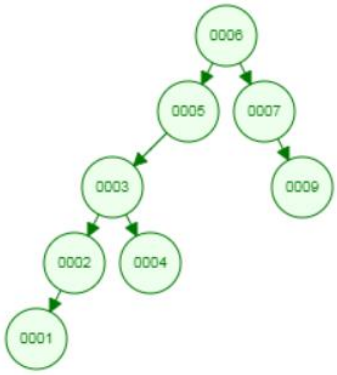
* 1. Insert the keys 2, 7, 1, 4, 3, 9, 5, 6 into an empty splay tree in the given order. Draw the tree after each insertion.

Solution:

Splay tree visualization: https://www.cs.usfca.edu/~galles/visualization/SplayTree.html

Resulting tree:

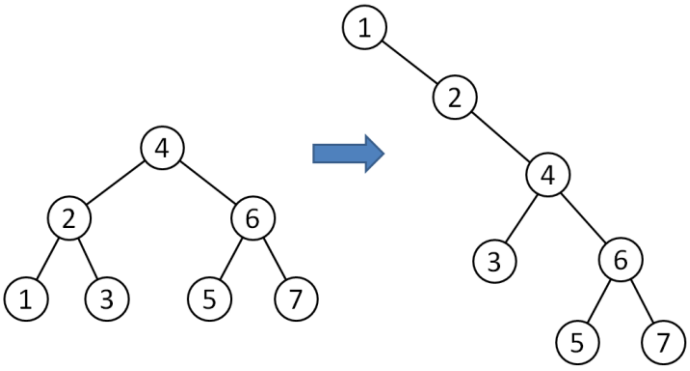


* 2. A splay tree contains $2^n - 1$ keys of values 1, 2, 3, ..., $2^n - 1$ and it is ideally balanced, i.e., its height is $n-1$. After calling Find(1), key 1 is moved to the root. What is the resulting height of the tree after the operation? Solve the problem for an even and odd n separately.

Solution:

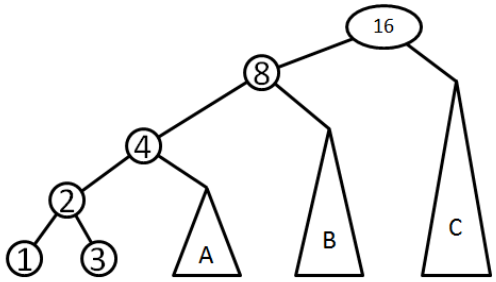
Let us assume that n is odd.

For n = 3, the initial and the obtained splay tree is:



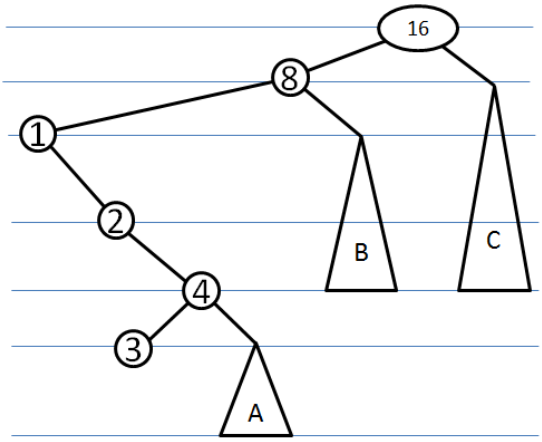
We can see that the height increased by 2.

For n = 5, the initial splay tree is:



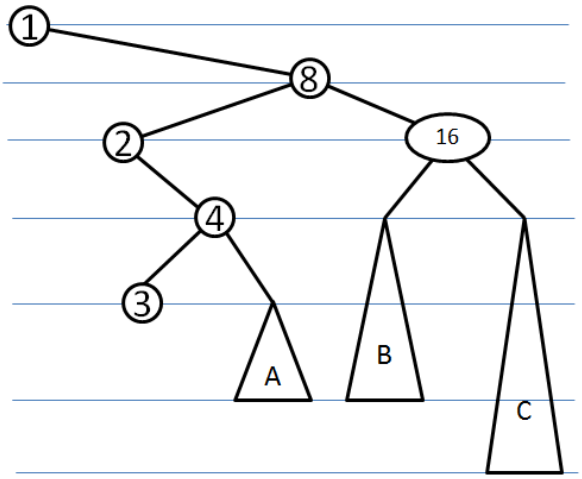
Note that subtree A has a height of 1, while subtree B and subtree C each have a height of 2.

After performing the first zig-zig rotation, we get:



The height of the tree increased by 2 (from height 4 to height 6).

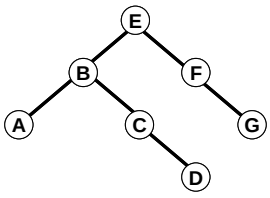
After performing the second zig-zig rotation, we get

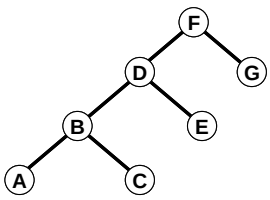


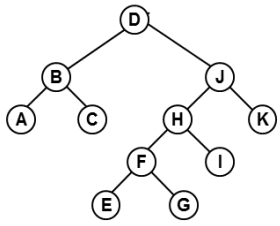
Now, the height remains 6, there is no increase. This can be generalized as follows: the first zig-zig rotation increases the height by 2, however, each subsequent zig-zig rotation maintains the same height. So, for an odd *n*, the resulting height is *n+1*.

Here we remind properties of the red-black tree (RB tree):
1. Every node is either red or black
2. Every leaf (nil) is black
3. If a node is red, then both its children are black
4. Every simple path from a node to a descendant leaf contains the same number of black nodes
5. Root is black

* 3. Color nodes of the trees in red or black so that the resulting tree is an RB tree. Note that empty (nil) leaves are not displayed.

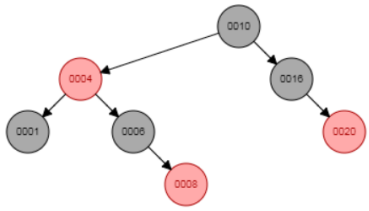
A. 

B. 

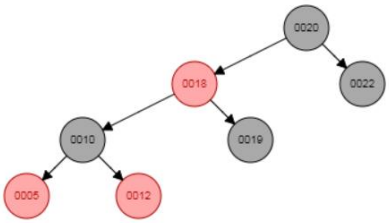
C. 

Solution:

A.



B.



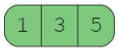
* 4. The black height of an RB-tree is 11. It is defined as the number of black nodes in any path from the root to an arbitrary nil leaf decreased by 1. What is the maximum possible number of A) black nodes, B) red nodes, C) all nodes in the tree?

Solution:
The number of nodes is maximum possible if the tree is ideally balanced and it contains red nodes at each odd level. There is 1 black root node (level 0), 2 red nodes (level 1), 4 black nodes (level 2), 8 red nodes (level 3), ..., 2^{20} black nodes (level 20), 2^{21} red nodes (level 21) and 2^{21} black nil nodes.

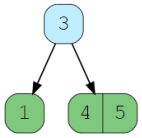
There are $1+2+...+2^{21}=2^{22}-1$ nodes in total. The number of red nodes is $2^1+2^3+2^5+...+2^{21}=2(1+4+4^2+...+4^{10})=2*(4^{11}-1)/(4-1)$. The number of black nodes can be expressed as the total number of nodes minus the number of red nodes.

* 5. Insert keys 3, 1, 5, 4, 2, 9, 10, 8, 7, 6 into an empty 2-3-4 tree in the given order. Draw the tree after each insertion.

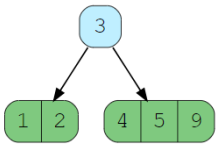
Solution:
Insert(3), Insert(1), Insert(5):



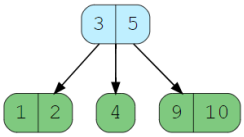
Insert(4):



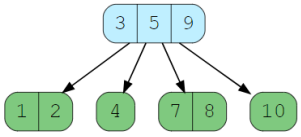
Insert(2), Insert(9):



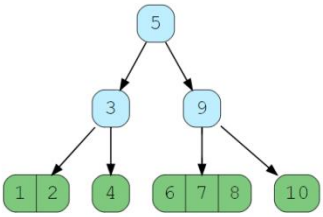
Insert(10):



Insert(8), Insert(7):



Insert(6):



6. An RB-tree contains 15 keys, its black depth is 2, i.e., there are 10 nodes. Values of the keys are 1, 2, 3, ..., 7, 8, 21, 22, ..., 26, 27. We insert additional three keys into the tree: 11, 12 and 13 (in this order). Draw the original tree and the modified tree after each insertion.

Solution:
Use the RB-tree visualization here: https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

7. Decide if there is a regular (each inner node has two children) binary tree whose nodes cannot be coloured to obtain an RB-tree. Give an example or explain why there is no such a tree.

Solution: In the tree bellow, the shortest path from the root to a leaf has 2 black nodes (note that, by the definition, the root and all leaves are black). The longest path must have the same number of black nodes. This implies that the longest path conatins a red node with a red child.