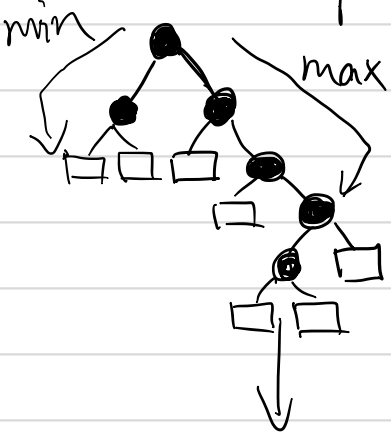
1.

Because red-black tree's property,

any root-leaf path of size k must have

> (max height from root to leave) / 2 (counting fake as black)

[square: fake node]



=> No way to just create a valid red black tree just by recolor

max root-leaf path = 6 (include fake node)

min root-leaf path = 3

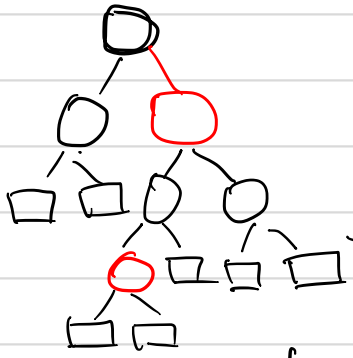
6/3 is not larger than 2, invalid!

As a result, the lowest rank of root in bst, is appear in the situation that

min root-leaf path · 2 + 1 = max root leaf path

That is, the node in longest path before fake node is red

Which is like:



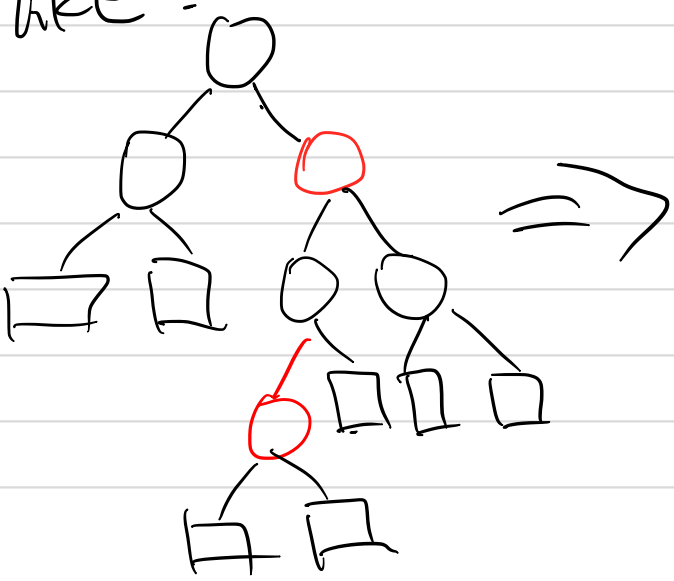
max path = 5
min path = 3
(count fakes)

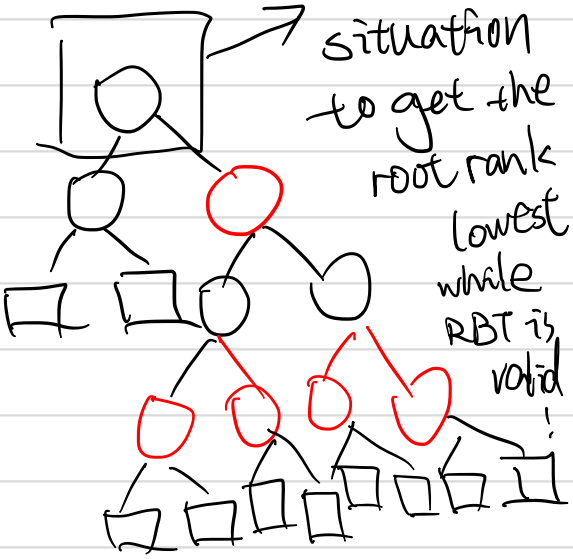
→ it is valid !

Because of binary tree property right > left, in order to have the rank of root as low as possible, we want to make sure to the right of the root, the nodes should be asfull as possible

That means, before the fake nodes on the right side, every node should be red !

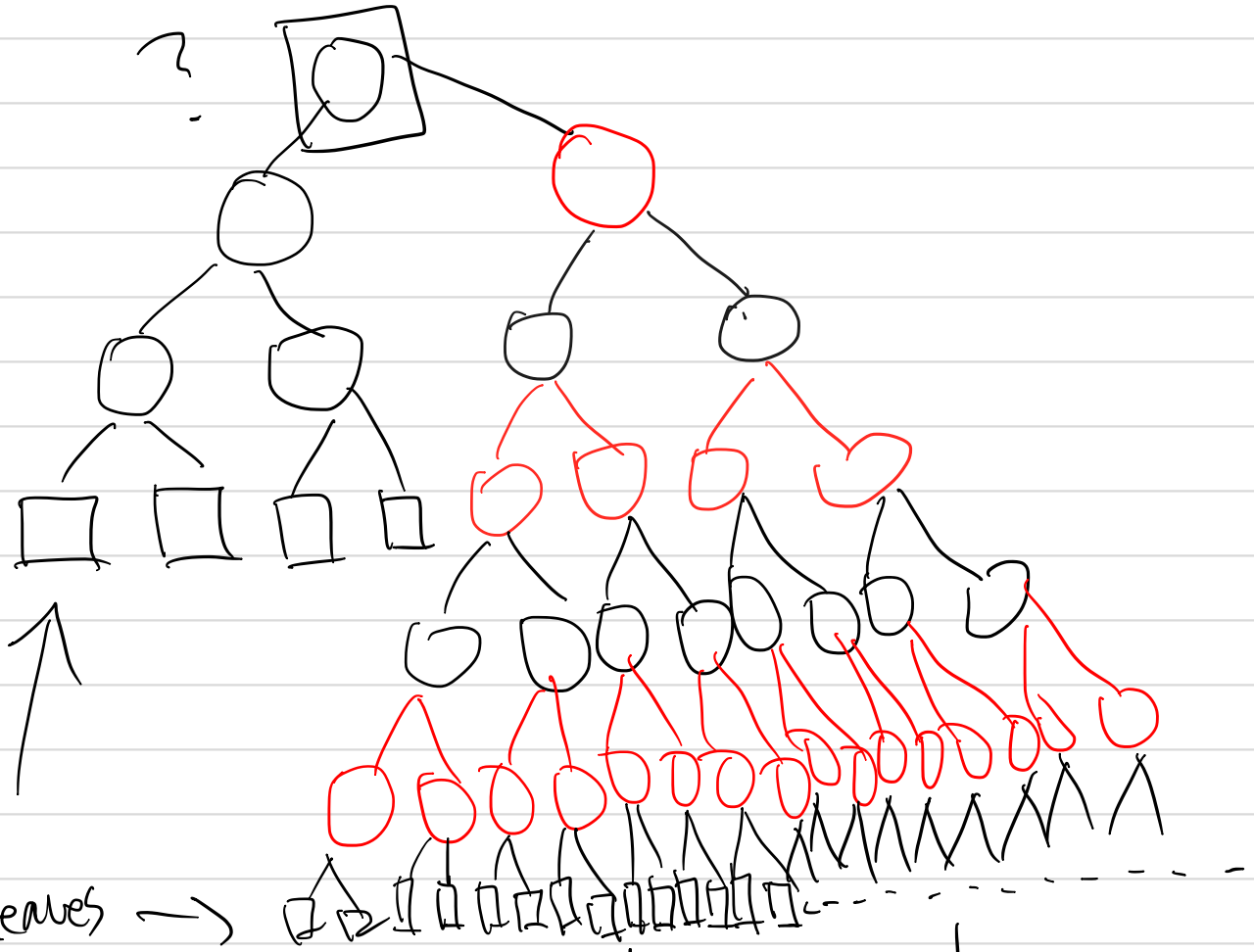
Which is like:





It is the situation to get the root rank lowest while RBT is valid !

So in the tree below, the rank of root is as low as possible. What is it relation to n? (Θ(?)) [while RBT valid]



(n nodes in total

Didn't count fake in n)

max path: 7

min path: 4

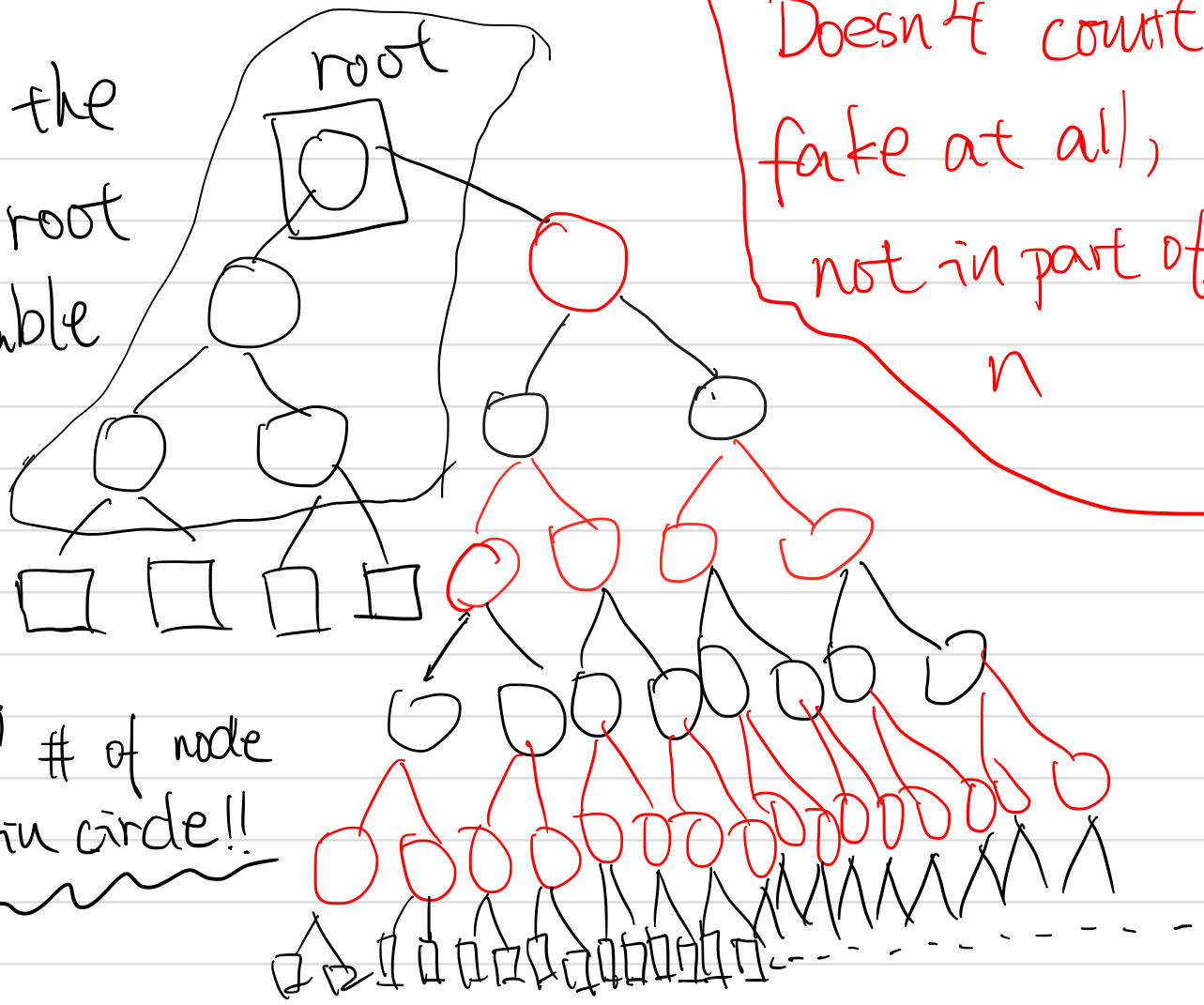
Valid!

1. take the rank of root as variable

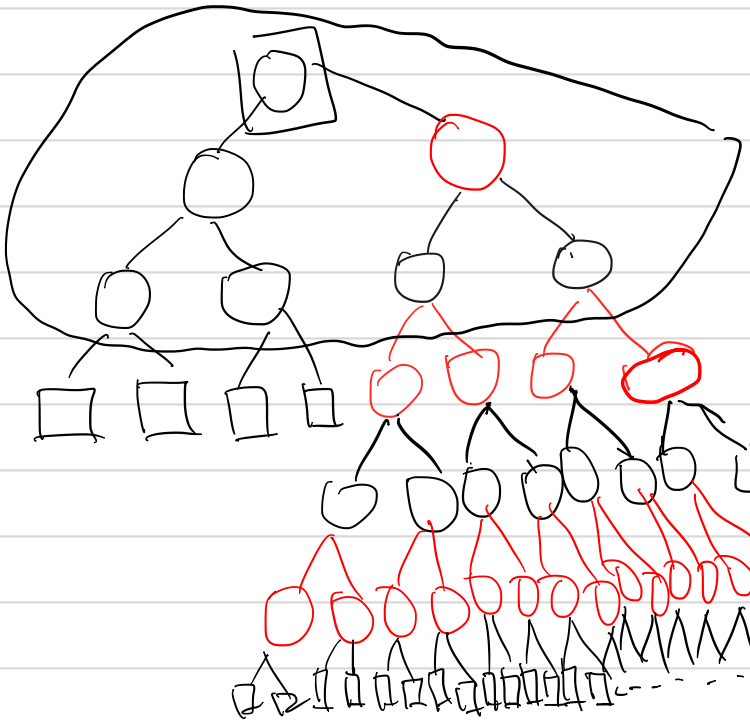
x

then

$x =$ # of node in circle!!



Doesn't count fake at all, not in part of n



2. This part in total, equals

$x + (x-1) = 2x-1$

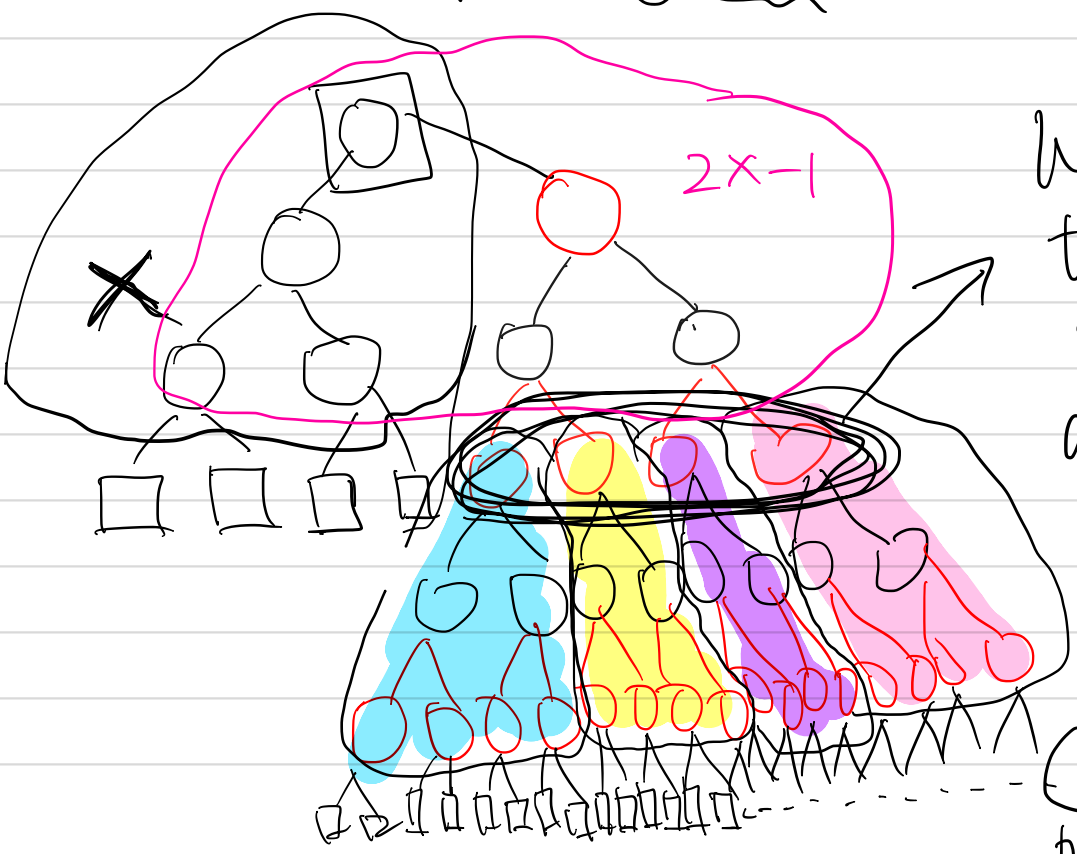
⇒ continue

Doesn't count fake at all, not in part of n!

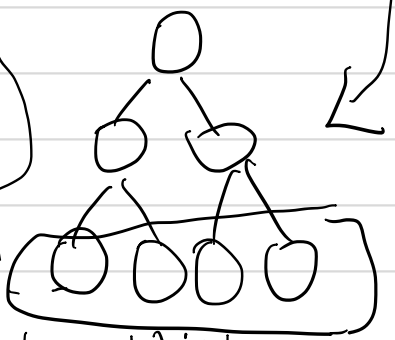
3.

each of them equal to the number in step 2, $2x-1$ as well

4.

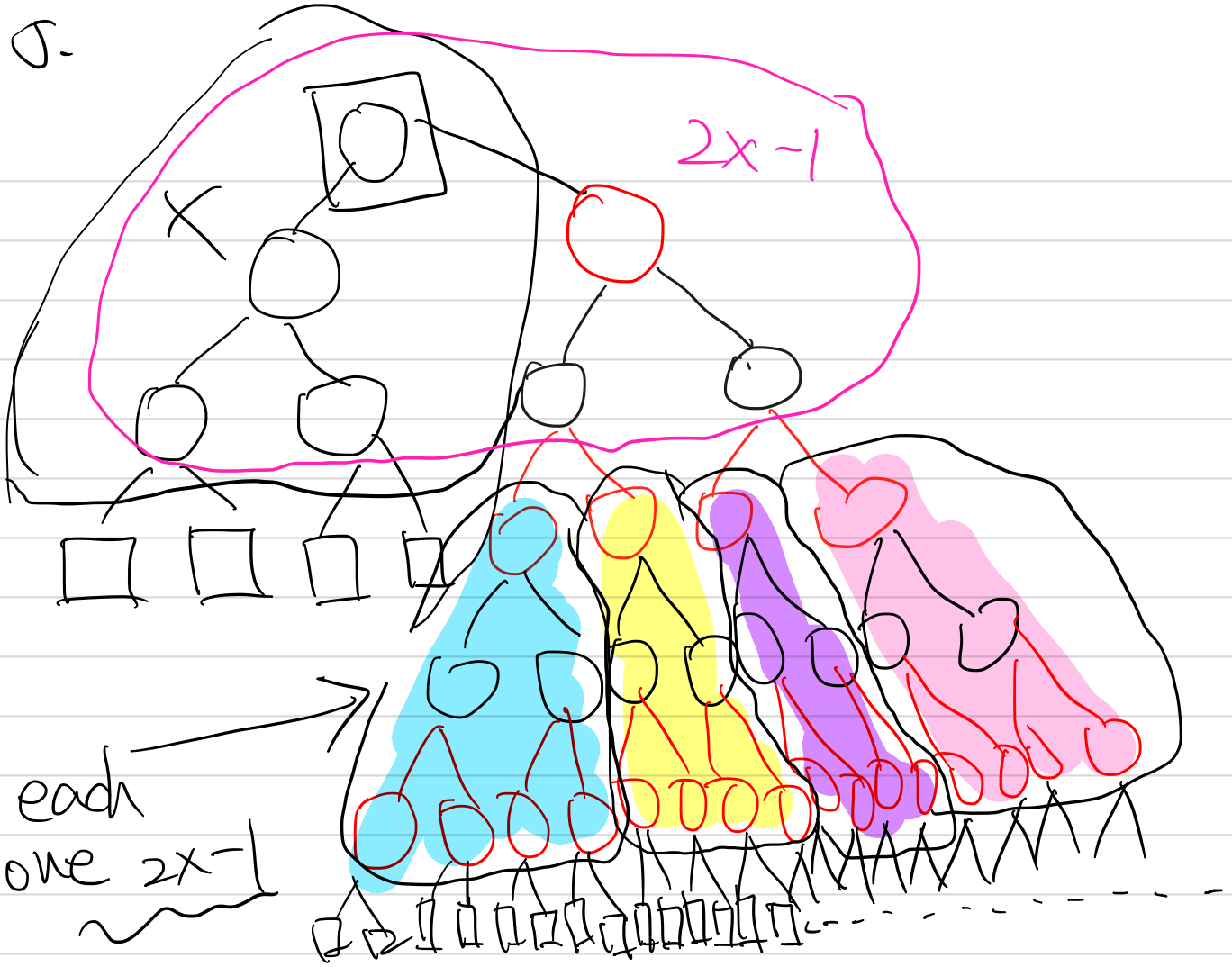


how many time color section appears?



$= \frac{(2x-1)+1}{2} = x$

5.



In total, the total number of node, in term of x, equals to

$$2x-1 + x \cdot (2x-1)$$

$$= (x+1)(2x-1)$$

$$= 2x^2 + x - 1 = n$$

$$rank = f(n)$$

$$\Downarrow$$

$$\frac{f(n)}{x} = \frac{n}{2x^2+x-1}$$

$$3x^2 > 2x^2+x-1$$

$$n < 3x^2$$

$$x^2 < 2x^2+x-1 \quad (x \geq 1)$$

$$n > x^2$$

$$n = \theta(rank^2)$$

$$rank = \theta(\sqrt{n})$$