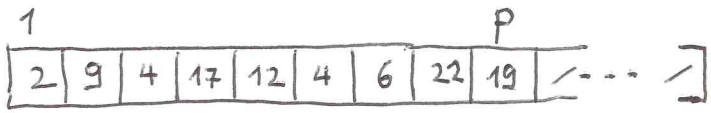
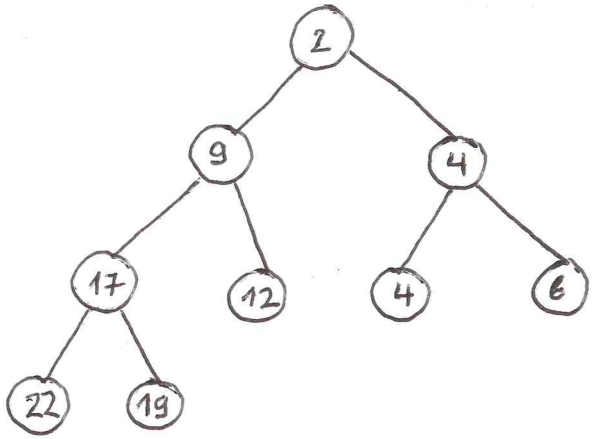
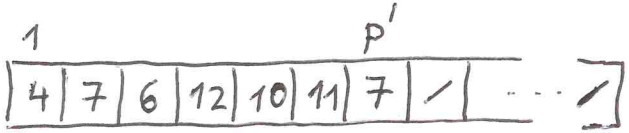
Binärheaps (AuD)





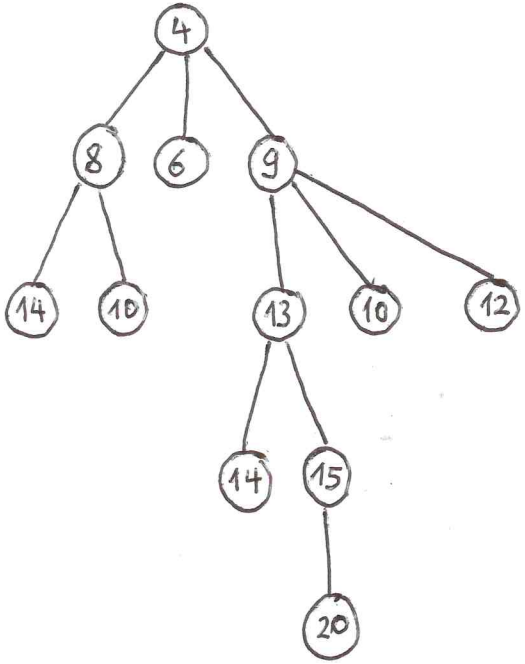
meld (H')

kombiniere mit

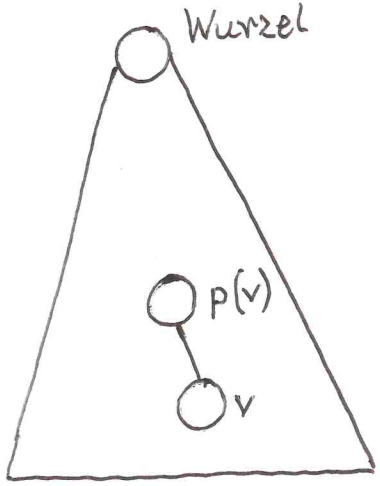


Lösung:
Vollständiger Neuaufbau.

Heapgeordneter Baum:



Schema:



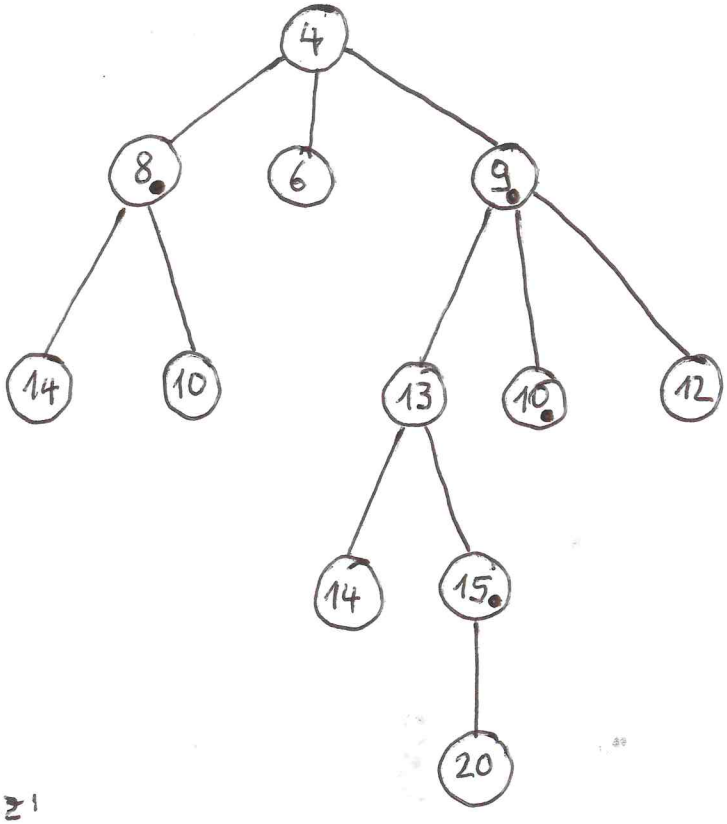
key(v)
$\geq$ key(p(v))

Konsequenz:
Schlüssel in Wurzel ist minimal

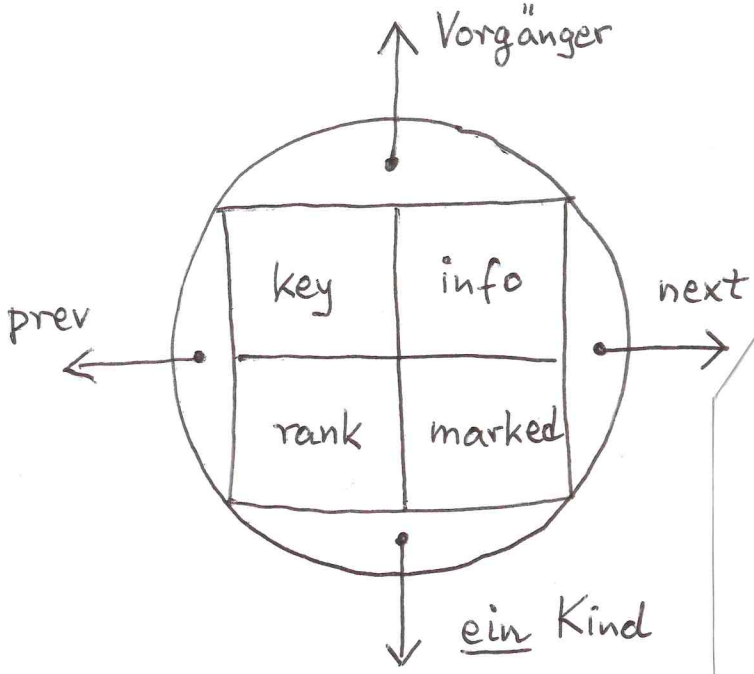
Zusätzlich bei F-Heaps:

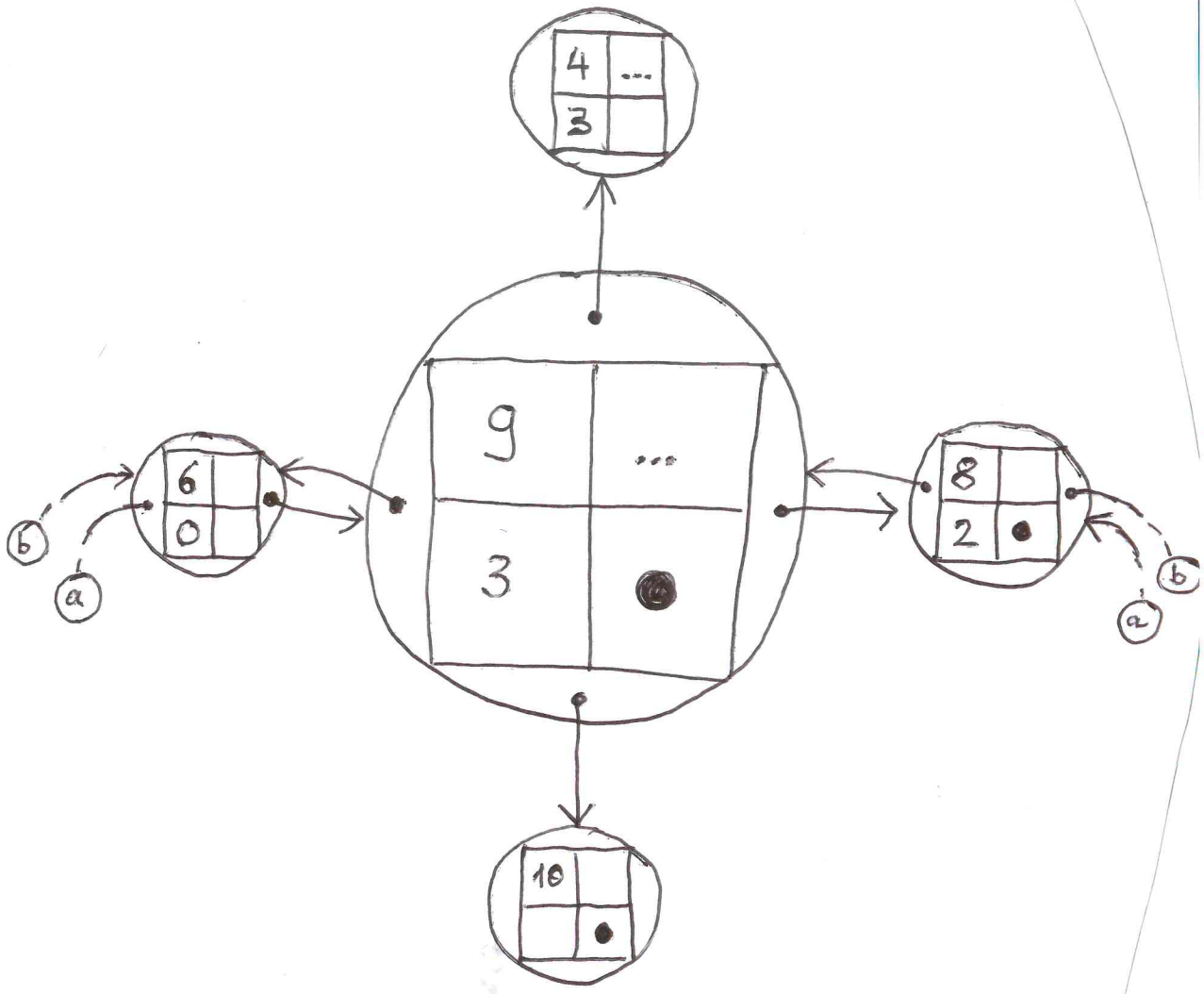
Knoten können „markiert" sein: 

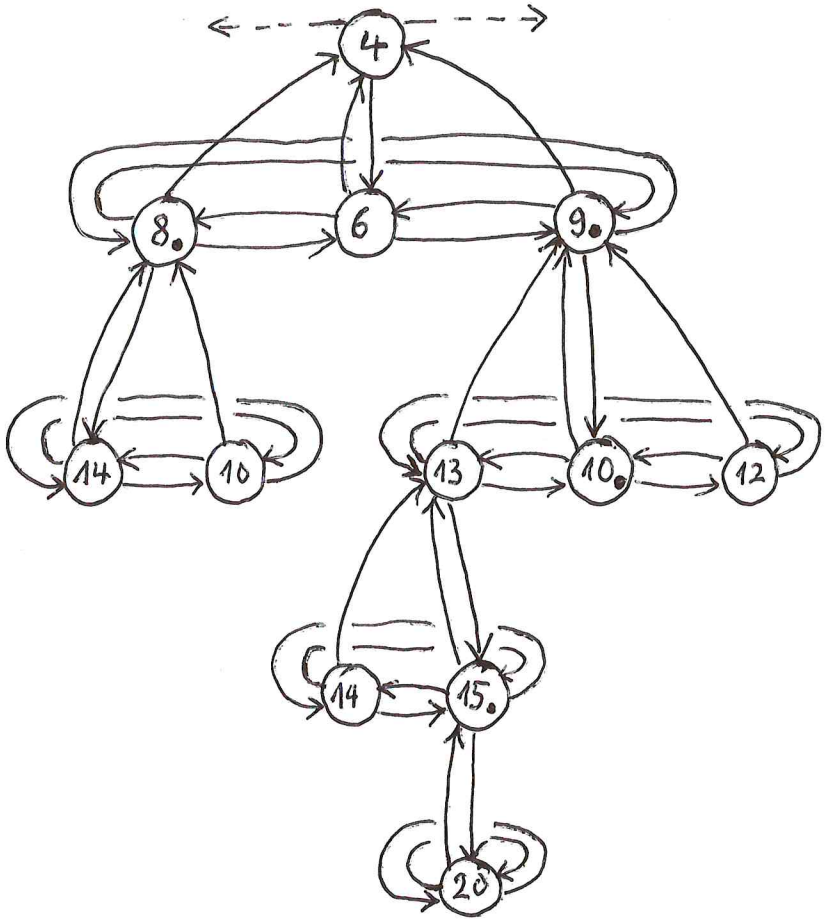
(Nur Nicht-Wurzeln)



Knotenformat:







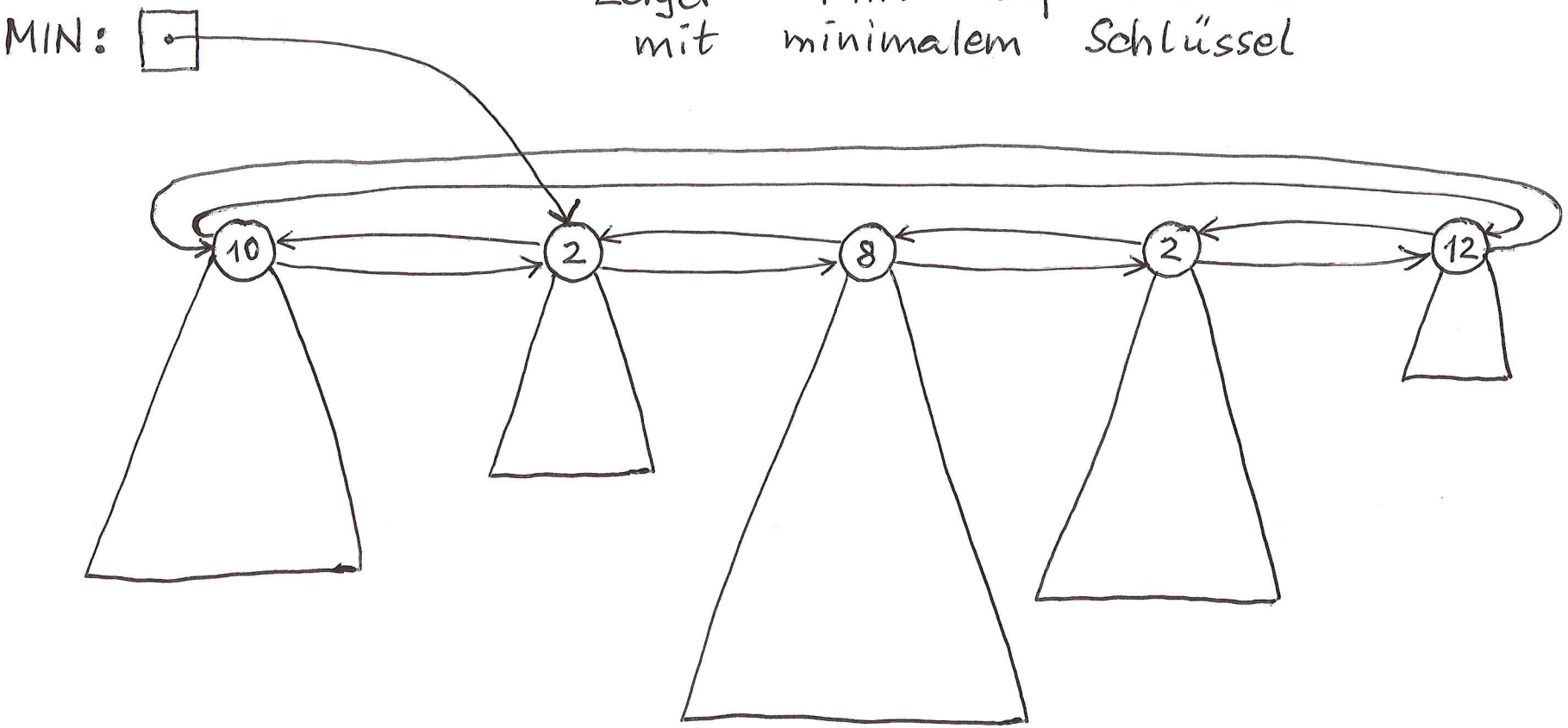
Details im Baum:

Kinder eines Knotens:
zirkulär doppelt verkettete Liste
Zeiger zu einem beliebigen Kind
Zeiger zum Vorgänger

Fibonacci- Heap :

Liste von Wurzeln heapgeordneter Bäume

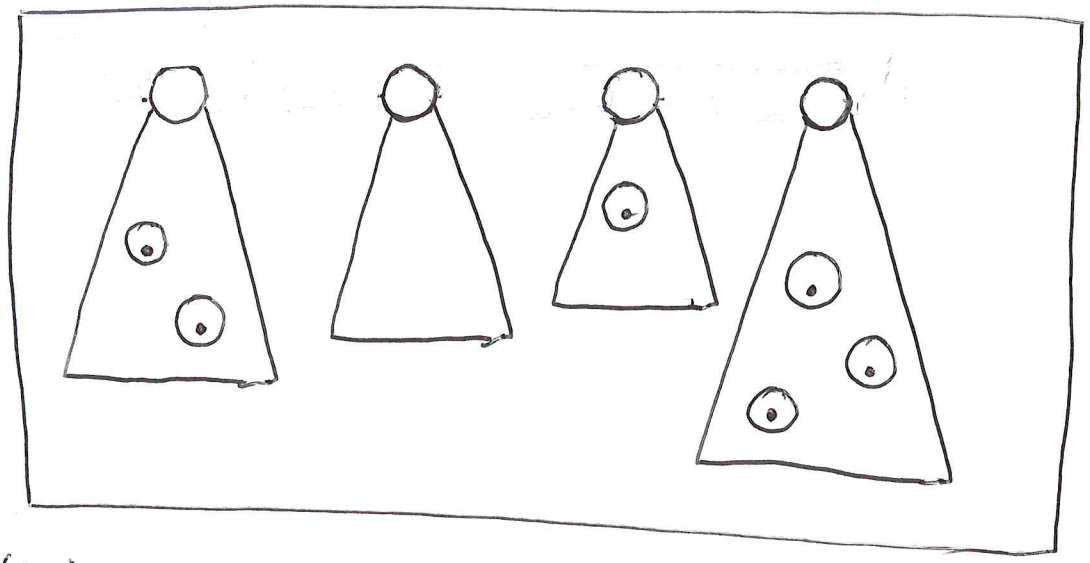
Zeiger MIN auf eine Wurzel mit minimalem Schlüssel



Potenzialfunktion

$\Phi(H) = \#\text{Wurzeln} + 2 \cdot \#(\text{markierte Knoten})$

H:



$\Phi(H) = 4 \text{ (Wurzeln)} + 2 \cdot 6 \text{ (markierte Knoten)}$

$= 16$

new():

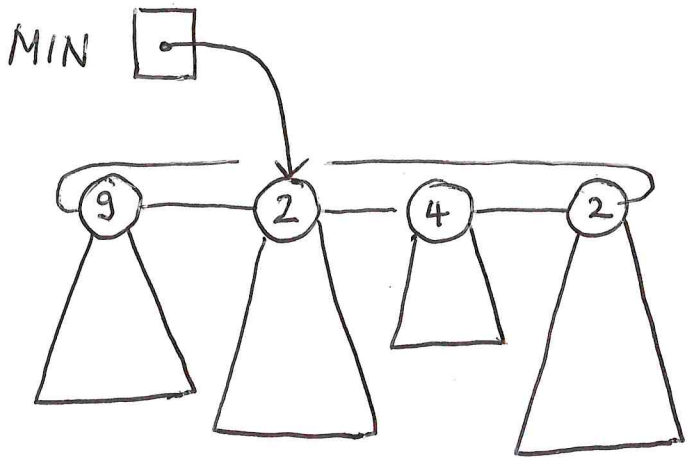
MIN: /

$c_{new} = 1$

$a_{new} = 1$

min():

return
Eintrag in Wurzel,
auf die MIN zeigt

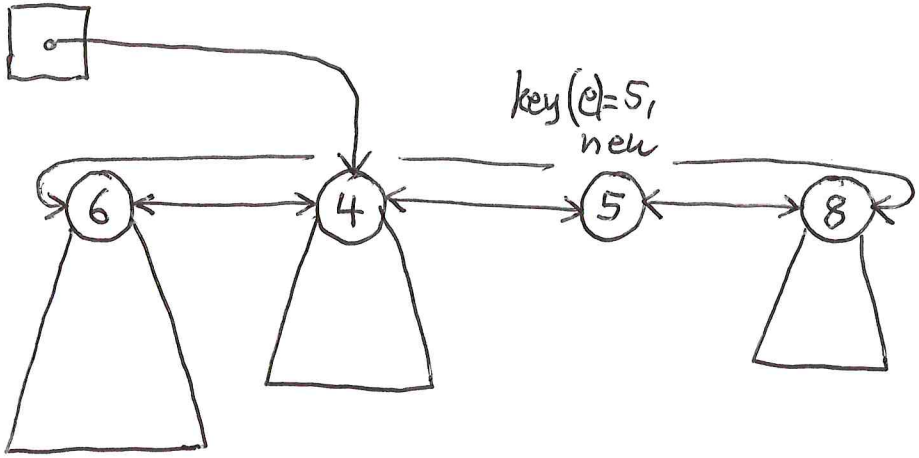


$c_{min} = 1$

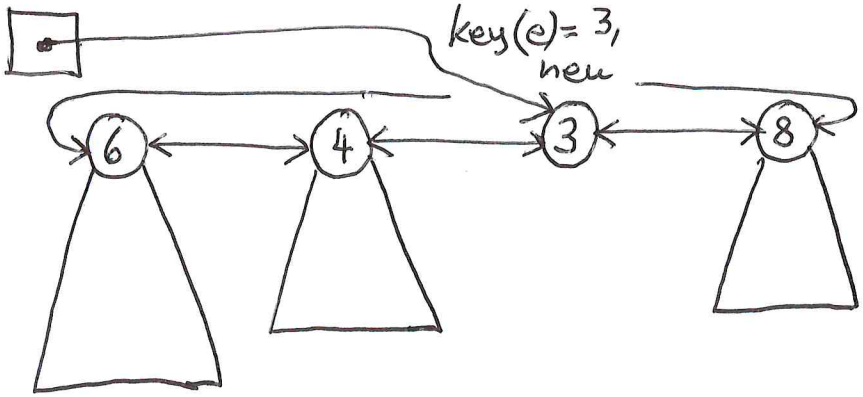
$a_{min} = 1$

insert(e):

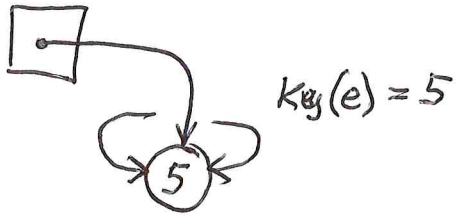
Falls $MIN \neq NIL$:
neuer Wurzelknoten für e
neben MIN-Knoten



Falls key(e) neues Minimum:
MIN-Zeiger umhängen.



Falls $MIN = NIL$:



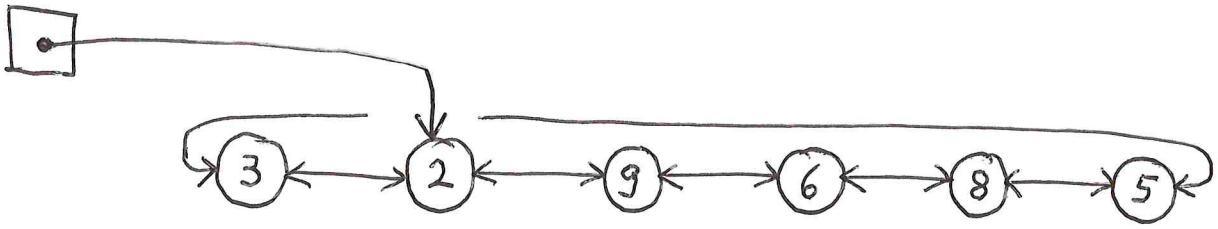
$$c_{insert} = 1$$

$$a_{insert} = 1 + 1$$

↑ 1 neue Wurzel

Heaperzeugung

makeHeap ({3, 2, 5, 8, 6, 9}) : mehrfaches insert

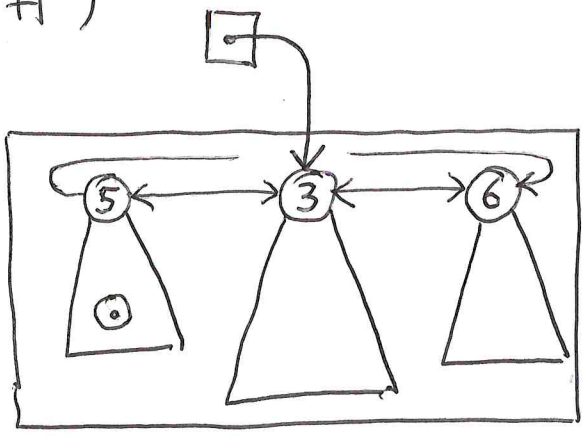


$c_{makeHeap} = n$ (= # Objekte)

$a_{makeHeap} = \underbrace{n}_{\text{echte Kosten}} + \underbrace{n}_{\text{Potenzial neu}} - \underbrace{0}_{\text{Potenzial alt}} = 2n$

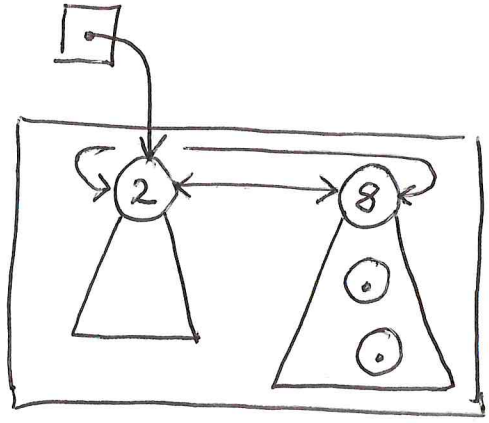
meld (H')

H:



Potenzial: $3+2=5$

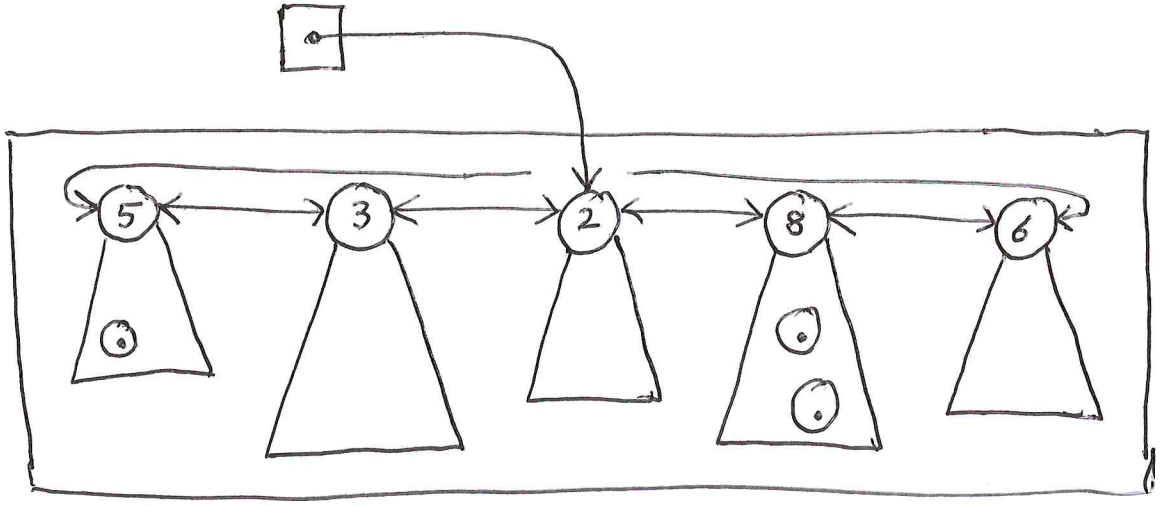
H':



Potenzial: $2+4=6$

Nachher:

H:



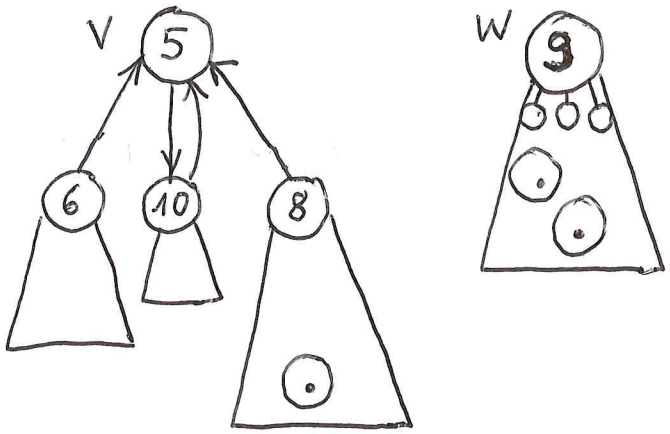
$c_{meld} = 1$

$a_{meld} = 1$

Potenzial: $5 + 2 \cdot 3 = 11$ $(= 5+6)$

join (v, w): Falls key(v) ≤ key(w):

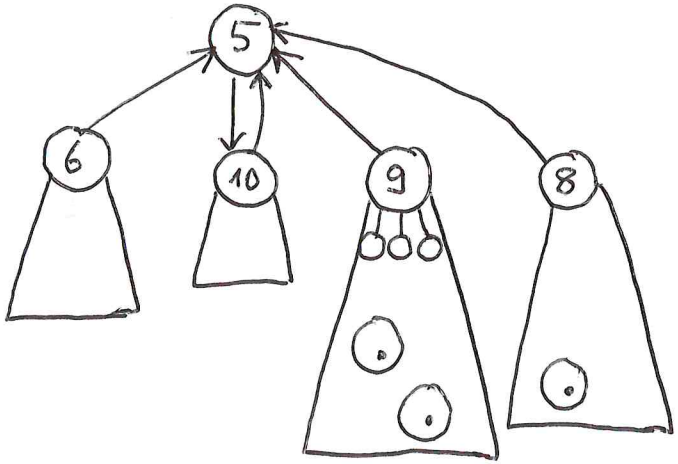
// v, w Wurzeln von gleichem Rang



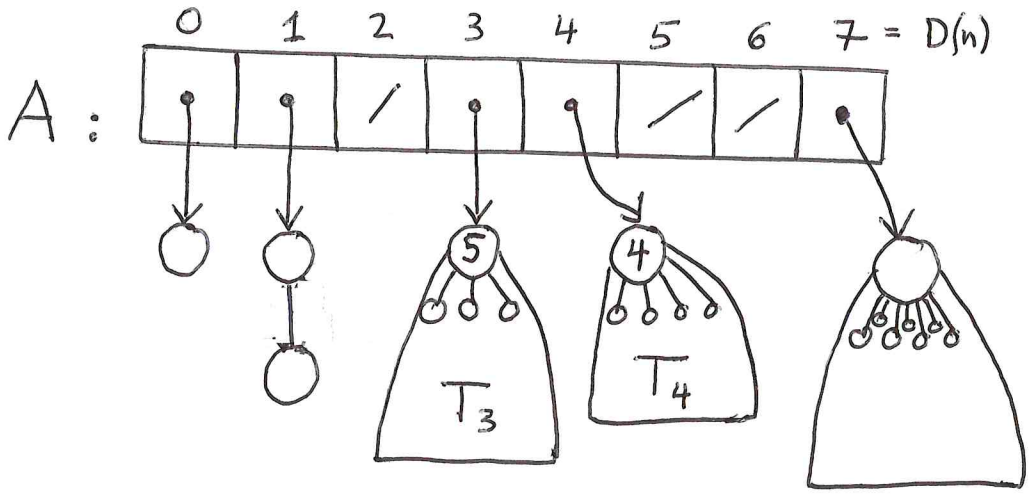
$c_{join} = 1$

$a_{join} = 1 + \underbrace{(-1)}_{\text{eine Wurzel weniger}} = 0$

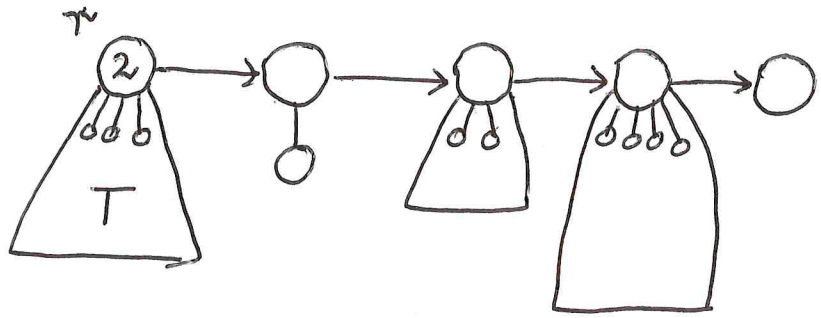
Nachher:



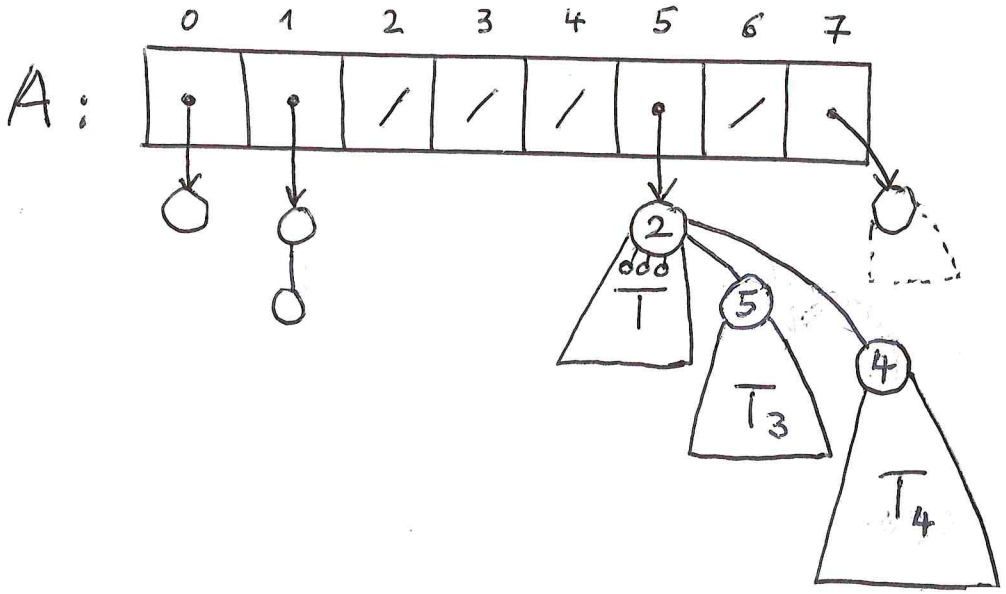
cleanup (L) // L ist Liste von Wurzeln (ohne MIN)



L : // Rest



Nachher:



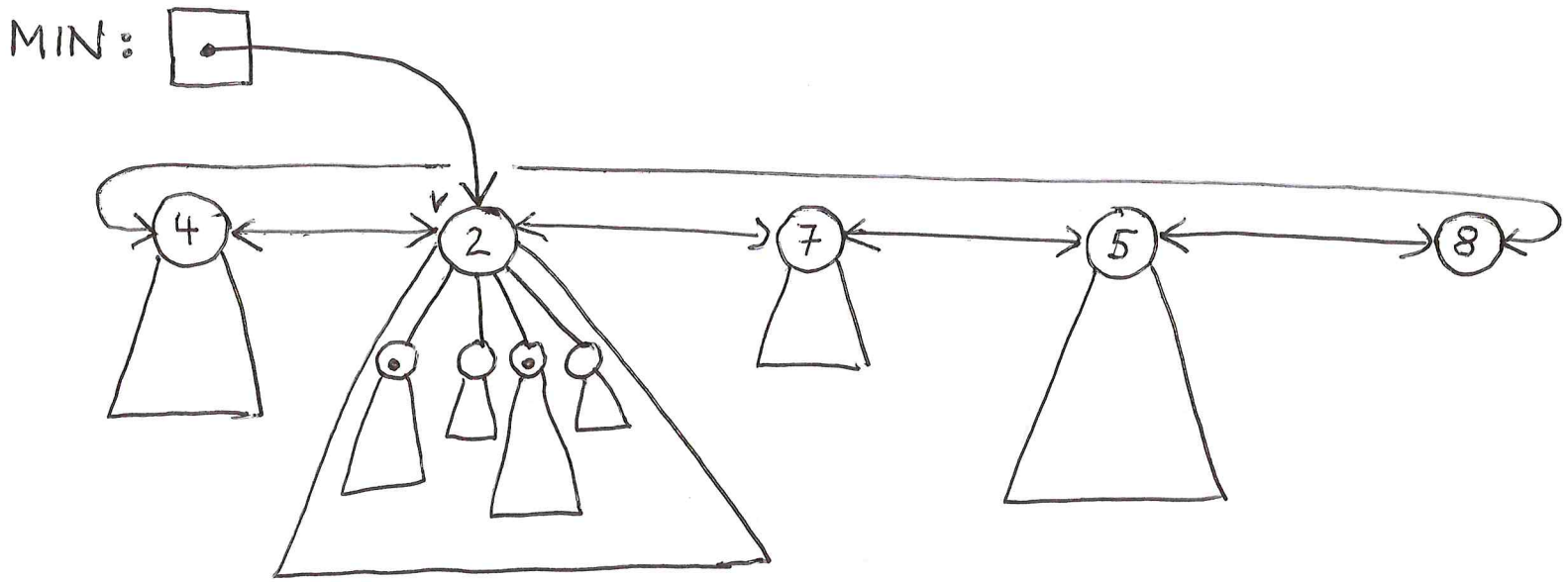
$$c_{cleanup} = D(n) + \ell + 1$$

$$a_{cleanup} = c_{cleanup} - \underbrace{(\ell - \ell')}_{\text{verschwundene Wurzeln}}$$

$$= D(n) + \ell' + 1$$

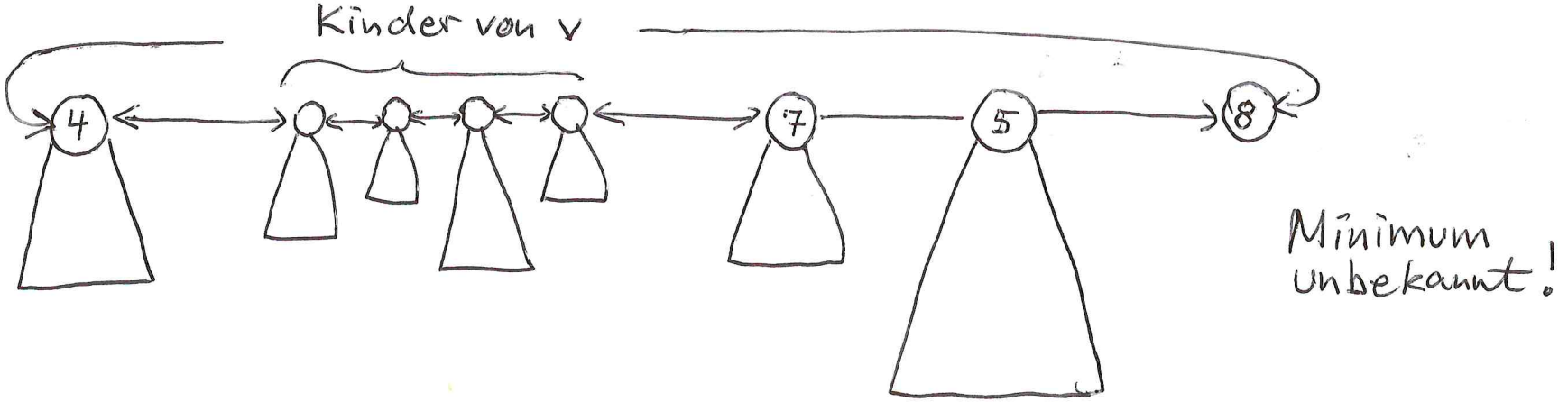
$$\leq 2(D(n) + 1).$$

deleteMin:



kleinste Wurzel v entfernen, Kinder werden unmarkierte Wurzeln

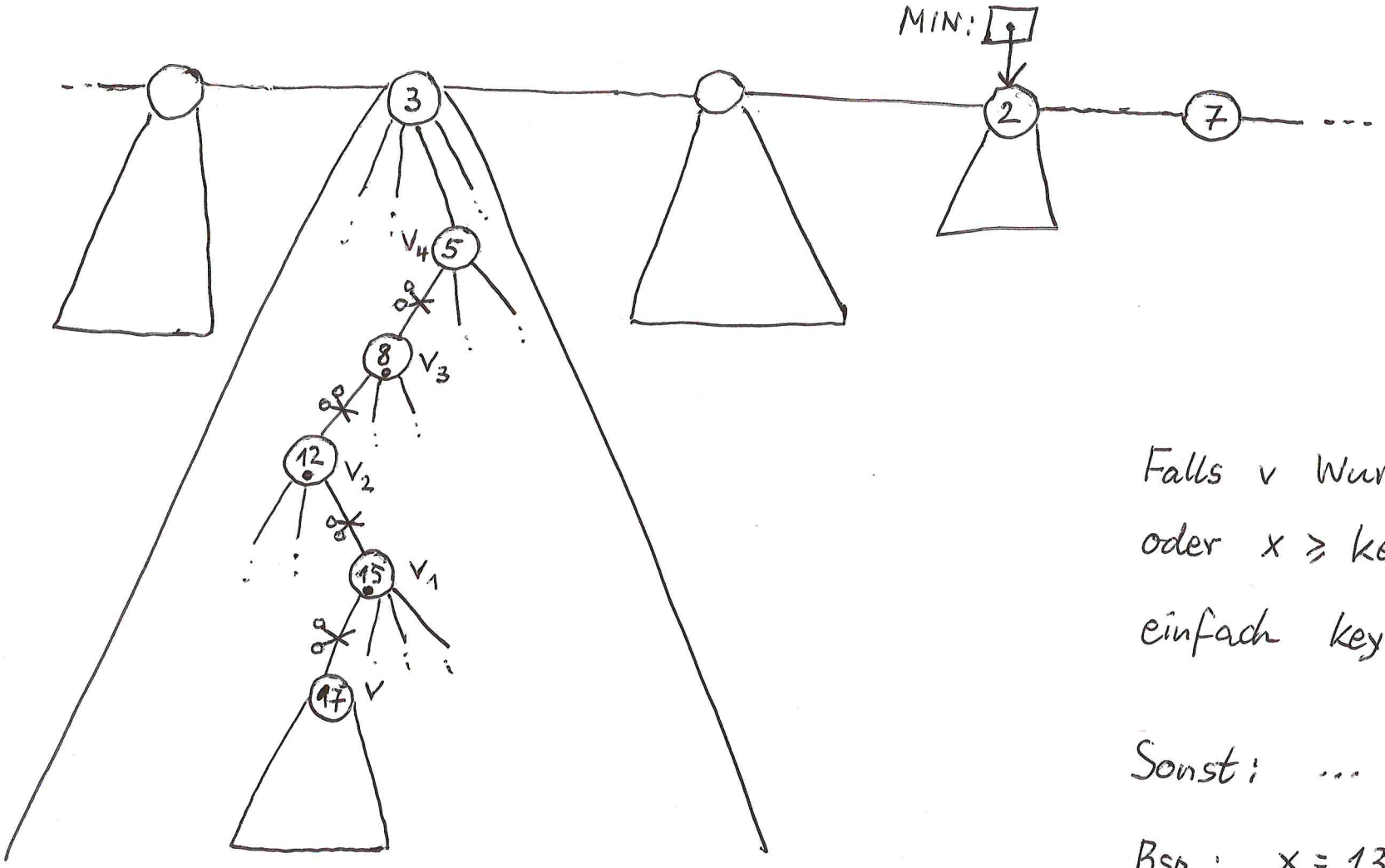
L:



Minimum unbekannt!

Nun: cleanup(L)

decreaseKey(v, x) // v: Knoten, x: Schlüssel



Falls v Wurzel
oder $x \geq key(p(v))$:
einfach key(v) auf x setzen.

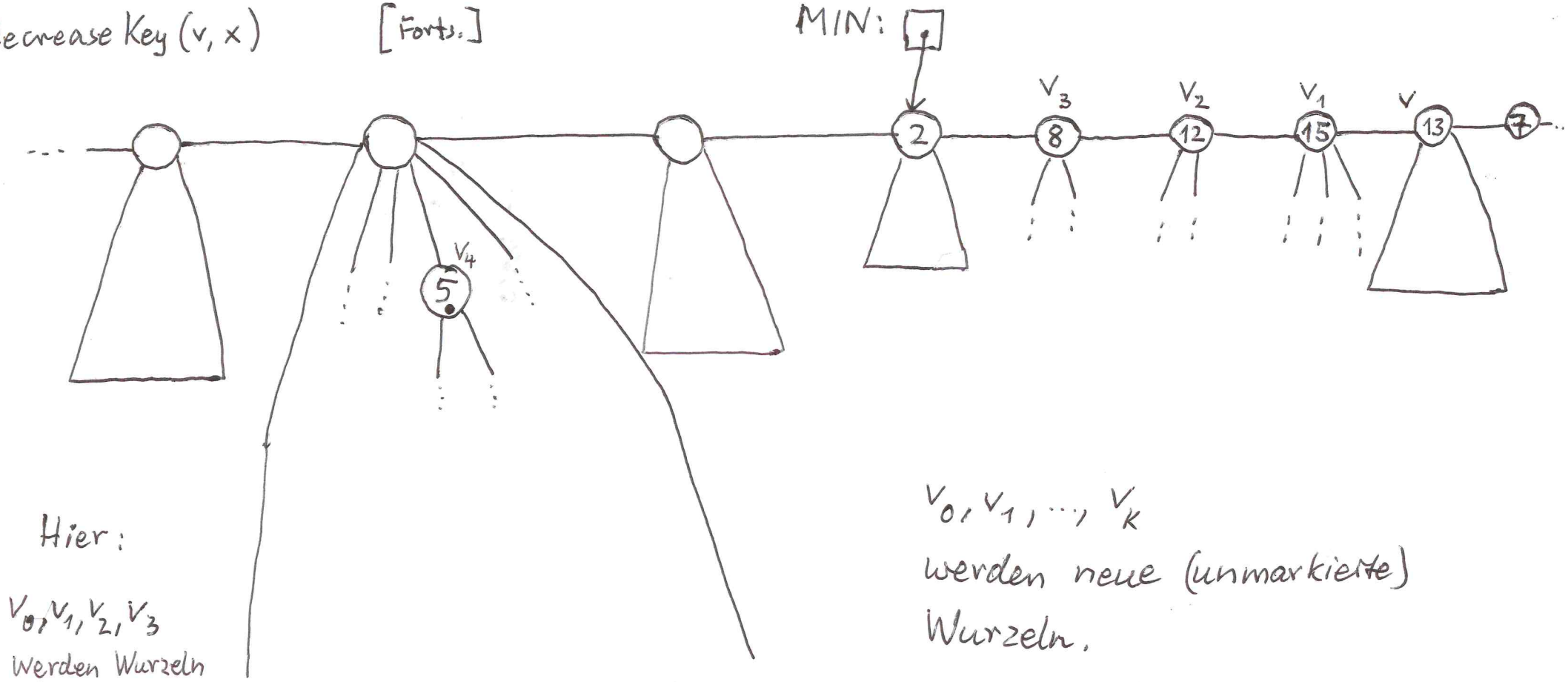
Sonst: ...

Bsp.: $x = 13$

✂: „Schnitt"

$v_0 := v$, $v_i = p(v_{i-1})$ für $i = 1, 2, 3, \ldots$
solange v_{i-1} markiert ist.

decrease Key (v, x) [Forts.]



Hier:

V_0, V_1, V_2, V_3
werden Wurzeln
V_4 wird neu markiert

$V_0, V_1, \ldots, V_k$
werden neue (unmarkierte) Wurzeln.

$V_{k+1} = p(V_k)$, bislang unmarkiert, wird markiert, wenn es keine Wurzel ist.

Echte Kosten: $c_{\text{decreaseKey}}(H, H') = k+1$

Amortisierte Kosten: $a_{\text{decreaseKey}}(H, H') = k+1 + \Phi(H') - \Phi(H)$

$\leq k+1 + \underbrace{k+1}_{\text{neue W.}} + 2 - 2k = 4.$

$+2$: Markierung V_{k+1}; $-2k$: Markierungen $V_1, \ldots, V_k$

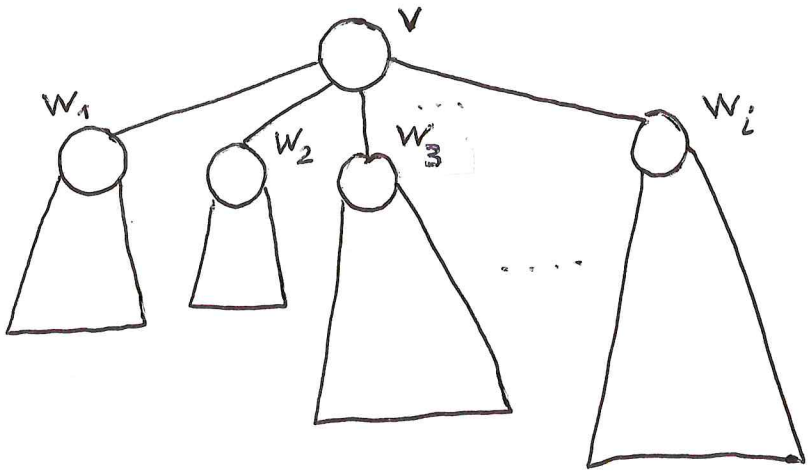
Beweis von Lemma 4.19

T_w = Teilbaum mit Wurzel w

I.A.: T_v hat Tiefe 0: v exakt $1 = F_2$ Knoten.

I.V.: Aussage richtig für alle w, wo T_w weniger tief als T_v.

I. Schritt:



$i = \text{rank}(v) \geq 1$

$w_1, \dots, w_i$: Kinder von v in der Reihenfolge, in der sie per join(v, w_i) oder join(w_i, v) als Kind an v angehängt wurden.

Für $j = 2, \dots, i$: $\text{rank}(w_j) \geqslant \underbrace{(j-1)}_{\text{bei join}(v,w_j)} - \underbrace{1}_{\text{evtl. Kind verloren}} = j-2$

$\overset{\text{I.V.}}{\Rightarrow}$ T_{w_j} hat $\geqslant F_j$ Knoten

$\Rightarrow$ T_v hat $\geqslant \underbrace{1}_{v} + \underbrace{1}_{w_1} + F_2 + \dots + F_i = F_{i+2}$ Knoten.