Algorithms

Solve any problem to achieve a rank

View Leaderboard

Topics: Depth First Search

Depth First Search

TUTORIAL PROBLEMS VISUALIZER BETA

Depth First Search (DFS)

The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

This recursive nature of DFS can be implemented using stacks. The basic idea is as follows:
Pick a starting node and push all its adjacent nodes into a stack.
Pop a node from stack to select the next node to visit and push all its adjacent nodes into a stack.
Repeat this process until the stack is empty. However, ensure that the nodes that are visited are marked. This will prevent you from visiting the same node more than once. If you do not mark the nodes that are visited and you visit the same node more than once, you may end up in an infinite loop.

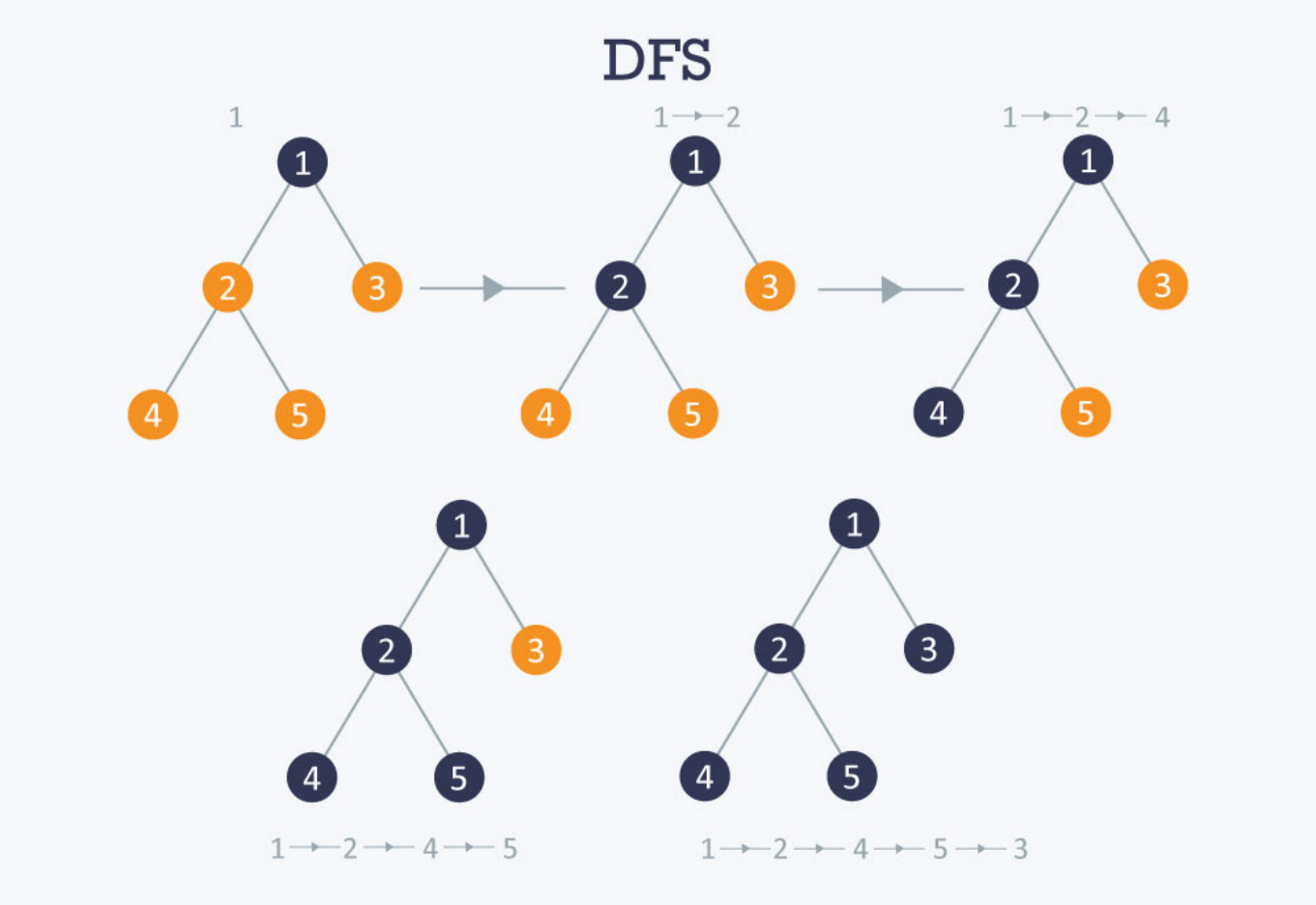
Pseudocode

```
    DFS-iterative (G, s):                                    //Where G is
graph and s is source vertex
      let S be stack
```

```
        S.push( s )               //Inserting s in stack
        mark s as visited.
        while ( S is not empty):
            //Pop a vertex from stack to visit next
            v  =  S.top( )
           S.pop( )
           //Push all the neighbours of v in stack that are not visited
          for all neighbours w of v in Graph G:
              if w is not visited :
                        S.push( w )
                       mark w as visited


DFS-recursive(G, s):
    mark s as visited
    for all neighbours w of s in Graph G:
        if w is not visited:
            DFS-recursive(G, w)
```

The following image shows how DFS works.



Time complexity $O(V + E)$, when implemented using an adjacency list.

?

Applications

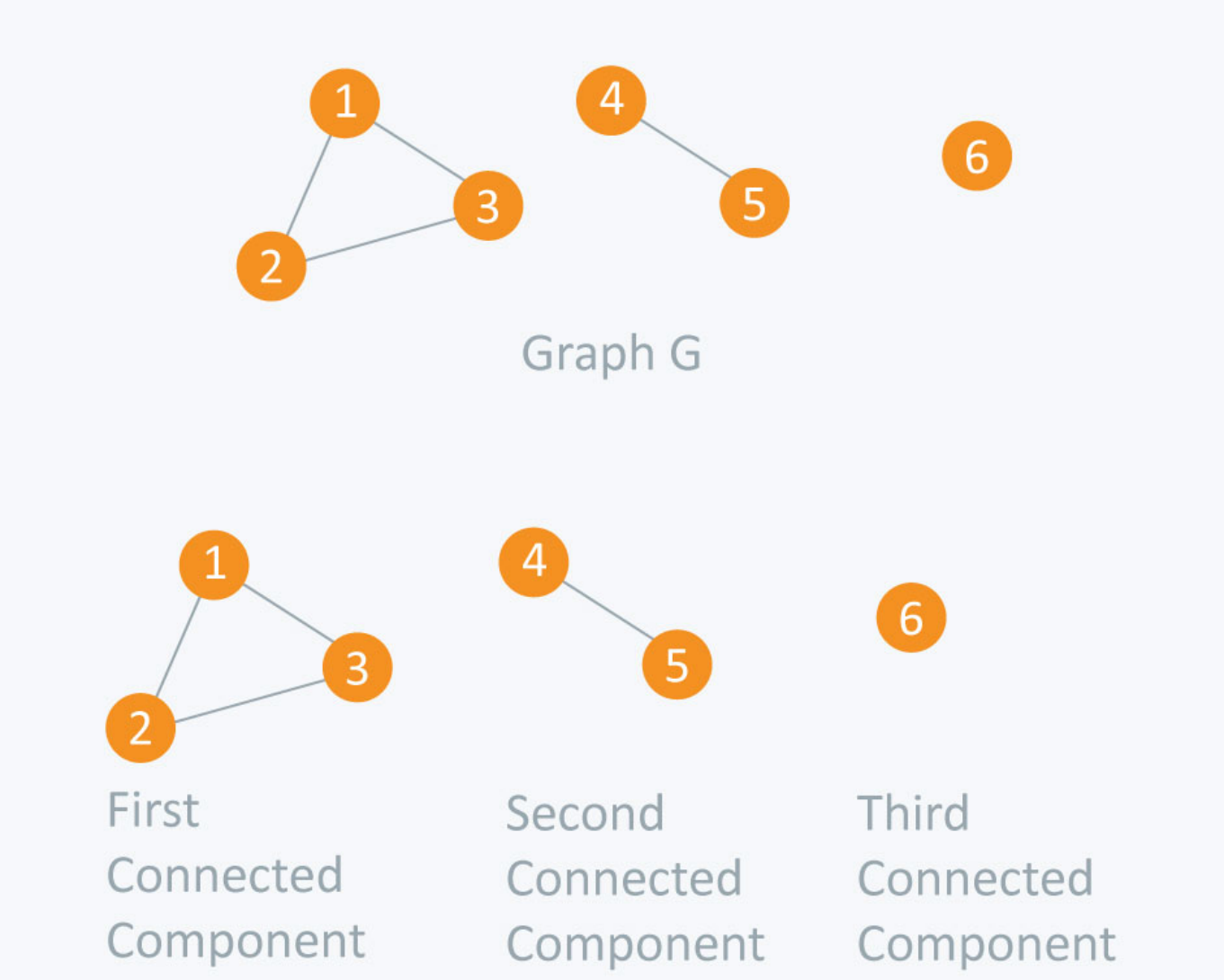
How to find connected components using DFS?

A graph is said to be disconnected if it is not connected, i.e. if two nodes exist in the graph such that there is no edge in between those nodes. In an undirected graph, a connected component is a set of vertices in a graph that are linked to each other by paths.

Consider the example given in the diagram. Graph G is a disconnected graph and has the following 3 connected components.

- First connected component is 1 -> 2 -> 3 as they are linked to each other
- Second connected component 4 -> 5
- Third connected component is vertex 6

In DFS, if we start from a start node it will mark all the nodes connected to the start node as visited. Therefore, if we choose any node in a connected component and run DFS on that node it will mark the whole connected component as visited.



Input File

```
6
4
1 2
2 3
1 3
4 5
```

Code

```
#include <iostream>
#include <vector>
 using namespace std;

 vector <int> adj[10];
 bool visited[10];

 void dfs(int s) {
     visited[s] = true;
     for(int i = 0;i < adj[s].size();++i)    {
      if(visited[adj[s][i]] == false)
          dfs(adj[s][i]);
     }
 }

 void initialize() {
     for(int i = 0;i < 10;++i)
      visited[i] = false;
 }

 int main() {
     int nodes, edges, x, y, connectedComponents = 0;
     cin >> nodes;                          //Number of nodes
     cin >> edges;                          //Number of edges
     for(int i = 0;i < edges;++i) {
      cin >> x >> y;
  //Undirected Graph
      adj[x].push_back(y);                    //Edge from vertex x to
vertex y
      adj[y].push_back(x);                    //Edge from vertex y to
vertex x
     }

     initialize();                           //Initialize all nodes a
```

```
not visited

        for(int i = 1;i <= nodes;++i) {
         if(visited[i] == false)     {
              dfs(i);
              connectedComponents++;
         }
        }
        cout << "Number of connected components: " << connectedComponents <<
endl;
        return 0;
    }
```

Output

Number of connected components: 3

Time complexity $O(V + E)$, when implemented using the adjacency list.

Contributed by: Prateek Garg

Did you find this tutorial helpful? YES NO

TEST YOUR UNDERSTANDING

Unreachable Nodes

You have been given a graph consisting of *N* nodes and *M* edges. The nodes in this graph are enumerated from *1* to *N* . The graph can consist of self-loops as well as multiple edges. This graph consists of a special node called the head node. You need to consider this and the entry point of this graph. You need to find and print the number of nodes that are unreachable from this head node.

Input Format

The first line consists of a 2 integers *N* and *M* denoting the number of nodes and edges in this graph. The next *M* lines consist of 2 integers *a* and *b* denoting an undirected edge between node *a* and *b*. The next line consists of a single integer *x* denoting the index of the head node.

Output Format:

You need to print a single integer denoting the number of nodes that are unreachable from the gi head node.

Constraints

$1 \leq N \leq 10^5$

$1 \leq M \leq 10^5$

$1 \leq x \leq N$

SAMPLE INPUT

```
10 10
8 1
8 3
7 4
7 5
2 6
10 7
2 8
10 9
2 10
5 10
2
```

SAMPLE OUTPUT

```
0
```

Enter your code or Upload your code as file. Save C (gcc 5.4.0)

```
/*
// Sample code to perform I/O:
#include <stdio.h>

int main(){
    int num;
    scanf("%d", &num);                          // Reading input from STDIN
    printf("Input number is %d.\n", num);       // Writing output to STDOUT
}

// Warning: Printing unwanted or ill-formatted data to output will cause the test cases t
*/

// Write your code here

```

?

1:1

Provide custom input

COMPILE & TEST SUBMIT

COMMENTS (71) SORT BY: Relevance

Login/Signup to Comment

Anand Kumar Edited a year ago

```
#include <bits/stdc++.h>
using namespace std;
vector <int> adj[100005];
bool visited[100005] = { false };
int ans = 0;
void dfs(int s){
visited[s] = true;
ans++;
for(int i = 0; i < adj[s].size(); ++i)
if(visited[adj[s][i]] == false)
dfs(adj[s][i]);
}
int main(){
int n, e, x, y;
cin >> n >> e;
for(int i = 0; i < e; i++){
cin >> x >> y;
adj[x].push_back(y);
adj[y].push_back(x);
}
int sd;
cin >> sd;
dfs(sd);
cout << (n - ans);
return 0;
}
```

12 votes ● Reply ● Message ● Permalink

Anurag Mishra 2 years ago

In first test case, Node 10 is not connecting to any node so all node is unreachable to it.Then the output of this case should be 99 instead of 0. Correct me if I am wrong.

4 votes ● Reply ● Message ● Permalink

Prateek Garg Moderator 2 years ago

You need to see the same for node 2, not for node 10.

1 vote ● Reply ● Message ● Permalink

Swati Lodha a year ago

Shouldn't the answer to first test case be 99? Node 10 is not connected to any other node all other nodes are unreachable to it.

0 votes ● Reply ● Message ● Permalink

goutham gundapu a year ago

first test case works if you consider the graph as undirected graph

▲ 1 vote ● Reply ● Message ● Permalink

Keshav Sharma 3 months ago

its connected to 52.... see the line number 11... graph is bi directional that is '52 is connected with 10' and '10 is connected with 52'

▲ 0 votes ● Reply ● Message ● Permalink

Keshav Sharma 3 months ago

its connected to 52.... see the line number 11... graph is bi directional that is '52 is connected with 10' and '10 is connected with 52'

▲ 0 votes ● Reply ● Message ● Permalink

maradana ravindra babu a year ago

it should be 99..

▲ 1 vote ● Reply ● Message ● Permalink

Honoya a year ago

that's true, first test case is 99

▲ 0 votes ● Reply ● Message ● Permalink

Sreerag Nair a year ago

He changed the input .....

▲ 0 votes ● Reply ● Message ● Permalink

goutham gundapu a year ago

first testcase works if you consider the graph as undirected graph

▲ 1 vote ● Reply ● Message ● Permalink

anirudd 8 months ago

thanks.

▲ 0 votes ● Reply ● Message ● Permalink

Devendra shaktawat 5 months ago

i have taken it as undirected , still my output is coming 99 , help

▲ 0 votes ● Reply ● Message ● Permalink

Mrinal Bhaskar 9 months ago

same thing was happening with me. but i realized my mistake. u must increase the length of the array taken to solve the question. like visited[100000]

▲ 0 votes ● Reply ● Message ● Permalink

Sandeep Dan a year ago

easy !!

▲ 1 vote ● Reply ● Message ● Permalink

xarilaos a year ago

apply dfs to head node, then count unvisited nodes.

▲ 3 votes ● Reply ● Message ● Permalink

Rohit a year ago

Finally , after a lot of struggle. AC

```
#include <iostream>
#include <bits/stdc++.h>
```

?

```
using namespace std;
bool mark[100000];
vector <int> v[100000];
int c=0;
void dfs(int s)
{
mark[s]=true;
for(vector<int>::iterator it=v[s].begin();it!=v[s].end();++it)
{
if(mark[*it]==false)dfs(*it);
}
}
int main()
{
memset(mark,false,sizeof(mark));
int nodes,edges,x,y,s;
cin>>nodes>>edges;
for(int i=1;i<=edges;i++)
{
cin>>x>>y;
v[x].push_back(y);
v[y].push_back(x);
}
cin>>s;
dfs(s);
for(int i=1;i<=nodes;i++)if(mark[i]==false)c++;
cout<<c;
return 0;
}
```

▲ 3 votes ● Reply ● Message ● Permalink

Mohammad Chand Alam a year ago

AC IN ONE GO!!

▲ 4 votes ● Reply ● Message ● Permalink

Abhay Nayar a year ago

In the tutorial above, shouldn't the definition of a disconnected graph be - "a graph where two nodes exist such that there is no path between them", instead of "edge" between them?

▲ 2 votes ● Reply ● Message ● Permalink

Ketul Shah a month ago

My solution for the given problem :-

```
#include <bits/stdc++.h>
#define N 100000
#define F(i,a,b) for(int i = (int) a ; i < (int) b ; ++i)
using namespace std;
int cnt = 0;
vector <int> adj[N];
bool visited[N];
void dfs(int s){
visited[s] = true;
F(i,0,adj[s].size()){
if(visited[adj[s][i]] == false){
cnt ++;
dfs(adj[s][i]);
}

}
}
```

?

```
int main(){
int nodes , edges , x, y , start ;
cin >> nodes >> edges ;
memset(visited , false , sizeof(visited[0]) * N);
F(i,0,edges){
cin >> x >> y;
adj[x].push_back(y);
adj[y].push_back(x);
}
cin >> start;
dfs(start);
cout << (nodes-cnt) - 1 << endl;
return 0;
}
```

▲ 1 vote ● Reply ● Message ● Permalink

Ketul Shah a month ago

In this line cout << (nodes-cnt) - 1 << endl;

-1 is for the head node which is already connected :)

▲ 1 vote ● Reply ● Message ● Permalink

Ravi sharma a year ago

Is the iterative algorithm correct ??

it does not give the same output as recursive?

▲ 1 vote ● Reply ● Message ● Permalink

Honoya ✎ Edited a year ago

you can do this problem in two ways:

-using stack

-using recursive

Both work

▲ 0 votes ● Reply ● Message ● Permalink

Mahidhar Bandaru a year ago

I don't think it is.

▲ 0 votes ● Reply ● Message ● Permalink

KUNWAR DESH DEEPAK SINGH a year ago

the iterative code is not working giving same result of bfs

▲ 1 vote ● Reply ● Message ● Permalink

Anuj Singh 2 months ago

use stack instead of queue

▲ 0 votes ● Reply ● Message ● Permalink

Aruna Maurya ✎ Edited a year ago

the code for dfs(for connected components) is giving runtime error

▲ 0 votes ● Reply ● Message ● Permalink

Ashish Nimbalkar a year ago

Make sure your adj and visited size is v+1

▲ 0 votes ● Reply ● Message ● Permalink

Vishnu Kumar V ✎ Edited a year ago

so u r traversing by using an for loop ..consider that i have 10 vertices and

what if they are not numbered from 1 to 10 ? what if they are from 10 to 20 ? ur for loop might gi ?

segmentation fault :/ coz no such vertex exist

0 votes ● Reply ● Message ● Permalink

Jeremias Serafim a year ago

got it on JavaScript :]

0 votes ● Reply ● Message ● Permalink

KyRillos BoshRa a year ago

yah! it should be 99 there's some thing wrong with the judge

0 votes ● Reply ● Message ● Permalink

Bhawana Sharma a year ago

What does adj[a].push_back(b) actually does.Does it mean adj[a]=b?
adj[10] is a 1-D vector but adj[s][i] is a 2-D vector?
I am not able to understand this concept.Somebody please help me with this.

0 votes ● Reply ● Message ● Permalink

Saravanan Ks 9 months ago

vector<int> adj ;
It means adj is a 1-D array
vector<int> adj[10];
It means each adj[i] can store 10 elements. So this is 2-D array

0 votes ● Reply ● Message ● Permalink

Rachit Yadav a year ago

its correct guys just take the graph as undirected

0 votes ● Reply ● Message ● Permalink

SATYAJEET BEHERA a year ago

```
#include<iostream>
#include <cstdlib>
#include<vector>
#include <queue>
using namespace std;
/*
*
*/
int v[100000]={0};
vector<int>adj[100000];
queue<int>q;
void bfs(int s){
int i ,j , k, l;
q.push(s);
v[s]=1;
while(!q.empty()){
k=q.front();
q.pop();
for(i=0;i<adj[k].size();i++){
if(v[adj[k][i]]!=1){
q.push(adj[k][i]);
v[adj[k][i]]=1;
}
}
}
}
int main() {
int nodes ,s ,edges , i ,j ,x ,y ,unvisited;
cin>>nodes>>edges;
for(i=0;i<edges;i++){
```

```
cin>>x>>y;
adj[x].push_back(y);
adj[y].push_back(x);
}
cin>>s;
bfs(s);
int visited=0;
for(i=0;i<100000;i++){
if(v[i]==1)
visited=visited+1;
}
unvisited=nodes-visited;
cout<<unvisited;
}
```

0 votes ● Reply ● Message ● Permalink

SATYAJEET BEHERA a year ago

can be done using bfs also just calculated the no visited nodes and subtract it from total no if nodes

0 votes ● Reply ● Message ● Permalink

distroter a year ago

first output should be 99

0 votes ● Reply ● Message ● Permalink

1504052_Avijeet a year ago

```
#include <iostream>
#include <vector>
using namespace std;
int nodes, edges, x, y, nco = 0;
vector <int> adj[10000+10];
bool visited[10000+10];
void dfs(int s) {
visited[s] = true;
for(int i = 0;i < adj[s].size();++i) {
if(visited[adj[s][i]] == false)
dfs(adj[s][i]);
}
}
void initialize() {
for(int i = 0;i <= nodes;++i)
visited[i] = false;
}
int main() {
cin >> nodes; //Number of nodes
cin >> edges;
nco=0; for(int i=0;i<10000+10;i++)
adj[i].clear(); //Number of edges
for(int i = 0;i < edges;++i) {
cin >> x >> y;
//Undirected Graph
adj[x].push_back(y); //Edge from vertex x to vertex y
adj[y].push_back(x); //Edge from vertex y to vertex x
}
initialize(); //Initialize all nodes as not visited
int st;
cin>>st;
dfs(st);
```

?

```
for(int i=1;i<=nodes;i++)
if(!visited[i])
nco++;
cout <<nco<< endl;
return 0;
}
```

0 votes ● Reply ● Message ● Permalink

Akash Singh sengar a year ago

java Solution :

```
// Java program to print DFS traversal from a given given graph
import java.io.*;
import java.util.*;

// This class represents a directed graph using adjacency list
// representation
class Graph
{
private int V; // No. of vertices
static int c=0;
ArrayList<Integer> al=new ArrayList();

// Array of lists for Adjacency List Representation
private LinkedList<Integer> adj[];

// Constructor
Graph(int v)
{
V = v;
adj = new LinkedList[v+1];
for (int i=1; i<=v; ++i)
adj[i] = new LinkedList();
}

//Function to add an edge into the graph
void addEdge(int v, int w)
{
adj[v].add(w); // Add w to v's list.
}

// A function used by DFS
void DFSUtil(int v,boolean visited[])
{
// Mark the current node as visited and print it
visited[v] = true;
//System.out.print(v+" ");
al.add(v);

// Recur for all the vertices adjacent to this vertex
Iterator<Integer> i = adj[v].listIterator();
while (i.hasNext())
{
int n = i.next();
if (!visited[n])
DFSUtil(n,visited);
}
}

// The function to do DFS traversal. It uses recursive DFSUtil()
void DFS(int point,int n)
```

?

```
{
// Mark all the vertices as not visited(set as
// false by default in java)
boolean visited[] = new boolean[V+1];

// Call the recursive helper function to print DFS traversal
// starting from all vertices one by one

DFSUtil(point, visited);
System.out.println(n-al.size());

}

public static void main(String args[])
{

Scanner in=new Scanner(System.in);
int n=in.nextInt();
int m=in.nextInt();
Graph g = new Graph(n);
int i=1;
while(i++<=m){
int x=in.nextInt();
int y=in.nextInt();
g.addEdge(x, y);
g.addEdge(y,x);
}

int point=in.nextInt();
//System.out.println("Following is Depth First Traversal");

g.DFS(point,n);
}
}
// This code is contributed by Aakash Sengar
```

▲ 0 votes ● Reply ● Message ● Permalink

bhaskar mishra a year ago

You can solve this problem by applying BFS or DFS, and they finish in pretty much the same time (They were both 0.22...). Is there any example of a question where there is an advantage to traversing the graph using DFS over BFS.

▲ 0 votes ● Reply ● Message ● Permalink

Jonathan Chan a year ago

```
//Objective C , Well, in fact ,it's C
#import <Foundation/Foundation.h>
#pragma mark - it only works when the grap is undirected
@interface DFS:NSObject{
@public
BOOL **_matrix;
BOOL *_visited;
int _vertice;
}
- (id)initWithSize: (int)nodes And: (int)edges;
- (void)getNode;
@end
@implementation DFS
- (id)initWithSize: (int)nodes And: (int)edges{
if (self = [super init]){
self->_vertice = nodes;
```

```
self->_visited = (BOOL *)malloc(_vertice * sizeof(BOOL));
self->_matrix = (BOOL **)malloc(_vertice * sizeof(BOOL *));
for (int i = 0; i < nodes; i++){
self->_matrix[i] = (BOOL *)malloc(_vertice * sizeof(BOOL));
for (int j = 0; j < nodes; j++)
*((*self->_matrix + i) + j) = NO;
}
for (int i = edges; i > 0; i--){
int x, y;
scanf("%d %d", &x, &y);
self->_matrix[x - 1][y - 1] = YES;
self->_matrix[y - 1][x - 1] = YES;
_visited[i - 1] = NO;
}
}
return self;
}
#pragma mark - recursive
- (void)dfs: (int)index{
if (index < 0 || index >= _vertice)
return;
for (int i = 0; i < _vertice; i++)
if (_matrix[index][i] && !_visited[i] && i != index){
_visited[i] = YES;
[self dfs:i];
}
}
- (void)getNode{
int head;
scanf("%d", &head);
[self dfs:head - 1];
int t = _vertice;
int n = 0;
while(t --){
if (!_visited[t])
n++;
}
printf("%d\n", n);
}
@end
int main(int argc, const char* argv[]){
NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
int nodes, edges;
scanf("%d %d", &nodes, &edges);
DFS *demo = [[DFS alloc] initWithSize:nodes And:edges];
[demo getNode];
[pool drain];
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Gokul Krishna G a year ago

http://ide.geeksforgeeks.org/ZESo9l

▲ 0 votes ● Reply ● Message ● Permalink

Akash Kandpal a year ago

http://ide.geeksforgeeks.org/Mgc1e2

▲ 0 votes ● Reply ● Message ● Permalink

?

Shubham Mishra a year ago

the accepted code is this
first try by your own

```
#include<bits/stdc++.h>
using namespace std;
void insert(vector<int>node[],int u,int v)
{
node[u].push_back(v);
node[v].push_back(u);
}
int DFS(vector<int>node[],int v)
{
int c=0;
bool visited[v]={false};
int x;
cin>>x;
stack<int>s;
s.push(x);
while(!s.empty())
{
int top=s.top();
s.pop();
if(visited[top]==false)
{
visited[top]=true;
c++;
}
for(vector<int>::iterator it=node[top].begin();it!=node[top].end();++it)
{
if(visited[*it]==false)
{
s.push(*it);
}
}
}
return c;
}
int main()
{
int edge,a,b,v;
cin>>v>>edge;
vector<int>node[1000000];
for(int i=0;i<edge;i++)
{
cin>>a>>b;
insert(node,a,b);
}
int c=DFS(node,v);
cout<<v-c;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Amulya Gaur a year ago

Nice dfs question

▲ 0 votes ● Reply ● Message ● Permalink

Akshive Pandey 10 months ago

```
#include <iostream>
#include <vector>
#include <stack>
using namespace std;
```

?

```
int main()
{
int N, M;
cin>>N>>M;
vector<int> adj[N+1];
int x, y;
for(int i = 0; i < M; i++){
cin>>x>>y;
adj[x].push_back(y);
adj[y].push_back(x);
}

int head;
cin>>head;
bool visited[N+1];
visited[0] = true;
for(int i = 1; i <= N; i++){
visited[i] = false;
}
stack<int> Stack;
Stack.push(head);
visited[head] = true;
int count = 0;
while(!Stack.empty()){
int v = Stack.top();
Stack.pop();
for(int i = 0; i < adj[v].size(); i++){
if(!visited[adj[v].at(i)]){
visited[adj[v].at(i)] = true;
Stack.push(adj[v].at(i));
count++;
}
}
}
cout<<N-1-count<<endl;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

jallarapu srinadh 8 months ago

Is the example code wrong ? because if it is given a 1->2->3 and 3 is not connected to 1 in return , then also it marks every node as true and includes 1->2->3 as connected part which is not as 3 is not connected to 1?
correct me if i am wrong

▲ 0 votes ● Reply ● Message ● Permalink

Xiaohua Yan 8 months ago

C++ solution:

```
/*
// Sample code to perform I/O:
cin >> name; // Reading input from STDIN
cout << "Hi, " << name << ".\n"; // Writing output to STDOUT
// Warning: Printing unwanted or ill-formatted data to output will cause the test cases to fail
*/
// Write your code here
#include <iostream>
#include <vector>
#include <stack>
using namespace std;
```

```
int numConnectedNodes(const vector<vector<int>>& neighbors, int node) {
int result = 0;

stack<int> S;
vector<bool> visited(neighbors.size());

S.push(node);
visited[node] = true;

while(!S.empty()) {
auto v = S.top();
S.pop();
++result;

for (auto n : neighbors[v]) {
if (!visited[n]) {
visited[n] = true;
S.push(n);
}
}
}
return result;
}
int main() {
int N;
cin >> N;

int M;
cin >> M;

vector<vector<int>> neighbors(N);

for (int i = 0; i < M; ++i) {
int n1;
cin >> n1;
int n2;
cin >> n2;
neighbors[n1 - 1].push_back(n2 - 1);
neighbors[n2 - 1].push_back(n1 - 1);
}

int query;
cin >> query;
cout << N - numConnectedNodes(neighbors, query - 1) << endl;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

nimesh jain 7 months ago

THIS WORKS FINE

```
#include <bits/stdc++.h>
using namespace std;
#define pb push_back
int ct=0;
vector <int > v[100005];
int vis[100005];
void dfs(int s)
{
//cout<<s<<" ";
ct++;
vis[s]=1;
```

?

```
int i;
for(i=0;i<v[s].size();i++)
{
if(vis[v[s][i]]==0)
dfs(v[s][i]);
}
}
int main() {
int n,e,i,x,y;
scanf("%d%d",&n,&e);
for(i=0;i<e;i++)
{
cin>>x>>y;
v[x].pb(y);
v[y].pb(x);
}
for(i=0;i<100005;i++)
vis[i]=0;
int start;
cin>>start;
dfs(start);
/*for(i=1;i<=n;i++)
{
if(vis[i]==0)
dfs(i);
}*/
int ans=n-ct;
cout<<ans<<endl;
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

nimesh jain 7 months ago

THIS WORKS CORRECT

```
#include <bits/stdc++.h>
using namespace std;
#define pb push_back
int ct=0; vector <int > v[100005]; int vis[100005];
void dfs(int s)
{ct++; int i; vis[s]=1;
for(i=0;i<v[s].size();i++)
{ if(vis[v[s][i]]==0)
dfs(v[s][i]);}}
int main() {
int n,e,i,x,y,start;
scanf("%d%d",&n,&e);
for(i=0;i<e;i++)
{ cin>>x>>y; v[x].pb(y); v[y].pb(x); }
for(i=0;i<100005;i++)
vis[i]=0;
cin>>start; dfs(start); int ans=n-ct; cout<<ans<<endl;
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Manveer Singh Edited 7 months ago

Hi guys I seriously feel DFS-iterative pseudocode is not correct. It is BFS. Please check it again @Prateek Garg

▲ 0 votes ● Reply ● Message ● Permalink

?

Ravi Kumar Tahlan 7 months ago

C++ working code
==========
```
#include<bits/stdc++.h>
using namespace std;
int num=0;
class Graph{
private:
int v;
list< int > *adj;
public:
void connect(int v, int w);
void DFS(int s);
void DFSUtil(int s,bool visited[]);

Graph(int n){
v=n;
adj = new list< int >[n+1];
}
};
void Graph::connect(int v, int w){
adj[v].push_back(w);
adj[w].push_back(v);
}
void Graph::DFSUtil(int s,bool visited[]){
visited[s]=true;
num+=1;

list< int >::iterator i;

for(i=adj[s].begin();i!=adj[s].end();i++){
if(!visited[*i]){
DFSUtil( *i , visited);
}
}
}
void Graph::DFS(int s){
bool *visited = new bool[v+1];

for(int i=0;i<=v;i++){
visited[i]=false;
}

DFSUtil(s, visited);
}
int main(){
int s,n,m,a,b;
cin>>n>>m;

Graph g1(n);

while(m--){
cin>>a>>b;
g1.connect(a,b);
}

cin>>s;

g1.DFS(s);
```

?

```
cout<<n-num;


return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Ravi Kumar Tahlan 7 months ago

C++ working code(Disjoint Sets)

==========

```
#include<bits/stdc++.h>
using namespace std;
int root(int ar[], int a){
while(ar[a]!=a){
a=ar[ar[ar[a]]];
}

return a;
}
void weighted_union(int ar[],int size[],int a, int b){
int root_a=root(ar,a);
int root_b=root(ar,b);

if(root_a == root_b)
return;

if(size[root_a] < size[root_b]){
ar[root_a]=root_b;
size[root_b]+=size[root_a];
}
else{
ar[root_b]=root_a;
size[root_a]+=size[root_b];
}
}
int main(){
int n,m,a,b,s,count=0;
cin>>n>>m;

int ar[n+1],size[n+1];

for(int i=0;i<=n;i++){
ar[i]=i;
size[i]=1;
}

while(m--){
cin>>a>>b;
weighted_union(ar,size,a,b);
}

cin>>s;
int root_s=root(ar,s);


for(int i=0;i<=n;i++)
if(root(ar,ar[i])==root_s)
count+=1;
```

?

```
cout<<n-count;

return 0;
}
```

0 votes ● Reply ● Message ● Permalink

Arvind Negi 6 months ago

This is the code for above problem

```
#include<bits/stdc++.h>
using namespace std;
int visited[100001];
void dfs(vector <int> adj[],int x)
{
visited[x]=true;
for(int i=0;i<adj[x].size();i++)
{
if(visited[adj[x][i]]==false)
dfs(adj,adj[x][i]);
}
}
int main()
{
int n,m,a,b,x,c,i;
cin>>n>>m;
vector <int> adj[n+1];
memset(visited,false,sizeof(visited));
for(i=0;i<m;i++)
{
cin>>a>>b;
adj[a].push_back(b);
adj[b].push_back(a);
}
cin>>x;
dfs(adj,x);
for(int i=1;i<=n;i++)
{
if(visited[i]==false)
c++;
}
cout<<c;
}
```

0 votes ● Reply ● Message ● Permalink

Shubham Khandodia 6 months ago

Vectors are not taught to us yet can we do it with some other method

0 votes ● Reply ● Message ● Permalink

Yash Jain 5 months ago

It should be specified that graph is undirected.

0 votes ● Reply ● Message ● Permalink

Angom Geetchandra 5 months ago

The iterative version of DFS given in the pseudocode doesn't seem right. For instance, on the image that follows, the code will mark 1 as visited, then 2 and then 3, instead of going down to 4. One way would be to mark 'v' as visited and push all unvisited neighbours 'w' but not mark them.

0 votes ● Reply ● Message ● Permalink

pratik jadhav 4 months ago

?

```
//javascript solution
process.stdin.resume();
process.stdin.setEncoding("utf-8");
var stdin_input = "";
process.stdin.on("data", function (input) {
stdin_input += input; // Reading input from STDIN
});
process.stdin.on("end", function () {
main(stdin_input);
});
function main(input) {
let [nodesedges,...rest]=input.split("\n");
let spliteden=nodesedges.split(" ");
let n=Number(spliteden[0]);
let m=Number(spliteden[0]);
var graph=[];
for(let r=0;r<n;r++){
let row=[];
for(let c=0;c<m;c++){
row.push(0);
}
graph.push(row)
}
let edgeArray=[];
for(let i=0;i<rest.length;i++){
edgeArray[i]=rest[i];
}
let x=Number(edgeArray.pop());
//console.log("x",x)
for(let itm of edgeArray){
let rowcol=[];
rowcol=itm.split(" ");
let rowno=Number(rowcol[0]);
let colno=Number(rowcol[1]);
for(let item of graph){
if(graph.indexOf(item)==rowno-1){
for(let item1 of item){
item[colno-1]=1;
}}}
for(let item of graph){
if(graph.indexOf(item)==colno-1){
for(let item1 of item){
item[rowno-1]=1;
}}}
}
var nodesLen = [];
for (var i = 0; i < graph.length; i++) {
nodesLen[i] = false;
}
//console.log(graph)
dfs(graph,(x-1),nodesLen);
//console.log(nodesLen)

let countOfUnvisited=0;
for(let i=0;i<nodesLen.length;i++){
if(!nodesLen[i]){
countOfUnvisited++;
}
}
```

```
console.log(countOfUnvisited);
}
function dfs(graph, root,nodesLen) {
nodesLen[root] = true;
//console.log("nodelen+root",nodesLen,root+1);
var current;
current = root;
var curConnected = graph[current];
// console.log("cuconect",curConnected)
var neighborIdx = [];
var idx = curConnected.indexOf(1);
while (idx != -1) {
neighborIdx.push(idx);
idx = curConnected.indexOf(1, idx + 1);
}

for (var j = 0; j < neighborIdx.length; j++) {
if (!nodesLen[neighborIdx[j]]) {

let ele=neighborIdx[j];
//console.log("recursion on ele",ele)
dfs(graph,ele,nodesLen)


}
}

}
```

▲ 0 votes ● Reply ● Message ● Permalink

Saumya Verma 4 months ago

```
#include<iostream>
#include<vector>
using namespace std;
vector<int> v[100000];
bool mark[100000];
void dfs(int s)
{
mark[s]=true;
for(int i=0;i<v[s].size();i++)
{
if(mark[i]==false)
{
dfs(v[s][i]);
}
}
}
int main()
{
int n,e,x,y,node;
cin>>n>>e;
for(int i=0;i<e;i++)
{
cin>>x>>y;
v[x].push_back(y);
v[y].push_back(x);
}
cin>>node;
dfs(node);
```

```
int count=0;
for(int i=0;i<n;i++)
{
if(mark[i]==false)
{
count++;
}
}
cout<<count;
return 0;
}
```

This code gives me an execution failed message. Can anyone please tell me why?

▲ 0 votes ● Reply ● Message ● Permalink

Shahrukh Khan 4 months ago

The very definition of disconnected graph is given wrong.

▲ 0 votes ● Reply ● Message ● Permalink

Ankush Kumar Singh 3 months ago

It is clearly mentioned in question that there is undirected edge between a and b.

▲ 0 votes ● Reply ● Message ● Permalink

Prateek Agarwal 3 months ago

```
#include <bits/stdc++.h>
using namespace std;
main()
{
vector<vector<int> > Graph;
vector<bool> Visited;
int Nodes;
cin >> Nodes;
int Edges;
cin >> Edges;
// Edges = Nodes - 1;
//According to Question (Not Always)
Graph.assign(Nodes + 1, vector<int>());
Visited.assign(Nodes + 1, false);
for (int i = 0; i < Edges; i++)
{
int a, b;
cin >> a >> b;
Graph[a].push_back(b);
Graph[b].push_back(a);
}
//DFS
stack<int> S;
int root;
cin >> root;
S.push(root);
Visited[root] = true;
while (!S.empty())
{
int f = S.top();
S.pop();
for (vector<int>::iterator itr = Graph[f].begin(); itr != Graph[f].end(); itr++)
{
if (!Visited[*itr])
{
S.push(*itr);
```

?

```
Visited[*itr] = true;
}
}
}
int count = 0;
for(int i=1; i<=Nodes;i++){
if(!Visited[i]){
count++;
}
}
cout << count << endl;
```

0 votes ● Reply ● Message ● Permalink

Ayush Mahajan 2 months ago

I think this question better belonged in the Disjoint-set problem set. It can very very easily be solved with it that DFS here is overkill I guess. Though my opinion.

0 votes ● Reply ● Message ● Permalink

Karan Jakhar 2 months ago

```
#include<bits/stdc++.h>
using namespace std;
int ans = 0;
void addEdge(vector<int>adj[],int u,int v)
{
adj[u].push_back(v);
adj[v].push_back(u);
}
void dfsutil(vector<int>adj[],int x,bool *visited)
{
visited[x] = true;
ans++;
for(int i = 0; i < adj[x].size(); i++)
if(!visited[adj[x][i]])
dfsutil(adj,adj[x][i],visited);
}

void dfs(vector<int>adj[],int x, int n)
{
bool visited[n+1] = {false};
ans = 0;
dfsutil(adj,x,visited);


}
int main()
{
int n,m;
cin>>n>>m;
int a,b,x;
vector<int>adj[n+1];
for(int i = 0; i < m; i++ )
{
cin>>a>>b;
addEdge(adj,a,b);
}
cin>>x;
dfs(adj,x,n);
cout<<n-ans<<endl;
```

?

```
return 0;
}
```

0 votes ● Reply ● Message ● Permalink

Dhrubodeep Basumatary 2 months ago

I tried using iterative way of doing it.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
int n,m,i,j,x,y,k,head,count=0;
cin>>n>>m;
vector<int>v[n+2];vector<bool>visited(n+2,false);
for(i=0;i<m;i++)
{
cin>>x>>y;
if(x!=y)
{
v[x].push_back(y);
v[y].push_back(x);
}
}
cin>>head;
stack<int>S;
S.push(head);
while(!S.empty())
{
int top=S.top();
S.pop();
if(visited[top]==false)
{
count++;
visited[top]=true;
}
for(j=0;j<v[top].size();j++)
{
if(visited[v[top][j]]==false)
{
S.push(v[top][j]);
}
}
}
cout<<n-count<<endl;
return 0;
}
```

0 votes ● Reply ● Message ● Permalink

Kunal Gupta 2 months ago

Code using BFS:

```
#include<bits/stdc++.h>
using namespace std;
int main() {

int n,m;
cin>>n>>m;
vector <int> V[n];
for(int i = 0;i<m;i++) {
int x,y;
cin>>x>>y;
V[x-1].push_back(y-1);
```

?

```
V[y-1].push_back(x-1);
}
int head;
cin>>head;
head--;
int vis[n];
for(int i = 0;i<n;i++)
{
vis[i] = 0;
}
int tv = 0;
queue <int> Q;
Q.push(head);
while(!Q.empty()) {

int node = Q.front();
Q.pop();
for(int i = 0;i<V[node].size();i++) {
if(vis[V[node][i]]==0) {
Q.push(V[node][i]);
vis[V[node][i]]=1;
tv++;
}
}
}
cout<<n-tv;

}
```

▲ 0 votes ● Reply ● Message ● Permalink

Sanyam Sood a month ago

please find the problem in code .. giving run time error

```
#include<bits/stdc++.h>
using namespace std;
void dfs(long long s);
long long visited[100005]={-1};
vector < long long > adj[100005];

int main()
{
long long n,m,x,a,b,i;
// cout<<"hello"<<endl;
cin>>n>>m;
while(m--)
{
// cout<<"hello"<<endl;
cin>>a>>b;
adj[a].push_back(b);
// cout<<"hello"<<endl;
adj[b].push_back(a);
}
cin>>x;
dfs(x);
long long count=0;
for(i=0;i<m;i++)
{
if(visited[i]==-1)
count++;
}
```

?

```
cout<<count;
}
void dfs(long long s)
{
long long i;
visited[s]=1;
for(i=0;i<adj[s].size();i++)
{
if(visited[i]==-1)
dfs(adj[s][i]);
}
}
```

0 votes ● Reply ● Message ● Permalink

Ömer AKALIN a month ago

Take care on array index vs node. (0/1)

0 votes ● Reply ● Message ● Permalink

tony_shashwat a month ago

```
#include<bits/stdc++.h>
using namespace std;
vector<int> adj[100001];
bool sptset[100001];
void initialize(bool sptset[]){
for(int i=0;i<=100000;i++)
sptset[i]=false;
}
void DFS(int src,bool sptset[]){

sptset[src]=true;
vector<int>::iterator i;
for(i=adj[src].begin();i!=adj[src].end();i++)
{
if(!sptset[*i])
DFS(*i,sptset);
}

}
int main(){
int n,m;
cin>>n>>m;
initialize(sptset);
int x,y;
for(int i=0;i<m;i++)
{
cin>>x>>y;
adj[x].push_back(y);
adj[y].push_back(x);
}
int src;
cin>>src;
DFS(src,sptset);

int count=0;
for(int i=1;i<=n;i++)
if(!sptset[i])
count++;

cout<<count<<endl;
```

?

```
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

DIVYANSHI MANGAL a month ago

Why is it showing In function ‘int dfs(int)’: /hackerearth/CPP14_24/s_03.cpp:16:15: warning: comparison between signed and unsigned integer expressions [-Wsign-compare] for (int i=0;i

```
#include <bits/stdc++.h>
using namespace std;
vector <int> adj[100005];
bool vis[100005];
void init(){
for (int i=0;i<100005;i++){
vis[i]=false;
}
}
int ans=0;
int dfs(int s){
vis[s]=true;
ans++;
for (int i=0;i<adj[s].size();++i){
if (vis[adj[s][i]]==false){
dfs(adj[s][i]);
}
}
return ans;
}
int main(){
int nodes,edges;
cin>>nodes>>edges;
int x,y;
for (int i=0;i<edges;i++){
cin>>x>>y;
adj[x].push_back(y);
adj[y].push_back(x);
}
init();
//int con=0;
int head;
cin>>head;
int o= dfs(head);

cout<<n-o<<endl;
return 0;}
```

▲ 0 votes ● Reply ● Message ● Permalink

Navneet Singh 18 days ago

```
/*
// Sample code to perform I/O:
cin >> name; // Reading input from STDIN
cout << "Hi, " << name << ".\n"; // Writing output to STDOUT
// Warning: Printing unwanted or ill-formatted data to output will cause the test cases to fail
*/
// Write your code here
#include<bits/stdc++.h>
using namespace std;
int main(){
```

?

```
int nodes;
int edges;
cin>>nodes>>edges;
vector<int> adj[nodes+1];
for(int i=0; i<edges; i++){
int x,y;
cin>>x>>y;
adj[x].push_back(y);
adj[y].push_back(x);
}
// cout<<adj[2].size();
int x;
cin>>x;
stack<int> s;
s.push(x);
int visited[nodes+1];
for(int i=1; i<sizeof(visited); i++){
visited[i]=0;
}
visited[x]=1;
while(!s.empty()){
int p=s.top();
// cout<<p<<" "<<adj[p].size()<<" ";
s.pop();
for(int i=0; i<adj[p].size(); i++){
if(visited[adj[p][i]]==0){
// cout<<"hi";
visited[adj[p][i]]=1;
s.push(adj[p][i]);
}
}
}
int count=0;
for(int i=1; i<=nodes; i++){
// cout<<"visited["<<i<<"]="<<visited[i];
if(visited[i]==0) count++;
}
cout<<count;
return 0;
}
```

why is it giving wrong answer?

▲ 0 votes ● Reply ● Message ● Permalink