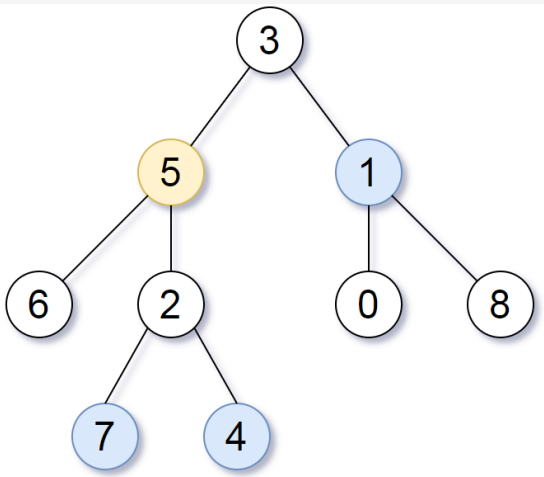
给定一个二叉树（具有根结点 `root`）， 一个目标结点 `target` ，和一个整数值 `K` 。
返回到目标结点 `target` 距离为 `K` 的所有结点的值的列表。 答案可以以任何顺序返回。

示例 1：

输入： `root = [3,5,1,6,2,0,8,null,null,7,4], target = 5, K = 2`

输出： `[7,4,1]`

解释：
所求结点为与目标结点（值为 `5`）距离为 `2` 的结点，
值分别为 `7`，`4`，以及 `1`



注意，输入的 `"root"` 和 `"target"` 实际上是树上的结点。
上面的输入仅仅是对这些对象进行了序列化描述。

提示：

1. 给定的树是非空的，且最多有 `K` 个结点。
2. 树上的每个结点都具有唯一的值 `0 <= node.val <= 500` 。
3. 目标结点 `target` 是树上的结点。
4. `0 <= K <= 1000`.

思路，该题的难点是，二叉树是单项链接的，子节点不能访问到父节点，那么我们先用map标记子节点的父节点，然后用深度遍历或者广度遍历进行距离的遍历，并将合适长度的添加进返回数组，需要了解的是，若是我是从子节点到父节点，那么父节点遍历时，不能在去子节点，防止数据重复。

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    void setParents(TreeNode* root,map<TreeNode*,TreeNode*> &bj,TreeNode*
parents){
        if(root==NULL){
            return;
        }
        if(parents!=NULL){
            bj[root]=parents;
        }
        if(root->left!=NULL){
            setParents(root->left,bj,root);
        }
        if(root->right!=NULL){
            setParents(root->right,bj,root);
        }
    }
    void dfs(TreeNode* root,TreeNode* parents,map<TreeNode*,TreeNode*>
&bj,vector<int> &result,int k){
        if(k==0){
            result.push_back(root->val);
            return;
        }
        if(root->left!=NULL&&root->left!=parents){
            dfs(root->left,root,bj,result,k-1);
        }
```

```
        if(root->right!=NULL&&root->right!=parents){
            dfs(root->right,root,bj,result,k-1);
        }
        if(bj[root]!=NULL&&bj[root]!=parents){
            dfs(bj[root],root,bj,result,k-1);
        }
    }
    vector<int> distanceK(TreeNode* root, TreeNode* target, int K) {
        map<TreeNode*,TreeNode*> parents;
        setParents(root,parents,NULL);
        vector<int> result;
        dfs(target,NULL,parents,result,K);
        return result;
    }
};
```