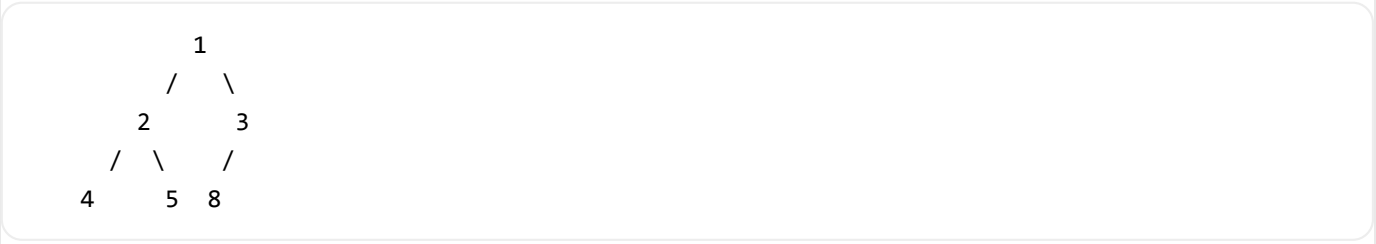
Print nodes at k distance from root

Given a root of a tree, and an integer k. Print all the nodes which are at k distance from root.

For example, in the below tree, 4, 5 & 8 are at distance 2 from root.



The problem can be solved using recursion. Thanks to eldho for suggesting the solution.

C

```
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
   int data;
   struct node* left;
   struct node* right;
};

void printKDistant(node *root , int k)
{
   if(root == NULL)
      return;
   if( k == 0 )
   {
      printf( "%d ", root->data );
      return ;
   }
   else
   {
      printKDistant( root->left, k-1 ) ;
      printKDistant( root->right, k-1 ) ;
   }
```

```
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode(int data)
{
  struct node* node = (struct node*)
                       malloc(sizeof(struct node));
  node->data = data;
  node->left = NULL;
  node->right = NULL;

  return(node);
}

/* Driver program to test above functions*/
int main()
{

  /* Constructed binary tree is
            1
          /   \
        2      3
      /  \    /
    4     5  8
  */
  struct node *root = newNode(1);
  root->left        = newNode(2);
  root->right       = newNode(3);
  root->left->left  = newNode(4);
  root->left->right = newNode(5);
  root->right->left = newNode(8);

  printKDistant(root, 2);

  getchar();
  return 0;
}
```

Run on IDE

Java

```
// Java program to print nodes at k distance from root

// A binary tree node
class Node {

    int data;
    Node left, right;

    Node(int item) {
        data = item;
        left = right = null;
    }
}

class BinaryTree {

    static Node root;

    void printKDistant(Node node, int k) {
```

```
        if (node == null) {
            return;
        }
        if (k == 0) {
            System.out.print(node.data + " ");
            return;
        } else {
            printKDistant(node.left, k - 1);
            printKDistant(node.right, k - 1);
        }
    }

    public static void main(String args[]) {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);
        tree.root.right.left = new Node(8);

        tree.printKDistant(root, 2);
    }
}

// This code has been contributed by Mayank Jaiswal
```

Run on IDE

The above program prints 4, 5 and 8.

Time Complexity: O(n) where n is number of nodes in the given binary tree.

Please write comments if you find the above code/algorithm incorrect, or find better ways to solve the same problem.

67 Comments Category: Trees

Related Posts:

- Check if removing an edge can divide a Binary Tree in two halves
- Check sum of Covered and Uncovered nodes of Binary Tree
- Lowest Common Ancestor in a Binary Tree | Set 2 (Using Parent Pointer)
- Construct a Binary Search Tree from given postorder
- BFS vs DFS for Binary Tree
- Maximum difference between node and its ancestor in Binary Tree
- Inorder Non-threaded Binary Tree Traversal without Recursion or Stack
- Check if leaf traversal of two Binary Trees is same?