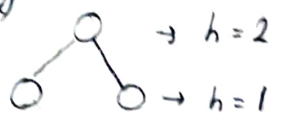
Let n_i be the number of nodes of ith height

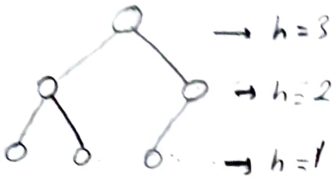
1) a)



$d = (h_{max} + 1) - h$

$d \times n_i \Rightarrow 1 \times 1 = 1$

$d \times n_i \Rightarrow 2 \times 2 = 4$



5 total depth

$(d \times n_i) + (d \times n_i) + (d \times n_i) = ?$

$(1 \times 1) + (2 \times 2) + (3 \times 3) = 14$

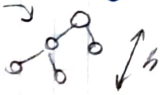
$$\text{formula} \Rightarrow \sum_{i=1}^{h_{max}} \left((h_{max} + 1) - i\right) * n_i$$

b) Time complexity of a binary search tree which has the structral property of being complete binary tree;

$T_B(n) = \Theta(1)$ → when root matches with target

$T_W(n) = \Theta(\log n)$ → because our tree is complete

$T(n) = O(\log n)$ ⇒ why $\log n$, because each time the values to be compared one halved



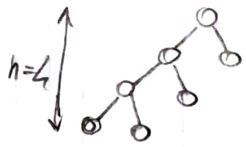
$O(\log n)$ is general time complexity of our algorithm and algorithm's basic operation is comparison so # of comparison = $\log n$

c) let h be the height of our tree;

→ Maximum number of nodes in a full binary tree is $(2^h - 1)$

minimum number of nodes in a full binary tree is $(2h - 1)$

$2^3 - 1 = 7$ nodes



$2.4 - 1 = 7$ nodes

If our tree has a total of N nodes;

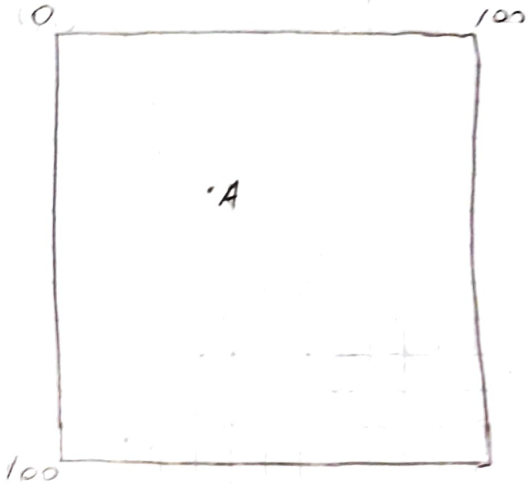
→ The number of interval nodes is $(N-1)/2$

→ The number of leaves is $(N+1)/2$

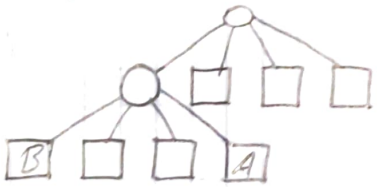
2) (30, 30), (20, 15), (50, 40), (10, 12), (40, 20), (20, 60), (15, 25)

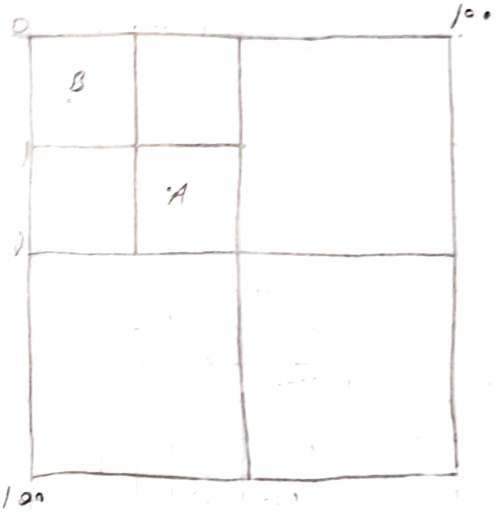
A B C D E F G

Tree ;

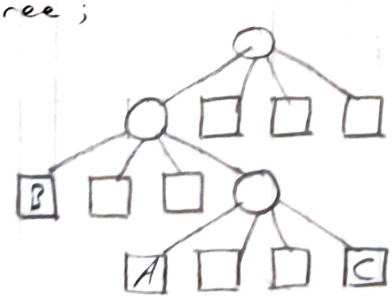


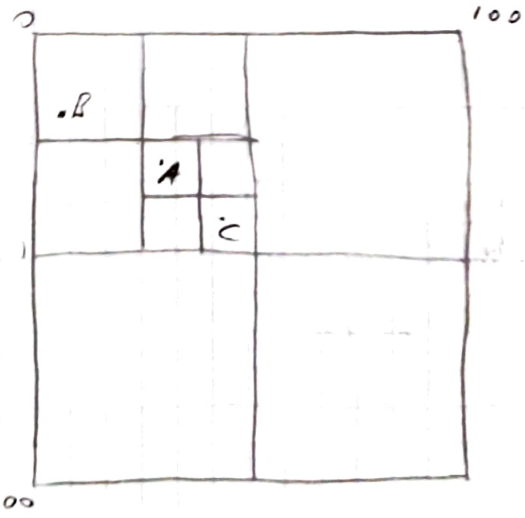
Tree ;



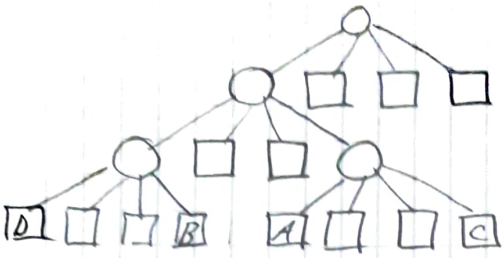


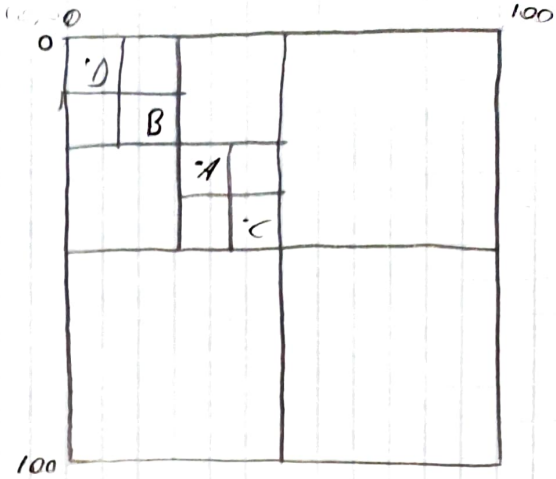
Tree ;



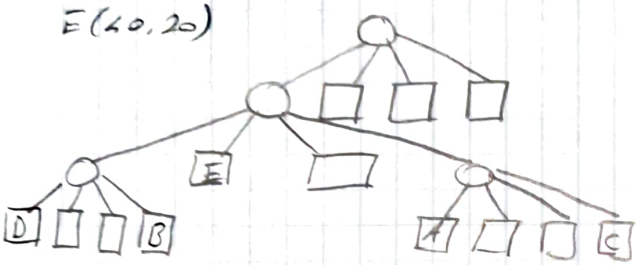


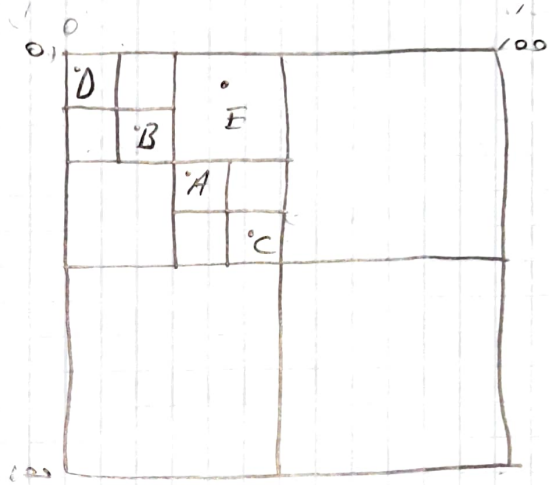
D (10, 12)



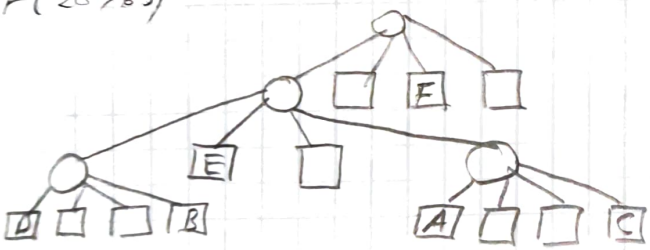


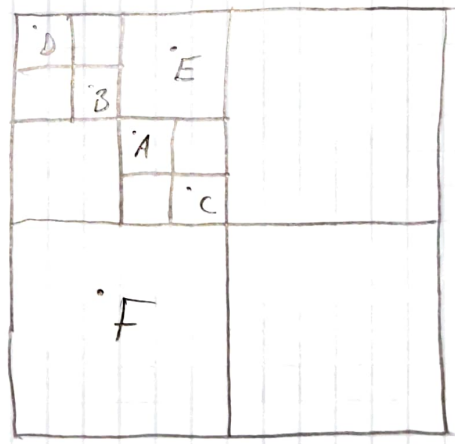
E (40, 20)





F (25, 60)





G (15, 25)