Nodes at a distance k in binary tree

You are given a function printKDistanceNodes which takes in a root node of a binary tree, a start node and an integer K. Complete the function to print the value of all the nodes (one-per-line) which are a K distance from the given start node in sorted order. Distance can be upwards or downwards.

algorithm | data-structures

asked **Oct 23 '11 at 8:02**

princess of persia
494 3 10
60% accept rate

1 What have you tried? – quasiverse Oct 23 '11 at 8:04

1 Is this a homework? – n.m. Oct 23 '11 at 8:11

feedback

4 Answers

There is at most one node at distance K which upwards - just start from the start node and move up along parents for K steps. Add this to a sorted data structure.

Then you need to add the downward nodes. To do that you can do a BFS with queue, where you store the depth together with the node when you insert it in the queue (the starting node is at level 0, it's children at level 1 and so on). Then when you pop the nodes if they are at level K add them to the sorted data structure. when you start poping nodes at level K+1 you can stop.

Finally print the nodes from the sorted data structure (they will be sorted).

EDIT: If there is no parent pointer:

Write a recursive function `int Go(Node node)` , which returns the depth of the start node with respect to the passed in node and -1 if the subtree of node doesn't contain start. The function will find the K-th parent as a side effect. Pseudo code:

```
static Node KthParent = null;
static Node start = ...;
static int K = ...;

int Go(Node node) {
    if (node == start) return 0;

    intDepth = -1;

    if(node.LeftChild != null) {
        int leftDepth = Go(node.LeftChild);
        if(leftDepth >= 0) intDepth = leftDepth+1;
    }
    if (intDepth < 0 && node.rightChild != null) {
        int rightDepth = Go(node.RightChild);
        if(rightDepth >= 0) intDepth = rightDepth+1;
    }

    if(intDepth == K) KthParent = node;

    return intDepth;
}
```

edited **Oct 23 '11 at 9:11**

answered **Oct 23 '11 at 8:11**

Petar Ivanov

13.8k 4 20

there is no parent pointer – princess of persia Oct 23 '11 at 8:30

feedback

```
private static int printNodeAtK(Node root, Node start, int k, boolean found){
        if(root != null){
            if(k == 0 && found){
                System.out.println(root.data);
            }
            if(root==start || found == true){
                int leftd = printNodeAtK(root.left, start, k-1, true);
                int rightd = printNodeAtK(root.right,start,k-1,true);
                return 1;
            }else{
                int leftd = printNodeAtK(root.left, start, k, false);
                int rightd = printNodeAtK(root.right,start,k,false);
                if(leftd == k || rightd == k){
                    System.out.println(root.data);
                }
                if(leftd != -1 && leftd > rightd){
                    return leftd+1;
                }else if(rightd != -1 && rightd>leftd){
                    return rightd+1;
                }else{
                    return -1;
                }
            }

        }
        return -1;
    }
```

answered **Jan 5 at 20:23**

jayesh
1

would you mind explaining ? – princess of persia Jan 6 at 12:53

feedback

```
private void printNodeAtN(Node root, Node start, int k) {
    if (root != null) {
        // calculate if the start is in left or right subtree - if start is
        // root this variable is null
        Boolean left = isLeft(root, start);
        int depth = depth(root, start, 0);

        if (depth == -1)
            return;
        printNodeDown(root, k);

        if (root == start)
            return;

        if (left) {
            if (depth > k) {
                // print the nodes at depth-k level in left tree
                printNode(depth - k - 1, root.left);
            } else if (depth < k) {
                // print the nodes at right tree level k-depth
                printNode(k - depth - 1, root.right);
            } else {
                System.out.println(root.data);
```

```
            }
        } else {
            // similar if the start is in right subtree
            if (depth > k) {
                // print the nodes at depth-k level in left tree
                printNode(depth - k - 1, root.right);
            } else if (depth < k) {
                // print the nodes at right tree level k-depth
                printNode(k - depth - 1, root.left);
            } else {
                System.out.println(root.data);
            }
        }
    }
}

    // print the nodes at depth - "level" from root
void printNode(int level, Node root) {
    if (level == 0 && root != null) {
        System.out.println(root.data);
    } else {
        printNode(level - 1, root.left);
        printNode(level - 1, root.right);
    }

}

// print the children of the start
void printNodeDown(Node start, int k) {
    if (start != null) {
        if (k == 0) {
            System.out.println(start.data);
        }
        printNodeDown(start.left, k - 1);
```

edited **Feb 7 at 12:15**

answered **Feb 7 at 1:41**

Vinay Kumar

1 1

feedback

---

typedef struct node

{

```
int data;
struct node *left;
struct node *right;
```

}node;

void printkdistanceNodeDown(node *n, int k)

{

```
 if(!n)
   return ;

 if(k==0)
 {
    printf("%d\n",n->data);
    return;
 }

 printkdistanceNodeDown(n->left,k-1);
 printkdistanceNodeDown(n->right,k-1);
```

}

void printkdistanceNodeDown_fromUp(node* target ,int *k)

{

```
 if(!target)
   return ;

   if(*k==0)
   {
      printf("%d\n",target->data);
      return;
   }
   else
   {
       int val=*k;
       printkdistanceNodeDown(target,val-1);
   }

}
```

int printkdistanceNodeUp(node* root, node* n , int* k)

```
{

 if(!root)
   return 0;

 if(root->data==n->data)
   return 1;

 int pl=printkdistanceNodeUp(root->left,n,k);
 int pr=printkdistanceNodeUp(root->right,n,k);

  if(pl )
  {
   (*k)--;

    if(*k==0)
     printf("%d\n",root->data);
    else
     printkdistanceNodeDown_fromUp(root->right,k);
    return 1;
 }

 if(pr )
 {
    (*k)--;
   if(*k==0)
     printf("%d\n",root->data);
   else
     printkdistanceNodeDown_fromUp(root->left,k);
    return 1;
 }

  return 0;

}
```

void printkdistanceNode(node* root, node* n , int k)

```
{

 if(!root)
   return ;

  int val=k;
  printkdistanceNodeUp(root,n,&k);
  printkdistanceNodeDown(n,val);

}
```

caller function : `printkdistanceNode(root,n,k);`

output will print all the nodes at a distance k from given node upward and downward.

edited **Feb 11 at 8:42**

answered **Feb 10 at 17:58**

Shirdi_Sai

1 1

feedback

Not the answer you're looking for? Browse other questions tagged algorithm data-structures or ask your own question.

question feed