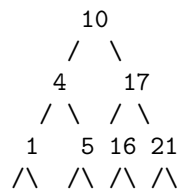




## Uitwerkingen

1. De getallen zijn labels van interne knopen en de bladeren zijn leeg.



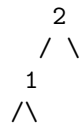
Ja, bijvoorbeeld ook de 'sliert van linksonder naar de wortel 1, 4, 5, 10, 16, 17, 21.

2. Ook in de plaatjes hieronder: de getallen zijn labels van interne knopen en de bladeren zijn leeg.

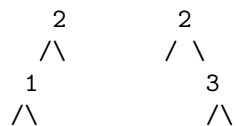
- (a) Ja, een boom met alleen een wortel, bijvoorbeeld:



- (b) Ja: een boom met alleen een wortel, maar bijvoorbeeld ook:

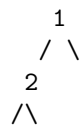


- (c) Nee: bijvoorbeeld de volgende binaire zoekbomen zijn geen min-heap:



De eerste voldoet niet aan de min-heap-eigenschap (maar is wel een complete binaire boom), en de tweede is geen complete binaire boom (maar voldoet wel aan de min-heap-eigenschap).

- (d) Nee: bijvoorbeeld de volgende min-heap is geen binaire zoekboom:

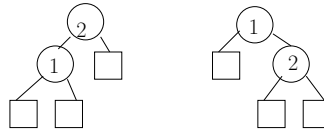




**Algorithm** findBiggest( $T, v$ ):

```
 $w := v$   
while  $T.isInternal(T.rightChild(w))$  do  
   $w := rightChild(w)$   
return  $w$ 
```

7. Een kleinste voorbeeld (met twee interne knopen) waaruit blijkt dat de volgorde waarin we getallen toevoegen aan een initieel lege binaire zoekboom uitmaakt voor het eindresultaat:



8. We gaan een priority queue implementeren met een binaire zoekboom. We bekijken de methoden van het ADT voor priority queues:

- `insertItem( $k, e$ )`: doen we met `insertItem` van de binaire zoekboom.
- `removeMin()`: We passen de procedure `findSmallest` hierboven aan door ook het verwijderen van de knoop  $w$  toe te voegen:

**Algorithm** removeMinBST( $T, v$ ):

```
 $w := v$   
while  $T.isInternal(T.leftChild(e))$  do  
   $w := leftChild(w)$   
 $x := key(w)$   
  removeElement $w$   
return  $x$ 
```

- `minElement()`: met `findSmallest` zoals hierboven.
- `minKey()`: met een aanpassing van `findSmallest`.

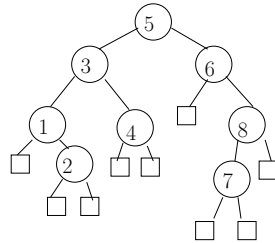
9. Er zijn 14 binairezoekbomen met keys 1, 2, 3, 4. Met 1 op de wortel: 5. Met 4 op de wortel: ook 5. Met 2 op de wortel: 2. Met 3 op de wortel: ook 2.

In het algemeen wordt het aantal binaire bomen met  $n$  knopen gegeven door het Catalan number van  $n$ :

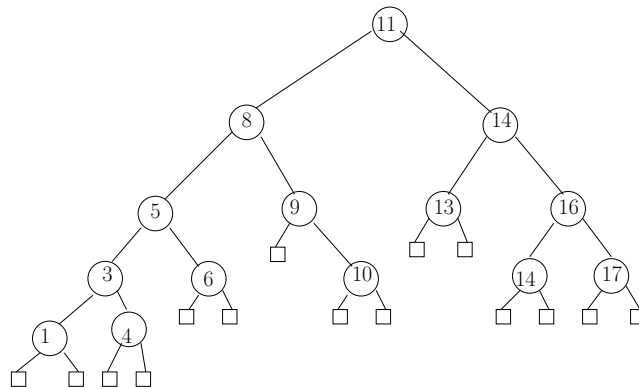
$$C_n = \frac{1}{n+1} \binom{2n}{n}$$

x

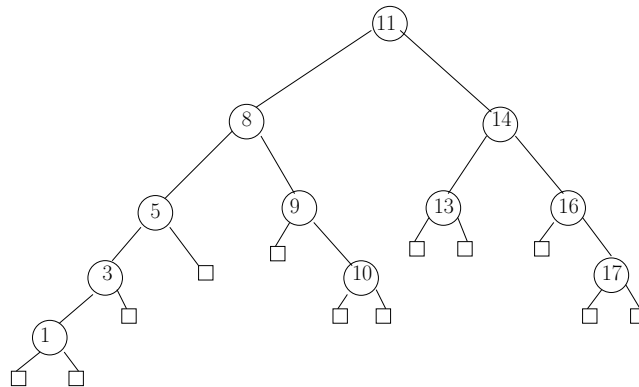
10. Een binaire zoekboom met 5, 6, 3, 8, 7, 4, 1, 2 een voor een toegevoegd:



11. We voegen eerst 4 toe, en vervolgens 14, dit levert de volgende boom:

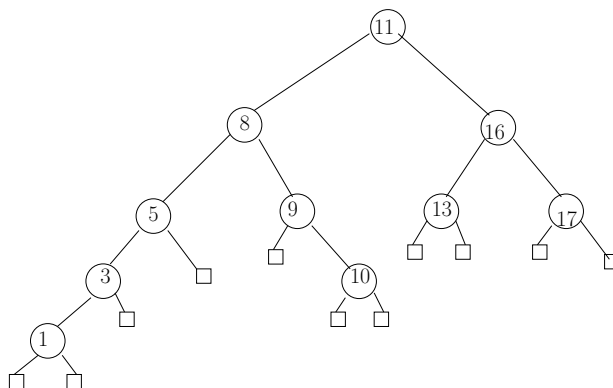


12. De eerste stap is het verwijderen van de knoop met key 6; dat is makkelijk met *removeAboveExternal*:



Vervolgens verwijderen we de knoop met key 14; de twee directe opvolgers zijn alletwee interne knopen, dus we zoeken de knoop met de kleinste key in de rechter subboom. Oftewel, de opvolger van 14 in een inorder traversal.

Dat is in dit geval de knoop met label 16. Die knoop wordt verwijderd met *removeAboveExternal*, en het item met key 16 wordt verplaatst naar de plek waar eerst 14 stond.



13. Inorder traversal is in  $\Theta(n)$ . Voor wat betreft het een-voor-een toevoegen: Best-case als we een complete binaire boom maken, dan

$$2^0 \cdot 1 + 2^1 \cdot 2 + 2^2 \cdot 3 + \dots + 2^h \cdot h$$

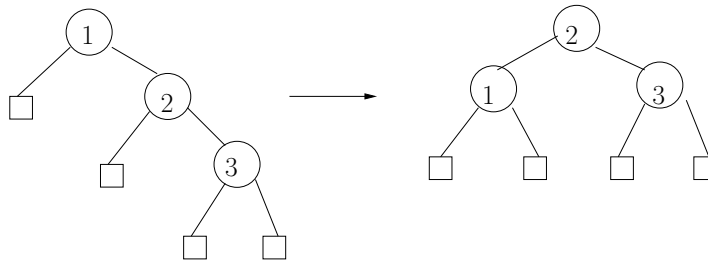
met  $h = \log(n)$ , dus  $\mathcal{O}(n \log(n))$ . Worst-case als we een sliert maken:

$$1 + 2 + \dots + n$$

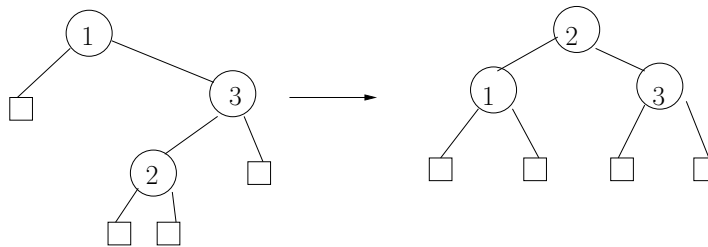
dat is in  $\mathcal{O}(n^2)$ . Dus in het best-case geval totaal  $\mathcal{O}(n \log(n))$ , en worst-case geval  $\mathcal{O}(n^2)$ .

Nog nakijken!

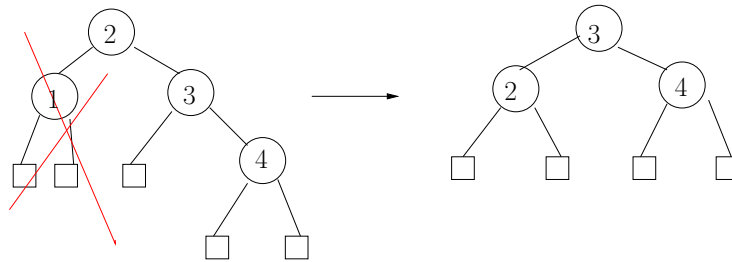
14. Zelfde als voor binaire zoekbomen.
15. Teken de AVL-boom die verkregen wordt door een item met key 52 toe te voegen aan de boom in Fig. 3.15b.  
Toevoegen.
16. Teken de AVL-boom die verkregen wordt door het item met key 62 te verwijderen uit de boom in Fig. 3.15b.  
Toevoegen.
17. Door toevoegen van een knoop met key 3 ontstaat een binaire zoekboom die geen AVL boom is; herbalanceren met een single rotation levert in één stap weer een AVL boom:



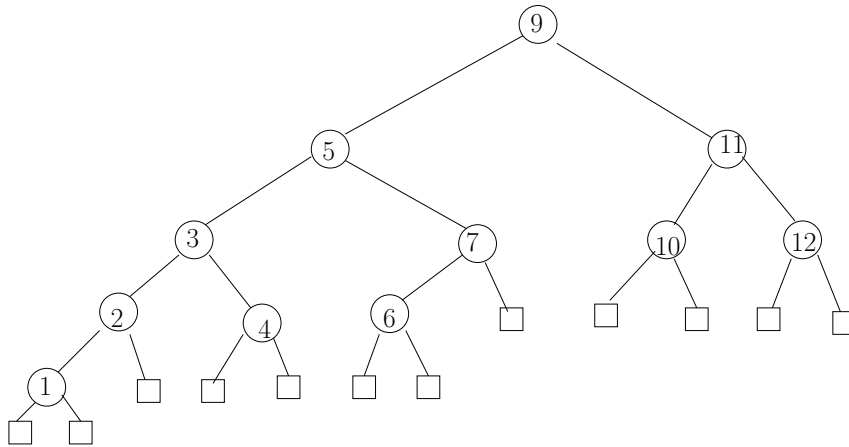
18. Door toevoegen van een knoop met key 2 ontstaat een binaire zoekboom die geen AVL boom is; herbalanceren met een double rotation levert in één stap weer een AVL boom:



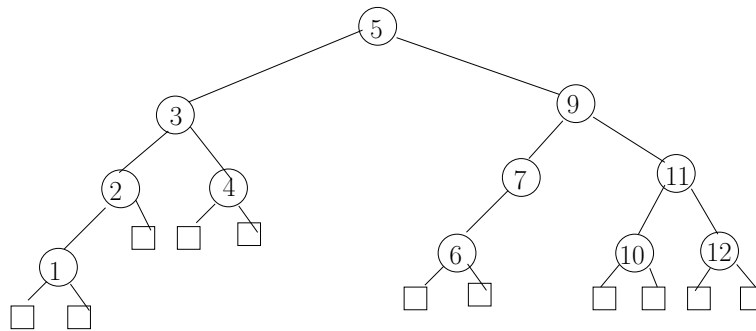
19. Door verwijderen van een knoop met key 1 ontstaat een binaire zoekboom die geen AVL boom is; herbalanceren met een single rotation levert in één stap weer een AVL boom:



20. Eerste stap: key 8 wordt verwijderd en op die plek wordt key 9 gezet; de knoop waar eerst key 9 stond wordt verwijderd.

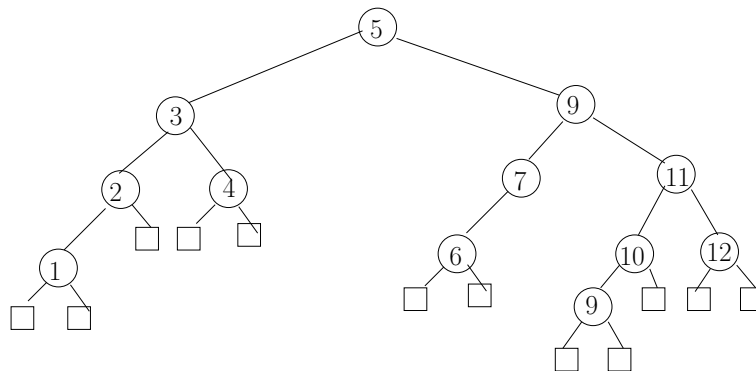


Vervolgens: de boom is uit balans en wordt dus geherbalanseerd.



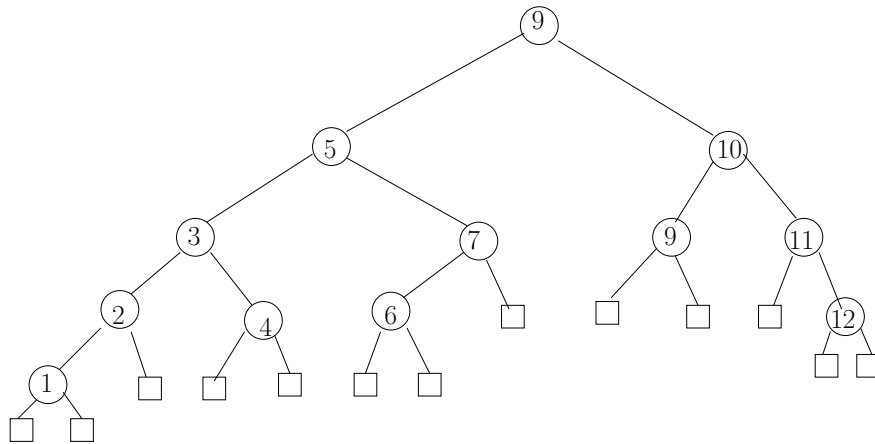
Na deze ene stap is de boom weer in balans.

We voegen nu een knoop met key 9 toe.

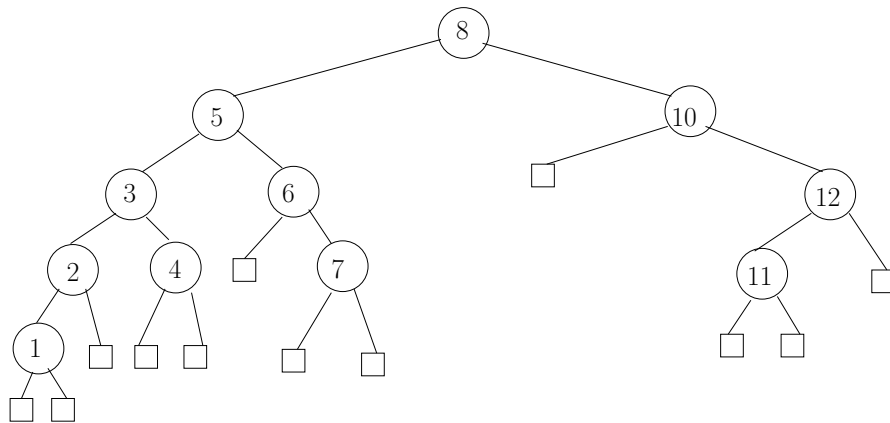


Herbalanseren is niet nodig.

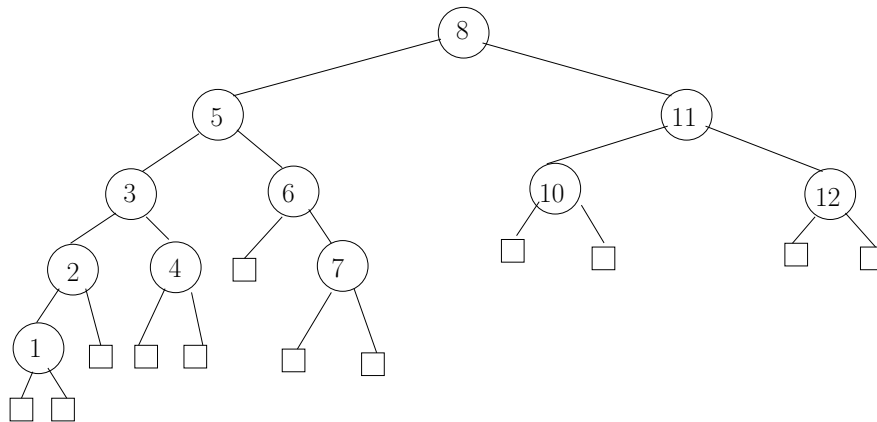
Eerst 9 toevoegen en dan 8 verwijderen leidt tot de volgende AVL boom:



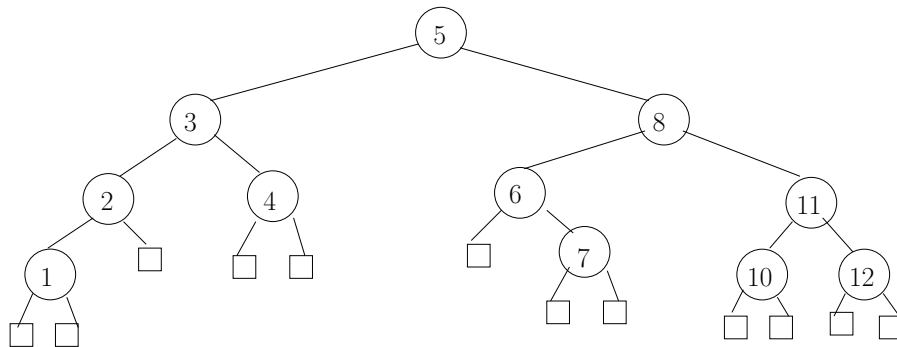
21. We verwijderen de knoop met key 9. Eerste stap levert de volgende boom op:



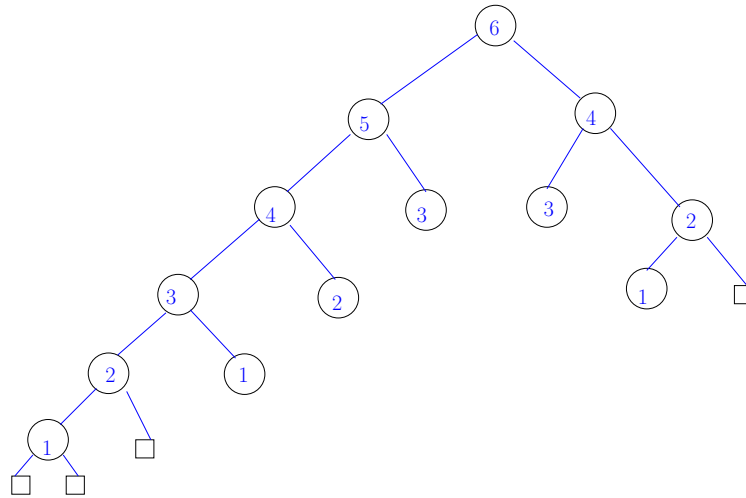
Vervolgens moeten we herbalanseren want de knoop met key 10 is niet in balans:



Omdat nu de knoop met key 8 niet in balans is moeten we nog een keer herbalanseren:

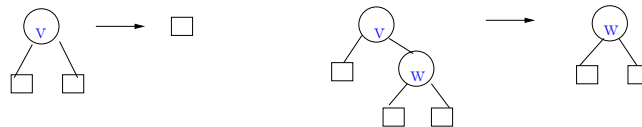


22. We geven voorbeeld waaruit blijkt dat het verschil in diepte tussen twee externe knopen in een AVL boom groter dan 2 kan zijn. In het volgende plaatje staat niet de hele boom getekend. Een vierkantje is een blad; een rondje is een interne knoop, en het getal geeft aan hoe hoog de daar beginnende subboom is. Het verschil in diepte tussen het blad uiterst links en het blad uiterst rechts is 3.

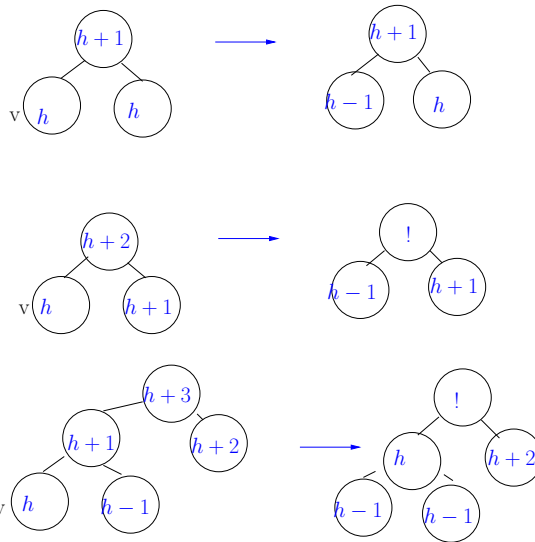


23. We laten zien: door het uitvoeren van de operatie *removeAboveExternal* in een AVL boom raakt ten hoogste één knoop uit balans.

We noemen de knoop die verwijderd wordt  $v$ . De subboom op  $v$  heeft hoogte 1 of hoogte 2, in beide gevallen wordt door het verwijderen van  $v$  de hoogte 1 minder.



We schrijven  $h$  voor de hoogte van de subboom op  $v$  voor verwijderen (dus  $h = 1$  of  $h = 2$ ). We onderscheiden drie gevallen. Als de sibling  $u$  van  $v$  dezelfde hoogte heeft, dan raakt er door verwijderen van  $v$  geen knoop uit balans. Als de sibling  $u$  van  $v$  hoogte  $h + 1$  heeft, dan raakt de voorganger van  $u$  en  $v$  uit balans (maar heeft nog wel dezelfde hoogte, dus erboven raakt niets uit balans.) Als de sibling  $u$  van  $v$  hoogte  $h - 1$  heeft, raakt mogelijk de voorganger van de voorganger van  $u$  en  $v$  uit balans (maar heeft nog wel dezelfde hoogte, dus erboven raakt niets uit balans.) Een plaatje (links-onder is knoop  $v$ ):



24. (\* extra opgave)

Bewijs: Na het toevoegen van een knoop aan een AVL boom is ten hoogste één herbalanceer-stap nodig om de balans te herstellen.

(Dit gaat met het onderscheiden van vier gevallen; in twee gevallen is precies één herbalansering nodig. Dit is geen volledige uitwerking van de opgave.)

25. Toevoegen.

26. We lossen de recurrente betrekking

$$T(n) = \begin{cases} 1 & \text{als } n = 1 \\ 2T(n-1) + 1 & \end{cases}$$

voor de torens van Hanoi op als volgt:

$$\begin{aligned} T(n) &= 2T(n-1) + 1 \\ &= 2(2T(n-2) + 1) + 1 \\ &= 4T(n-2) + 2 + 1 \\ &= 4(2T(n-3) + 1) + 2 + 1 \\ &= 8T(n-3) + 4 + 2 + 1 \\ &= \dots \\ &= 2^i T(n-i) + 2^{i-1} + \dots + 2^0 \end{aligned}$$

Substitueer:  $n = i + 1$  oftewel  $i = n - 1$ . Dan vinden we:

$$\begin{aligned} T(n) &= 2^{n-1} + 2^{n-2} + \dots + 2^0 \\ &= 2^n - 1. \end{aligned}$$