

# Übung 14

## Damen und Heaps

Das N-Damen-Problem

Aufgabenstellung

Lösung

Optimierung

Datenstrukturen: Heaps

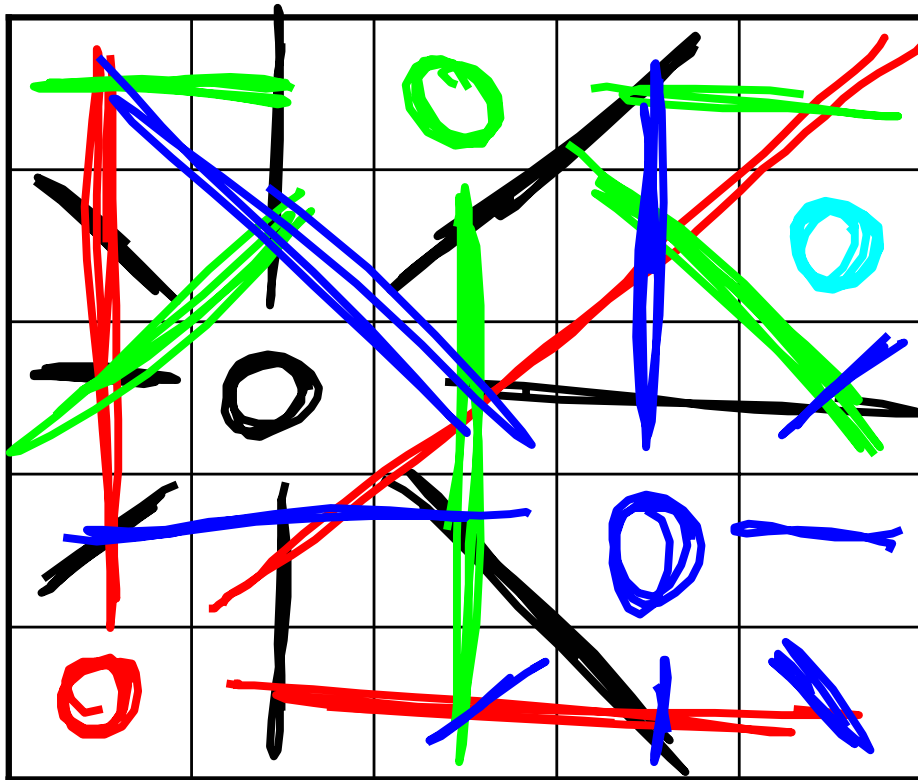
Die Heap-Eigenschaft

Erzeugen von Heaps

Manipulieren von Heaps

Anwendungsgebiete



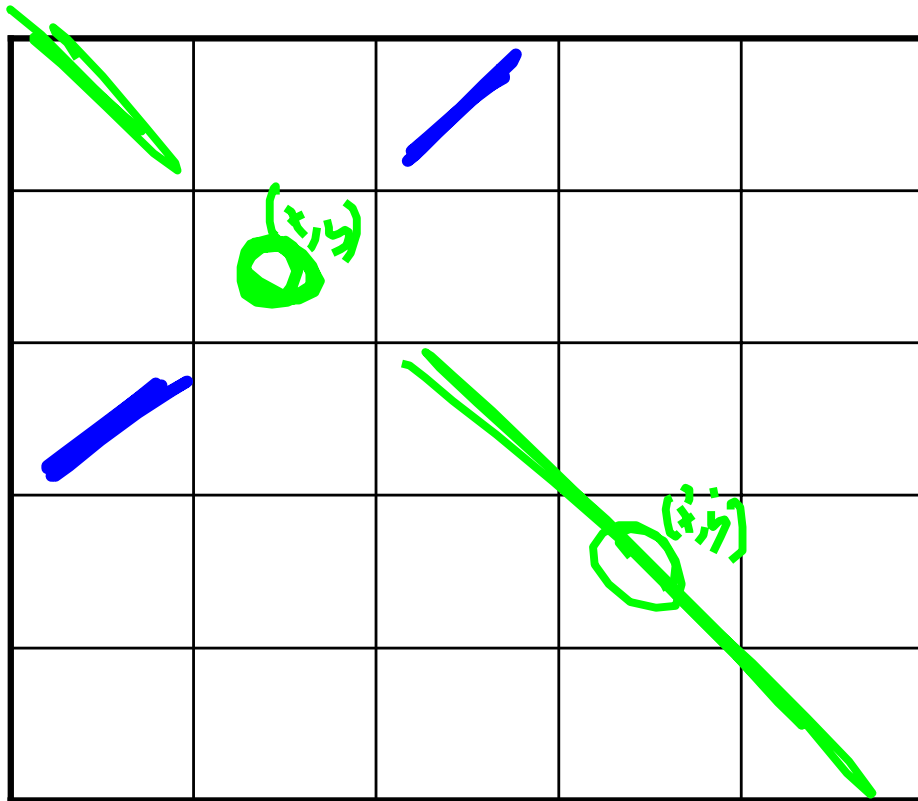



$$\binom{n^2}{n-1}$$

$$n^2 \cdot (n^2 - 1) \dots$$

$$(n^2 - n - 1)$$





$$y = y'$$

$$x = x'$$

$$\{(x+k, y+k)\}$$

$$\{(x+k, y-k)\}$$

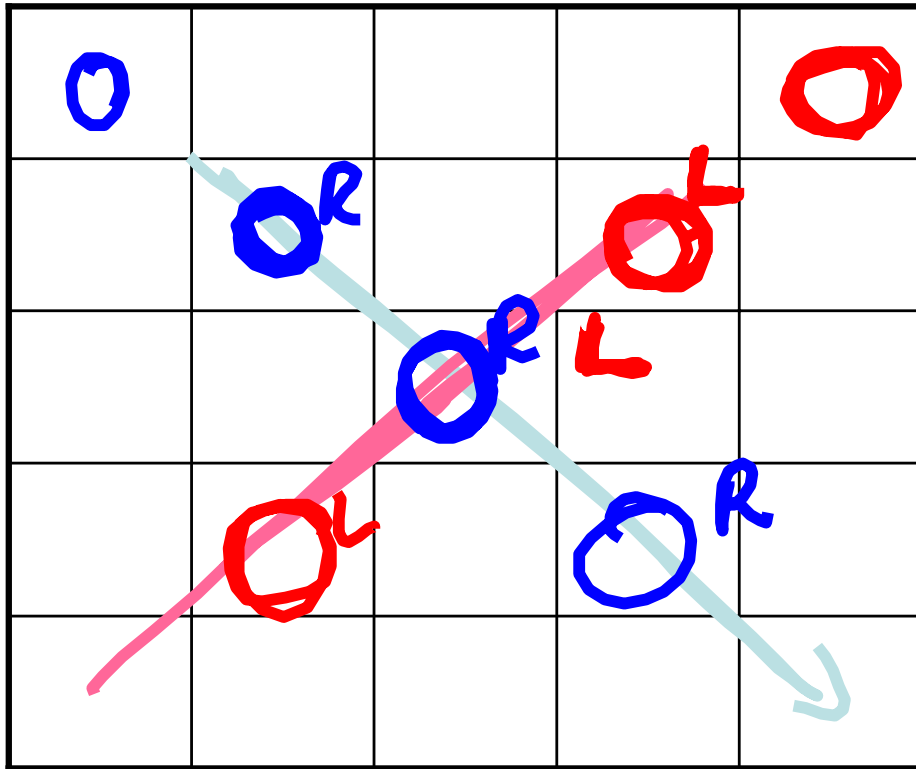
$$abs(x' - x) = abs$$

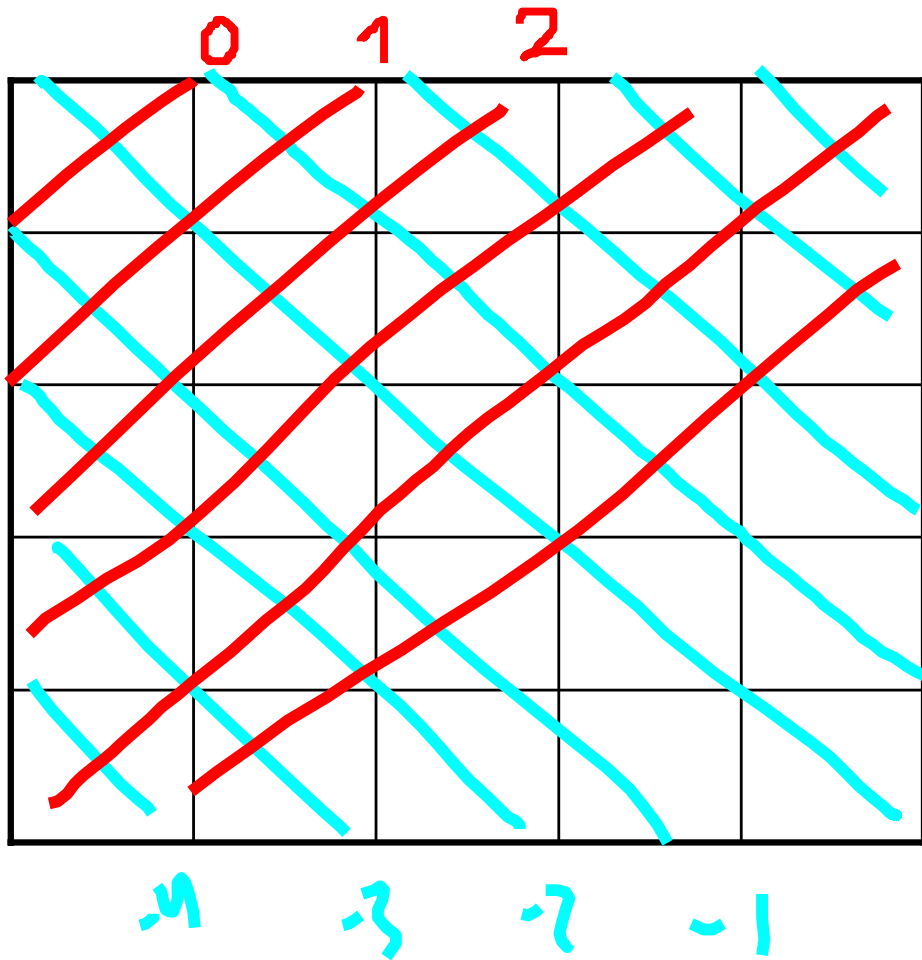
$$(y' - y)$$

$$y' = y + k$$

$$x' = x + k \Rightarrow (x' - x) = (y' - y)$$



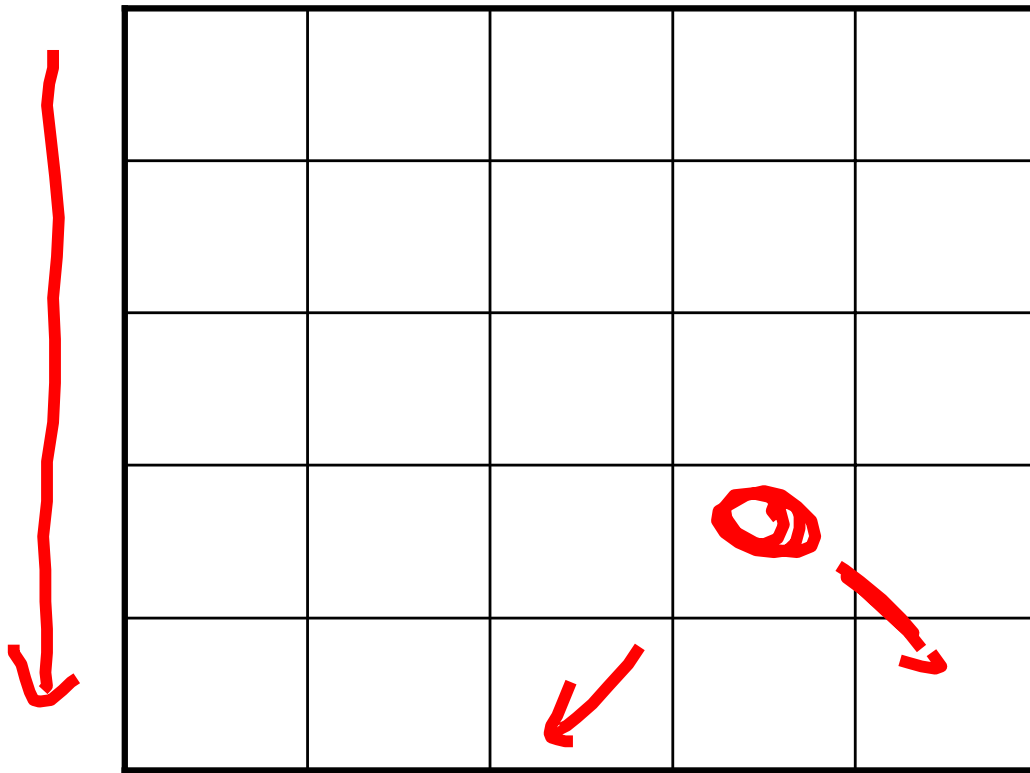




$$(x - y)$$

$$(x + y)$$





$|s|$     $|c|$     $|r|$

$|s|/|r|$

,

.

$n-1$     $n-1$     $n-1$

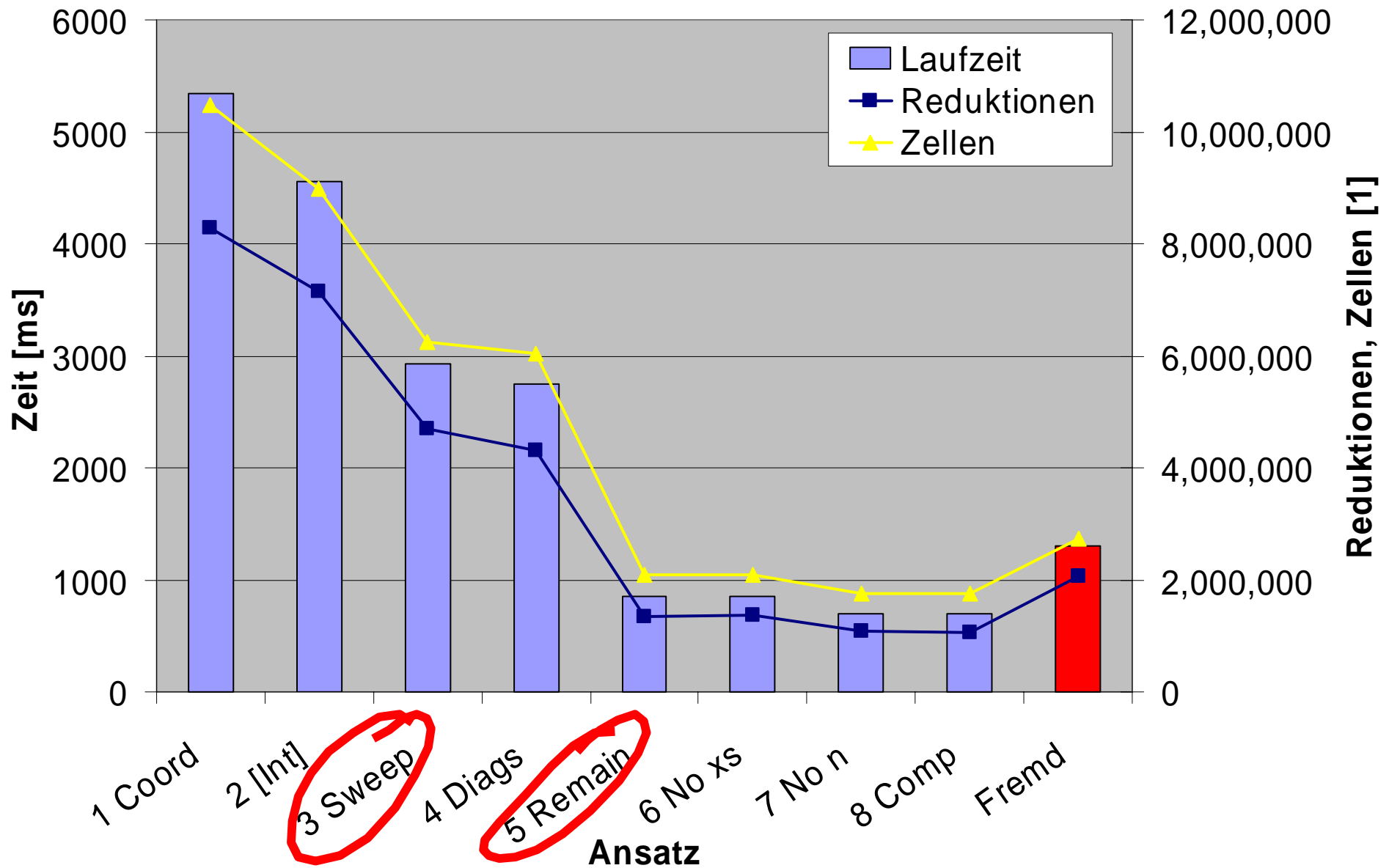


Kand:  $n$



bel

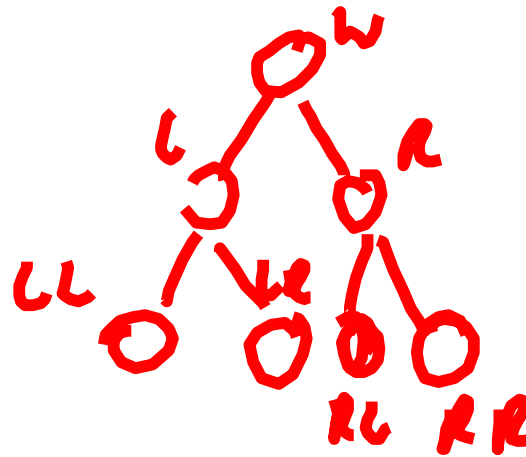




<b>Ansatz</b>	<b>Laufzeit</b>	<b>Reduktionen</b>	<b>Zellen</b>	<b>gcs</b>
1 Coord	5347	8291395	10483353	4
2 [Int]	4557	7137730	8981331	3
3 Sweep	2934	4687397	6251856	2
4 Diags	2744	4312074	6044058	2
5 Remain	851	1350865	2085263	0
6 No xs	851	1355514	2089761	0
7 No n	701	1072406	1765739	0
8 Comp	701	1062245	1752955	0
Fremd	1302	2054658	2739829	1

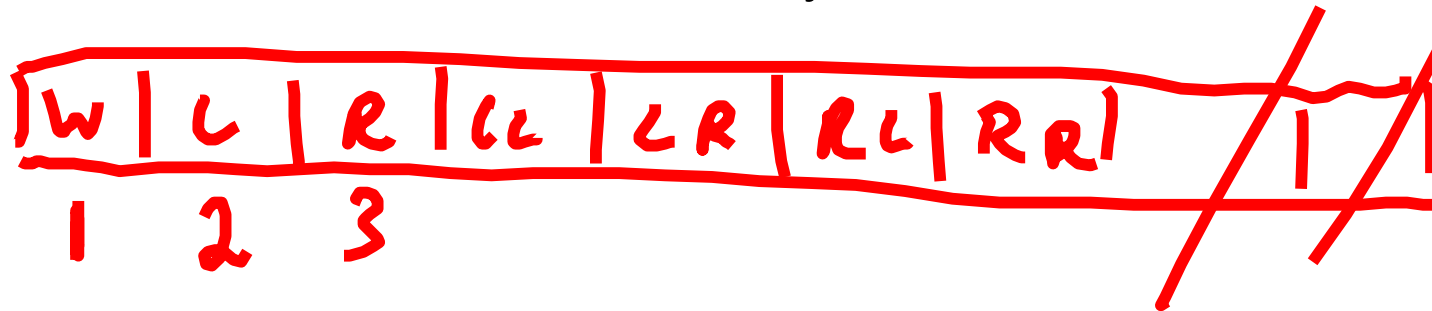


- Bekannt: Binärbäume



vollst.

- Speichern von Binärbäumen in Arrays



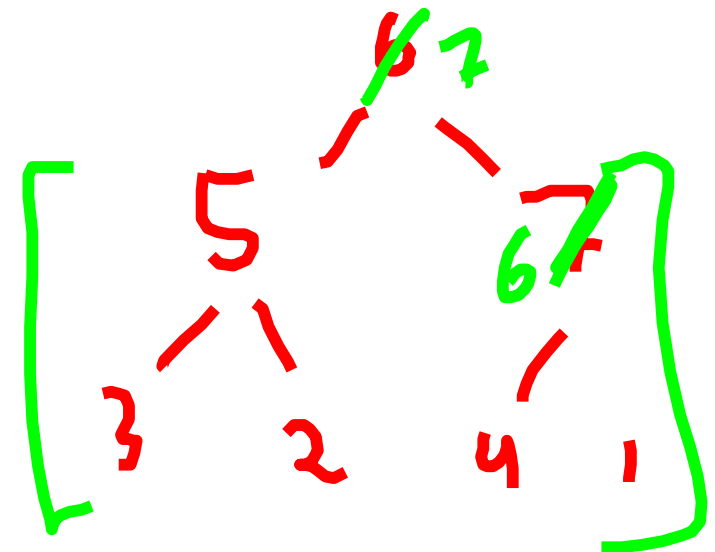
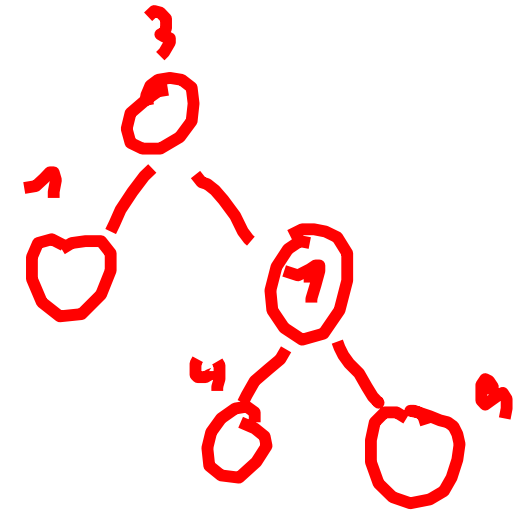
$$\text{left}(i) = 2 * i$$

$$\text{right}(i) = 2 * i + 1$$

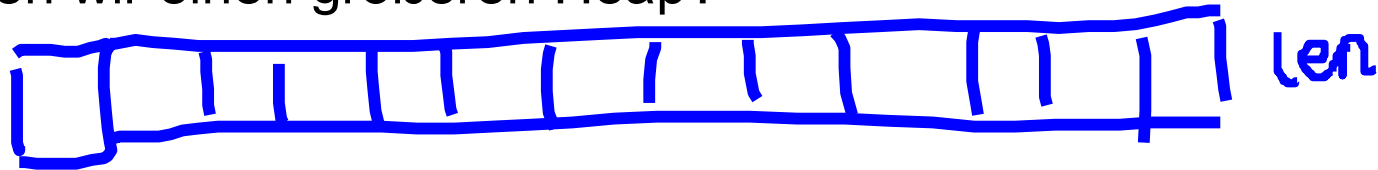
$$\text{parent}(i) = \lfloor i / 2 \rfloor$$



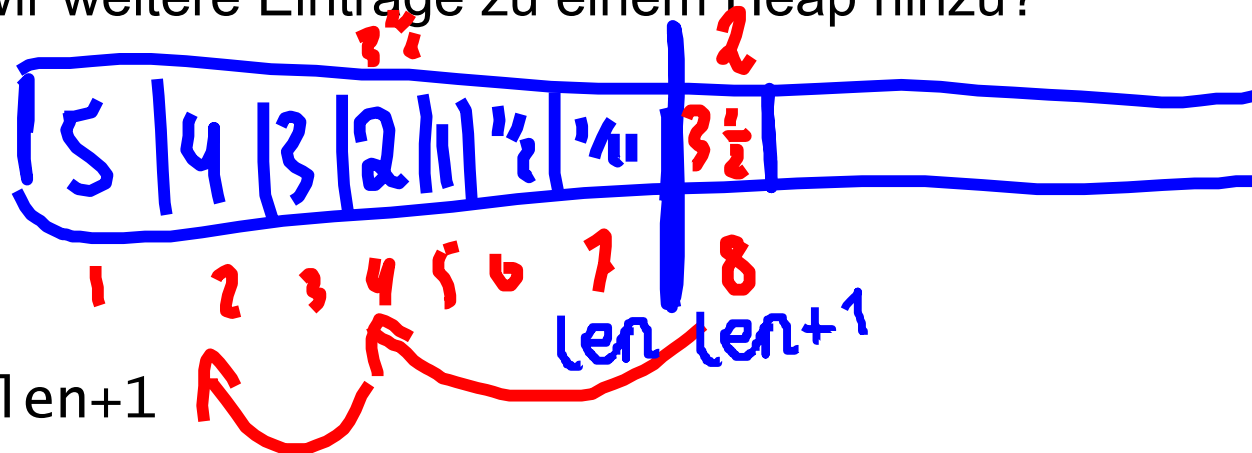
- Binäre Suchbäume:
  - Linkes Kind  $\leq$  Vater  $\leq$  Rechtes Kind
- Heap-Eigenschaft:
  - Kinder  $\leq$  Vater (  $A[i] \leq A[p(i)]$  )
- Wie erreichen wir diese Eigenschaft?
  - Annahme:  $(A, l(i))$  und  $(A, r(i))$  sind Heaps
  - heapify A i
    - Sei  $j \in \{i, l(i), r(i)\}$   
(Kinder soweit vorhanden)  
mit  $A[j]=\max$
    - Wenn  $i \neq j$ 
      - Tausche  $A[i], A[j]$
      - heapify A j



- Heaps mit einem Element erfüllen stets die Heap-Eigenschaft
- Wie erzeugen wir einen größeren Heap?



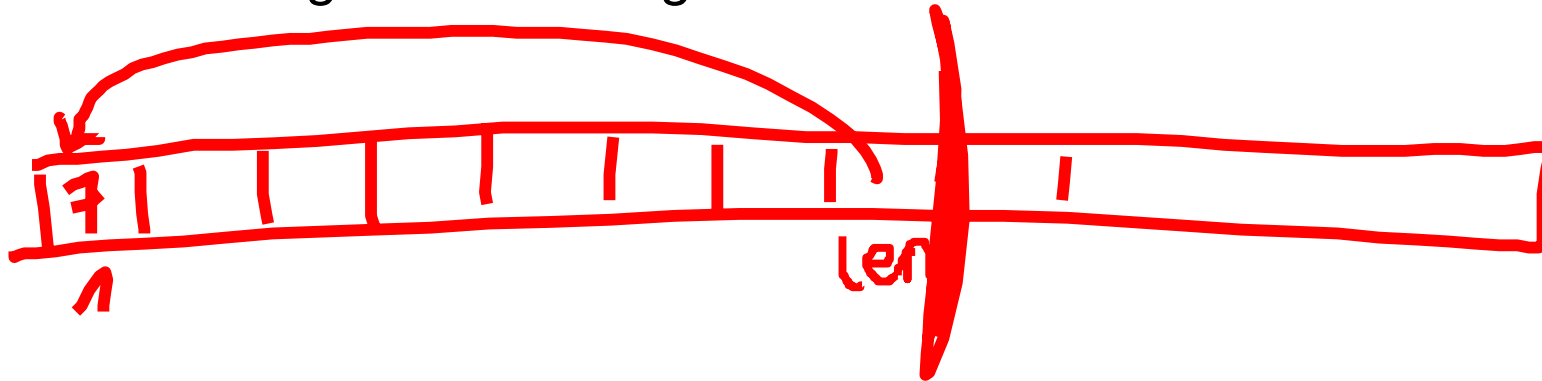
- Für  $j = \lfloor \text{len} / 2 \dots 1$ : heapify A j
- Wie fügen wir weitere Einträge zu einem Heap hinzu?



- $i = \text{len} = \text{len} + 1$
- $A[i] = w$
- while  $i > 0 \ \&\& \ A[(p)i] < A[i]$ 
  - Tausche  $A[p(i)], A[i]$
  - $i = p(i)$



- Entfernen des größten Eintrags

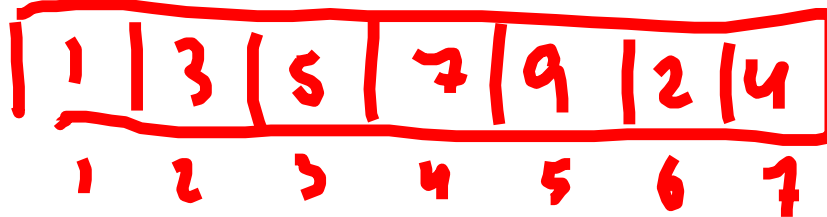


- $A[1]=A[\text{len}]$
- $\text{len}=\text{len}-1$
- `heapify A 1`

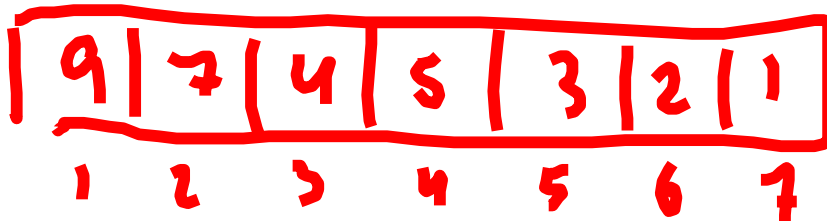
- → Heaps implementieren Prioritätsschlangen



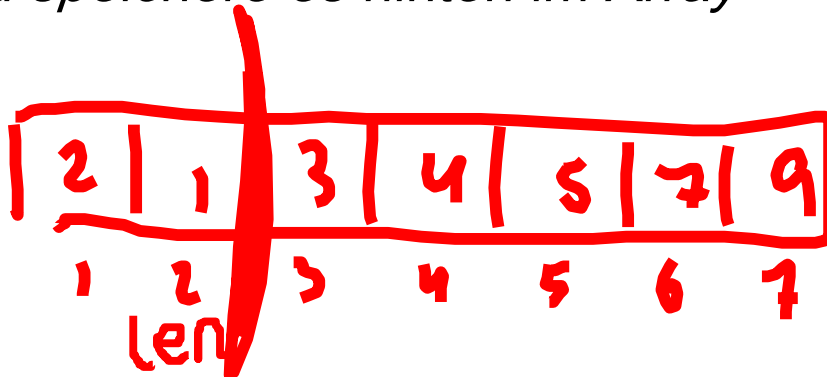
- Gegeben ein Array



- Stelle Heap-Eigenschaft her



- Entferne fortwährend das größte Element aus Heap  
*und speichere es hinten im Array*



- Wenn Heap leer, gib sortiertes Array zurück.

