1\right) = \Omega(n \cdot \log n)$$

Erwartungswert

$X(a_1, \ldots, a_n) :=$ Tiefe des Blattes, zu dem $(a_1, \ldots, a_n)$ führt.

Zufallsvariable

$$E[X] = \underbrace{\sum_{(a_1, \ldots, a_n)} X(a_1, \ldots, a_n) \cdot \frac{1}{n!}}_{n! \text{ Summanden.}}$$

Frage: Was ist $X(a_1, \ldots, a_n)$???

3.7.17

auch

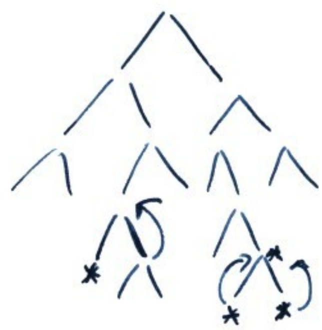
$$E[X] = \underbrace{\sum_{\substack{b \text{ Blatt} \\ \text{im Baum} \\ \underline{\text{und}}\ b \text{ kommt} \\ \text{vor.}}} \text{Tiefe}(b) \cdot \frac{1}{n!}}_{n! \text{ Summanden}}$$

Das jetzt nach unten wegschätzen.

Ziel: Verkleinerung der Summe

$$\sum_{b,\dots} \text{Tiefe}(b)$$

durch Transformation des Baumes, wo tote Blätter gelöscht sind.



* Tote Blätter

$$E[X] \geq \sum_{\substack{b \\ \text{im Baum,} \\ \text{wo jedes} \\ \text{Blatt vorkommt}}} \text{Tiefe}(b) \cdot \frac{1}{n!}$$

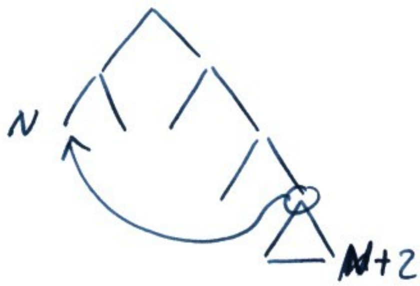
<u>Transformation:</u> Baum so, dass Blätter nur noch in 2 Tiefen vorkommen.

$N,\ N+1$

$$2^t \leq n! \leq 2^{t+1}$$

$N = t$

$\Rightarrow$ fangen in tiefster Tiefe an. Falls tiefste Tiefe $\geq N+2$, dann haben wir Blatt in Tiefe $\leq N$ (durch abzählen, wenn alle anderen in Tiefe $(N+1)$ sind, haben wir zu viele Blätter!)



Summe der Blatttiefen wird nicht größer.

$\cancel{\leq (N+1) - 2(N+2) + 2(N+1) - N}$

Sei K Tiefe der beiden Blätter $K \geq N+2$, H Tiefe des Blattes, wo die hinkommen.

~~Ä~~ Änderung: $-2K + (K-1) - H + 2(H+1)$

$$= -K - 1 + H + 1$$

$$= H - K$$

$$\leq 0$$

funktioniert solange es Blätter in Tiefe $\geq N+2$ gibt!

3.7.17

also: $E[X] \geq \sum_{\substack{b \\ \text{Blatt, im transformierte} \\ \text{Baum}}} \lfloor \log(n!) \rfloor \cdot \frac{1}{n!}$

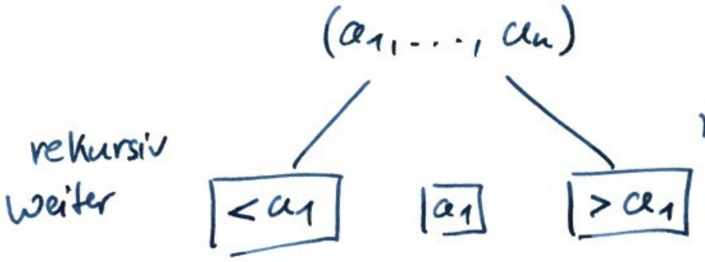
$= \lfloor \log(n!) \rfloor$

QuickSort

Satz: Erwartungswert der #Vergleiche von Quicksort ist $O(n \cdot \log n)$.

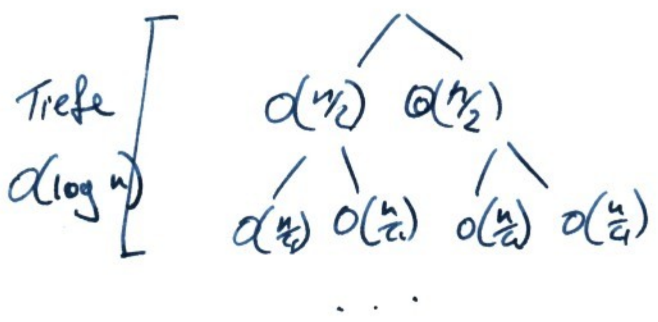
(Damit im ~~m~~ Mittel optimal)

einfachste Variante: Pivotelement ist a_1, d.h. das ~~#~~ erste in zu Sortierenden.

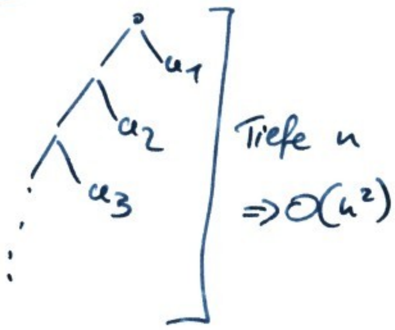


$\Rightarrow n$ Vergleiche

best-case



worst-case



3.7.17

Beweis: $X_n(a_1, \dots, a_n)$ = # Vergleiche bei $(a_1, \dots, a_n)$

$n \to$ ~~La~~ Länge der Folge

$$E[X_n] = \sum_{(a_1,\dots,a_n)} X_n(a_1, \dots, a_n) \cdot \frac{1}{n!}$$

$$X_n(a_1,\dots,a_n) = \underbrace{(n+1)}_{\text{(Vergleiche vor Rekursion)}} + \text{Rest, der von } a_1 \text{ abhängt.}$$

Aufteilen anhand von a_1, dann ist $E[X]$

$$E[X_n] = \cancel{E}\Bigg(\sum_{\substack{(a_1,\dots,a_n)\\ a_1=1}} X_n(a_1,\dots,a_n)$$

$$+ \sum_{\substack{(a_1,\dots,a_n)\\ a_1=2}} X_n(a_1,\dots,a_n)$$

$$+ \dots$$

$$+ \sum_{\substack{(a_1,\dots,a_n)\\ a_1=n}} X_n(a_1,\dots,a_n)\Bigg) \cdot \frac{1}{n!}$$

---

einzelne Summen

$$\sum_{\substack{(a_1,\dots,a_n)\\ a_1=1}} X_n(a_1,\dots,a_n) = \sum_{(a_2,\dots,a_n)} X_{n-1}(a_2,\dots,a_n) + (n-1)$$

$$\sum_{\substack{(a_1,\dots,a_n)\\ a_1=n}} X_n(a_1,\dots,a_n) = \sum_{(a_2,\dots,a_n)} X_{n-1}(a_2,\dots,a_n) + (n-1)$$

$$\sum_{\substack{(a_1,\dots,a_n)\\ a_1=2}} X_n(a_1,\dots,a_n) = \sum_{\substack{(a_1,\dots,a_n)\\ a_1=2}} \Big(X_{n-2}\big(\overset{b_1,\dots,b_{n-2}}{\cancel{a_2,\dots,a_n \text{ ohne } a_i=1}}\big) + (n-1) \Big)$$

$(b_1,\dots,b_{n-2})$ ist Teil >2 von $(a_1,\dots,a_n)$

$$\sum_{\substack{(a_1,\dots,a_n)\\ a_1=n-1}} X_n(a_1,\dots,a_n) = \sum_{\substack{(a_1,\dots,a_n)\\ a_1=n-1}} \Big(X_{n-2}\big(\overset{b_1,\dots,b_{n-2}}{\cancel{a_2,\dots,a_{n+1} \text{ ohne } a_i=n}}\big) + (n-1) \Big)$$

$(b_1,\dots,b_{n-2})$ ist Teil <2

Wie, wenn es zweimal in die Rekursion geht?

<u>Annahme</u>: $a_1 = k \qquad 3 \le k \le n-2$

$\hookrightarrow$ k'tes Element der sortierten Folge. (genauer)

$$X_n(a_1,\dots,a_n) = X_{k-1}(<k \text{ in alter Reihung}) + X_{n-k}(>k \text{ in alter Reihung}) + (n-1)$$

$(k, a_2, \dots, a_n)$

$\boxed{<k}$ $\boxed{k}$ $\boxed{>k}$

in alter Reihung.

3.7.17

$$\sum_{\substack{(a_1,\dots,a_n)\\ a_1=k}} X_n(a_1,\dots,a_n) =$$

$$\sum_{\substack{a_1=k,(a_1,\dots,a_n),(b_1,\dots,b_{k-1})\\ b_j<k}} \left(X_{k-1}(b_1,\dots,b_{k-1}) + \cancel{(k-1)}\right)$$

ist Anordnung d. El. $< k$

$$+ \sum_{\substack{a_1=k\,(a_1,\dots,a_n),\ (c_1,\dots,c_{n-k})\\ c_j>k}} \left(X_{n-k}(c_1,\dots,c_{n-k}) + \right.$$

ist Anordnung d. El. $> k$

$$\left. + \cancel{(n-1)}\right)$$

$$\sum_{\substack{(a_1,\dots,a_n)\\ a_1=k\\ (b_1,\dots,b_{k-1}),\\ (c_1,\dots,c_{n-k})}} \left(X_{k-1}(b_1,\dots,b_{k-1}) + X_{n-k}(c_1,\dots,c_{n-k}) + (n-1)\right)$$

- $(b_1,\dots,b_{k-1})$ ist Teil von $(a_1,\dots,a_n)$ $<k$
- $(c_1,\dots c_{n-k})$ ist Teil von $(a_1,\dots,a_n)$ $>k$

6.7.17

Erwartungswert Laufzeit Quicksort

Zufallsvariable $X_n(a_1,\ldots,a_n) \Leftrightarrow$ Laufzeit auf Folge mit n Elementen
($\Rightarrow$ alle Vergleiche d. Elemente)

$$E[X_n] = O(n \cdot \log n)$$

$$E[X_n] = \frac{1}{n!} \sum_{(a_1,\ldots,a_n)} X_n(a_1,\ldots,a_n)$$

$n!$ Summanden

$X_n(a_1,\ldots,a_n) = (n-1) + \ldots$ hängt von a_1 ab! Fallunterscheidung.

$$= X_{n-1}(\underbrace{a_2,\ldots,a_n}_{n-1 \text{ Elem.}}) + n-1 \quad \text{wenn } a_1 = 1$$

tritt in der ganzen Summe $(n-1)!$ mal auf! alle Perm. von $(a_2,\ldots,a_n)$

analog wenn $a_1 = n$

$X_n(a_1, a_2, \ldots, a_n)$, wenn $a_1 = 2$

$$= X_{n-2}(\underbrace{a_2,\ldots,a_n \text{ ohne } 1}_{n-2 \text{ Elemente}}) + n-1$$

(analog für $a_1 = n-1$)

Jede Permutation von $3,\ldots,n$ tritt genau $n-1$ mal auf!, wenn wir über alle ~~(in der gesamten Summe)~~ $a_2,\ldots,a_n$ gehen.

6.7.17

$X_n(a_1, a_2, \ldots, a_n)$ wenn $a_1 = 3$

$= X_2$ (Elemente 1,2 aus $a_2, \ldots, a_n$)

$+ X_{n-3}$ ($a_2, \ldots, a_n$ ohne 1,2) $+ n-1$

$X_n(a_1, \ldots, a_n) =$ X_{k-1} (El. $1, \ldots, k-1$ aus $a_2, \ldots, a_n$)

$a_1 = k$

$+ X_{n-k}$ ($a_2, \ldots, a_n$ ohne $1, \ldots, k-1$)

$+ n-1$

tritt genau $\binom{n-1}{2} \cdot (n-3)!$ mal auf, wenn wir über alle $a_2, \ldots, a_n$ gehen.

tritt $\binom{n-1}{n-3} \cdot 2!$ mal auf.

Möglichkeite für festes $(b_1, \ldots, b_{k-1})$

$\binom{n-1}{k-1} \cdot (n-k)!$

f. $(b_1, \ldots, b_{n-k})$

$\binom{n-1}{n-k} \cdot (k-1)!$

$$\binom{n-1}{k-1} \cdot (n-k)!$$

$$= \frac{(n-1)!}{(k-1)!\,(n-1-(k-1))!} \cdot (n-k)!$$

$$= \frac{(n-1)!}{(k-1)!}$$

$$\binom{n-1}{\cancel{k-1}\ \cancel{k-1}\ n-k} \cdot (\cancel{n-k}\ k-1)! = \cancel{\frac{(n-1)!}{(k-1)!}}\ \frac{(n-1)!}{(n-k)!}$$

6.7.17

$$\sum_{(a_1,\dots,a_n)} \left(X_n(a_1,\dots,a_n) \cdot \frac{1}{n!} \right)$$

$$= \frac{1}{n!} \cdot \Bigg(\sum_{\substack{(a_2,\dots,a_n)\\ \text{ohne } 1}} \big(X_{n-1}(a_2,\dots,a_n) \big) + \sum_{\substack{(a_2,\dots,a_n)\\ \text{ohne } 2}} X_{n-2}(\text{„}a_2,\dots,a_n \text{ ohne } 1\text{"})$$

$$\vdots$$

$$+ \boxed{\sum_{\substack{(a_2,\dots,a_n)\\ \text{ohne } k}} X_{k-1}(\dots)} + \boxed{\sum_{\substack{(a_2,\dots,a_n)\\ \text{ohne } k}} X_{n-k}(\dots)}$$

$$+ \dots$$

$$+ \sum_{\substack{(a_2,\dots,a_n)\\ \text{ohne } n}} X_{n-1}(a_2,\dots,a_n \text{ ohne } n) \Bigg) + \underbrace{\frac{1}{n!} \cdot n! \cdot (n-1)}_{n-1}$$

erster Summand:

$$= \frac{1}{n} \cdot \frac{1}{(n-1)!} \sum_{\substack{(a_2,\dots,a_n)\\ \text{ohne } 1}} X_{n-1}(a_2,\dots,a_n)$$

$$= \frac{1}{n} E[X_{n-1}]$$

$$\frac{1}{n!} \sum_{(b_1,\dots,b_{k-1})} X_{k-1}(b_1,\dots b_{k-1}) \cdot \frac{(n-1)!}{(k-1)!}$$

$$= \frac{1}{n} E[X_{k-1}]$$

analog

$$= \frac{1}{n} E[X_{n-k}]$$

6.7.17

<u>alles:</u>

$$\frac{1}{n}\Big(E[X_{n-1}] + E[X_{n-2}] +$$

$$+ E[X_2] + E[X_{n-3}]$$

$$+ E[X_3] + E[X_{n-4}]$$

$$+ \ \ldots$$

$$+ E[X_{n-3}] + E[X_2] \qquad \leftarrow \text{Pivot } n-2$$

$$+ E[X_{n-2}] + E[X_{n-1}]$$

$$= \frac{2}{n} \sum_{k=2}^{n-1} E[X_k] + (n-1) \qquad E[X_2] = 1$$

$$* \ E[X_n] = \qquad n \geq 3$$

6.7.17

1. Versuch: Einsetzen

$$E[X_n] = \frac{2}{n}\Big(E[X_{n-1}] + \quad \dots \Big) + n - 1$$

$$\frac{2}{n-1}\Big(E[X_{n-2}] + \dots \Big) + n - 2$$

2. Versuch:

für $n \geq 2$:

$$E[X_{n+1}] = \frac{2}{n+1} \cdot \Big(E[X_2] + \dots + E[X_n] \Big) + n$$

$$E[X_n] = \frac{2}{n} \cdot \Big(E[X_2] + \dots + E[X_{n-1}] \Big) + n - 1$$

$$\Rightarrow \quad (n+1)\, E[X_{n+1}] = 2 \Big(E[X_2] + \dots + E[X_n] \Big) + n^2 + n$$

$$n\, E[X_n] = 2 \Big(E[X_2] + \dots + E[X_{n-1}] \Big) + n^2 - n$$

also:

$$(n+1)\, E[X_{n+1}] - n\, E[X_n] = 2 E[X_n] + 2n$$

$$(n+1)\, E[X_{n+1}] = (n+2)\, E[X_n] + 2n$$

dann:

$$\boxed{\begin{array}{c} E[X_{n+1}] = \dfrac{n+2}{n+1} E[X_n] + \dfrac{2n}{n+1} \\ n \geq 2 \qquad E[X_2] = 1 \end{array}}$$

$$\frac{n}{n+1} = \left(1 - \frac{1}{n+1}\right)$$

Es wird höchstens größer, wenn wir das $-\frac{1}{n+1}$ weglassen. 6.7.17

mod. E-Wert:

$$\boxed{\begin{aligned} E[X_{n+1}] &= \frac{n+2}{n+1} E[X_n] + 2 \\ E[X_2] &= 1 \le 2 \quad / \quad n \ge 2 \end{aligned}}$$

$$E[X_5] = \frac{6}{5} \cdot E[X_4] + 2$$

$$= \frac{6}{5}\left(\frac{5}{4} E[X_3] + 2\right) + 2$$

$$= \frac{6}{5}\left(\frac{5}{4}\left(\frac{4}{3}\underbrace{E[X_2]}_{\le 2} + 2\right) + 2\right) + 2$$

$$= \frac{6}{5}\left(\frac{5}{4}\left(\frac{4}{3} \cdot 2 + 2\right) + 2\right) + 2$$

$$= \frac{6}{3} \cdot 2 + \frac{6}{4} \cdot 2 + \frac{6}{5} \cdot 2 + 2$$

<u>allgemein</u>: $\cancel{E[X_n] \le 2 \cdot \sum_{i=3}^{n+1} \frac{n+1}{i}}$

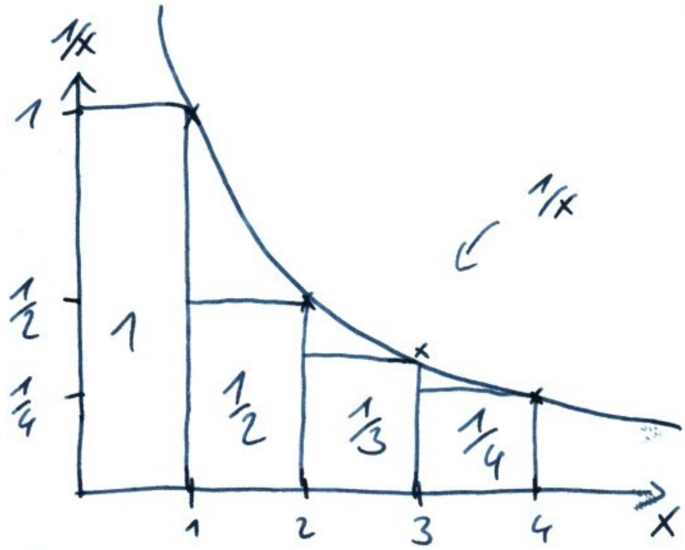
$$E[X_{n+1}] \le 2 \sum_{i=3}^{n+2} \frac{n+2}{i}$$

$$= \cancel{2n} \; 2(n+2) \sum_{i=3}^{n+2} \frac{1}{i}$$

6.7.17

$$2\,(n+2)\cdot\sum_{i=3}^{n+2}\frac{1}{i}$$

Suchen obere Schranke!



$$\sum_{i=3}^{n+2}\frac{1}{i}\leq\int_{2}^{n+2}\frac{1}{x}\,dx=\ln x\Big|_{2}^{n+2}$$

$$=\ln(n+2)-\ln(2)$$

<u>also:</u> $2(n+2)\cdot\ln(n+2)$

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

10.7.17

7. Hashing

- wie am Anfang der Vorlesung, Dictionary-Operationen:
  - Finden
  - Einfügen
  - Löschen

- Zeitaufwand bei n solchen Operationen:
  - mit balancierten Suchbäumen $O(n \cdot \log n)$
    (AVL, Rot-Schwarz, B-Baum)
  - mit Spley-Bäumen $O(n \cdot \log n)$

- Geht das besser? Zumindest unter gewissen Annahmen?

$\Rightarrow$ Führt zu Hashing mit average-case $O(1)$ pro Operation.

- Elemente werden über Schlüssel angesprochen.
  U Universum, Menge aller möglichen Schlüssel.
  z.B. $U = \mathbb{N}$.

- davon (Annahme!) treten nur wenige auf,
  S Menge der auftretenden Schlüssel.

$\cancel{\#S \lll \#U}$ $\boxed{\#S \lll \#U}$

- S kennt man meist vorher nicht!

10.7.17

- nehmen Tabelle T für die zu speichernden Schlüsselwerte. $\Rightarrow$ Hashtabelle

- T ist indiziert von $0, \ldots, m-1$

  $\#T = m$, können also m ~~Werte~~ Schlüsselwerte direkt speichern.

- Hashfunktion

$$h: U \to T \quad \text{bzw.} \; \{0, 1, \ldots, m-1\}$$

- Kollision: zwei Schlüssel kommen auf denselben Platz in T, d.h.

$$s, s' \in S \quad \text{und} \quad h(s) = h(s').$$

$\Rightarrow$ falls $\#S > m$, dann immer Kollisionen (klar)

Was ist bei $\#S \leq m$?

Wenn h auf S injektiv ist, keine Kollisionen.
Das ist i. A. unrealistisch, da S ja vorher nicht bekannt ist.

Operationen wären hier aber sehr einfach.

10.7.17

Eine einfache Hashfunktion

Sei $U = \mathbb{N}$, (jede abzählbare Menge lässt sich notfalls (vorher) nach $\mathbb{N}$ abbilden)

$\#T = m$

Dann können wir als Hashfunktion ~~die Divi~~ den Divisionsrest nehmen.

$$\boxed{h(s) := s \bmod m}$$

Was ist eine sinnvolle Wahl für m?

⇒ Wollen möglichst wenige Kollisionen haben, also auf jeden Fall $m \geq \#S$!

⇒ und sonst? Wähle m als Primzahl! ~~Warum?~~ Warum?

z.B. $S = \{0, 2, 4, 6, \dots\}$ $\boxed{m = 30 = 2 \cdot 3 \cdot 5}$

- nur gerade Zahlen
  - ⇒ es werden nur gerade Einträge von T getroffen! die ungeraden bleiben ungenutzt!
- genauso bei Vielfachen von 3, 5 od. Kombinationen davon!
- $S = \{0, \dots, 29\}$ kein Problem.

~~⇝ Diese~~

10.7.17

⇝ Diese Problematik trifft bei m Primzahl nicht auf!

m hat dann keine Teiler (außer sich selbst)
⇒ Äquivalenzklassen, Gruppentheorie.

Ab wann ist mit Kollisionen zu rechnen?

⇒ Geburtstagsparadoxon.

Bsp. ~~m=5~~ $m=13$ ⇒ bei 5 Elementen ist die Wkt. einer Kollision schon $> \frac{1}{2}$!

- allgemeine Berechnung ?!

$$P(\text{keine Kollision}) = \prod_{i=1}^{n-1} \frac{m-i}{m} = \dots$$

- Erwartungswert der # Kollisionen bei ~~m, n~~ Tabellengröße m und n eingetragenen Elementen.

- Belegungs- bzw. Lastfaktor $\beta = \frac{n}{m}$.

10.7.17

Umgang mit Kollisionen

- Überhanglisten, Hashing mit Verkettung
- Open Hashing

↳ bei Kollision neuen Platz suchen, nach einer best. Vorschrift, z.B. Lineares Sondieren.

mittlere Länge der Listen?

13.7.17

Noch eine Beobachtung

Anzahl der Plätze in der Hashtabelle T, die nicht direkt von der Hashfunktion h getroffen werden.

Prob[Platz wird nicht getroffen] $= 1-\frac{1}{m}$

Prob[Platz wird nie getroffen] $= (1-1/m)^n$

$$= \Big(\underbrace{\big(1-\tfrac{1}{m}\big)^m}_{\approx e^{-1}}\Big)^{\frac{n}{m}}$$

$$\approx e^{-\beta}$$

$\Rightarrow$ Erwartungswert: $m \cdot e^{-\beta}$ Plätze werden nie getroffen

$\Rightarrow$ nur ~~1 m~~ $m - me^{-\beta} = m(1-e^{-\beta})$

Plätze werden „gehasht".

$\beta = 1 \Rightarrow e^{-\beta} = e^{-1} = 0{,}368$ $\quad$ $1-e^{-\beta} = 0{,}632$

$\downarrow$ es könnte jeder Platz getroffen werden

$\downarrow$ etwa 2/3 der Plätze getroffen

$\beta = \frac{1}{2}$ $\quad$ $1-e^{-\frac{1}{2}} = 0{,}393$

etwas mehr als $\frac{1}{3}$ d. Plätze, dort dann Kollisionen.

13.7.17

Umgang mit Kollisionen

Einfachste Variante, Überhanglisten.

zu jeden Platz ex. Liste mit den dort gehashten Elementen.
(Algorithmus f. Einf., Löschen, Finden ist klar)

Zugriffszeiten:

Def.: $A(m,n) \Rightarrow$ Zeit falls Element gefunden

$A'(m,n) \Rightarrow$ Zeit falls Element nicht gefunden

(Bedeutung von m, n wie vorher:
m größe von T
n # Elemente in T)

Frage: (mittlere) Länge der Listen?

X = Länge einer Überhangliste.

oder mit Binomialverteilung direkt

$X = x_1 + \dots + x_n$

$x_i = 1 \Leftrightarrow$ i-tes gehashtes Element kommt in diese Liste.

$\text{Prob}[x_i = 1] = \frac{1}{m}$ $\quad E[x_i] = \frac{1}{m}$

$E[X] = E[\sum x_i] = \sum E[x_i] = n \cdot \frac{1}{m} = \beta$

Also: $A' = 1 + \beta = 1 + \frac{n}{m}$

Falls das gesuchte Element nicht in T ist, beträgt die Suchzeit im Mittel

$$\boxed{1 + \beta}$$

Und wenn das Gesuchte da ist?

Dazu eine Beobachtung:

- Die Elemente in T wurden in einer best. Reihenfolge eingetragen.

  $s_1, s_2, \dots, s_n$

- Beim Eintragen von s_i waren $i-1$ Elemente in T, s_i selbst aber noch nicht.

  Dabei wurden alle Elemente der Liste, in ~~durchlaufen, in d~~ der s_i landet, durchlaufen. (um sicherzustellen, dass s_i noch nicht dabei ist)

- Suchen wir am Ende wieder nach s_i, so werden genau dieselben Elemente wie oben durchlaufen, bis schließlich s_i gefunden wird, also ist die

  ~~A~~ Suchzeit von s_i $= A'(m, i-1)$

13.7.17.

Die mittlere Suchzeit ist dann ~~also die~~ der Mittelwert alle $A'(m,i)$

$$A(m,n) = \frac{1}{n}\sum_{i=0}^{n-1} A'(m,i)$$

$$= \frac{1}{n}\sum_{i=0}^{n-1}\left(1+\frac{i}{m}\right)$$

$$= \frac{1}{n}\left(\underbrace{\sum_{i=0}^{n-1} 1}_{n} + \underbrace{\sum_{i=0}^{n-1}\frac{i}{m}}_{\frac{n(n-1)}{2m}}\right)$$

$$= 1 + \frac{1}{n}\cdot\frac{n(n-1)}{2m}$$

$$= 1 + \frac{n-1}{2m} = 1 + \frac{n}{2m} - \frac{1}{2m} \leq 1 + \frac{\beta}{2}$$

irgendwo Logisch, es wird im Mittel die halbe Liste durchsucht.

13.7.17

<u>andere Frage:</u>

Wie lange dauert es, ~~d.h.~~ bzw. wie groß muß B werden, damit im Mittel in <u>jeder</u> Liste mind. ein Element ist?

(Coupon Collector Problem)

X = # Versuche, bis alle Plätze mind. einmal vorkommen.

Teilen X in Abschnitte ein: • Abschnitt endet, wenn ein neuer Platz belegt wird.
Zählen die Versuche in jeden Abschnitt.

$$X = X_1 + X_2 + \dots + X_m$$

X_1: #Versuche bis 1 Platz belegt

X_2: #Versuche bis 2. Platz benutzt.

X_m: # Vers., bis m-ter Platz benutzt.

innerhalb X_i: • $i-1$ Plätze bereits belegt

• $m-(i-1)$ bis jetzt unangetastete Plätze.

$$\text{Prob}[\text{neuer Platz}] = \frac{m-(i-1)}{m} = \underbrace{1-\frac{i-1}{m}}_{:=p_i}$$

$$E[X] = E\left[\sum_{i=1}^{m} X_i\right] = \sum_{i=1}^{n} E[X_i]$$

13.7.17

Was ist $E[x_i]$?

$= \frac{1}{p_i}$