Programming problem 1: Tìm chu trình Euler va đường đi Euler (nếu có) trên đồ thị vô hướng (bắt buộc phải Run trước khi Evaluate để tránh treo tiến trình = tránh 0 điểm khi bị treo tiến trình): 2.5đ

#include<bits/stdc++.h>

#define max 100

using namespace std;

class graph{

public:

int a[max][max],n,start;

void readdata(){

cin>>n>>start;

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

for(int j=1;j<=n;++j) cin>>a[i][j];

}

void Euler\_Cycle(int u){

stack<int> Stack, CE;

Stack.push(u);

while(!Stack.empty()){

int s=Stack.top(),empty=1;

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

if(a[s][i]==1){

Stack.push(i);

a[s][i]=a[i][s]=0;

empty=0;

break;

}

}

if(empty==1){

Stack.pop();

CE.push(s);

}

}

while(!CE.empty()){

cout<<CE.top()<<" ";

CE.pop();

}

}

bool ok(){

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

int cnt=0;

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

if(a[i][j]) cnt++;

}

if(cnt%2!=0) return 0;

}

return 1;

}

bool ok1(){

int s,d;

d=0;

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

s=0;

for(int j=1;j<=n;++j)s+=a[i][j];

if(s%2){

d++;

}

}

if(d!=2) return 0;

return 1;

}

};

int main(){

graph g;

g.readdata();

if(g.ok()||g.ok1())

g.Euler\_Cycle(g.start);

else cout<<"no Euler cycle";

}

## Programming problem 2: Xây dựng cây bao trùm DFS sử dụng ngăn xếp (bắt buộc phải Run trước khi Evaluate để tránh treo tiến trình = tránh 0 điểm khi bị treo tiến trình): 2.5đ

#include<bits/stdc++.h>

using namespace std;

#define max 100

struct edge{

int start, end;

};

class graph{

int x[max][max], visited[max],n,e;

edge DFSTr[max];

public:

int root;

void readdata(){

cin>>n>>root;

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

for(int j=1;j<=n;++j) cin>>x[i][j];

}

void init(){

for(int i=0;i<=n;++i) visited[i]=0;

e=0;

}

void DFStree(int r){

stack<int> st;

st.push(r);

visited[r]=1;

while(!st.empty()){

int u=st.top();

st.pop();

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

if(x[u][i]&&!visited[i]){

visited[i]=1;

if(u<i){

DFSTr[e].start=u;

DFSTr[e].end=i;

}else{

DFSTr[e].end=u;

DFSTr[e].start=i;

}

st.push(u);

st.push(i);

e++;

break;

}

}

}

}

void KetQua(){

if(e==n-1){

cout<<"Stack DFS tree"<<endl;

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

cout<<DFSTr[i].start<<" "<<DFSTr[i].end<<endl;

}

}else{

cout<<"No spanning tree"<<endl;

}

}

};

int main(){

graph g;

g.readdata();

g.init();

g.DFStree(g.root);

g.KetQua();

}

## Programming problem 3: Xây dựng cây bao trùm BFS sử dụng hàng đợi (bắt buộc phải Run trước khi Evaluate để tránh treo tiến trình = tránh 0 điểm khi bị treo tiến trình): 2.5đ

#include<bits/stdc++.h>

using namespace std;

#define max 100

struct edge{

int start,end;

};

class graph{

int x[max][max],visited[max],n,e;

edge BFSTr[max];

public:

int root;

void readdata(){

cin>>n>>root;

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

for(int j=1;j<=n;++j) cin>>x[i][j];

}

void init(){

for(int i=0;i<=n;++i) visited[i]=0;

e=0;

}

void BFStree(int r){

queue<int> qe;

qe.push(r);

visited[r]=1;

while(!qe.empty()){

int u=qe.front();

qe.pop();

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

if(x[u][i]&&!visited[i]){

qe.push(i);

visited[i]=1;

if(u<i){

BFSTr[e].start=u;

BFSTr[e].end=i;

}else{

BFSTr[e].end=u;

BFSTr[e].start=i;

}

e++;

}

}

}

}

void KetQua(){

if(e==n-1){

cout<<"Queue BFS tree"<<endl;

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

cout<<BFSTr[i].start<<" "<<BFSTr[i].end<<endl;

}

}else cout<<"No spanning tree"<<endl;

}

};

int main(){

graph g;

g.readdata();

g.init();

g.BFStree(g.root);

g.KetQua();

}

## Programming problem 4: Xây dựng cây bao trùm bé nhất theo thuật toán Prim (bắt buộc phải Run trước khi Evaluate để tránh treo tiến trình = tránh 0 điểm khi bị treo tiến trình): 2.5đ

#include<bits/stdc++.h>

using namespace std;

#define max 100

struct Edge{

int head,tail,w;

};

int n,s,a[max][max];

bool used[max];

void readdata(){

cin>>n>>s;

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

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

cin>>a[i][j];

}

memset(used,0,sizeof(used));

}

void Prim(int u){

vector<Edge> MST;

int dH=0;

used[u]=1;

bool lienthong=1;

while(MST.size()<n-1){

int min\_w=INT\_MAX;

int x,y;

bool empty=1;

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

if(used[i]){

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

if(a[i][j]&&!used[j]){

empty=0;

int trongso=a[i][j];

if(trongso<min\_w){

min\_w=trongso;

x=j;

y=i;

}

}

}

}

}

if(empty){

cout<<"No spanning tree"<<endl;

lienthong=0;

break;

}

if(x<y)MST.push\_back({x,y,min\_w});

else MST.push\_back({y,x,min\_w});

dH+=min\_w;

used[x]=1;

}

if(lienthong){

cout<<"dH = "<<dH<<endl;

for(int i=0;i<MST.size();++i){

cout<<MST[i].head<<" "<<MST[i].tail<<endl;

}

}

}

int main(){

readdata();

Prim(s);

}

## Programming practice & homework 6.2: Shortest path using Dijkstra algorithm

#include<bits/stdc++.h>

using namespace std;

#define max 100

typedef long long ll;

class graph{

ll d[max],n,a[max][max],previous[max];

public:

ll s;

void readdata(){

cin>>n>>s;

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

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

cin>>a[i][j];

if(a[i][j]==0) a[i][j]=INT\_MAX;

}

}

void init(){

for(int i=0;i<=n;++i){

previous[i]=0;

}

}

void Dijkstra(int s){

vector<int> T;

for(int i=1;i<=n;++i) T.push\_back(i);

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

d[v]=INT\_MAX;

}

d[s]=0;

// for(int i=0;i<T.size();++i) cout<<T[i]<<" ";

while(!T.empty()){

int miN=INT\_MAX,u=T[0],pos=0;

for(int i=0;i<T.size();++i){

if(d[T[i]]<miN){

miN=d[T[i]];

u=T[i];

pos=i;

}

}

T.erase(T.begin()+pos);

for(int i=0;i<T.size();++i){

if(d[T[i]]>d[u]+a[u][T[i]]){

d[T[i]]=d[u]+a[u][T[i]];

previous[T[i]]=u;

}

}

}

}

void Ketqua(){

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

int u=s,v=i;

if(s==i){

cout<<"K/c "<<s<<" -> "<<v<<" = "<<d[v]<<"; ";

cout<<s<<" <- "<<s<<endl;

}else if(previous[v]!=0&&d[v]!=INT\_MAX){

cout<<"K/c "<<s<<" -> "<<v<<" = "<<d[v]<<"; ";

while(u!=v){

cout<<v<<" <- ";

v=previous[v];

}

cout<<s<<endl;

}else{

cout<<"K/c "<<s<<" -> "<<v<<" = INF"<<endl;

}

}

}

// void in(){

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

//// for(int j=1;j<=n;++j) cout<<a[i][j]<<" "

//// ;cout<<endl;

//// }

// for(int i=1;i<=n;++i) cout<<previous[i]<<" ";

// cout<<endl;

// for(int i=1;i<=n;++i) cout<<d[i]<<" ";

// }

};

int main(){

graph g;

g.init();

g.readdata();

g.Dijkstra(g.s);

g.Ketqua();

// g.in();

}