11.04.13

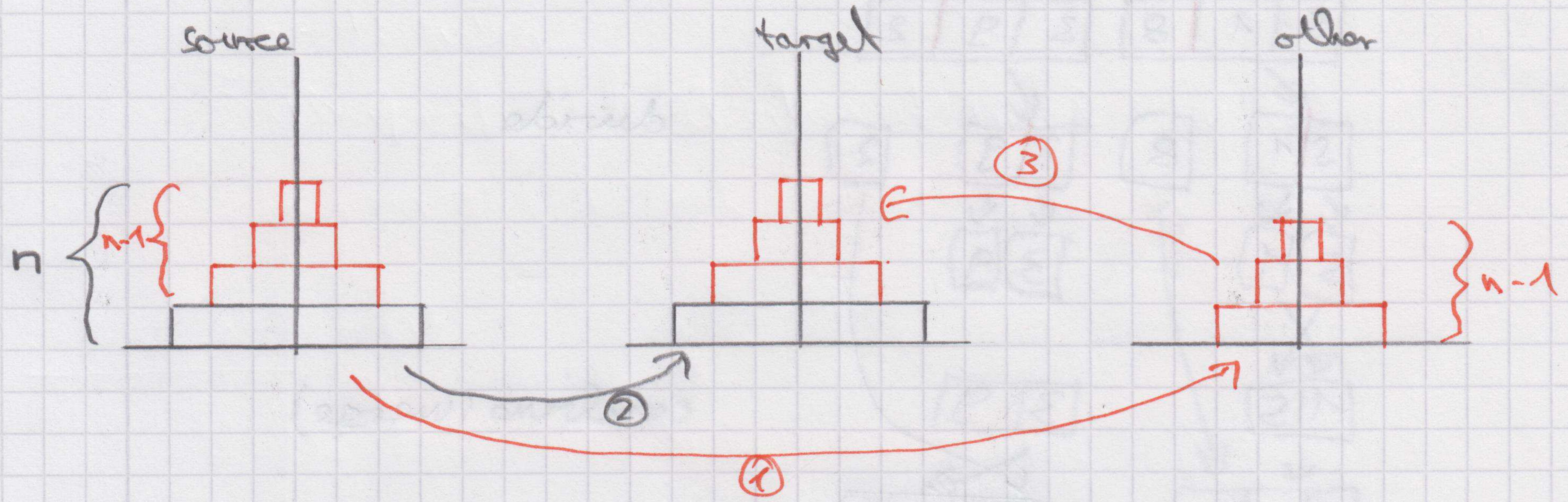
Rekursive Lösung Türme von Hanoi

Aufgabe:

hanoi(n, source, target, other)

← Anzahl Scheiben

Quelle   Ziel   Hilfe



① hanoi(n-1, source, other, target)

② Lege obere Scheibe von source nach target

③ hanoi(n-1, other, target, source)

• Anzahl Umschichtungen

$$T(n) = T(n-1) + 1 + T(n-1) = 2\cdot \underbrace{T(n-1)}_{2\cdot T(n-2)+1} + 1$$

$$= 2^2 \cdot \underbrace{T(n-2)}_{2T(n-3)+1} + 2 + 1$$

$$= 2^3 \cdot T(n-3) + 2^2 + 2 + 1$$

• Annahme (n=4):

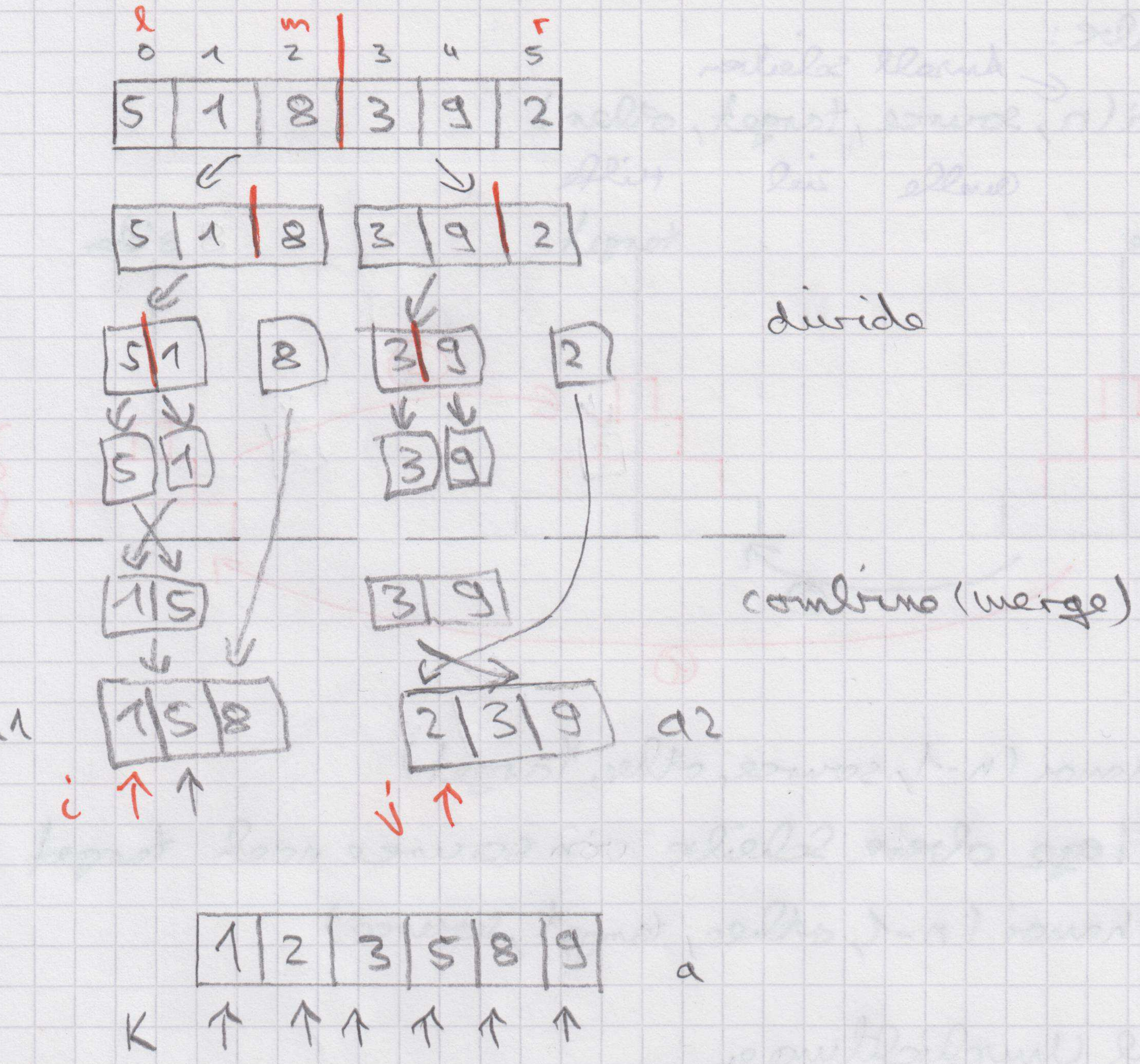
$$= 2^3 \cdot \underbrace{T(1)}_{1} + 2^2 + 3$$

$$\Rightarrow T(4) = 2^3 + 2^2 + 2 + 1 = \sum_{k=0}^{4-1} 2^k$$

• Vermutung: $T(n) = \sum_{k=0}^{n-1} 2^k \; (\forall n \in \mathbb{N}) = 2^n - 1$

↳ leicht beweisbar durch vollständige Induktion?

Bsp. (Merge Sort)



• Laufzeit-Analyse: $n = 2^k$

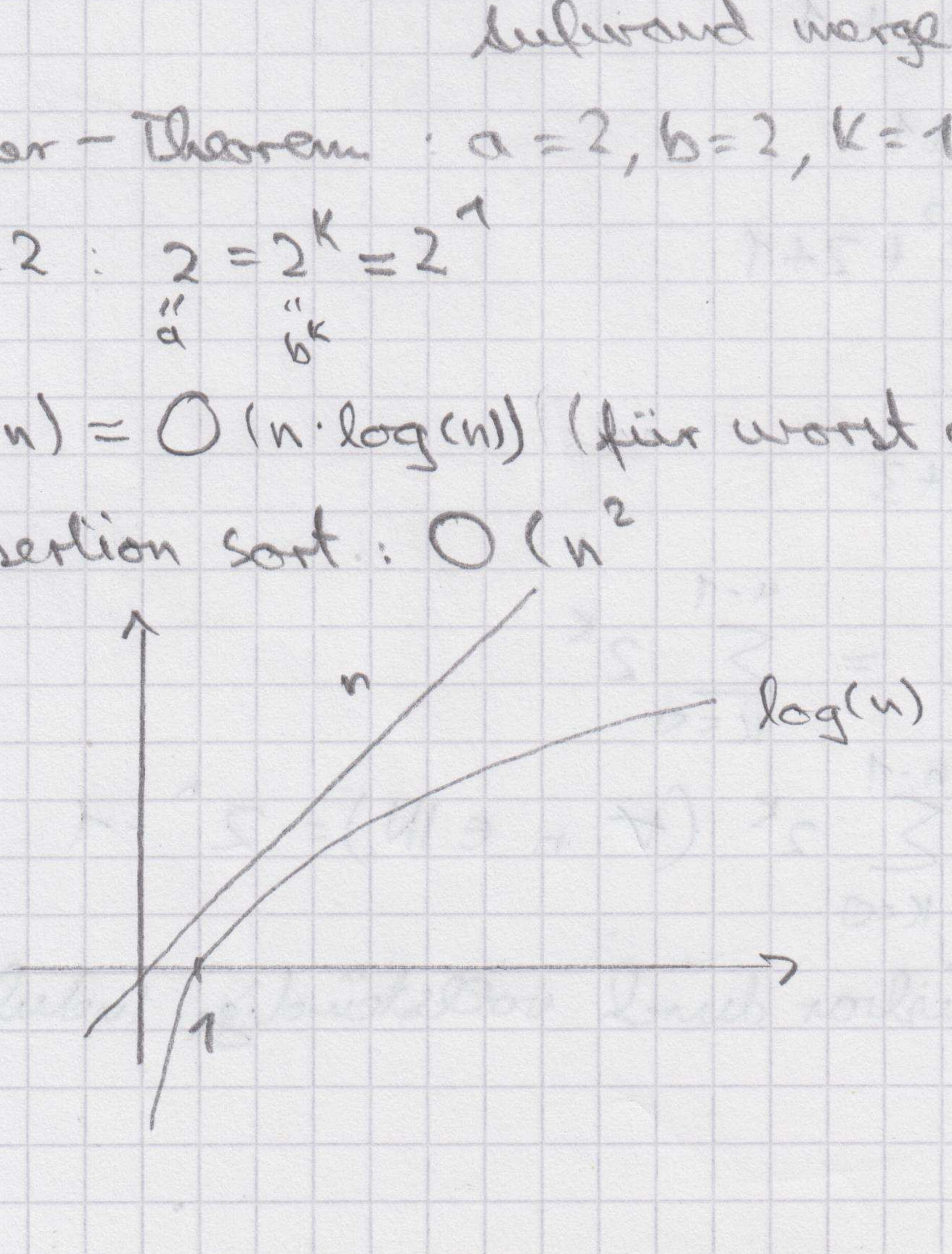
$T(n) = 2 \cdot T(\frac{n}{2}) + \underbrace{\overbrace{C \cdot n}^{\Theta(n^1)}}_{\text{Aufwand merge}}$

Master-Theorem: $a = 2, b = 2, k = 1$

Fall 2: $\underset{a}{2} = \underset{b^k}{2^k} = 2^1$

$\Rightarrow T(n) = O(n \cdot \log(n))$ (für worst case)

Insertion sort: $O(n^2$

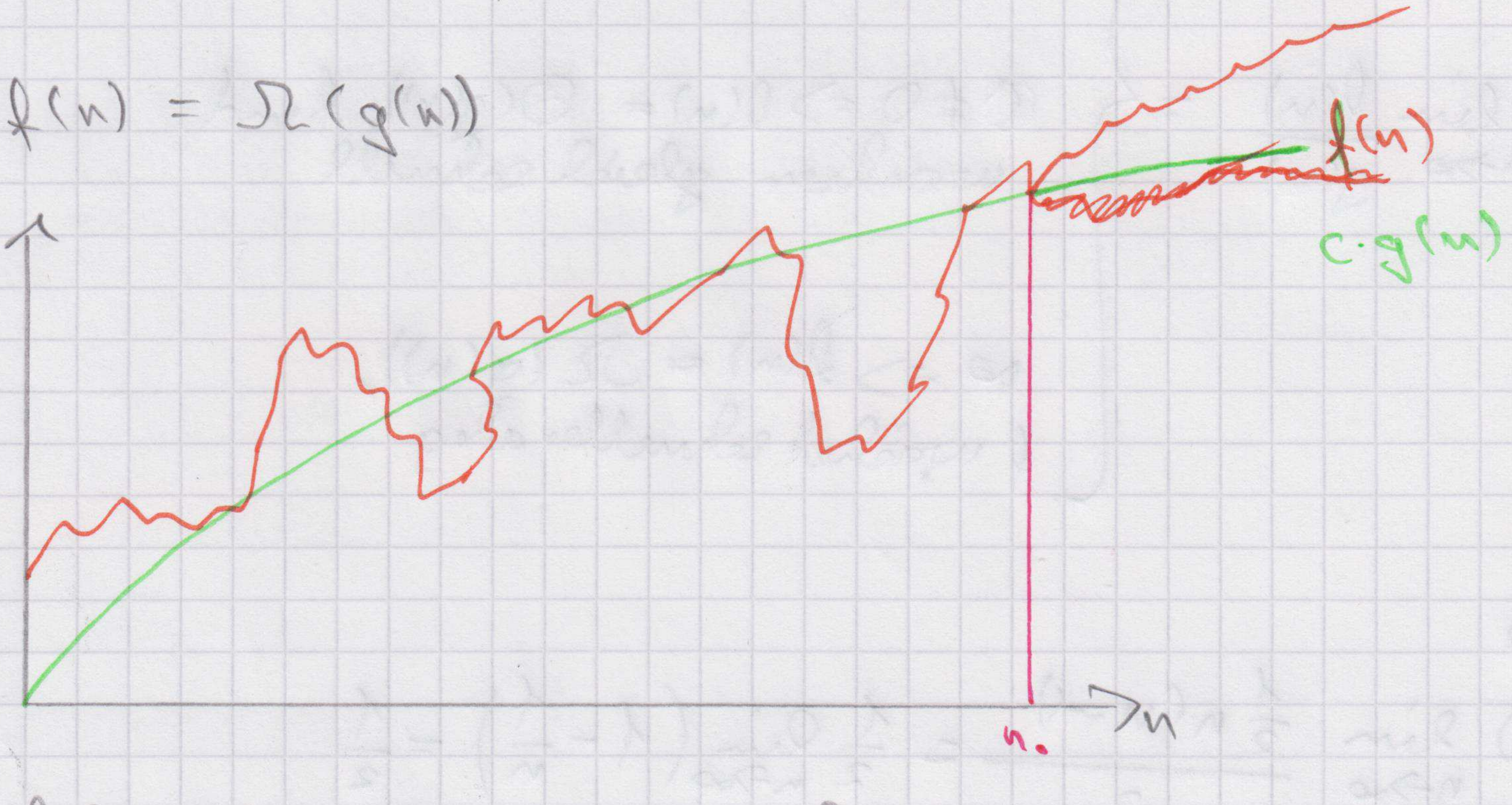


Bilder zur Asymptotischen Notation

1) $f(n) = O(g(n))$

- f bleibt ab n_0 unterhalb von $g(n)$
- dient als obere Schranke
- z.B. MergeSort benötigt höchstens $O(n \cdot \log(n))$ Operationen

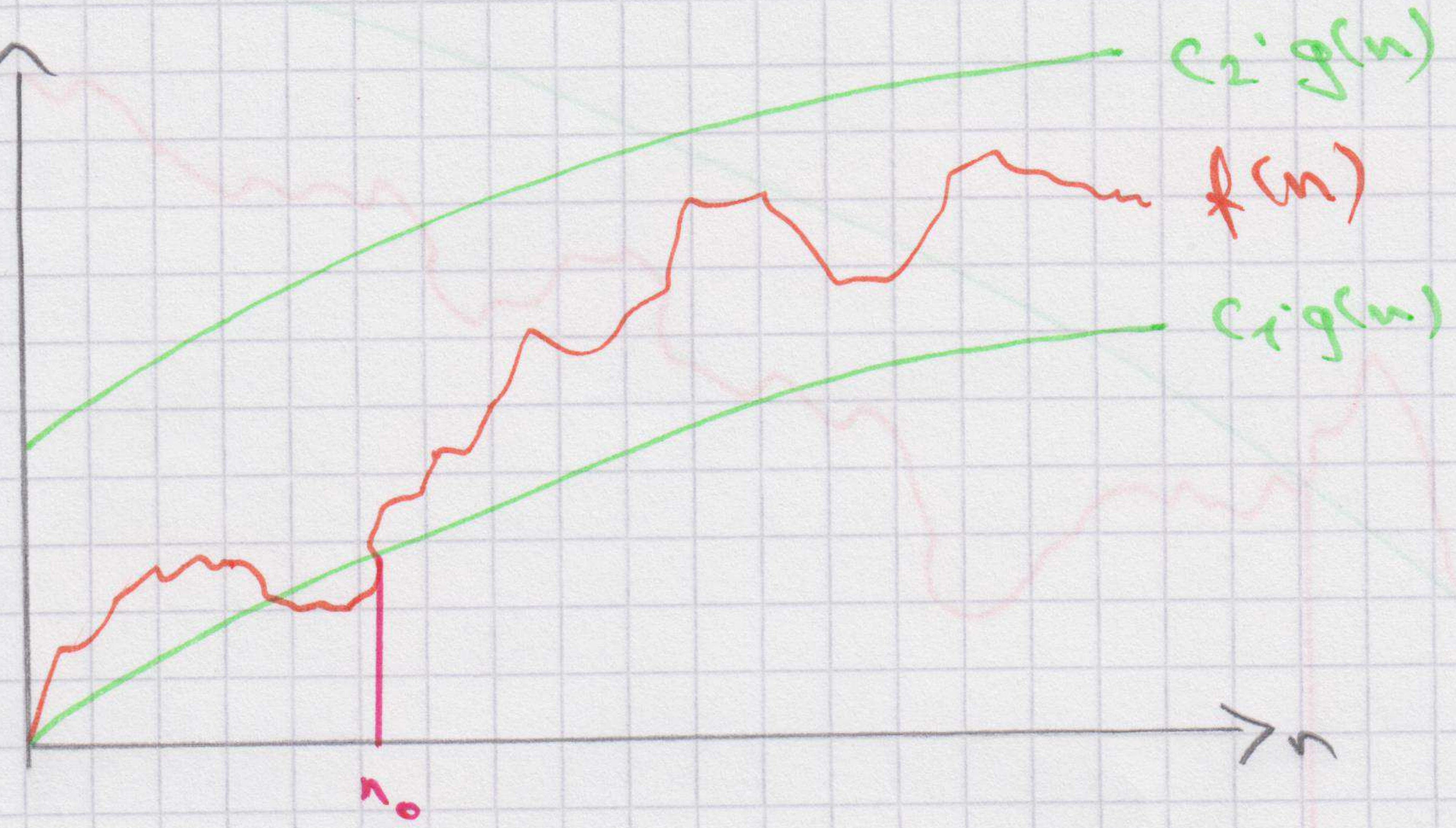
2) $f(n) = \Omega(g(n))$



- f bleibt ab n_0 immer oberhalb von $c \cdot g(n)$ (untere Schranke)
- z.B. jedes Sortierverfahren braucht mind. $\Omega(n)$ Operationen (jedes Element muss 1x angefasst werden)

3) Asympt. Äquivalenz:

$f(n) = \Theta(g(n))$



ab n_0 bleibt $f(n)$ zwischen $c_1 \cdot g(n)$ und $c_2 \cdot g(n)$

Kriterium:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 \Rightarrow f(n) = O(g(n)), \text{ denn } g(n) \text{ wächst schneller als } f(n) \\ c \neq 0 \Rightarrow f(n) = \Theta(g(n)), \ f \text{ und } g \text{ wachsen gleich schnell} \\ \infty \Rightarrow f(n) = \Omega(g(n)), \ f \text{ wächst schneller als } g \end{cases}$$

Bsp

a) $$\lim_{n \to \infty} \frac{\frac{1}{2} n(n-1)}{n^2} = \frac{1}{2} \lim_{n \to \infty} \left(1 - \frac{1}{n}\right) = \frac{1}{2}$$

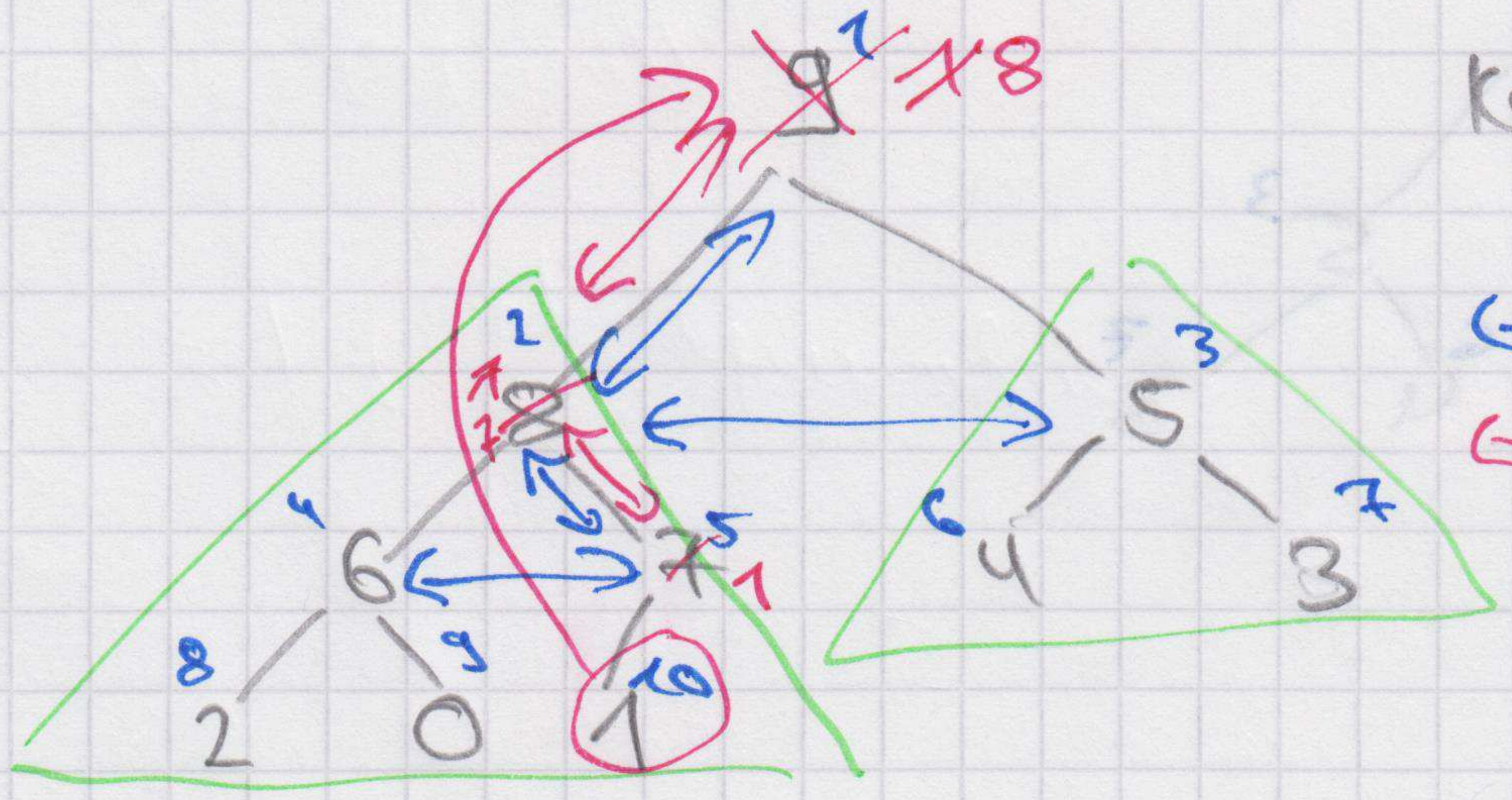
$$\Rightarrow \frac{1}{2} n(n-1) = \Theta(n^2)$$

b) $$\lim_{n \to \infty} \frac{\log_2(n)}{\sqrt{n}} = \lim_{n \to \infty} \frac{\frac{\ln(n)}{\ln(2)}}{\sqrt{n}} = \lim_{n \to \infty} \frac{2}{\ln(2)} \cdot \frac{1}{\sqrt{n}} = 0$$

$$\text{L'Hos} = \lim_{n \to \infty} \frac{\frac{1}{\ln(2)} \cdot \frac{1}{n}}{\frac{1}{2\sqrt{n}}} = \lim_{n \to \infty} \frac{2}{\ln(2)} \cdot \frac{\sqrt{n}}{n}$$

$$\Rightarrow \log_2(n) = O(\sqrt{n})$$

Bsp: Maximum Heap



Kein Binärer Suchbaum!

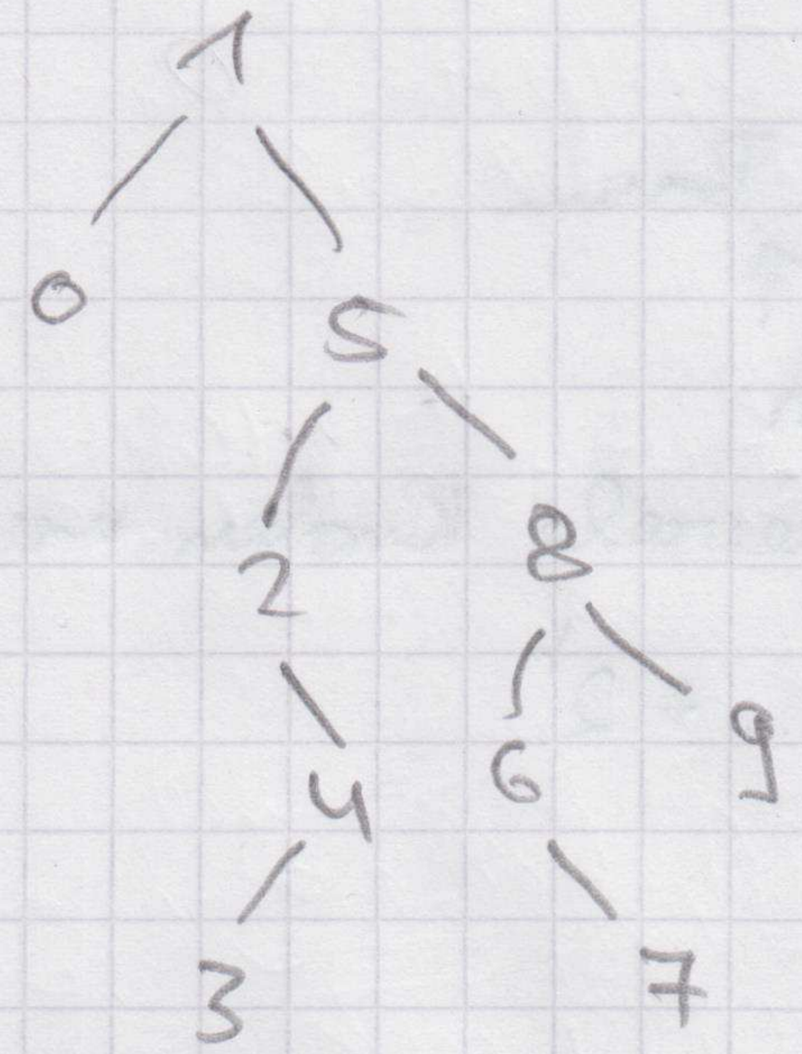
⟷ Vergleiche

⟷ swap

↳ Einschub: Binärer Suchbaum

- linker Sohn ist immer kleiner als der Vater
- rechter Sohn ≥ Vater

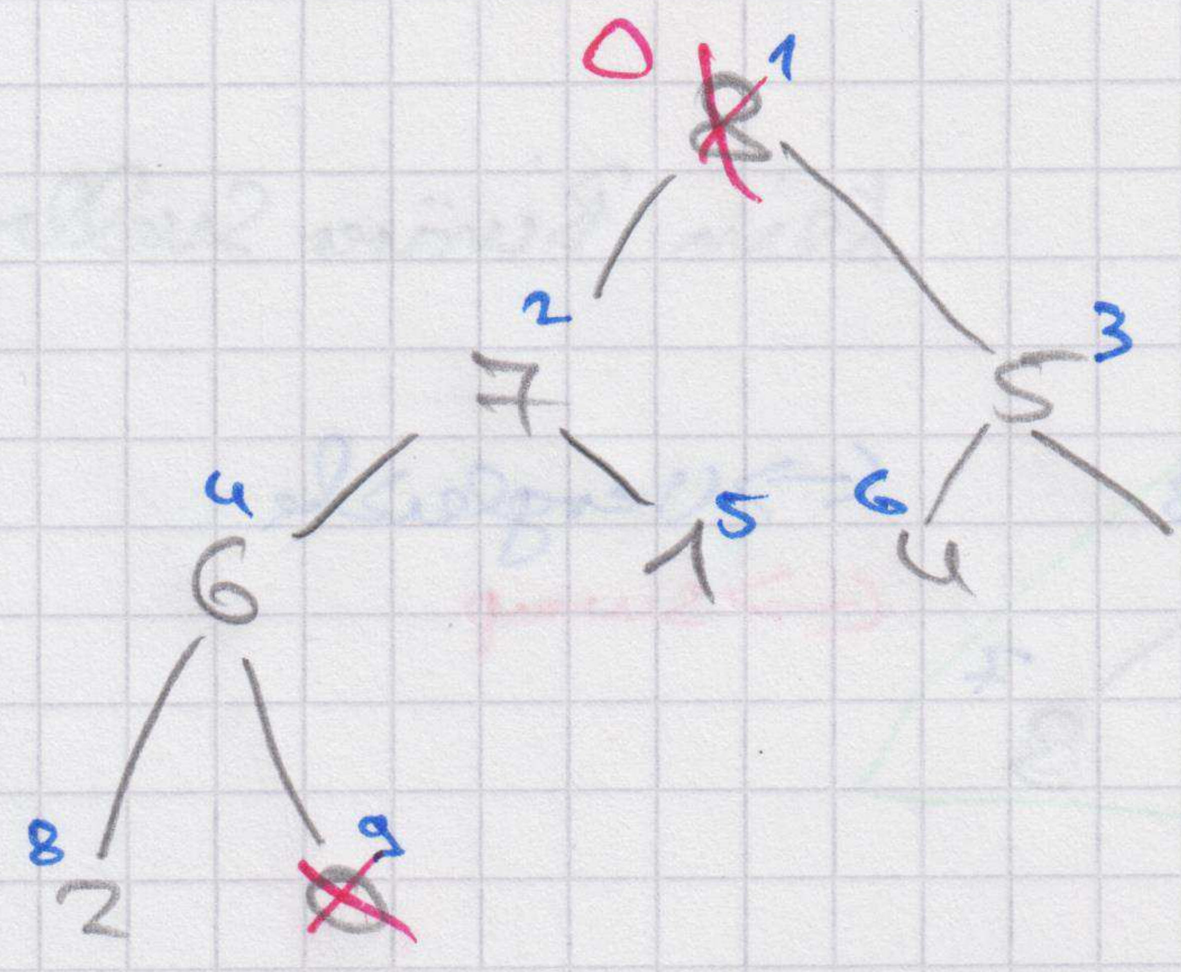
z.B. 1, 0, 5, 2, 8, 4, 6, 9, 3, 7

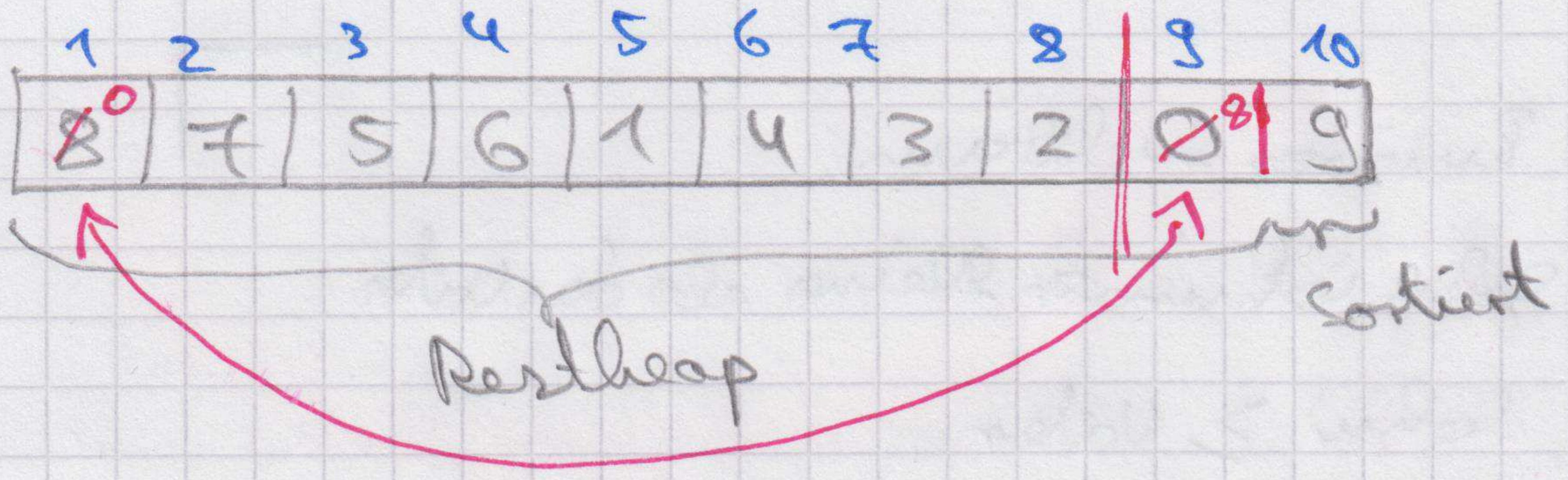


Speicherung in Feld mit Hilfe der Level-Nummerierungen

K								last	
1	2	3	4	5	6	7	8	9	10
~~9~~	8	5	6	7	4	3	2	0	~~1~~
1									9

Heapeigenschaften herstellen durch Heapify:





· Aufwand Heapify:

$$\begin{pmatrix} 1\times \text{swap} \\ 2\times \text{Vergleich} \end{pmatrix} \cdot \underbrace{\text{Höhe Baum}}_{h}$$

Baum der Höhe h hat wieviele Knoten maximal:

$$1 + 2 + 2^2 + 2^4 + \dots + 2^h$$

$$\sum_{k=0}^{h} 2^k = \frac{2^{h+1}-1}{2-1} = n$$

$$\Rightarrow 2^{h+1} = n-1$$

$$h+1 = \log_2(n-1)$$

$$h = \log_2(n-1) - 1$$

$$= O(\log(n))$$

$$\log_2(n) = \frac{\log(n)}{\log(2)} = \underbrace{\frac{1}{\log(2)}}_{\text{Konstante}} \cdot \log(n)$$

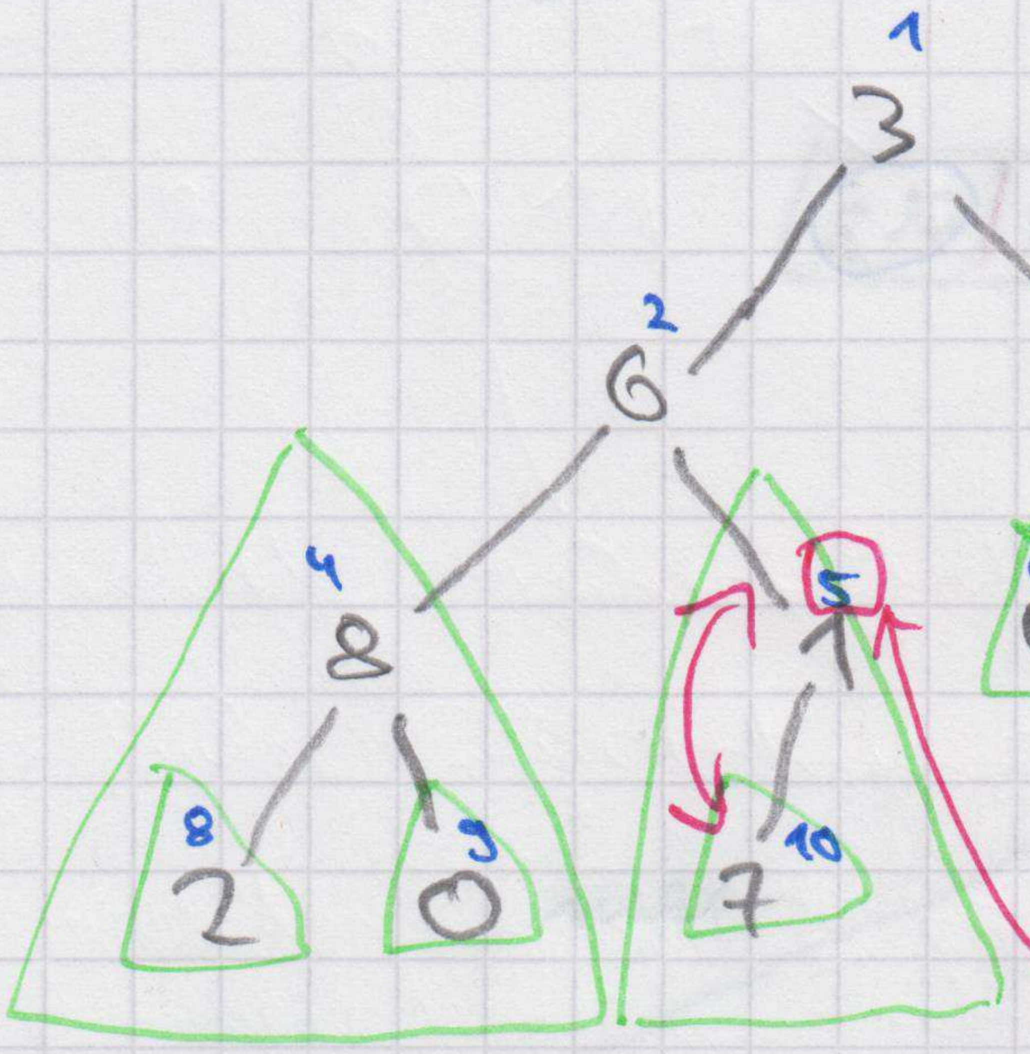
$$O(\log_2(n)) = O(\log(n))$$

$$\hookrightarrow c \cdot O(\log(n)) = O(\log(n))$$

Erstellung Startheap

Start-Feld:

1	2	3	4	5	6	7	8	9	10
3	6	5	8	1	4	9	2	0	7

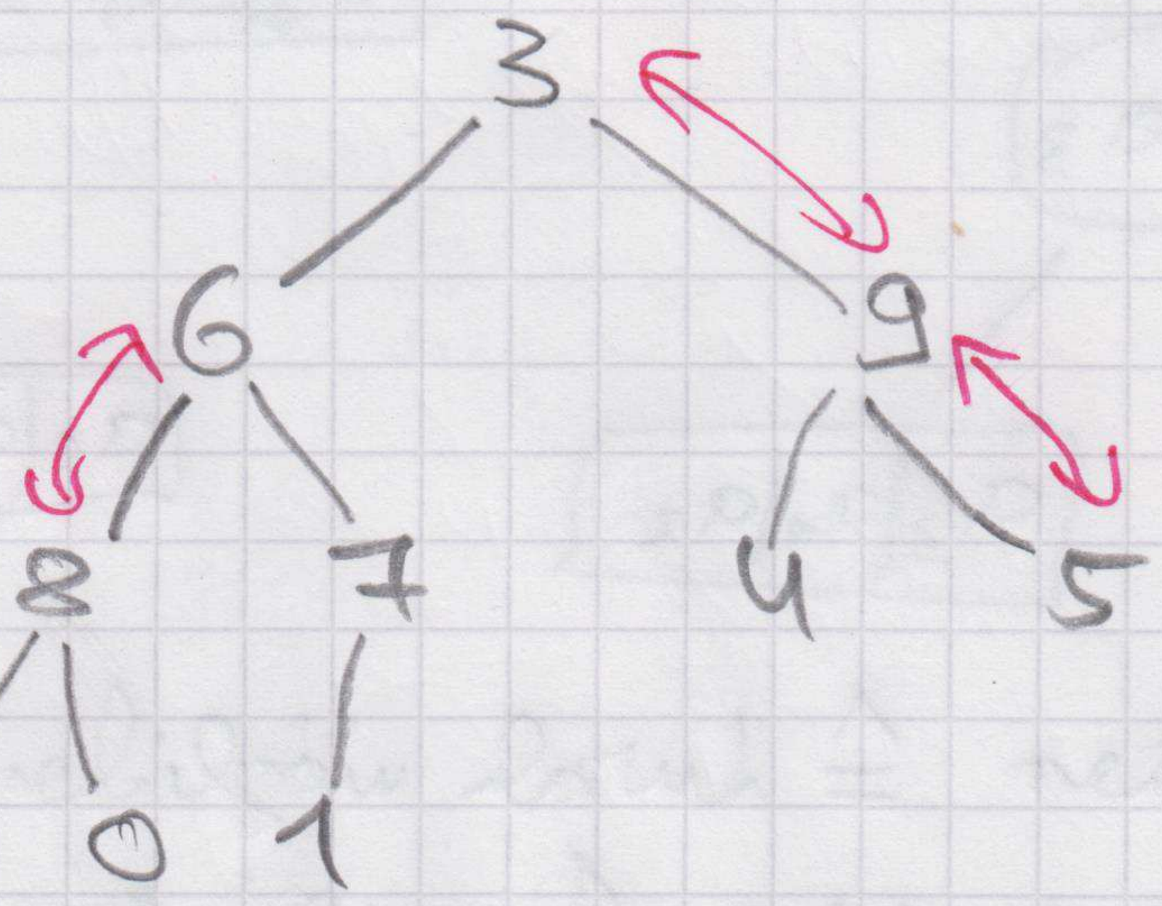


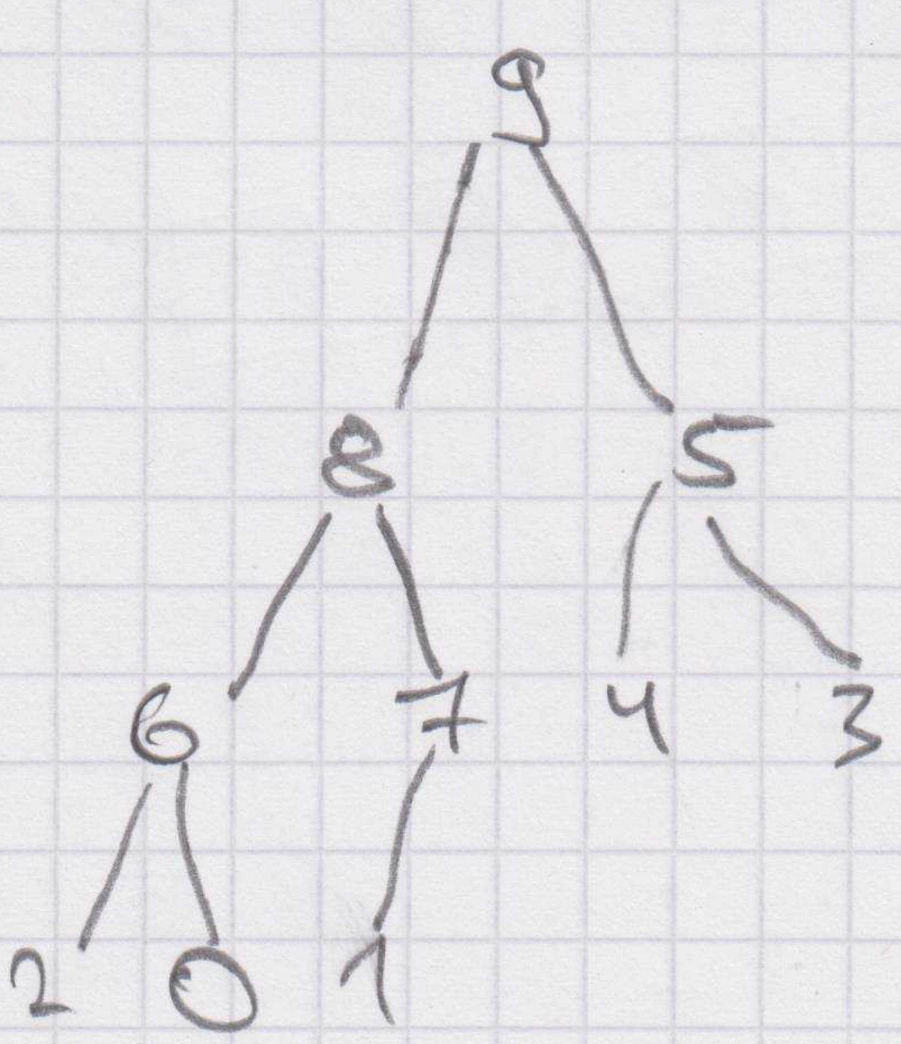
$5 = \frac{n}{2} = \frac{10}{2}$

ist schon heap

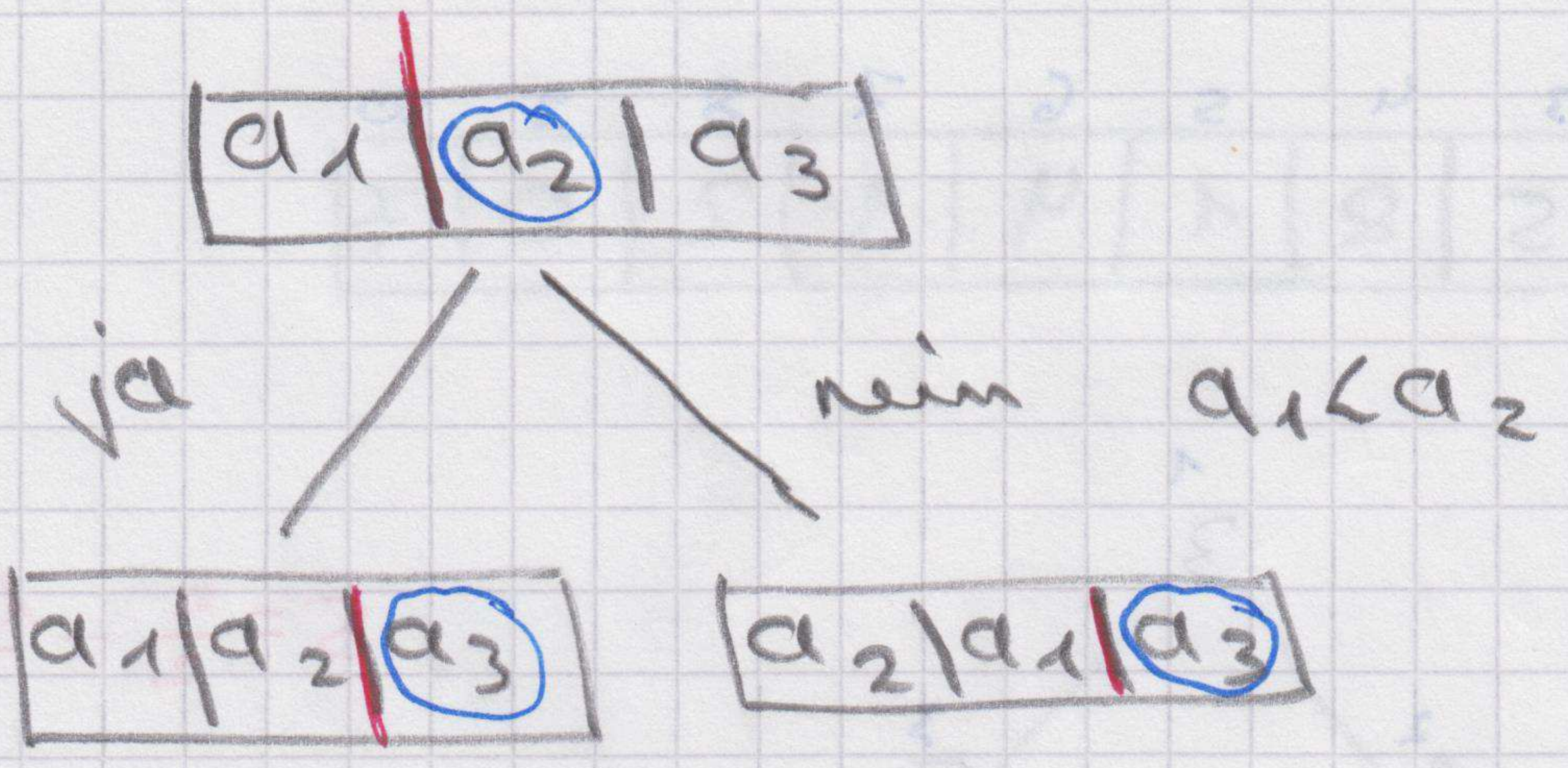
erstes nicht Blatt an dem die Heap-Eigenschaft verletzt sein kann

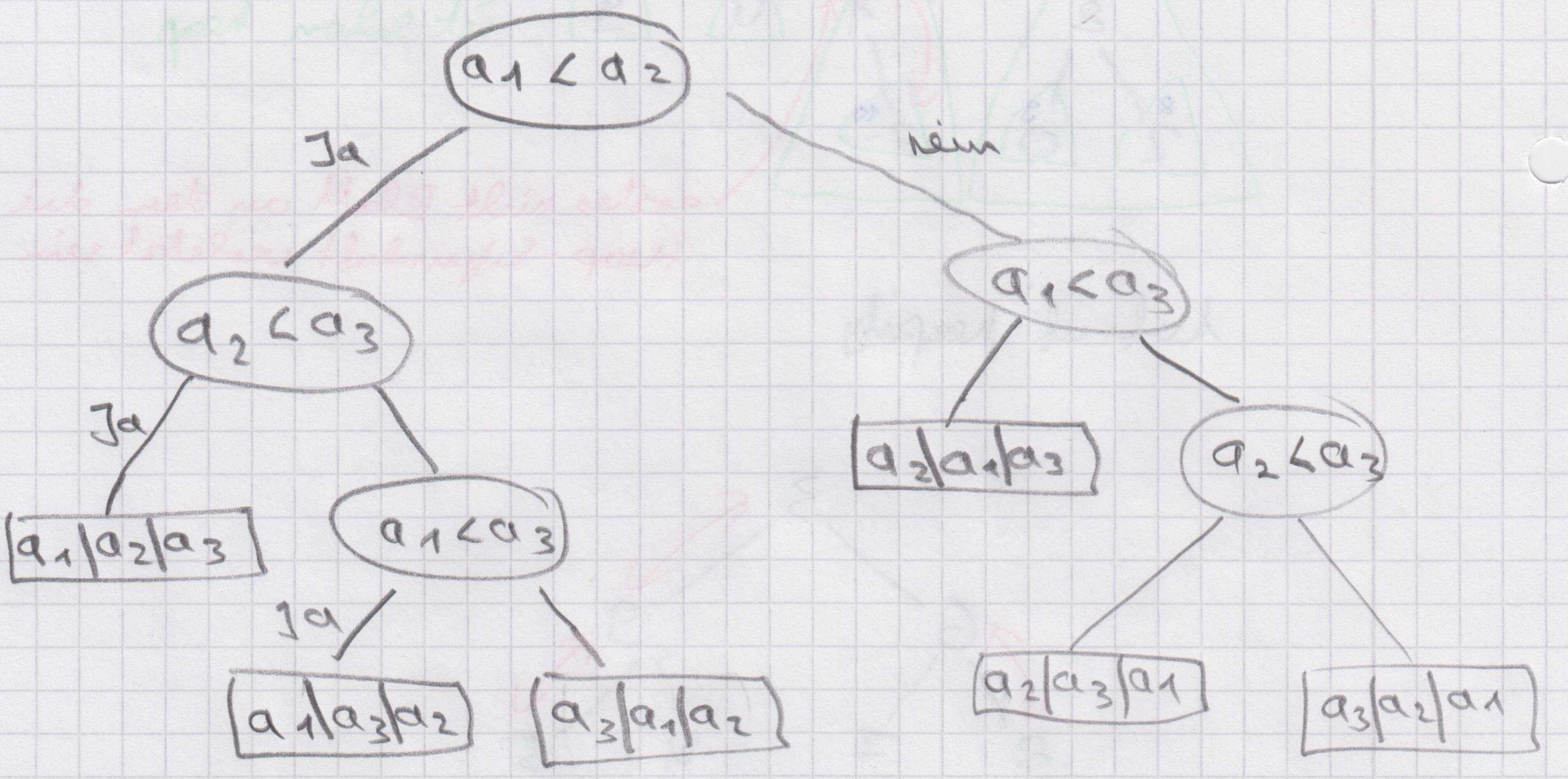
Aufruf heapify





Entscheidungsbaum für Insertion-Sort





hat $6 = 3!$ Blätter $\triangleq$ Anzahl möglicher Anordnungen

Anzahl Vergleiche im WC $\triangleq$ Höhe des Baumes

Anzahl Blätter $b \leq 2^h$

$\log_2(b) \leq h$

$\Rightarrow \lceil \log_2(b) \rceil \leq h$

$b = n!$

Bsp Count-Sort

ist stabil

a

0	1	2	3	4	5	6	7
2	5	3	0	2	3'	0	3''

start

$K = 5$

c

0	1	2	3	4	5
2	0	2	3	0	1

unter Index i steht drin, wie oft i vorkommt

2 2 4 7 7

c

0	1	2	3	4	5
2	2	4	7	7	8
~~1~~		3	~~6~~		7
0			~~5~~		
			4		

unter Index i steht, wieviel Elemente $\leq$ i sind

Ergebnisfeld b

5	2	8	4	6	3	1	7
0	0	2	2	3	3'	3''	5
0	1	2	3	4	5	6	7