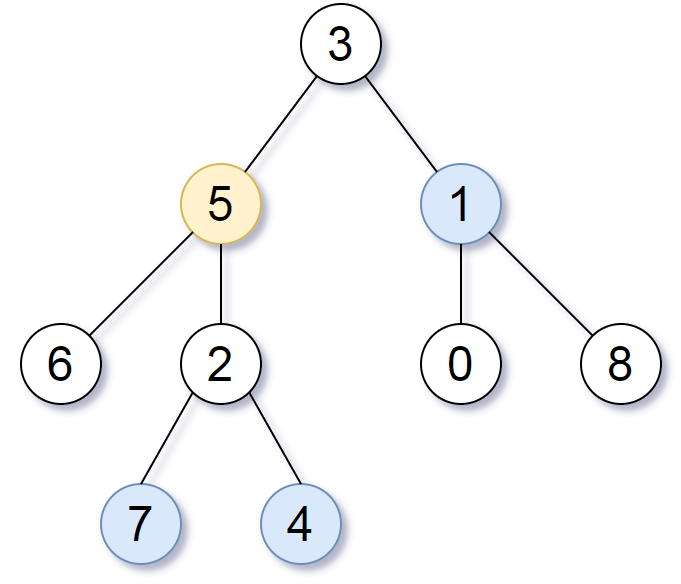
Given the root of a binary tree, the value of a target node target, and an integer k, return *an array of the values of all nodes that have a distance*k*from the target node.*

You can return the answer in **any order**.

**Example 1:**



**Input:** root = [3,5,1,6,2,0,8,null,null,7,4], target = 5, k = 2

**Output:** [7,4,1]

Explanation: The nodes that are a distance 2 from the target node (with value 5) have values 7, 4, and 1.

**Example 2:**

**Input:** root = [1], target = 1, k = 3

**Output:** []

Solution:

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode(int x) { val = x; }

\* }

\*/

class Solution {

private void markParents(TreeNode root, Map<TreeNode, TreeNode> parent\_track) {

Queue<TreeNode> queue = new LinkedList<TreeNode>();

queue.offer(root);

while(!queue.isEmpty()) {

TreeNode current = queue.poll();

if(current.left != null) {

parent\_track.put(current.left, current);

queue.offer(current.left);

}

if(current.right != null) {

parent\_track.put(current.right, current);

queue.offer(current.right);

}

}

}

public List<Integer> distanceK(TreeNode root, TreeNode target, int k) {

Map<TreeNode, TreeNode> parent\_track = new HashMap<>();

markParents(root, parent\_track);

Map<TreeNode, Boolean> visited = new HashMap<>();

Queue<TreeNode> queue = new LinkedList<TreeNode>();

queue.offer(target);

visited.put(target, true);

int curr\_level = 0;

while(!queue.isEmpty()) { /\*Second BFS to go upto K level from target node and using our hashtable info\*/

int size = queue.size();

if(curr\_level == k) break;

curr\_level++;

for(int i=0; i<size; i++) {

TreeNode current = queue.poll();

if(current.left != null && visited.get(current.left) == null) {

queue.offer(current.left);

visited.put(current.left, true);

}

if(current.right != null && visited.get(current.right) == null ) {

queue.offer(current.right);

visited.put(current.right, true);

}

if(parent\_track.get(current) != null && visited.get(parent\_track.get(current)) == null) {

queue.offer(parent\_track.get(current));

visited.put(parent\_track.get(current), true);

}

}

}

List<Integer> result = new ArrayList<>();

while(!queue.isEmpty()) {

TreeNode current = queue.poll();

result.add(current.val);

}

return result;

}

}

T.c:= O(N), S.C:= O(N)