Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

GRAPH

Name:	Wilson Vidyut Doloy
Reg No:	19BCE1603
Subject:	CSE2003 Data Structures and Algorithms Lab
Slot:	L67+L68 (Prof. Sandeep Kumar)
Date:	16 October 2020

Q2. You are given an undirected weighted graph of `n` nodes (0-indexed), represented by an edge list where `edges[i] = [a, b]` is an undirected edge connecting the nodes `a` and `b` with a probability of success of traversing that edge `succProb[i]`.

Given two nodes `start` and `end`, find the path with the maximum probability of success to go from `start` to `end` and return its success probability.

If there is no path from `start` to `end`, **return 0**. Your answer will be accepted if it differs from the correct answer by at most **1e-5**.

CODE:

```
#include <stdio.h>
struct mygraph
{
   double next;
   double val;
}mygraph;
```

```
void help(int n, struct mygraph** graph, int index, int end, double *state, double current, double *xx,int *col)
{
    if(current<=0.00005) return;
    if(state[index]>=current)
    {
        return;
    }
    state[index] = current;
    if(index == end)
    {
        if(*xx<current) *xx = current;
        return ;
    }

    int i;
    for(i=0;i<col[index];i++)
    {
            help(n, graph, graph[index][i].next, end, state, current*graph[index][i].val, xx,col);
    }
}

double maxProbability(int n, int** edges, int edgesSize, int* edgesColSize, double* succProb, int succProbSize, int start, int end){

    double *state = (double*)calloc(sizeof(double) , n);

    struct mygraph **graph = ( struct mygraph**)calloc(sizeof( struct mygraph*), n);
```

```
    int *col  = (int*)calloc(sizeof(int),n);
    int i;
    for(i=0;i<edgesSize;i++)
    {
        int temp0 = edges[i][0];
        int temp1 = edges[i][1];
        if(col[temp0]==0)
        {
            graph[temp0] = ( struct mygraph*)malloc(sizeof( struct mygraph) );
        }
        else
            graph[temp0] = ( struct mygraph*)realloc(graph[temp0],sizeof( struct mygraph)
*(1+col[temp0]));
        if(col[temp1]==0)
        {
            graph[temp1] = ( struct mygraph*)malloc(sizeof( struct mygraph));


        }
        else
            graph[temp1] = ( struct mygraph*)realloc(graph[temp1],sizeof( struct mygraph)
*(1+col[temp1]));
        graph[temp0][col[temp0]].next=temp1;
        graph[temp0][col[temp0]++].val  = succProb[i];
        graph[temp1][col[temp1]].next=temp0;
        graph[temp1][col[temp1]++].val  = succProb[i];
    }
    double max=-1;
    help(n, graph, start, end, state, 1, &max,col);
    return max==-1?0:max;
```

}

OUTPUT:


C
Autocomplete
struct mygraph
{
double next;
double val;
}mygraph;
void help(int n, struct mygraph** graph, int index, int end, double *state, double current,
double *xx,int *col)
{
if(current<=0.00005) return;
if(state[index]>=current)
{
return;
}
state[index] = current;
if(index == end)
{
if(*xx<current) *xx = current;
return ;
Testcase
Run Code Result
Debugger
Accepted
Runtime: 0 ms
Your input
3
[[0,1],[1,2],[0,2]]
Output
0.25000
Diff
Expected
0.25000
Console
How to create a testcase
Run Code
Submit
ENG
IN
20:08
16-10-2020


C
Autocomplete
struct mygraph
{
double next;
double val;
}mygraph;
void help(int n, struct mygraph** graph, int index, int end, double *state, double current,
double *xx,int *col)
{
if(current<=0.00005) return;
if(state[index]>=current)
{
return;
}
state[index] = current;
if(index == end)
{
if(*xx<current) *xx = current;
return ;
Testcase
Run Code Result
Debugger
Accepted
Runtime: 0 ms
Your input
3
[[0,1],[1,2],[0,2]]
Output
0.30000
Diff
Expected
0.30000
Console
How to create a testcase
Run Code
Submit
ENG
IN
20:09
16-10-2020


C
Autocomplete
struct mygraph
{
double next;
double val;
}mygraph;
void help(int n, struct mygraph** graph, int index, int end, double *state, double current,
double *xx,int *col)
{
if(current<=0.00005) return;
if(state[index]>=current)
{
return;
}
state[index] = current;
if(index == end)
{
if(*xx<current) *xx = current;
return ;
Testcase
Run Code Result
Debugger
Accepted
Runtime: 0 ms
Your input
3
[[0,1]]
Output
0.00000
Diff
Expected
0.00000
Console
How to create a testcase
Run Code
Submit
ENG
IN
20:09
16-10-2020

Q1. Given a directed acyclic graph (**DAG**) of `n` nodes labelled from 0 to n – 1, find all possible paths from node `0` to node `n - 1`, and return them in any order.

The graph is given as follows: `graph[i]` is a list of all nodes you can visit from node `i` (i.e., there is a directed edge from node `i` to node `graph[i][j]`).

CODE:

```
#include <stdio.h>
int path[16];
int *ret[7000];
void helper(int** graph,int cur,int *idx,int graphSize,int i,int *graphColSize,int **returnColumnSizes)
{
    path[i]=cur;
    if(cur==graphSize-1)
    {
        ret[*(idx)]=calloc(16,sizeof(int));
        returnColumnSizes[0][*(idx)]=i+1;
        memcpy(ret[(*idx)++],path,sizeof(int)*(i+1));
    }
    else
        for(int j=0;j<graphColSize[cur];j++)
        {
            helper(graph,graph[cur][j],idx,graphSize,i+1,graphColSize,returnColumnSizes);
        }
}
```

```
int** allPathsSourceTarget(int** graph, int graphSize, int* graphColSize, int* returnSize, int**
returnColumnSizes){

    returnColumnSizes[0]=malloc(sizeof(int)*7000);

    int idx=0;

    helper(graph,0,&idx,graphSize,0,graphColSize,returnColumnSizes);

    *returnSize=idx;

    return ret;

}
```

OUTPUT:

```
int path[16];
int *ret[7000];
void helper(int** graph,int cur,int *idx,int graphSize,int i,int *graphColSize,int
**returnColumnSizes)
{
    path[i]=cur;
    if(cur==graphSize-1)
    {
        ret[*(idx)]=calloc(16,sizeof(int));
        returnColumnSizes[0][*(idx)]=i+1;
        memcpy(ret[(*idx)++],path,sizeof(int)*(i+1));
    }
    else
        for(int j=0;j<graphColSize[cur];j++)
        {
            helper(graph,graph[cur][j],idx,graphSize,i+1,graphColSize,returnColumnSizes);
        }
}

int** allPathsSourceTarget(int** graph, int graphSize, int* graphColSize, int* returnSize,
int** returnColumnSizes){
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 0 ms

Your input [[1,2], [3], [3], []]

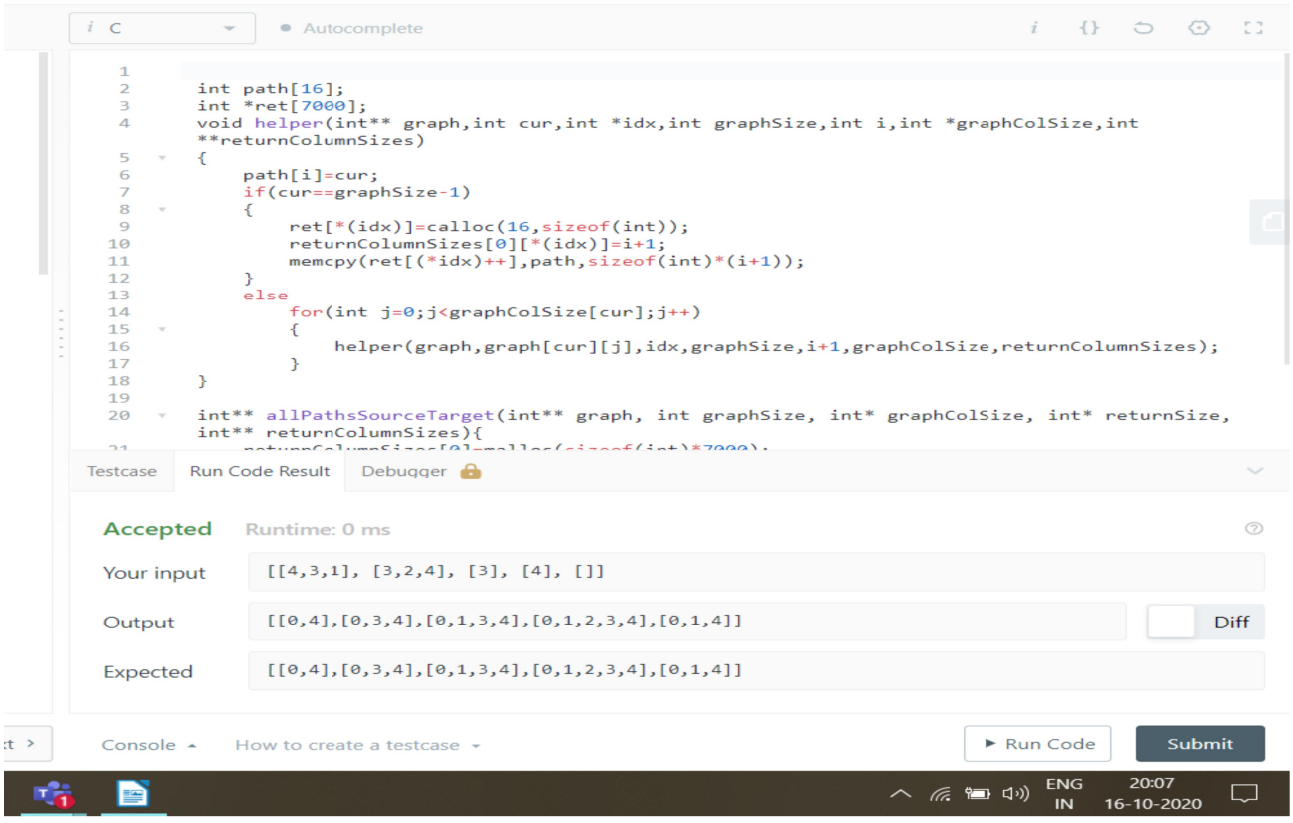
Output [[0,1,3],[0,2,3]]

Expected [[0,1,3],[0,2,3]]

Diff

Console | How to create a testcase | Run Code | Submit

ENG IN 20:06 16-10-2020



```
int path[16];
int *ret[7000];
void helper(int** graph,int cur,int *idx,int graphSize,int i,int *graphColSize,int
**returnColumnSizes)
{
    path[i]=cur;
    if(cur==graphSize-1)
    {
        ret[*(idx)]=calloc(16,sizeof(int));
        returnColumnSizes[0][*(idx)]=i+1;
        memcpy(ret[(*idx)++],path,sizeof(int)*(i+1));
    }
    else
        for(int j=0;j<graphColSize[cur];j++)
        {
            helper(graph,graph[cur][j],idx,graphSize,i+1,graphColSize,returnColumnSizes);
        }
}

int** allPathsSourceTarget(int** graph, int graphSize, int* graphColSize, int* returnSize,
int** returnColumnSizes){
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 0 ms

Your input [[4,3,1], [3,2,4], [3], [4], []]

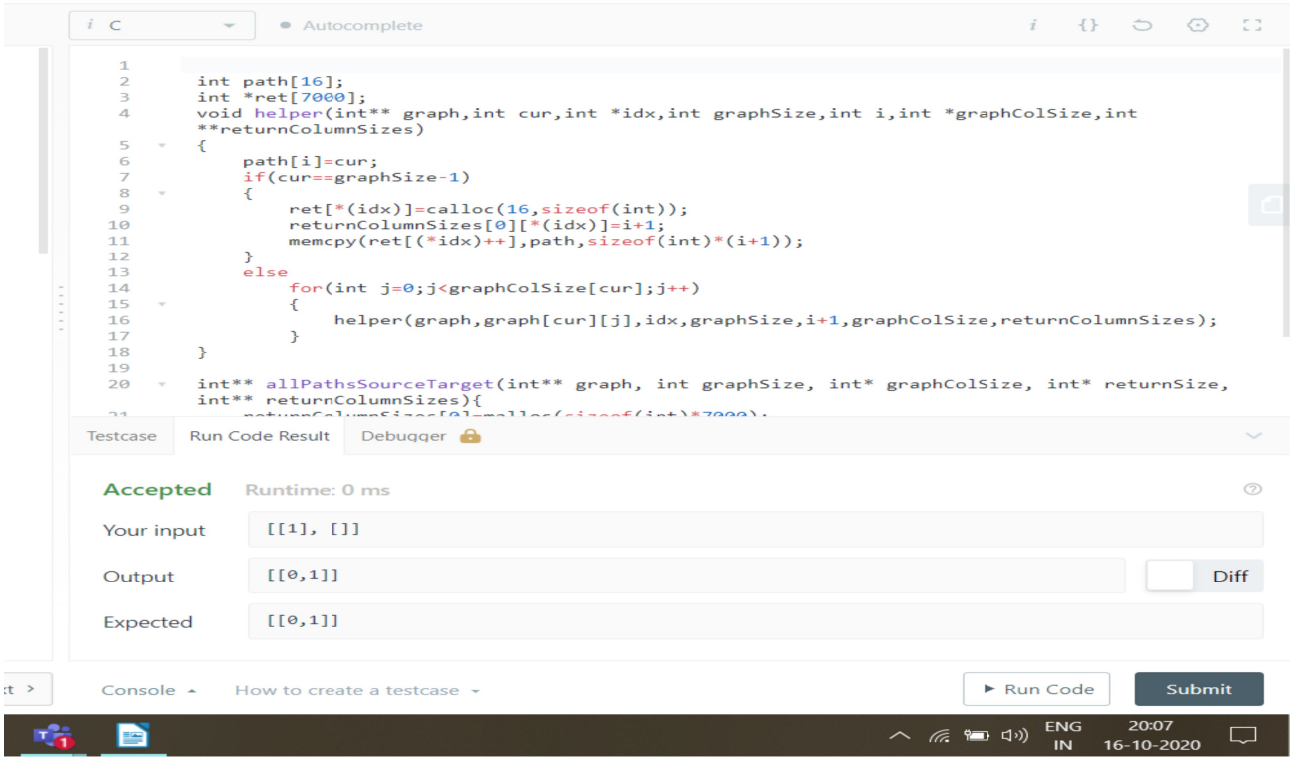
Output [[0,4],[0,3,4],[0,1,3,4],[0,1,2,3,4],[0,1,4]]

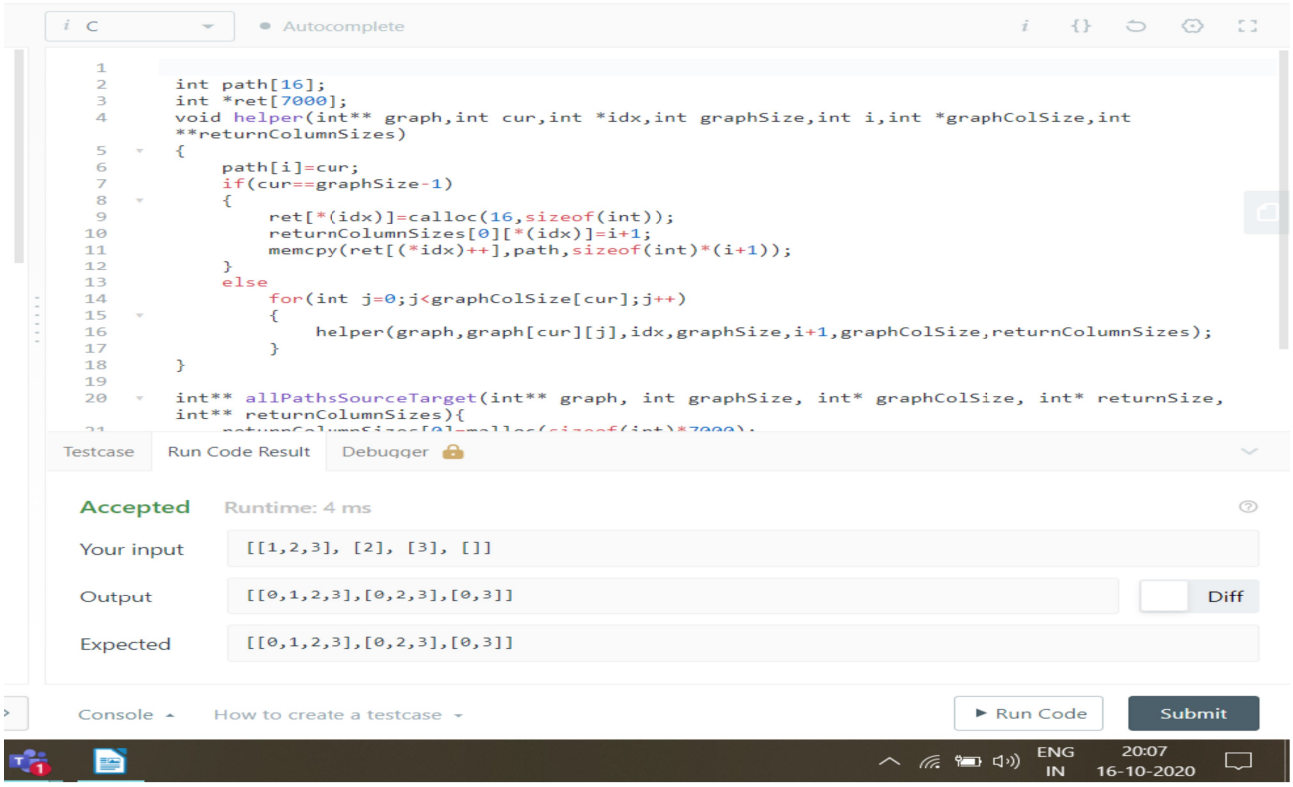
Expected [[0,4],[0,3,4],[0,1,3,4],[0,1,2,3,4],[0,1,4]]

Diff

Console | How to create a testcase | Run Code | Submit

ENG IN 20:07 16-10-2020


C
Autocomplete
int path[16];
int *ret[7000];
void helper(int** graph,int cur,int *idx,int graphSize,int i,int *graphColSize,int
**returnColumnSizes)
{
path[i]=cur;
if(cur==graphSize-1)
{
ret[*(idx)]=calloc(16,sizeof(int));
returnColumnSizes[0][*(idx)]=i+1;
memcpy(ret[(*idx)++],path,sizeof(int)*(i+1));
}
else
for(int j=0;j<graphColSize[cur];j++)
{
helper(graph,graph[cur][j],idx,graphSize,i+1,graphColSize,returnColumnSizes);
}
}
int** allPathsSourceTarget(int** graph, int graphSize, int* graphColSize, int* returnSize,
int** returnColumnSizes){
Testcase
Run Code Result
Debugger
Accepted
Runtime: 0 ms
Your input
[[1], []]
Output
[[0,1]]
Diff
Expected
[[0,1]]
Console
How to create a testcase
Run Code
Submit
ENG
IN
20:07
16-10-2020


C
Autocomplete
int path[16];
int *ret[7000];
void helper(int** graph,int cur,int *idx,int graphSize,int i,int *graphColSize,int
**returnColumnSizes)
{
path[i]=cur;
if(cur==graphSize-1)
{
ret[*(idx)]=calloc(16,sizeof(int));
returnColumnSizes[0][*(idx)]=i+1;
memcpy(ret[(*idx)++],path,sizeof(int)*(i+1));
}
else
for(int j=0;j<graphColSize[cur];j++)
{
helper(graph,graph[cur][j],idx,graphSize,i+1,graphColSize,returnColumnSizes);
}
}
int** allPathsSourceTarget(int** graph, int graphSize, int* graphColSize, int* returnSize,
int** returnColumnSizes){
Testcase
Run Code Result
Debugger
Accepted
Runtime: 4 ms
Your input
[[1,2,3], [2], [3], []]
Output
[[0,1,2,3],[0,2,3],[0,3]]
Diff
Expected
[[0,1,2,3],[0,2,3],[0,3]]
Console
How to create a testcase
Run Code
Submit
ENG
IN
20:07
16-10-2020