CSE246

Section 4

Fall 2023

**Lab Task - 05**

**Topic:** MCM Calculation

Unweighted Shortest Path – Undirected

Bicoloring

Tree Diameter for an undirected tree graph

Multiple shortest path existence

Submitted By

|  |  |
| --- | --- |
| **Name** | **ID** |
| B M Shahria Alam | 2021-3-60-016 |
| Golam Kibria | 2021-3-60-215 |
| Sidratul Moontaha | 2021-3-60-048 |
| MD Imran Khan | 2021-3-60-206 |

**Problem 1**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int mcm(int i, int j, vector<int> &arr, vector<vector<int>> &dp)

{

if(i==j) return 0;

if(dp[i][j]!=-1) return dp[i][j];

int minimum=100000;

for(int k=i; k<j; k++)

{

int r1 = arr[i-1] \* arr[k] \* arr[j] + mcm(i, k, arr, dp) + mcm(k+1,j,arr,dp);

minimum = min(r1, minimum);

}

dp[i][j]=minimum;

return dp[i][j];

}

int main()

{

int n;

cin>>n;

vector<int>arr(n+1);

int a,b;

cin >> a >> b;

arr[0] = a;

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

{

cin>> a >> b;

arr[i] = a;

}

arr[n]=b;

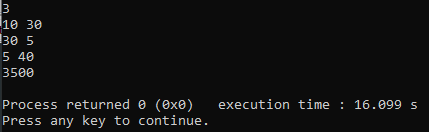
vector<vector<int>>dp(n+1, vector<int>(n+1, -1));

cout<<mcm(1,n,arr,dp)<<endl;

return 0;

}

**OUTPUT:**



**Problem 2**

**Solution:**

#include <bits/stdc++.h>

using namespace std;

vector<int> graph[100000];

int pre[100000];

int dis[100000];

void edge(int s, int d)

{

graph[s].push\_back(d);

graph[d].push\_back(s);

}

bool BFS(int s, int d, int v)

{

list<int> queue;

bool visit[v];

for (int i=0; i<v; i++)

{

visit[i] = false;

dis[i] = INT\_MAX;

pre[i] = -1;

}

visit[s] = true;

dis[s] = 0;

queue.push\_back(s);

while (!queue.empty())

{

int u = queue.front();

queue.pop\_front();

for (int i = 0; i < graph[u].size(); i++)

{

if (visit[graph[u][i]] == false)

{

visit[graph[u][i]] = true;

dis[graph[u][i]] = dis[u] + 1;

pre[graph[u][i]] = u;

queue.push\_back(graph[u][i]);

if (graph[u][i] == d)

return true;

}

}

}

return false;

}

void shortestPath(int s, int d, int v)

{

if(BFS(s, d, v)==false)

{

cout<<"No path";

return;

}

vector<int> path;

int temp2 = d;

path.push\_back(temp2);

while(pre[temp2] != -1)

{

path.push\_back(pre[temp2]);

temp2 = pre[temp2];

}

cout<<dis[d];

}

int main()

{

int v,e,x,y,s,d;

cin>>v>>e;

v+=1;

while(e--)

{

cin>>x>>y;

edge(x,y);

}

cin>>s>>d;

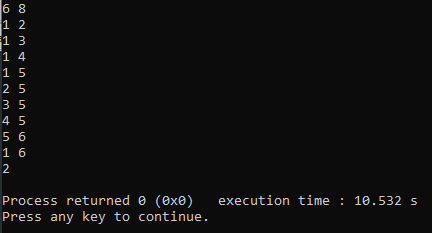
shortestPath(s,d,v);

cout<<endl;

return 0;

}

**OUTPUT**

****

**Problem 3**

**Solution:**

#include<bits/stdc++.h>

#include<vector>

using namespace std;

int main()

{

int v,e;

cin >> v>>e;

vector<vector<int>>graph;

vector<int>color;

bool visit[100000];

int x,y,i;

graph.resize(e);

color.resize(e, -1);

memset(visit, 0, sizeof(visit));

for(int i=0; i<e; i++)

{

graph[i].resize(e);

}

for(int i=0; i<e; i++)

{

cin>>x>>y;

x--;

y--;

graph[x][y]=1;

graph[y][x]=1;

}

bool res;

color[0]=1;

queue <int> q;

q.push(0);

while (!q.empty())

{

int temp=q.front();

q.pop();

for (i=0; i<e; i++)

{

if (graph[temp][i] && color[i] == -1)

{

color[i]=1-color[temp];

q.push(i);

res=true;

}

else if (graph[temp][i] && color[i]==color[temp]) res=false;

}

}

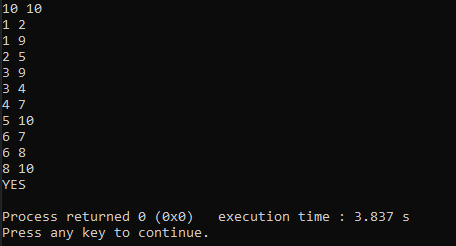
if(res) cout<<"YES"<<endl;

else cout<<"NO"<<endl;

return 0;

}

**OUTPUT:**



**Problem 4**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

vector<vector<int>> graph;

vector<int> h;

int n,e;

int height(int curr, int p)

{

for(auto it:graph[curr])

{

if(it == p)

{

continue;

}

h[curr]=max(h[curr], height(it, curr));

}

h[curr]+=1;

return h[curr];

}

int diameter(int curr, int p)

{

int mx1, mx2, mx\_subtree;

mx1 = mx2 = mx\_subtree = INT\_MIN;

for(auto it:graph[curr])

{

if(it==p)

{

continue;

}

if(mx1<h[it])

{

mx2=mx1;

mx1=h[it];

}

else

{

mx2=h[it];

}

}

for(auto it:graph[curr])

{

if(it==p)

{

continue;

}

mx\_subtree = max(mx\_subtree,diameter(it, curr));

}

return max(mx\_subtree, mx1 + mx2 + 1);

}

int main()

{

int u,v;

cin>>n>>e;

graph.resize(n+1);

h.resize(n+1, 0);

for(int i=0; i<e ; i++)

{

cin >>u>>v;

graph[u].push\_back(v);

graph[v].push\_back(u);

}

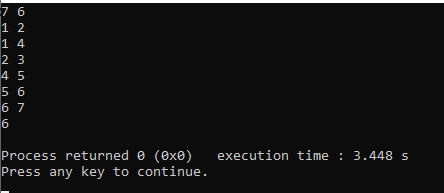
height(1, -1);

cout <<diameter(1, -1)-1 <<endl;

return 0;

}

**OUTPUT:**



**Problem 5**

**Solution:**

#include<bits/stdc++.h>

#include<string.h>

using namespace std;

const int N = 1e5+9;

vector<int>v[N],dis,vis;

int n,m,s;

int main()

{

cin>>n>>m;

for(int i=1,a,b;i<=n;i++)

{

cin>>a>>b;

v[a].push\_back(b);

v[b].push\_back(a);

}

dis=vector<int>(n+1,1e9);

vis=vector<int>(n+1);

cin>>s;

dis[s]=0;

queue<int>q;

q.push(s);

vis[s] = 1;

while(!q.empty())

{

int now = q.front();

q.pop();

for(auto x:v[now])

{

if(vis[x])

{

if(dis[now]+1==dis[x])

{

puts("YES");

exit(0);

}

}

else

{

vis[x]=1;

dis[x]=dis[now]+1;

q.push(x);

}

}

}

puts("NO");

}

**OUTPUT:**

