Leetcode May Challenge DAY: 07 [BFS Approach]

1. Python

```
class Solution:
    def isCousins(self, root: TreeNode, x: int, y: int) -> bool:
        stack = [(root, 0, root.val)]
        nx = None
        ny = None
        for node, level, parent in stack:
            if node.left: stack.append((node.left, level+1, node.val))
            if node.right: stack.append((node.right, level+1, node.val))
            if node.val == x:
                nx = (node, level, parent)
            if node.val == y:
                ny = (node, level, parent)

            if nx and ny and nx[1] == ny[1] and nx[2] != ny[2]:
                return True
        return False
```

2. C++

```
bool isCousins(TreeNode* root, int x, int y) {
    unordered_map<int,TreeNode*> map;
    map[x]=root;
    map[y]=root;
    queue<TreeNode*> q;
    q.push(root);
    while(!q.empty())
    {
      TreeNode* t=q.front();
      q.pop();
      if(t->left)
      {
        if(t->left->val==x)map[x]=t;
        else if(t->left->val==y)map[y]=t;
        q.push(t->left);
      }
      if(t->right)
      {
        if(t->right->val==x)map[x]=t;
        else if(t->right->val==y)map[y]=t;
        q.push(t->right);
      }
    }
    unordered_map<int,int> map1;
    int level=0;
    q.push(root);
    while(!q.empty())
```

```
        {
            int n=q.size();
            level++;
            for(int i=0;i<n;i++)
            {
                TreeNode* t=q.front();
                q.pop();
                if(t->val==x)map1[x]=level;
                else if(t->val==y)map1[y]=level;
                if(t->left)q.push(t->left);
                if(t->right)q.push(t->right);
            }
        }
        return map[x]!=map[y]&&map1[x]==map1[y];
    }
```

3. JAVA

```
class Solution {
    public boolean isCousins(TreeNode root, int x, int y) {
        if(root == null || root.val == x || root.val == y) {
            return false;
        }

        Map<Integer, int[]> map = new HashMap<>();
        Queue<TreeNode> q = new LinkedList<>();
        q.offer(root);
        int level = 0;
        while(!q.isEmpty()) {
            int size = q.size();
            while(size-- != 0) {
                TreeNode parent = q.poll();
                int val = parent.val;
                if(parent.left != null) {
                    int leftVal = parent.left.val;
                    map.put(leftVal, new int[2]);
                    map.get(leftVal)[0] = val;
                    map.get(leftVal)[1] = level;
                    q.offer(parent.left);
                }

                if(parent.right != null) {
                    int rightVal = parent.right.val;
                    map.put(rightVal, new int[2]);
                    map.get(rightVal)[0] = val;
                    map.get(rightVal)[1] = level;
                    q.offer(parent.right);
                }
            }

            level++;
        }

        int[] p1 = map.get(x);
        int[] p2 = map.get(y);

        if(p1[0] != p2[0] && p1[1] == p2[1]) {
            return true;
        }

        return false;
    }
}
```