Q7: (refer to figure 1)

A binary tree has a red root path, if there is a path in the tree from the root *x* to a leaf, where all nodes on the path from *x* to the leaf has the colour red. Construct an algorithm REDROOTPATH(*x*), which returns the number 1 if the binary tree with root *x* has a red root path. If such a path does not exist, the algorithm must return 0. Give the time complexity of your solution. Higher marks shall be given for the most efficient and algorithm.

[5 marks]

Answer

- If we want to determine whether a path consisting of only red nodes from root to a leaf exists, we should only explore in those nodes which are red.

Algorithm:

Start with root x.

- If x.left and x.right are null. (it means x is a leaf)
  - **Return 1.**
- If left node of root is red
  - REDROOTPATH(r.left)
- If right node of root is red
  - REDROOTPATH(r.right)
- If no such leaf is found
  - **Return 0.**

Code in Java: [we suppose there is a map colour[node] which gives us the color (RED).]

```
int REDROOTPATH(x){
      if ( x.left == null && x.right == null )
            return 1;
      if ( colour[x.left] == RED )
            REDROOTPATH(x.left);
      if ( colour[x.right] == RED )
            REDROOTPATH(x.right);
      // if not found
      return 0;
}
```

- Time complexity of this algorithm is O(n), with n being the number of nodes in the binary tree, as each node is visited at most once.