Problem 1.

Honors Data Structures

Theoretical homework 2

a) Let's create a table to see where our general hunch for solving this problem will come from:

k	L
0	1
1	2
2	4
3	4
4	8
5	8
6	8
7	8
8	16
9	16
10	16
11	16
12	16
...	...

From the table to the left we can see easily that L doubles every time when k is in the form $k = 2^x$ for some const. $x \in \mathbb{N}_0$. From here we can easily generalize this observation by induction:

1) Base case: $k = 1$, $k = 2$.

2) Hypothesis: Every time $k = 2^x$ for some natural number x (or $x = 0$) L doubles.

3) Let's assume that the hypothesis is true for $1 \le k \le 2^s$ for some natural number s.

4) Proof: If we had doubled L the last time when $k = 2^s$, then L will not need to double for $k \in [2^s; 2^{s+1} - 1]$ because $L = 2 \times k = 2 \times 2^s = 2^{s+1}$. => If k becomes 2^{s+1}, then L needs to double. => Induction is prooved!

Now, let's assume that there is such a constant m for which $2^m \le n < 2^{m+1}$. It's trivial why we can always find such a number (i.e. n will always be between something and something bigger).

Now, for n pushes, we would need n amount of cost only for the pushing. We would need an additional $1+2+4+\dots+2^m$ amount of cost for the copying.

Let's say that $1+2+4+8+\dots+2^m = a$

$\Rightarrow 2a = 2+4+8+\dots+2^m+2^{m+1} = 2(1+2+4+\dots+2^m)$

$\Rightarrow 2a - a = (2+4+8+\dots+2^{m+1}) - (1+2+4+\dots+2^m) =$

$= \cancel{2}+\cancel{4}+\cancel{8}+\cancel{16}+\dots+\cancel{} +2^{m+1} -1 - \cancel{2} - \cancel{4} - \cancel{8} - \cancel{16} - \dots - \cancel{2^m} =$

$= \boxed{2^{m+1}-1} \Rightarrow \boxed{a = 2^{m+1}-1}$ $\Rightarrow$ Our total cost would be: $a+n = 2^{m+1}-1+n < 2^{m+1}+n$

However, we know that $2^m \leq n < 2^{m+1}$

$n \geq 2^m \Leftrightarrow 2n \geq 2^{m+1} \Rightarrow 2n+n \geq 2^{m+1}+n$

$\Rightarrow 3n \geq n+2^{m+1} > n+2^{m+1}-1 = a+n$

$\Rightarrow 3n \geq$ total cost for n pushes!

b) We can use the Banker's method to prove what we want. If we "take a loan" of 3 coins every time we push, we can see that we'll always have ≥ 0 coins in the bank. That's because we are paying 1 coin for every

Problem 1 continue:

push operation & keeping 2 coins when we don't don't have to allocate more memory and we're paying this + (2k) coins when we do have to allocate more memory because of the previous doublings and this one. This follows from the induction from part 1a) since we know that if $n \in [2^m; 2^{m+1})$ we'll need $1+2+4+\ldots+2^m = 2^{m+1}-1$ allocations cost $\approx 2^{m+1}$ allocations cost. Thus, when we reach k, we'll need (2k) coins to cover the allocations up to now. However, for the k pushes we've "saved" (2k) coins because for each of the k pushes we "save" 2 coins. Thus, at the worst case, i.e. when we pay the most we'll be left with at least $2k - 2k = 0$ coins. => We'll always have non-negative amount of coins in the bank

c) If we increase the size by a constant c, then we won't have a linear solution because: ~~assume~~ we'll have to do $\left\lfloor \frac{n}{c} \right\rfloor$ doublings and since c is just a constant we know that $O\left(\left\lfloor \frac{n}{c} \right\rfloor\right) = n$. Since we also have n pushes, then the complexity will not be linear. Let's assume that ~~...~~ $d \in \mathbb{N}, d \in [0; c)$ and $(n-d) \vdots c$

Problem 2.

We're given a full binary tree of N nodes.

In order for a tree to be full, we need every node:

I: either to be leaf;

II: or to have exactly 2 children

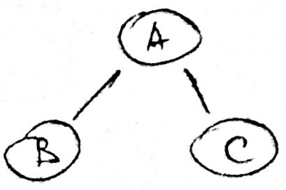
Now, let's look at the "smallest" possible full binary trees:

1) If we have a tree with only 1 node: A

This tree satisfies the condition since it has 1 node and this node is a leaf. => For this case: $N=1$.

$\frac{N-1}{2} = \frac{1-1}{2} = \frac{0}{2} = 0$; Since our single node is a leaf node, it's true that we have 0 internal nodes.

2) If we have a tree with only 3 nodes -> 1 root node and 2 leaf nodes



=> Here we have 2 leaf nodes and 1 internal node for this tree: $N=3$; $\frac{N-1}{2} = \frac{3-1}{2} = \frac{2}{2} = 1$

=> The condition is satisfied here too.

~~Hence~~ We can use the above 2 cases as a base case for our proof by induction!

$\Rightarrow$ Let $n-d = \cancel{bb}\ cb$ $\Rightarrow n = bc + d$

Every c we have to double:

$$\Rightarrow c + 2c + 3c + \dots + bc =$$

$$= c(1+2+\dots+b) = c\,\frac{b(b+1)}{2} =$$

$$= \left(\frac{b+1}{2}\right) bc = \left(\frac{b+1}{2}\right)(n-d)$$

However: $n = bc + d \Rightarrow \boxed{\frac{n-d}{c} = b}$

$$\Rightarrow \frac{b+1}{2} = \left(\frac{\frac{n-d}{c}+1}{2}\right)(n-d) = \left(\frac{n-d+c}{2c}\right)(n-d)$$

In this answer we have to include n pushes

$\Rightarrow$ The total cost would be now $\left(\frac{n-d}{2c} + \frac{1}{2}\right)(n-d) + n =$

$$= \frac{(n-d)^2}{2c} + \frac{n-d}{2} + n = \frac{(n-d)^2}{2c} + \frac{3n-d}{2}$$

Since c, d are constants, we know that:

$O\left(\frac{(n-d)^2}{2c} + \frac{3n}{2} - \frac{d}{2}\right) = O(n^2)$ since n^2 is the highest power $\Rightarrow$ Not linear!

Statement: A full binary tree of N nodes has $\left(\frac{N-1}{2}\right)$ internal nodes.

Base case: $N=1$ and $N=3$ shown above

Inductive step:

1) Hypothesis: Assume the statement holds for all <u>odd</u> integers $1 \leq N \leq 2k+1$ for $k \in \mathbb{N}_0$

2) We need to show that it works for $\boxed{2k+3}$

Since we know from the hypothesis that the statement holds for $N = 2k+1$, let's start with a tree that has $N = 2k+1$ nodes and insert 2 more.

For $N = 2k+1$ we have $\left(\frac{N-1}{2}\right) = \frac{2k+1}{2} = k$ internal nodes.

We want to proove that for a tree with $N = 2k+3$ nodes, we would have $\left(\frac{N-1}{2}\right) = \frac{2k+3-1}{2} = \frac{2k+2}{2} = (k+1)$ internal nodes.

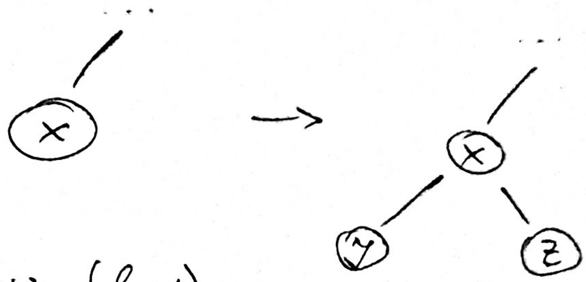
~~The~~ Starting with a tree with $(2k+1)$ nodes we ~~can~~ ~~add~~ must add 2 nodes to reach $(2k+3)$ nodes.

We can add 2 nodes either by:

1) Adding 1 node to 2 leaf nodes

2) Adding 2 nodes to the same leaf node

However, the 1^{st} case will produce a tree that's not full. => Only the 2^{nd} case can happen. The second case will produce 2 new leaf nodes and 1 new internal node:



=> Our internal nodes are now $(k+1)$ and our total number of nodes is $(2k+1)+2=\boxed{2k+3}$ => The induction statement holds for $N=2k+3$ too! => Statement holds for any odd, natural N!

Note: we did not assess the case where we insert nodes internally, rather than on the leaf nodes. However, since we used strong induction we can safely assume WLOG that all tree configurations which have $N \leq 2k+1$ nodes satisfy the condition. And since we're choosing a randomly selected leaf node to "attach" the 2 new nodes to, this assumption is plausible.

Note 2: We proved the induction only for odd N. However, if a tree has an even number of nodes, then it cannot be full. We can prove the claim that for every odd N there exists at least 1 full binary tree. This can be done by simple induction:

1) Base: $N = 1$: A -> works since that's a leaf node

2) Hypothesis: Assume that there exists at least 1 full binary tree for all odd integers $1 \leq N \leq 2k+1$

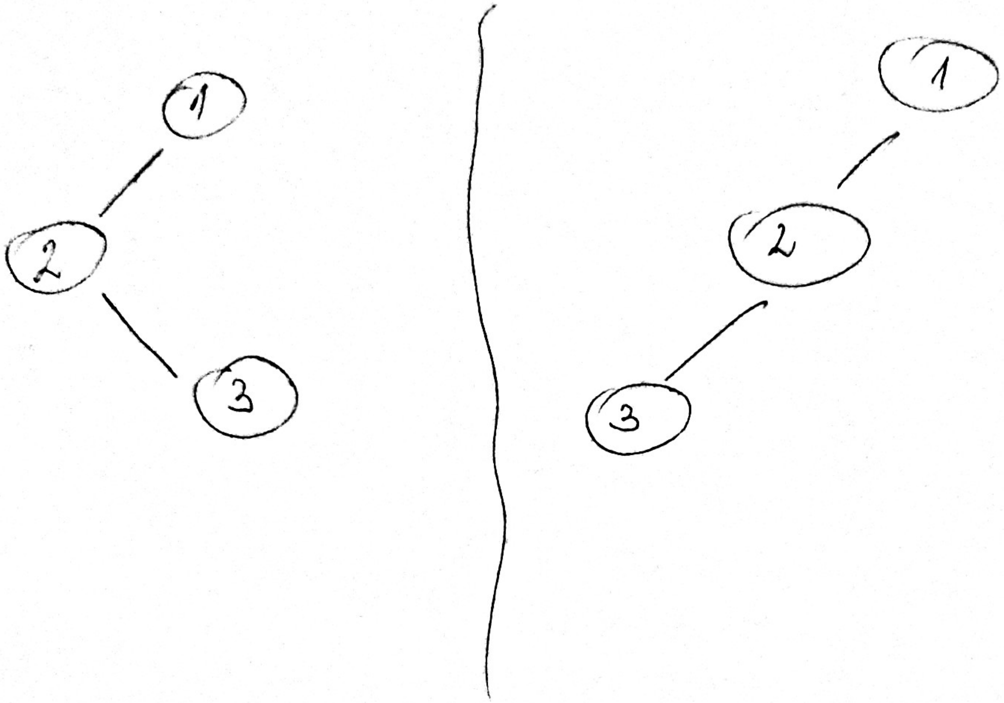
3) We need to show that there exists at least 1 full binary tree with $N = 2k+3$ nodes. Well, that's easy. Just add a pair of nodes to a random leaf and we're done! => Induction holds.

However, if we try to take any tree with odd number of nodes and add a single node to any of its leaves, the tree will <u>not</u> be full anymore

=> Only odd N trees can be full, thus completing our analysis for all $N \in \mathbb{N}$.

Problem 3.

We're going to provide a proof by contradiction:



For both trees we can see that:

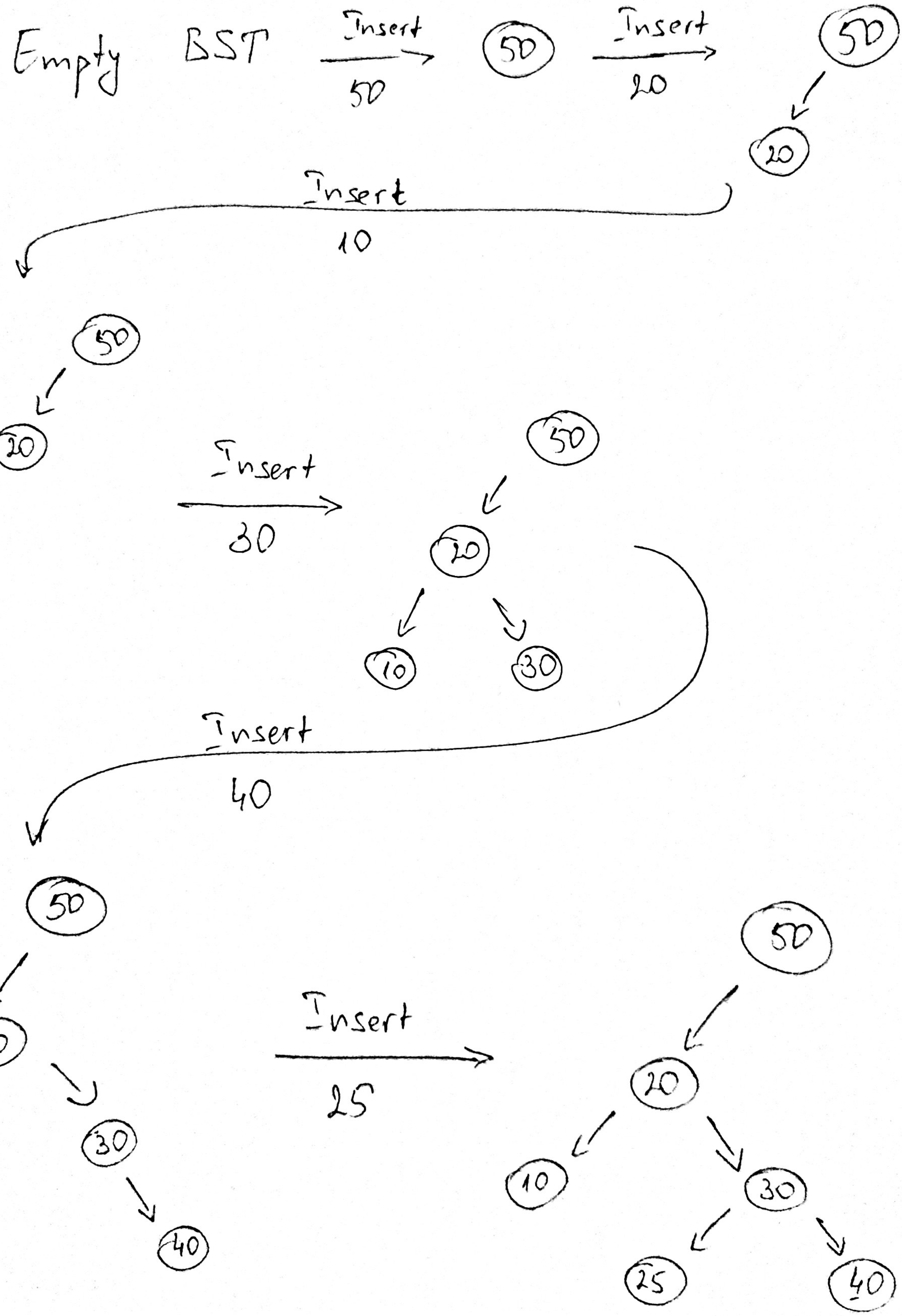
1) Preorder: 1 2 3

2) Postorder: 3 2 1

=> Contradiction!

Problem 4.

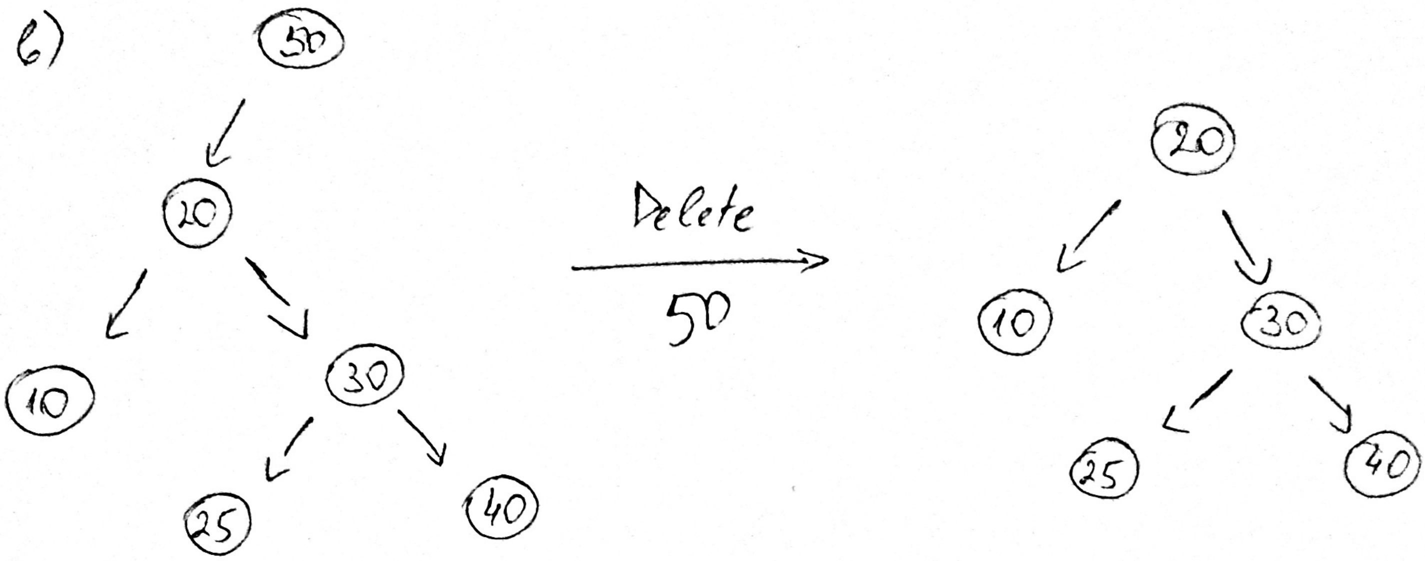
a)



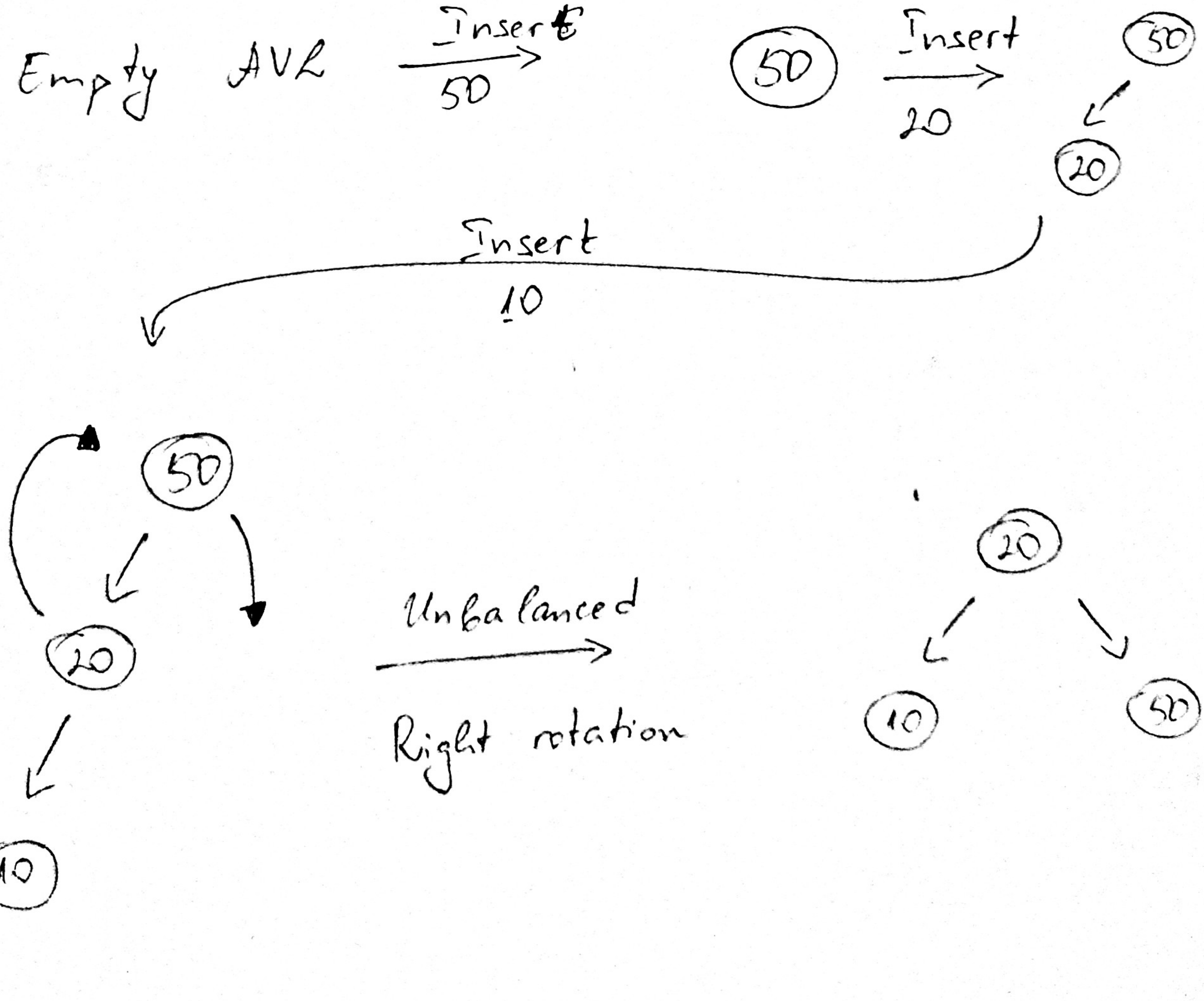
Empty BST
Insert 50
50
Insert 20
50
20
Insert 10
50
20
10
Insert 30
50
20
10
30
Insert 40
50
20
10
30
40
Insert 25
50
20
10
30
25
40

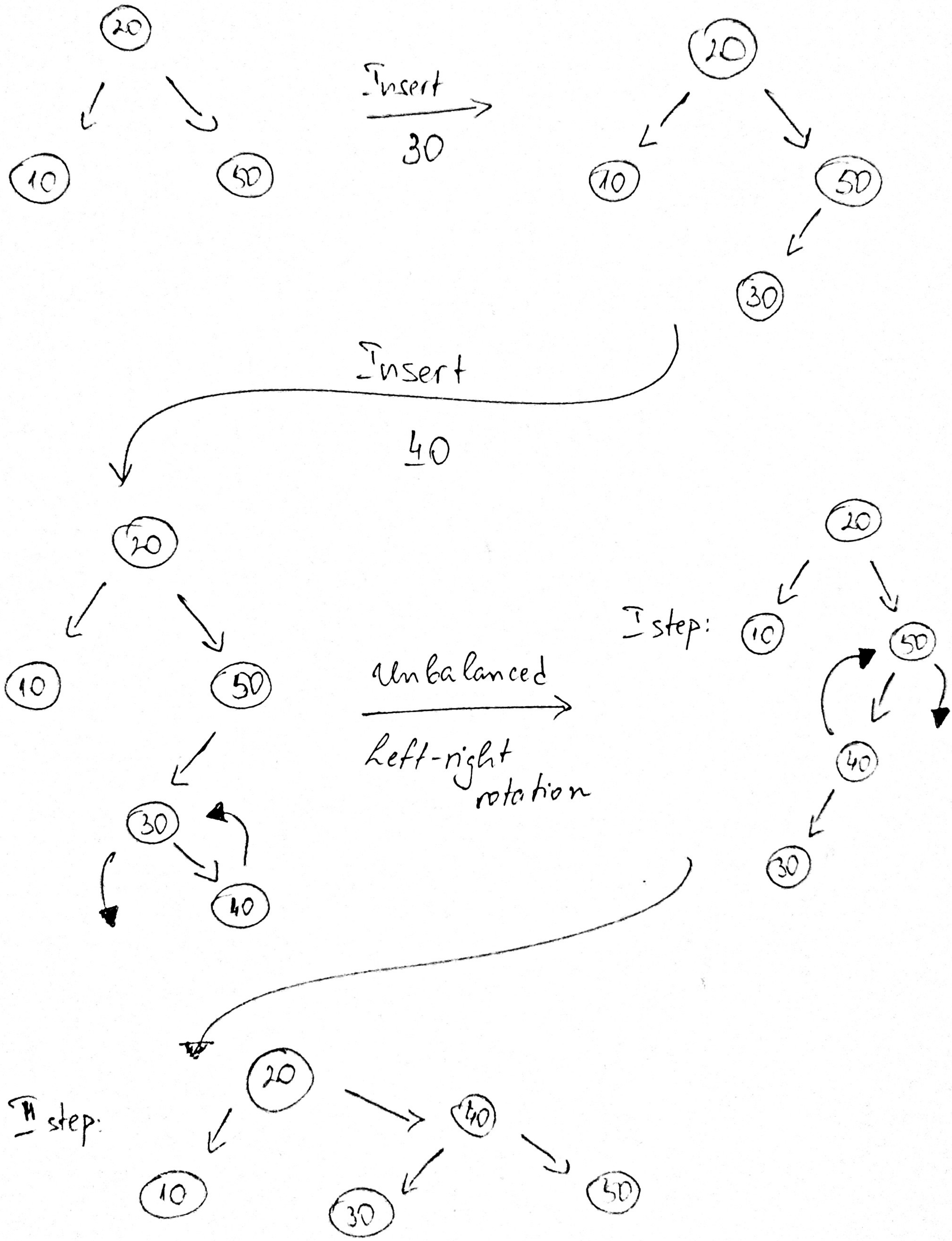
Problem 4:

b)

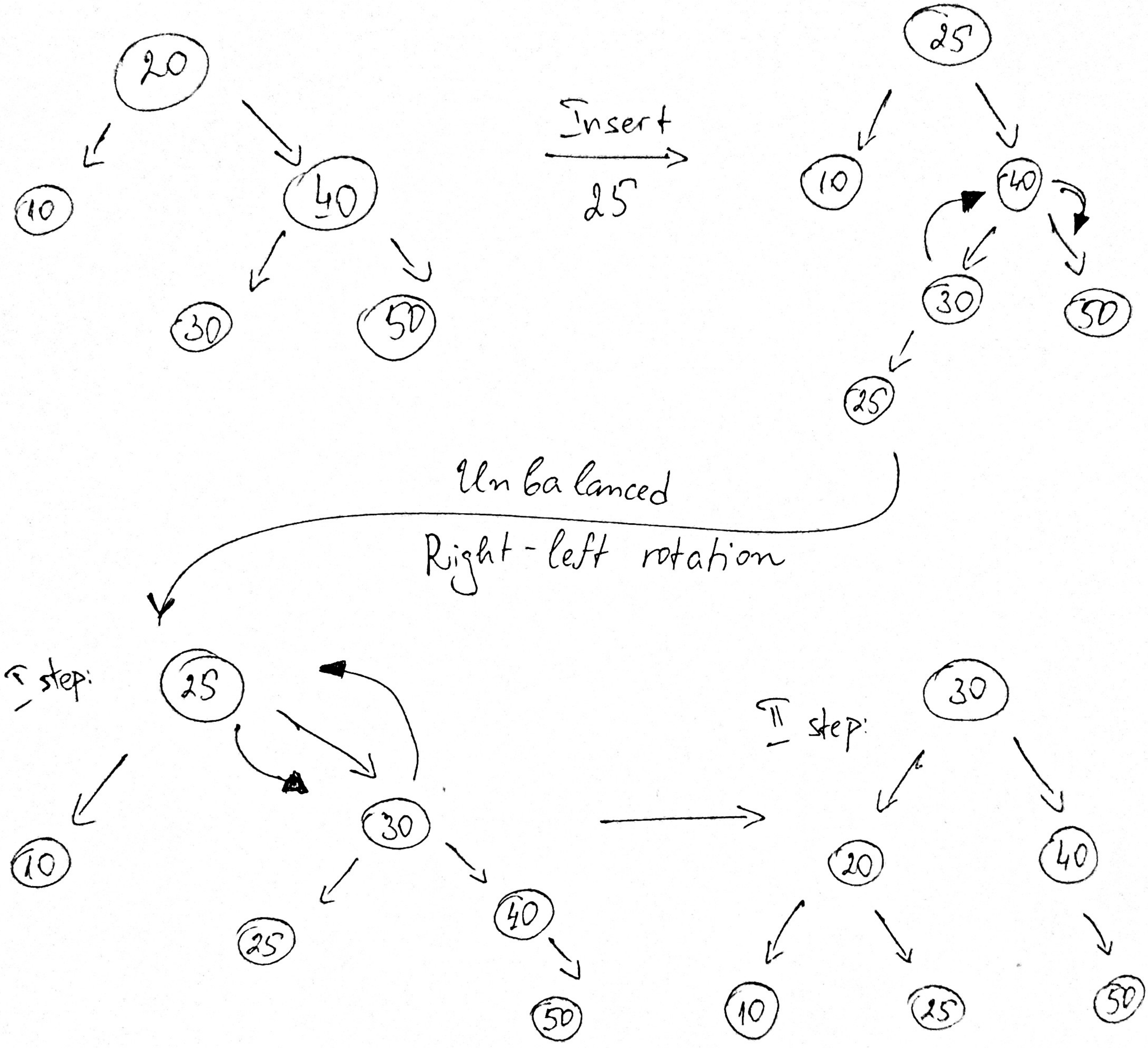

50
20
10
30
25
40
Delete
50
20
10
30
25
40

c)


Empty AVL
Insert
50
50
Insert
20
50
20
Insert
10
50
20
10
Unbalanced
Right rotation
20
10
50


20
10
50
Insert
30
20
10
50
30
Insert
40
20
10
50
30
40
Unbalanced
Left-right
rotation
I step:
20
10
50
40
30
II step:
20
10
40
30
50

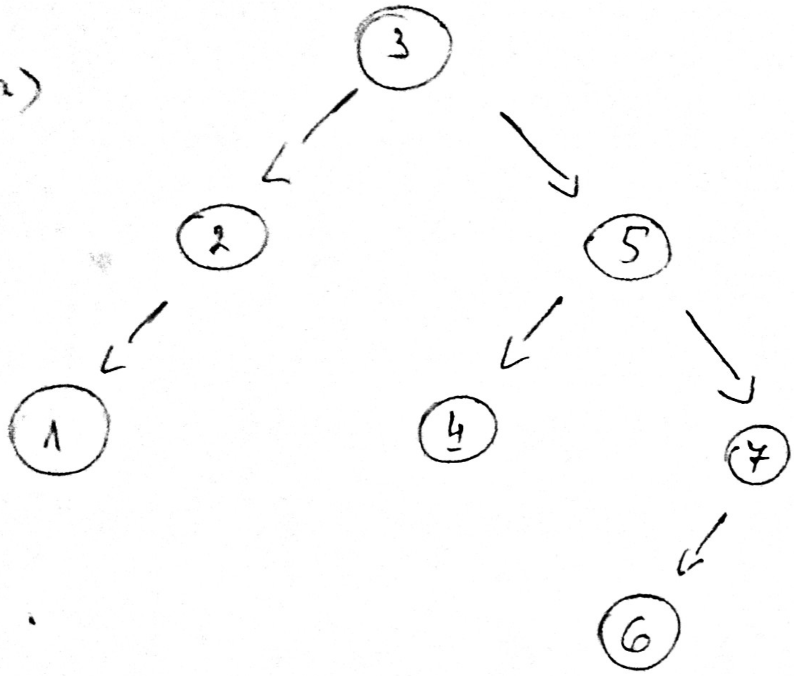
Problem 4c) continue:



Our final balanced AVL tree

Problem 5:

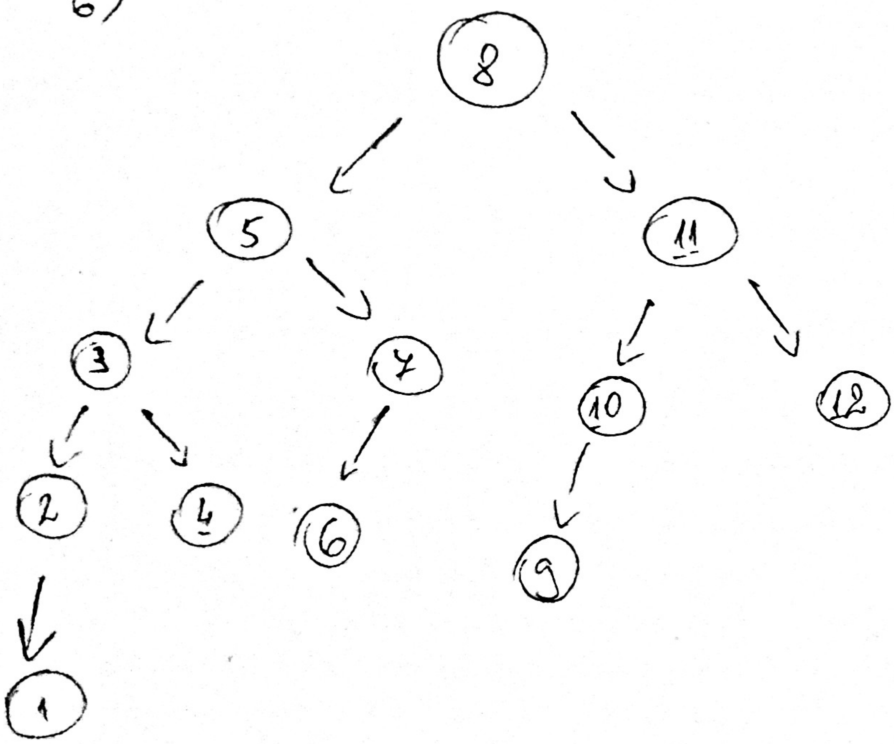
a)



height: 3

Min.# nodes: 7

b)

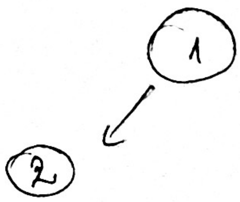


height: 4

Min. # of nodes: 12

Problem 5. c) To get an AVL tree with the smallest possible number of nodes, we need to make the tree as "unbalanced" as possible. => This will occur when for every height h we should have 2 subtrees: one with height $(h-1)$ and the other with height $(h-2)$. This way the tree will gain max. height and will remain AVL.

=> We get a recurrence relation: $f(h) = 1 + f(h-1) + f(h-2)$

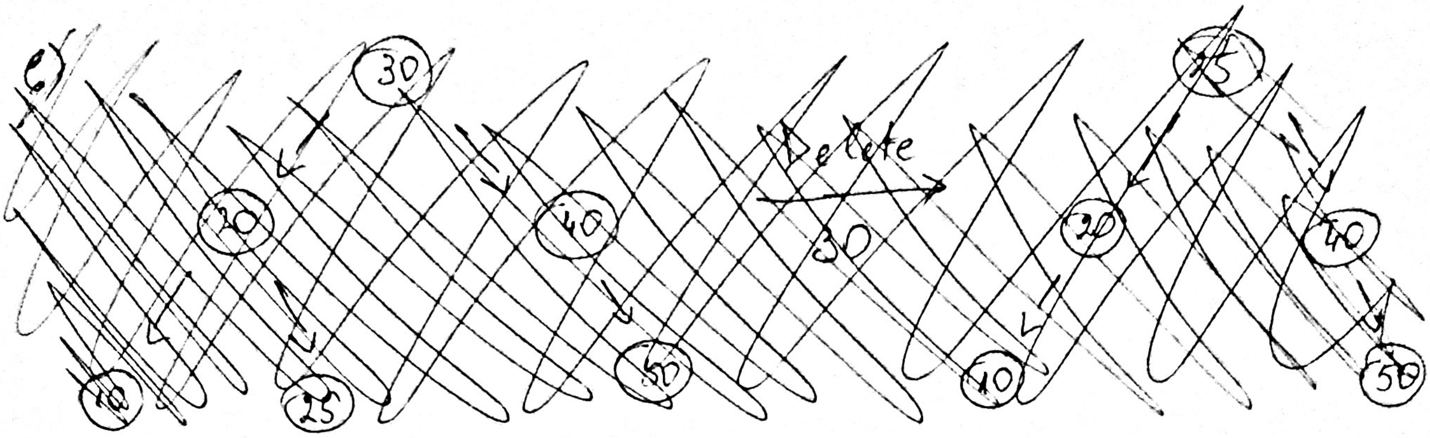
(We add one for the root node). For $h=0$ and $h=1$ we have the trees:  and  $\Rightarrow \begin{cases} f(0) = 1 \\ f(1) = 2 \end{cases}$

$\Rightarrow f(2) = f(0) + f(1) + 1 = 1 + 2 + 1 = 4$; $f(3) = f(2) + f(1) + 1 = 4 + 2 + 1 = 7$; $f(4) = f(3) + f(2) + 1 = 7 + 4 + 1 = 12$, which confirms our drawings' correctness.; Since we have a recurrence relation, we can set $f(h) = a_n$; $f(h-1) = a_{n-1}$; $f(h-2) = a_{n-2}$

$\Rightarrow a_n = a_{n-1} + a_{n-2} + 1$; Now, if we add 1 to both sides:

$a_n + 1 = (a_{n-1} + 1) + (a_{n-2} + 1)$; We can now set $b_n = a_n + 1$:

$\Rightarrow b_n = b_{n-1} + b_{n-2}$; It's easy to recognize that this is the Fibonacci sequence! It's easy to get the characteristic polynomial: $x^2 - x - 1$ and characteristic equation: $x^2 - x - 1 = 0$



$$x^2 - x - 1 = 0 \Rightarrow \left(x - \frac{1}{2}\right)^2 - \frac{1}{4} - 1 = \left(x - \frac{1}{2}\right)^2 - \frac{5}{4} = 0$$

$$\Rightarrow \left(x - \frac{1}{2} - \frac{\sqrt{5}}{2}\right)\left(x - \frac{1}{2} + \frac{\sqrt{5}}{2}\right) = 0 \Rightarrow \text{We have:}$$

$$b_n = a\left(\frac{1-\sqrt{5}}{2}\right)^n + b\left(\frac{1+\sqrt{5}}{2}\right)^n$$

We know that $a_0 = f(0) = 1$; $a_1 = f(1) = 2 \Rightarrow b_0 = 2$; $b_1 = 3$

$$b_0 = a + b = 2 \; ; \; b_1 = a\left(\frac{1-\sqrt{5}}{2}\right) + b\left(\frac{1+\sqrt{5}}{2}\right) = 3$$

$$b = 2 - a \; ; \; b_1 = a\left(\frac{1-\sqrt{5}}{2}\right) + (2-a)\left(\frac{1+\sqrt{5}}{2}\right) = 3$$

$$\Rightarrow \cancel{\frac{1}{2}}a - \frac{\sqrt{5}}{2}a + (1+\sqrt{5}) - \cancel{\frac{a}{2}} - \frac{\sqrt{5}}{2}a = 3$$

$$\Rightarrow 1 + \sqrt{5} - \sqrt{5}a = 3 \Rightarrow \sqrt{5} - 2 = \sqrt{5}a \Rightarrow \begin{cases} a = 1 - \dfrac{2}{\sqrt{5}} \\ b = 1 + \dfrac{2}{\sqrt{5}} \end{cases}$$

$\Rightarrow$ Finally we get: $f(n) = a_n = b_n - 1 =$

$$\boxed{= \left(1 - \frac{2\sqrt{5}}{5}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n + \left(1 + \frac{2\sqrt{5}}{5}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n - 1}$$