Chitraarth Singh.

Assignment - 3

1. Problem 12.3-2.

Soln :- When inserting a node into binary search tree at depth 'd' we compare 'n' nodes and insert n^{th} node as leaf.
While performing search operation we compare 'd' times to examine the right position and then compare node at the branch with 'n'. Therefore we examine ~~d~~ d + 1 nodes. Hence number of nodes in searching for a value in the tree is one plus the number of nodes examined when the value was first inserted. i.e. there will be d+1 nodes.

2. Problem 13-1-5.

Soln:- The property 5 of red black tree says:
For each node, all simple paths from the node to the descendant leaves contain the same no. of black nodes.
If 'h' is the height of the ~~black~~ ~~xxx~~ tree and bh(x) is the height of the black nodes, the longest path contain at least every node as black and the shortest path contain at most every node as black. If we insert red node after every black node, then ~~the~~ both the paths contain equal number of black nodes.
Therefore, longest simple path from a node x will be 2bh(x) of the shortest simple path from node x to descendant leaf.

3. Problem 13.3-4

Soln:- If we consider the scenario in which we set T.nil.color to RED, then z is its parent and z.p is its grandparent. The grandparent z.p ~~[illegible]~~ must be the root.
Now, ~~[illegible]~~ according to property ~~1~~ 2, ~~if~~ if z.p is the root then it has to be BLACK.
∴ z.p.color = BLACK
The RB-INSERT-FIXUP (T, z), the while loop only runs if z.p.color == RED. Then the condition that z.p.color == BLACK will terminate the loop.

4. Problem 13.4-6.

Soln:- Lets consider, if(w.color == RED) then case 1 will occur. Now, if this condition is true the control will go inside the if loop.

From the algorithm if w = x.p.right and w.color == RED then x.p has to be BLACK as per the property 4 which says that if a node is red, then both its children are black and no child-parent nodes can be red.

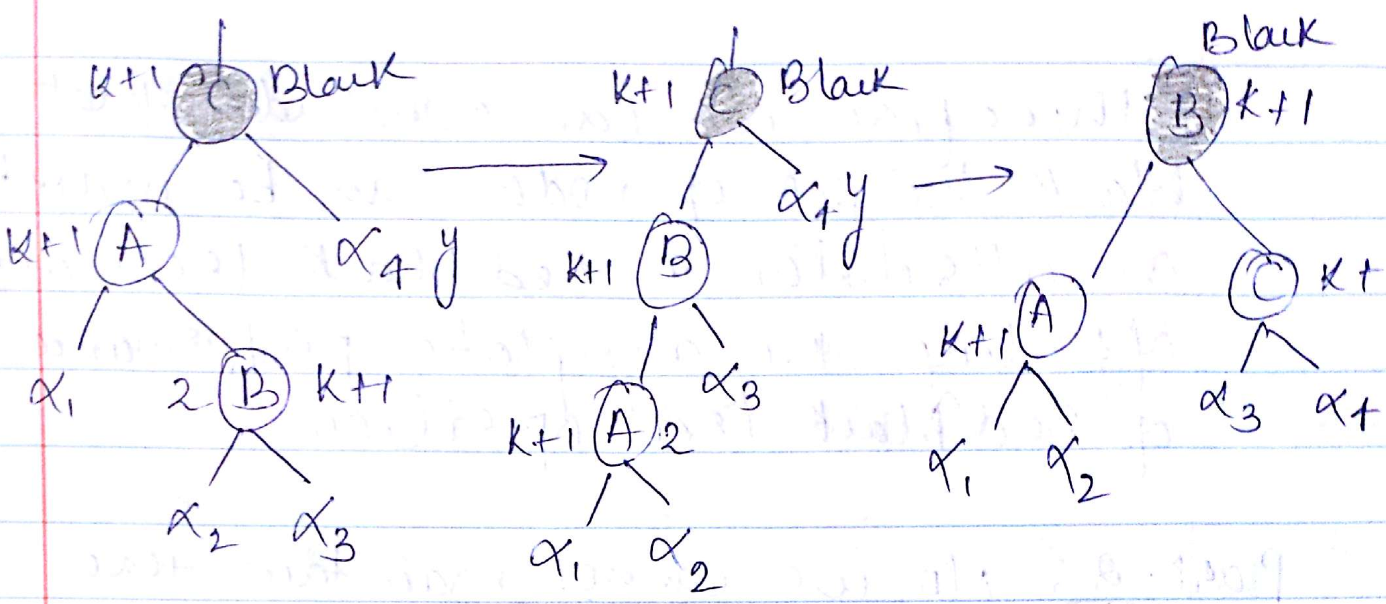
Therefore, at the start of case 1, x.p must be BLACK.

5. Problem 14.2-2.

Soln:- Yes, we can maintain black-heights as attributes in the node of R-B tree without affecting the asymptotic performance of the red-black tree operations. By Theorem 14.1, we can say that the height of a node can be computed from the information at the node & its two children. The black height can also be computed using only one child's information which means that the black-height of a node is equal to the black height of red child ~~node~~ or black height of black ~~node~~ child + 1.

Lets consider RB-INSERT-FIXUP & RB-DELETE-FIXUP to show that insertion & deletion doesnot affect the asymptotic performance.

RB-INSERT-FIXUP



In the above cases when z's uncle y is black, & z is a right child and when z's uncle y is black, & z is a left child, we find that the subtrees $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ of black height K, even with color changes & rotations, the black height of nodes A, B and C remain the same (K+1).

∴ RB-INSERT-FIXUP maintains its original O(lg n) time

Similarly, for RB-DELETE-FIXUP we can show that even with color changes and rotations the black height remains the same.

∴ RB-DELETE-FIXUP also maintains its original O(lg n) time.

Therefore, we can conclude that black height of nodes can be maintained as attributes in red black trees without affecting the asymptotic performance of red black tree operations.

Part-2 :- No, we cannot maintain the node depths without affecting the asymptotic performance. The depth of the node depends the depth of ~~the~~ its parent. The depth of the tree must be updated if the depth of ~~a~~ the node above it changes. If root node changes it will cause other n-1 nodes to be updated. Therefore operations on the tree that change node depths ~~original~~ will simply not run in its original time which shows that depth of nodes will affect the asymptotic performance.