Data Structures

Solve any problem to achieve a rank

View Leaderboard

Topics: Binary/ N-ary Trees

Binary/ N-ary Trees

TUTORIAL PROBLEMS

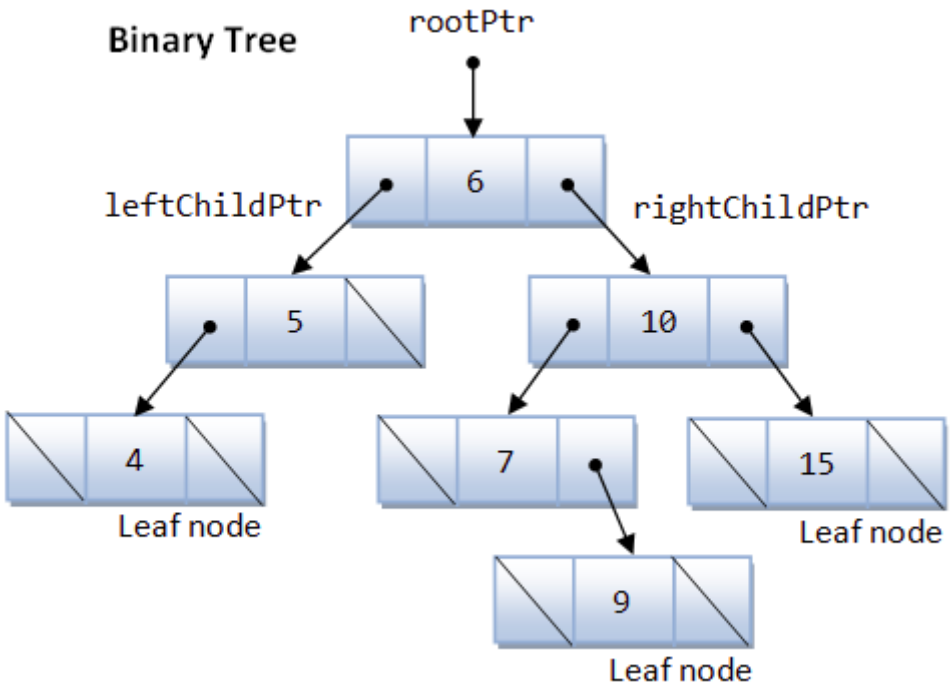
A binary tree is a structure comprising nodes, where each node has the following 3 components:

1. Data element: Stores any kind of data in the node
2. Left pointer: Points to the tree on the left side of node
3. Right pointer: Points to the tree on the right side of the node

As the name suggests, the **data** element stores any kind of data in the node.
The **left** and **right** pointers point to binary trees on the left and right side of the node respectively.

If a tree is empty, it is represented by a **null** pointer.

The following image explains the various components of a tree.



Commonly-used terminologies

- **Root**: Top node in a tree
- **Child**: Nodes that are next to each other and connected downwards
- **Parent**: Converse notion of child
- **Siblings**: Nodes with the same parent
- **Descendant**: Node reachable by repeated proceeding from parent to child
- **Ancestor**: Node reachable by repeated proceeding from child to parent.
- **Leaf**: Node with no children
- **Internal node**: Node with at least one child
- **External node**: Node with no children

Structure code of a tree node

In programming, trees are declared as follows:

```
    struct node
    {
          int data;                    //Data element
          struct node * left;            //Pointer to left node
          struct node * right;           //Pointer to right node
    };
```

Creating nodes

Simple node

```
    struct node root;
```

Pointer to a node

```
struct node * root;
root=(node * )malloc(sizeof(node));
```

In this case, you must explicitly allocate the memory of the node type to the pointer (preferred method).

Utility function returning node

```
struct node * newnode(int element)
{
     struct node * temp=(node * )malloc(sizeof(node));
     temp->data=element;
     temp->left=temp->right=NULL;
     return temp;
}
```

Maximum depth/height of a tree

The idea is to do a post-order traversal and maintain two variables to store the left depth and right depth and return max of both the depths.

```
int maxDepth(struct node* node)
{
     if (node==NULL)
          return 0;
     else
     {
           /* compute the depth of each subtree */
           int lDepth = maxDepth(node->left);
           int rDepth = maxDepth(node->right);

           /* use the larger one */
           if (lDepth > rDepth)
                 return(lDepth+1);
           else
                return(rDepth+1);
    }
}
```

Time complexity

$O(n)$

Application of trees

1. a Manipulate hierarchical data
2. Make information easy to search (see tree traversal)
3. Manipulate sorted lists of data

4. Use as a workflow for compositing digital images for visual effects
5. Use in router algorithms

Contributed by: Vaibhav Tulsyan

Did you find this tutorial helpful? YES NO

TEST YOUR UNDERSTANDING

Binary Tree

Given a binary tree which has T nodes, you need to find the diameter of that binary tree. The diameter of a tree is the number of nodes on the longest path between two leaves in the tree.

Input:
First line contains two integers, T and X, number of nodes in the tree and value of the root.
Next $2 \times (T-1)$ lines contain details of nodes.
Each detail of node contains two lines. First lines contains a string and second line contains an integer, which denotes the path of the node and the value of the node respectively.

String consists of only L or R. L denotes left child and R denotes right child. (Look at the sample explanation for more details)

Output:
Print the diameter of the binary tree.

Constraints:
$1 \leq T \leq 20$
$1 \leq valueofnodes \leq 20$

SAMPLE INPUT

```
5 1
L
2
R
3
LL
4
LR
5
```

SAMPLE OUTPUT

4

Enter your code or Upload your code as file. Save C (gcc 4.8.2)

```c
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}

```

1:1

Provide custom input

COMPILE & TEST SUBMIT

Press Ctrl-space for autocomplete suggestions. POWERED BY code table

Need Help ?

In case you feel you are stuck with the problem, you can view our editorial.
Remember this is just to help you out, in order to learn you should try your best.

VIEW EDITORIAL

COMMENTS (18) SORT BY: Relevance

Join Discussion...

Cancel Post

Branislav Petrovič 2 months ago

I think, there is wrong output for input#2. Longest path is for example between nodes (5,7) or (5,11). There are 8 nodes on these paths.

15 1
LLL
2
LL
3
L
4
RR
5
R
6
LLRLL
7
LLR
8
LLRL
9
LLRLR
10
LRRRR
11
LRRL
12
LRRR
13
LRR
14
LR
15

▲ 4 votes ● Reply ● Message ● Permalink

Sandeep Gupta a month ago

U are wrong.

▲ 0 votes ● Reply ● Message ● Permalink

Sandeep Gupta a month ago

Draw the graph, and you will know.

▲ 0 votes ● Reply ● Message ● Permalink

Branislav Petrovič a month ago

Thanks, I know now.

▲ 0 votes ● Reply ● Message ● Permalink

homputr 3 months ago

> struct node* temp = (struct node*) malloc(sizeof(node));
should be:
"struct node* temp = (struct node*) malloc(sizeof(struct node));"

▲ 2 votes ● Reply ● Message ● Permalink

Joker Knight 3 months ago

why is the above code giving runtime error

▲ 0 votes ● Reply ● Message ● Permalink

homputr 3 months ago

compilation error

▲ 0 votes ● Reply ● Message ● Permalink

hac_123 2 months ago

Without using any pointer any tree concept

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
int n,k,l;
cin>>n>>k;
string a[20];

for(int i=0;i<n-1;i++)
{
cin>>a[i]>>l;
}
int count = 0;
for(int i=0;i<n-1;i++)
for(int j=0;j<n-1;j++)
for(int u=0;u<a[i].length()&&u<a[j].length();u++)
{
if(i == j)
break;
if( a[i][u] == a[j][u])
continue;
else
{
if(count < (a[i].length()+a[j].length()-2*u + 1) )
{
count = a[i].length()+a[j].length()-2*u + 1;

}
break;
}

}
cout<<count<<endl;
return 0;
}
```

▲ 1 vote ● Reply ● Message ● Permalink

Siddhant a month ago

Can you explain your code?

▲ 0 votes ● Reply ● Message ● Permalink

Siddhant a month ago

How did you think of it?

▲ 0 votes ● Reply ● Message ● Permalink

Sonali Agrawal 15 days ago

Note:
1. LL can come before L in input
2. diameter may not include root (input#2)

▲ 1 vote ● Reply ● Message ● Permalink

Rohit Chhillar 3 months ago

i dint get how the value of root is updating with ptr in this editorial.

▲ 0 votes ● Reply ● Message ● Permalink

Branislav Petrovič Edited a month ago

If pointer of node is ptr_name then value of left part the node you get by command:
prt_name->left

My hint:
struct node * root_ptr = newnode(root);

```
struct node * temp_node_ptr;
// in loop - for every "word" (L,RLR,LLR,LL,...)
temp_node_ptr=root_ptr;
for(j=0;arr[j];j++)
{
if(arr[j]=='L' && arr[j+1]!='\0')
temp_node_ptr=temp_node_ptr->left;
if(arr[j]=='R' && arr[j+1]!='\0')
temp_node_ptr=temp_node_ptr->right;
if(arr[j]=='L' && arr[j+1]=='\0')
temp_node_ptr->left=newnode(k);
if(arr[j]=='R' && arr[j+1]=='\0')
temp_node_ptr->right=newnode(k);
}
```

0 votes ● Reply ● Message ● Permalink

shubhra garg 2 months ago

The value of lDepth and rDepth should be incremented. How come "Maximum depth/height of a tree"section is written not properly?

0 votes ● Reply ● Message ● Permalink

Branislav Petrovič 2 months ago

Hi, there are used recursion in function maxDepth. You must put top node of tree/subtree in which you want to find depth. You must put pointer of node to the argument of the function (maxDepth(pointer_of_node)).

0 votes ● Reply ● Message ● Permalink

Salil Kaul 22 days ago

I am not able to understand how the tree is laid out what is the meaining of LL ?? e.g. for the given data 2 is the left child of root , 3 is the right child of node having value 2 what is the meaning of LL in respect to 3 ? Is four the right child or the left child ??

0 votes ● Reply ● Message ● Permalink

Akash Srivastava 19 days ago

```
/* IMPORTANT: Multiple classes and nested static classes are supported */
/*
* uncomment this if you want to read input.
//imports for BufferedReader
import java.io.BufferedReader;
import java.io.InputStreamReader;
//import for Scanner and other utility classes
import java.util.*;
*/
import java.util.Scanner;
class TestClass {


public static int[] tree = new int[1048577];
public static int res = 0;
public static void main(String[] args) {


Scanner sc = new Scanner(System.in);

int n = sc.nextInt();
tree[1] = sc.nextInt();

for(int i=1 ; i < n ; i++){

String code = sc.next();
int index = getIndex(code);
if(index <= 1048576){
tree[index] = sc.nextInt();
```

```
}else{
System.out.println("ERROR : index == " + index);
}

}

maxDia(1);
System.out.println(res);
}
private static void maxDia(int root) {

if(tree[root] == 0){
return;
}

int ml = getHeight(2*root);
int mr = getHeight(2*root+1);

int restemp = ml+mr+1;
if(res < restemp){
res = restemp;
maxDia(2*root);
maxDia(2*root +1);
}else{
return;
}

}
private static int getHeight(int i) {
if(tree[i] == 0){
return 0;
}


int lh = getHeight(2*i);
int rh = getHeight(2*i + 1);

if(lh > rh){
return lh+1;
}else{
return rh +1;
}
}
private static int getIndex(String i) {
int index = 1;

for(Character c : i.toCharArray()){
if(c.equals('L')){
index = index*2;
}else{
index = index*2+1;
}
}
return index;
}
}
```

▲ 0 votes ● Reply ● Message ● Permalink

rossion 11 days ago

the diameter of a tree: https://www.youtube.com/watch?v=i9nVJDr4HmA

▲ 0 votes ● Reply ● Message ● Permalink