Custom Tree Problem

You are given a set of links, e.g.

```
a ---> b
b ---> c
b ---> d
a ---> e
```

Print the tree that would form when each pair of these links that has the same character as start and end point is joined together. You have to maintain fidelity w.r.t. the height of nodes, i.e. nodes at height n from root should be printed at same row or column. For set of links given above, tree printed should be –

```
-->a
   |-->b
   |   |-->c
   |   |-->d
   |-->e
```

Note that these links need not form a single tree; they could form, ahem, a forest. Consider the following links

```
a ---> b
a ---> g
b ---> c
c ---> d
d ---> e
c ---> f
z ---> y
y ---> x
x ---> w
```

The output would be following forest.

```
-->a
   |-->b
   |   |-->c
   |   |   |-->d
```

```
    |   |   |   |-->e
    |   |   |-->f
    |-->g

-->z
    |-->y
    |   |-->x
    |   |   |-->w
```

You can assume that given links can form a tree or forest of trees only, and there are no duplicates among links.

Solution: The idea is to maintain two arrays, one array for tree nodes and other for trees themselves (we call this array forest). An element of the node array contains the TreeNode object that corresponds to respective character. An element of the forest array contains Tree object that corresponds to respective root of tree.

It should be obvious that the crucial part is creating the forest here, once it is created, printing it out in required format is straightforward. To create the forest, following procedure is used –

```
Do following for each input link,
1. If start of link is not present in node array
     Create TreeNode objects for start character
     Add entries of start in both arrays.
2. If end of link is not present in node array
     Create TreeNode objects for start character
     Add entry of end in node array.
3. If end of link is present in node array.
     If end of link is present in forest array, then remove it
     from there.
4. Add an edge (in tree) between start and end nodes of link.
```

It should be clear that this procedure runs in linear time in number of nodes as well as of links – it makes only one pass over the links. It also requires linear space in terms of alphabet size.

Following is Java implementation of above algorithm. In the following implementation characters are assumed to be only lower case characters from 'a' to 'z'.

```
// Java program to create a custom tree from a given set of links.

// The main class that represents tree and has main method
public class Tree {

    private TreeNode root;

    /* Returns an array of trees from links input. Links are assumed to
       be Strings of the form "<s> <e>" where <s> and <e> are starting
       and ending points for the link. The returned array is of size 26
       and has non-null values at indexes corresponding to roots of trees
       in output */
    public Tree[] buildFromLinks(String [] links) {

        // Create two arrays for nodes and forest
        TreeNode[] nodes = new TreeNode[26];
        Tree[] forest = new Tree[26];

        // Process each link
        for (String link : links) {

            // Find the two ends of current link
            String[] ends = link.split(" ");
            int start = (int) (ends[0].charAt(0) - 'a'); // Start node
            int end   = (int) (ends[1].charAt(0) - 'a'); // End node

            // If start of link not seen before, add it two both arrays
            if (nodes[start] == null)
            {
                nodes[start] = new TreeNode((char) (start + 'a'));

                // Note that it may be removed later when this character is
                // last character of a link. For example, let we first see
                // a--->b, then c--->a.  We first add 'a' to array of trees
                // and when we see link c--->a, we remove it from trees array.
                forest[start] = new Tree(nodes[start]);
            }

            // If end of link is not seen before, add it to the nodes array
            if (nodes[end] == null)
                nodes[end] = new TreeNode((char) (end + 'a'));

            // If end of link is seen before, remove it from forest if
            // it exists there.
            else forest[end] = null;

            // Establish Parent-Child Relationship between Start and End
            nodes[start].addChild(nodes[end], end);
        }
        return forest;
    }

    // Constructor
    public Tree(TreeNode root) { this.root = root;  }

    public static void printForest(String[] links)
    {
        Tree t = new Tree(new TreeNode('\0'));
        for (Tree t1 : t.buildFromLinks(links)) {
           if (t1 != null)
           {
              t1.root.printTreeIdented("");
```

```
                System.out.println("");
            }
        }
    }

    // Driver method to test
    public static void main(String[] args) {
        String [] links1 = {"a b", "b c", "b d", "a e"};
        System.out.println("------------ Forest 1 ----------------");
        printForest(links1);

        String [] links2 = {"a b", "a g", "b c", "c d", "d e", "c f",
                            "z y", "y x", "x w"};
        System.out.println("------------ Forest 2 ----------------");
        printForest(links2);
    }
}

// Class to represent a tree node
class TreeNode {
    TreeNode []children;
    char c;

    // Adds a child 'n' to this node
    public void addChild(TreeNode n, int index) { this.children[index] = n;}

    // Constructor
    public TreeNode(char c) { this.c = c;  this.children = new TreeNode[26];}

    // Recursive method to print indented tree rooted with this node.
    public void printTreeIdented(String indent) {
            System.out.println(indent + "-->" + c);
            for (TreeNode child : children) {
              if (child != null)
                child.printTreeIdented(indent + "   |");
            }
    }
}
```

Output:

```
------------ Forest 1 ----------------
-->a
   |-->b
   |   |-->c
   |   |-->d
   |-->e

------------ Forest 2 ----------------
```

```
-->a
   |-->b
   |   |-->c
   |   |   |-->d
   |   |   |   |-->e
   |   |   |-->f
   |-->g

-->z
   |-->y
   |   |-->x
   |   |   |-->w
```

Exercise:
In the above implementation, endpoints of input links are assumed to be from set of only 26 characters. Extend the implementation where endpoints are strings of any length.

This article is contributed by **Ciphe**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Spend less time debugging, more time coding.
Get Visual Studio Code FREE
Microsoft

8 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?

Atul
Just to mention question assumes that the links are provided in order they are traversed from top.
Guest
I am modifying the preorder traversal to print the tree using a tree information in the array tr[1024][3]
inp[][2] to take in the edges
tr[256][3] to store the tree parent and two children
chk[256][3] to find out the root nodes of the forest trees.

```
#include
using namespace std;
char tr[256][3] = {0};
char inp[1024][2] = {0};
int chk[256][2] = {0};
void in(char arr[][2], int n)
{ for(int i = 0; i < n ; i++)
{ inp[i][0] = arr[i][0];
inp[i][1] = arr[i][1];
}
}
void populate_tree(int n)
{ for(int i = 0; i < n; i++)
{ tr[int(inp[i][0])][0] = inp[i][0];
if(tr[int(inp[i][0])][1] == 0)
{ tr[int(inp[i][0])][1] = inp[i][1];
}
else
{ tr[int(inp[i][0])][2] = inp[i][1];
}
}
}
void print_tree(char c,int d)
{ if(c == 0)
{ return;
}
if(tr[int(c)][1] == 0 && tr[int(c)][2] == 0)
{
for(int i = 0 ; i <d ; i++)
{ cout<<" |";
}
cout<"<<c<<endl;
return;
}
for(int i = 0 ; i <d ; i++)
{ cout<<" |";
}
cout<"<<c<<endl;
print_tree(tr[int(c)][1],d+1);
print_tree(tr[int(c)][2],d+1);
}
int main()
{ char c[][2] = {{'a','b'},{'a','g'},{'b','c'},{'c','d'},{'d','e'},{'c','f'},{'z','y'},{'y','x'},{'x','w'}};
int len = (sizeof(c)/2*sizeof(char));
in(c,len);
populate_tree(len);
for(int i = 0 ; i < len; i++)
{ chk[int(inp[i][0])][0] = 1;
chk[int(inp[i][1])][1] = 1;
}
int h;
for( h = 0 ; h < 256; h++)
{ if(chk[h][0] != 0 && chk[h][1] == 0 && (tr[h][1] || tr[h][2]))
{ print_tree(char(h),0);
}
}
// print_tree(char(h),0);
return 0;
}
```

jeswanth

my code that works like preorder like traversal of the Binary tree .. can be found at http://ideone.com/HPB2wE
Assuming only two edges are possible from each node.
inp[1024][2] to take in the edges
 tr[256][3] to store the tree parent and two children
chk[256][3] to find out the root nodes of the forest trees.
// uses three 2-D arrays

Nitin Sharma

I have taken an array of nodes which will contain address of root of a tree.
ALGORITHM :-
1.call search function and pass it root and start point.
2.If root is null , then create a node and put its entry in node array.
3.Now add the end point of this start point as a child of start point.
4.Repeat above points.

```
/*CODE*/
/*CUSTOM TREE PROBLEM*/
#include
#include
typedef struct node
{
char ch;
struct node *left,*right;
}node;
node* newnode(char value)
{
node *tmp;
tmp = (node *)malloc(sizeof(node *));
if(tmp == NULL)
{
printf("Memory Underflow.n");
exit(0);
}
tmp->ch = value;
tmp->left = NULL;
tmp->right = NULL;
}
node * search(node *root,char value)
{
node *ptr;
if(root == NULL)
{
ptr = NULL;
}
else if(root->ch == value)
{
ptr = root;
}
else
{
ptr = search(root->left,value);
if(ptr == NULL)
{
ptr = search(root->right,value);
}
}
return ptr;
}
void print(node *root, int tab)
{
if(root == NULL)
{
return;
}
int i=0;
for(i=0;i%cn",root->ch);
print(root->left,tab+1);
print(root->right,tab+1);
}
void main()
{
int i,j,k=0;
node* root[10]={NULL};
node *found;
char array[2][9] = {{'a','a','b','c','d','c','z','y','x'},{'b','g','c','d','e','f','y','x','w'}};
for(i=0;ileft == NULL)
{
found->left = newnode(array[1][i]);
}
else
{
found->right = newnode(array[1][i]);
}
}
}
for(i=1;i<=k;i++)
{
print(root[i],0);
}
}
```

http://anshulsalvo.blogspot.com Anshul

```
package geeksforgeeks.random;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
public class CustomTree {
public static HashMap> map = new HashMap>();
public static Set childs = new HashSet();
public static Set parents = new HashSet();
public static Set mainparents = new HashSet();
public static void printList(String parent, ArrayList childList,String spc){
if(childList == null)
return;
for (Iterator iterator = childList.iterator(); iterator.hasNext();) {
String child = (String) iterator.next();
System.out.println( spc + parent +"—>"+ child);
if(map.get(child) != null)
printList(child, map.get(child),spc+" ");
}
}
public static void generateMap (String arr[]){
for (int i = 0; i < arr.length; i++) { String parent = arr[i].split(",")[0]; String child = arr[i].split(",")[1];
if(!map.containsKey(parent)){ ArrayList l = new ArrayList();
l.add(child);
map.put(parent, l);
}
else{
map.get(parent).add(child);
}
parents.add(parent);
childs.add(child);
}
for (Iterator iterator = parents.iterator(); iterator.hasNext();) {
String parent = (String) iterator.next();
if(!childs.contains(parent))
mainparents.add(parent);
}
for (Iterator iterator = mainparents.iterator(); iterator.hasNext();) {
String parent = (String) iterator.next();
printList(parent,map.get(parent),"");
}
}
public static void main(String[] args) {
String arr [] = new String[9];
arr[0] = "a,b";
arr[1] = "a,g";
arr[2] = "b,c";
arr[3] = "c,d";
arr[4] = "d,e";
arr[5] = "c,f";
arr[6] = "z,y";
arr[7] = "y,x";
arr[8] = "x,w";
generateMap(arr);
}
}
```

http://www.spiderman.com spiderman

How does it ensure the first followed children of each node is with maximum depth while printing each tree ?
In the first example, what will be the output if the input is String [] links1 = {"a b", "a e", "b c", "b d"};

```
/* Paste your code here (You may delete these lines if not writing code) */
```