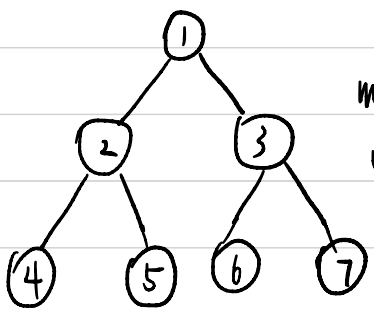
EL504 Homework 3

1.

(a) True

(b) True



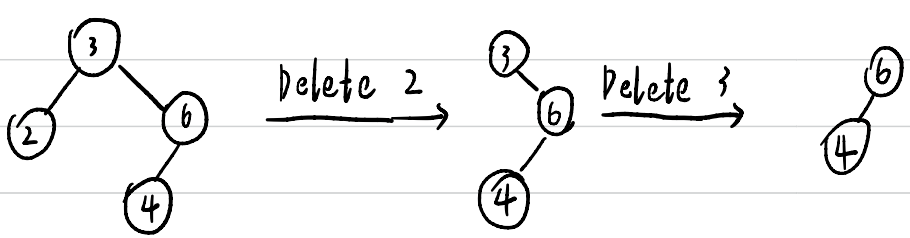
start with the root node the minimum height of the tree is 0 and the maximum height of the tree is 1. So the value of $h=1$. Put the value of h in the given formula $2^{h+1}-1$.

For the tree above, $2^{2+1}-1 = 2^3-1 = 8-1 = 7$

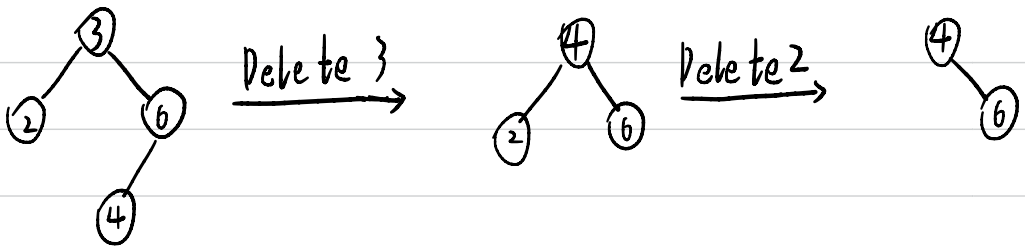
So the statement is correct.

(c) False

For example, First delete 2 then delete 3



First delete 3 then delete 2



So the statement is false

(d) False

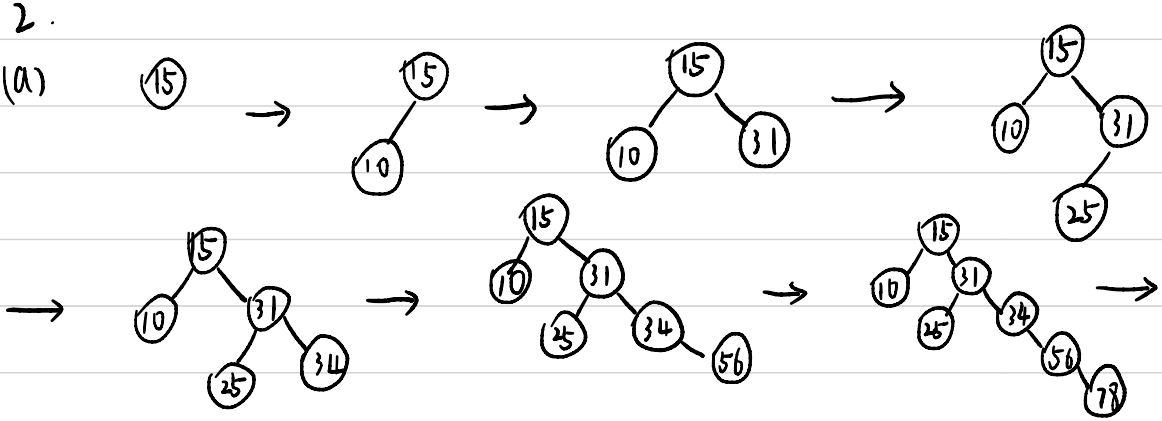
Inserting one number in binary min-heap will take O(logn) time.

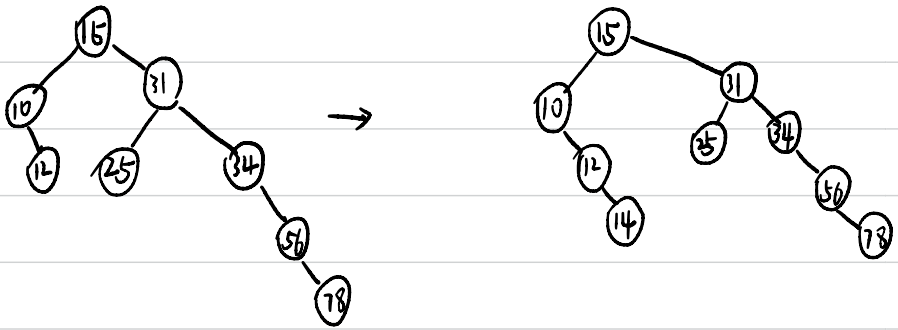
(e) True

Because it is the min-heap property.

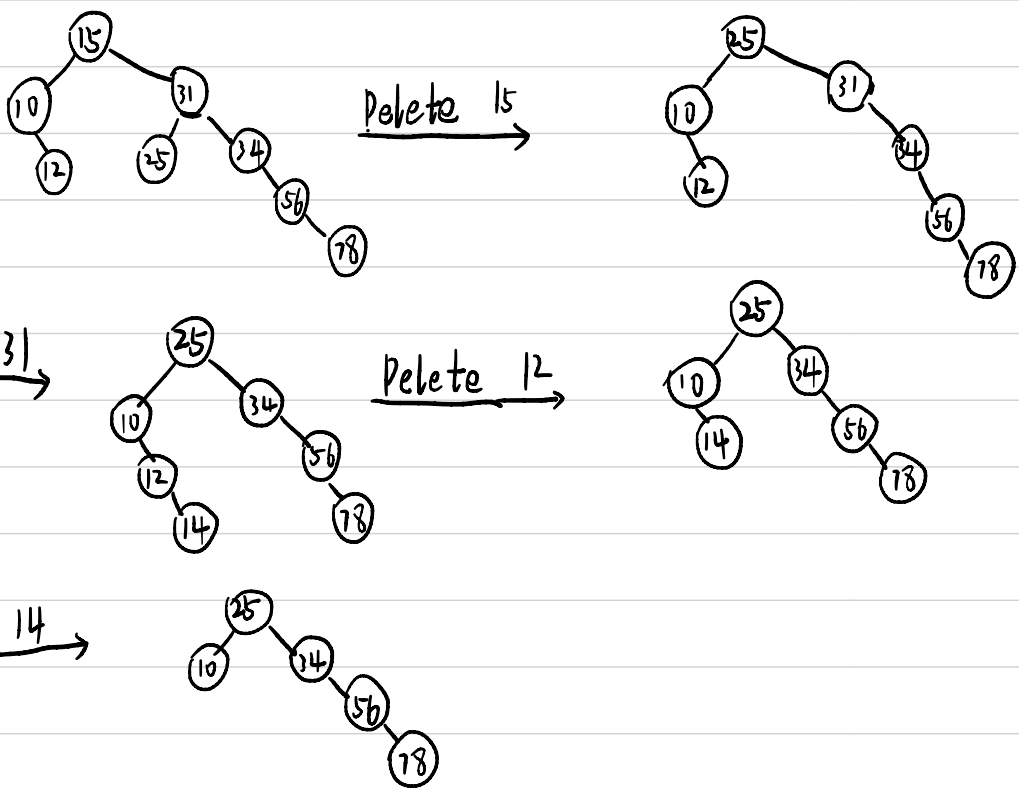
2.

(a)

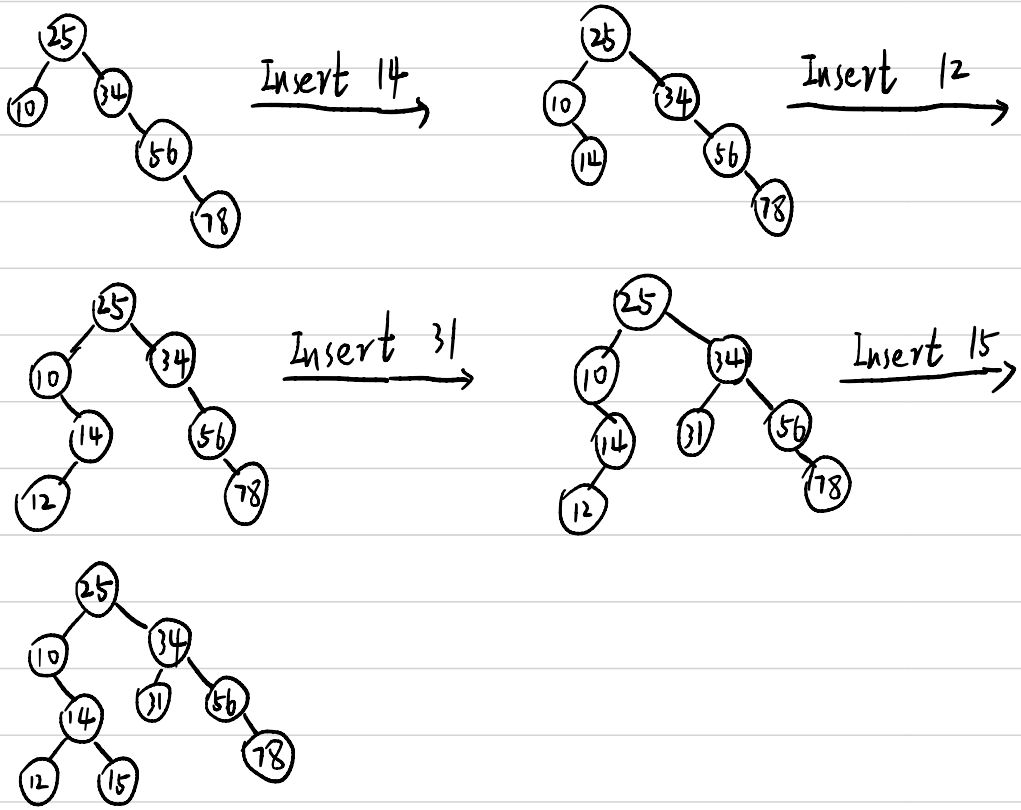



15
10
31
12
25
34
56
78
15
10
31
25
34
12
56
14
78

(b)


15
10
31
12
25
34
56
78
Delete 15
25
10
31
12
34
56
78
Delete 31
25
10
34
12
56
14
78
Delete 12
25
10
34
14
56
78
Delete 14
25
10
34
56
78

(c)



3.

(a)

6.1-3: If the largest element in the subtree were somewhere orther than the root, it has a parent that is in the subtree. So, it is larger than it's parent, so, the heap property is violated at the

parent of the maximum element in the subtree.

6.1-4: In any of the leaves, that is, elements with index $\lfloor n/2 \rfloor + k$, where $k \geq 1$. that is, in the second half of the heap array.

6.1-6: No. Since the parent of 7 is 6. This violates the max-heap property.

6.3-3: We know that the leaves of a heap are the nodes indexed by

$$\lfloor n/2 \rfloor + 1, \lfloor n+2 \rfloor + 2, \cdots, n$$

Note that those elements corresponds to the second half of the array (plus the middle element if n is odd). Thus, the number of leaves in any heap of size n is $\lceil n/2 \rceil$. Let's prove by induction. Let n_h denote the number of nodes at height h. The upper bound holds for the base since $n_0 = \lceil n/2^0 \rceil$ is exactly the number of leaves in a

heap of size n.
Now assume it holds for $h-1$. We have prove that it also holds for h. Note that if n_{h-1} is even each node at height h has exactly two children, which also implies $n_h = [n_{h-1}/2] + 1 = [n_{h-1}/2]$. Thus, we have

$$n_h = \left[\frac{n_{h-1}}{2}\right] \leq \left[\frac{1}{2} \cdot \left[\frac{n}{2^{(h-1)}+1}\right]\right]$$

$$= \left[\frac{1}{2} \cdot \left[\frac{n}{2^h}\right]\right]$$

$$= \left[\frac{n}{2^{h+1}}\right]$$

which implies that it holds for h.

12.3-2:
Number of nodes examined while searching also includes the node which is searched for, which isn't the case when we inserted it.

12.3-3

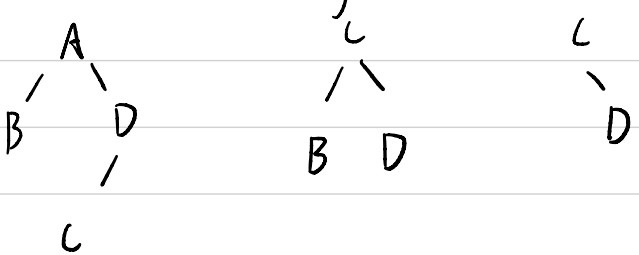
· The worst-case is that the tree formed has height n because we were inserted them in already sorted order. This will result in a runtime of $\Theta(n^2)$

· The best-case is that the tree formed is approximately balanced. This will mean that the height doesn't exceed $O(\lg n)$. Note that it can't have a smaller height, because a complete binary tree of height h only has $\Theta(2^h)$ elements. This will result in a runtime of $O(n \lg n)$.

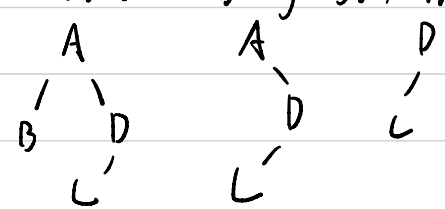
12.3-4

No, giving the following counterexample

· Delete A first, then delete B



· Delete B first, then delete A



4.

(a) False, it takes $n \log n$ time.

(b) False, it's a complete binary tree.

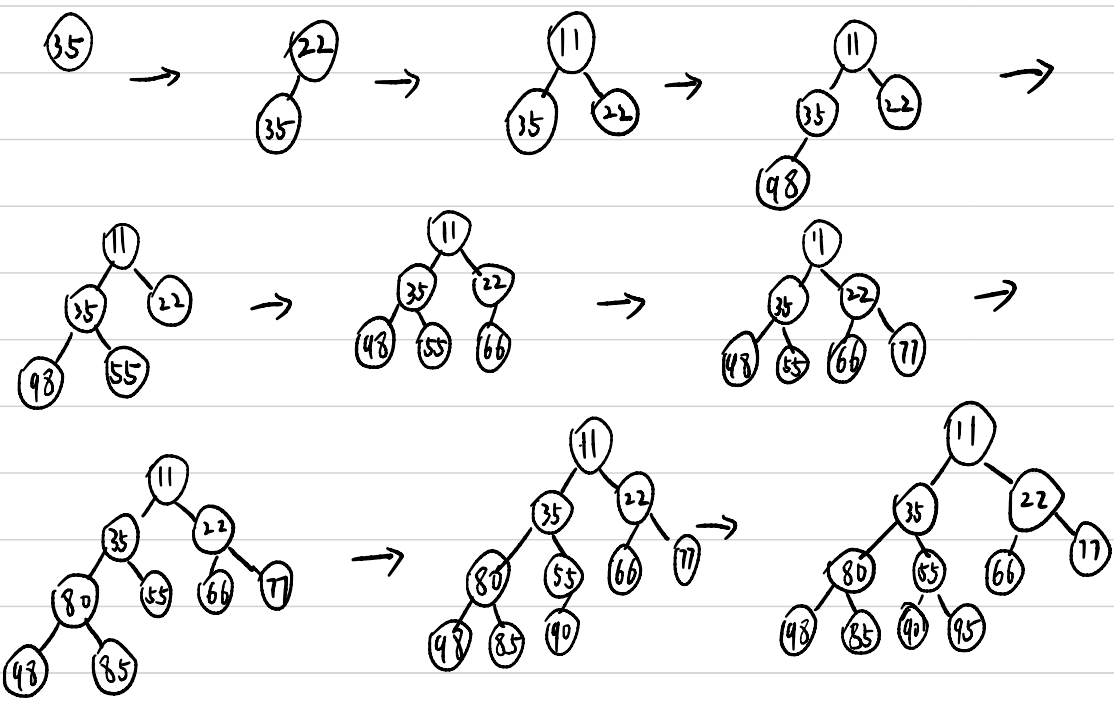
(c) true

(d) true

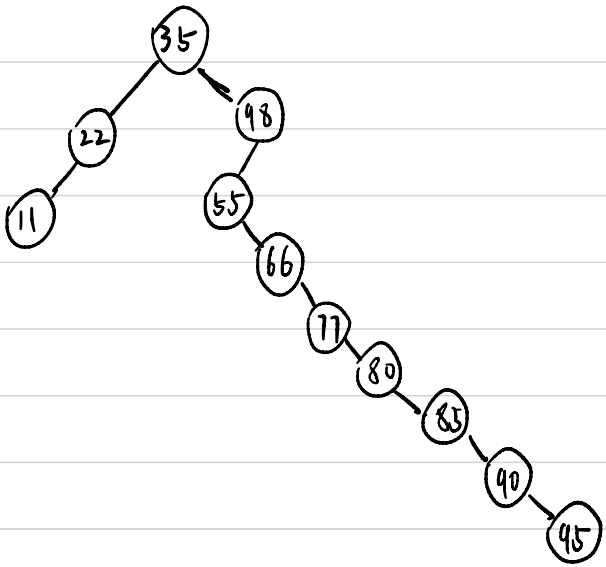
(e) true

5.

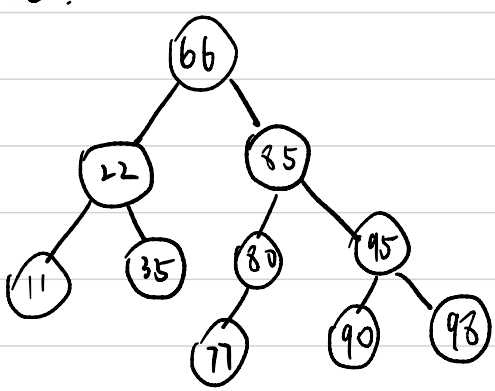
(a)



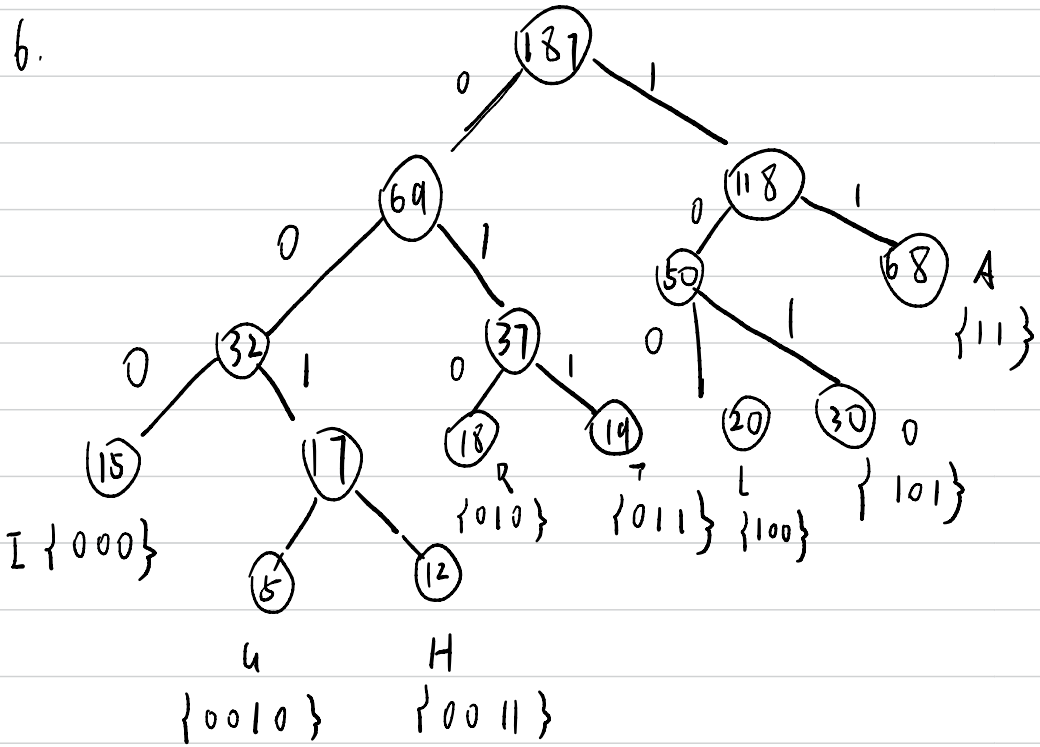
(b) BST tree:



(c) AVL tree:



6.



(a) Minimum external path length

$20\times3 + 5\times4 + 30\times3 + 18\times3 + 15\times3 + 19\times3 + 12\times4 + 68\times2$

$= 510$

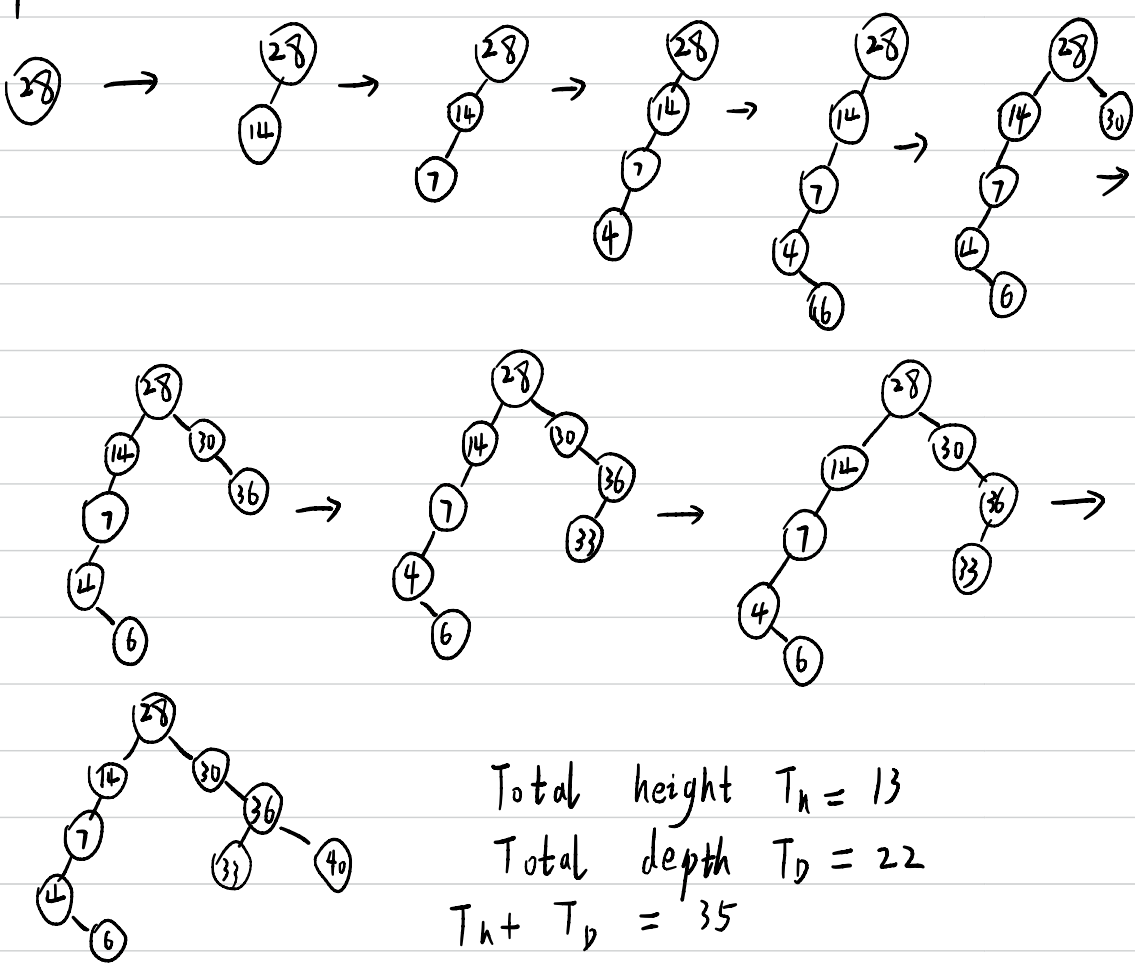
(b) Number of bit for each letter is given in image like for

A - 2bits, L - 4bits, O - 3bits, R - 3bits, I - 3bits, T

- 3 bits H - 4 bits

Compute the average number of bit per symbol = 510 / 187 = 2.72, yes it is less than 3.

7.

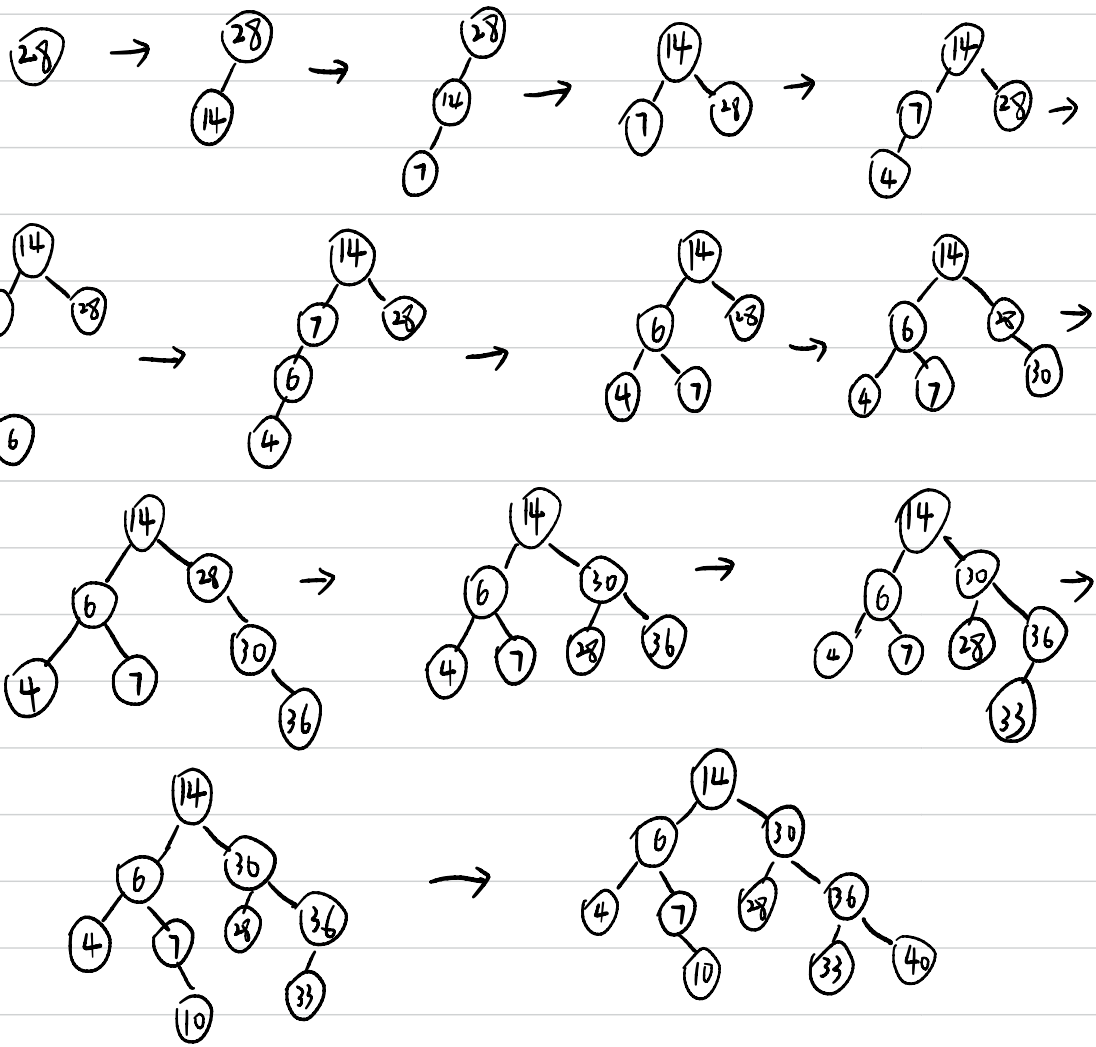


Total height $T_h = 13$

Total depth $T_D = 22$

$T_h + T_D = 35$

(c)



Total height = 9 = T_H

Total depth = 19 = T_D

$T_H + T_D = 28$