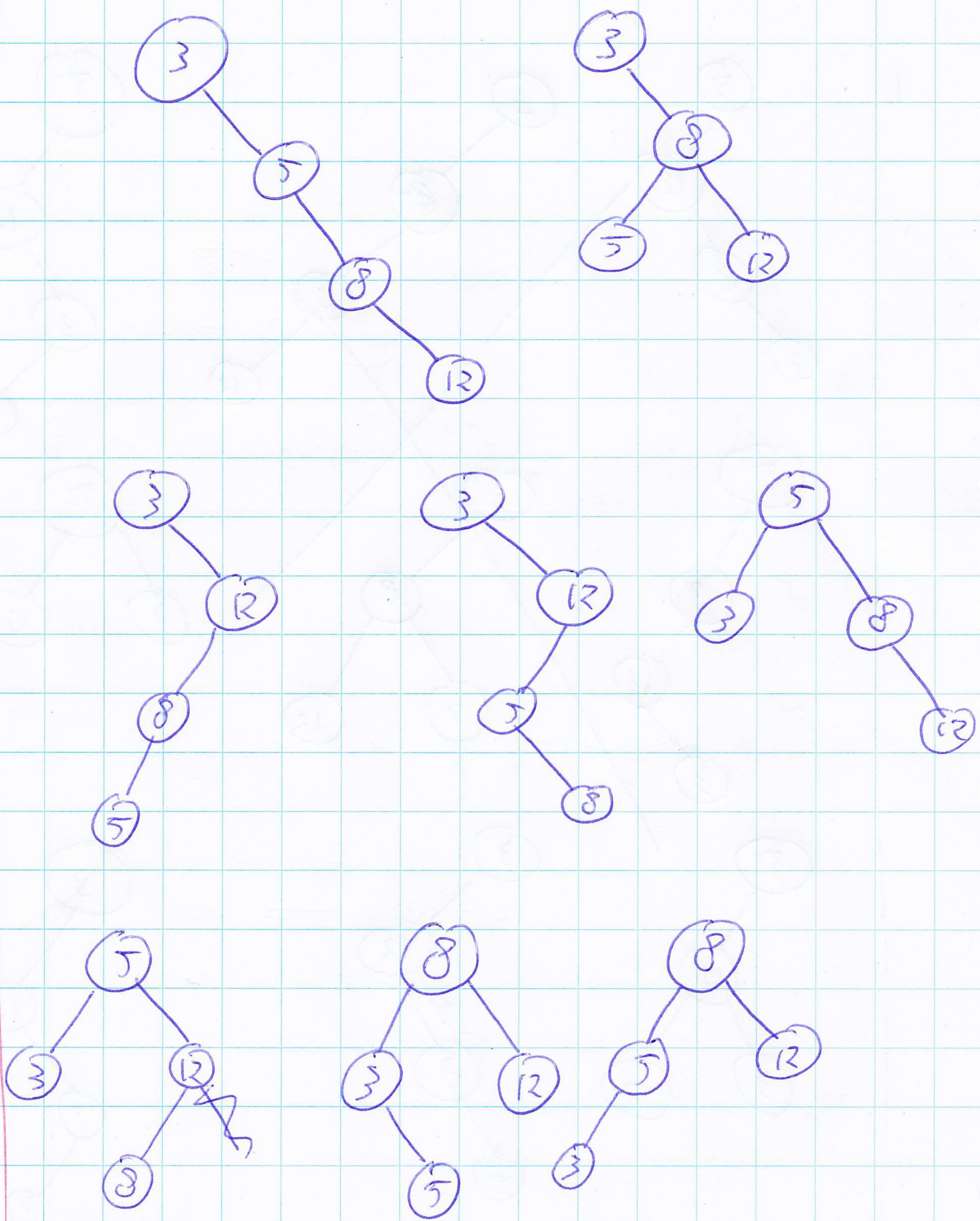
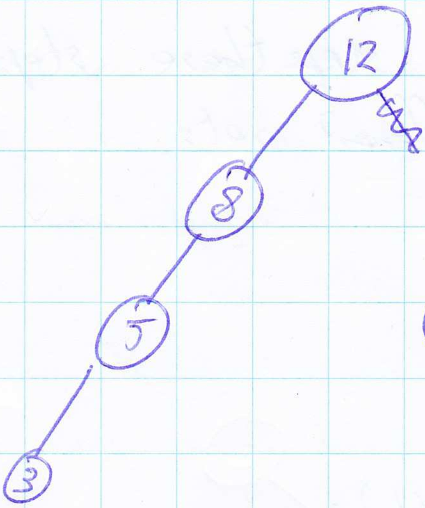
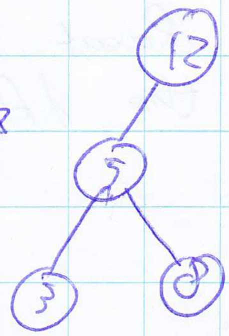
VII

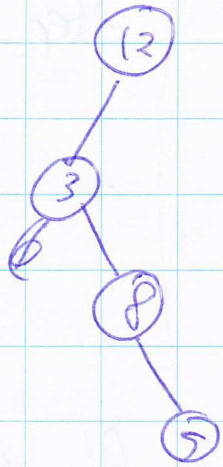
Algoritmen en Datastructuren

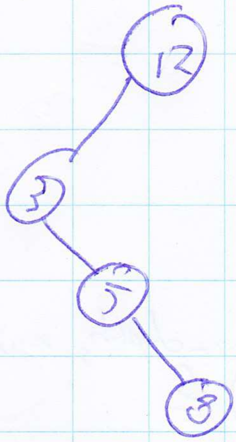
1.











2.

1. When inserting them all after eachother (in order), the next node will always be right of the previous one.

2. Step 1: calculate the middle element of $[1, \ldots, 2^n - 1]$

   ↳ $\frac{2^n - 1}{2}$

   take this element as root node

   Step 2: Split the set of elements into two sets with

   set 1 : $\{ x \mid x < \frac{2^n - 1}{2} \}$

   set 2 : $\{ x \mid x > \frac{2^n - 1}{2} \}$

Step 3 : Repeat ~~step~~ these steps with
the different sets.

3) Algorithm (x, depth) :

max_depth ← 0
min_depth ← 0

find_max_and_min (x, 0, max_depth, min_depth)
if (|max_depth - min_depth| <= 1) {
    return true
else return false

pass by reference

find_max_and_min (x, depth, &max_depth, &min_depth)

if (x != NIL) {
    depth ← depth + 1
    if (max_depth < depth) {
        max_depth ← depth
    find_max_and_min (x.left, depth, max_depth, min_depth)
    find_max_and_min (x.right, depth, max_depth, min_depth)
~~else{~~
else {
    if (min_depth > depth) {
        min_depth ← depth
    }}

This runs in $O(n)$ because each node in the binary tree will be checked in find_max_and_min(...)

4

1. $2^{h+1} - 1 = n$

2.

$2^{m} = n \longrightarrow \log_2(n) = m$

$$nodes = 2^{height+1} - 1$$

$$nodes + 1 = 2^{height+1}$$

$$\downarrow$$

$$height + 1 = \log_2(nodes + 1)$$

$$height = \underbrace{\lceil \log_2(nodes + 1) \rceil}_{} - 1$$

↳ naar boven afronden want height is een geheel getal

3.

5. The root node has no parent, so this is one NIL pointer. For every extra node, this node will delete one NIL pointer and add two, this means each extra added node will create one extra NIL pointer. By definition of a binary tree, each ~~ex~~ next layer can house n+1 nodes, this is where the n+1 comes from.

6.