Programming practice & homework 5.1: Xây dựng cây bao trùm DFS de quy

#include<bits/stdc++.h>

using namespace std;

#define max 100

struct edge{

int dau,cuoi,w;

};

class Graph{

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

edge T[max];

public:

int gets(){

return s;

}

void init(){

for(int i=1;i<=n;++i) visited[i]=0;

neT=0;

}

void readdata(){

cin>>n>>s;

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

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

}

void treedfs(int u){

visited[u]=1;

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

if(a[u][v]&&!visited[v]){

neT++;

if(u>v){

T[neT].dau=v;

T[neT].cuoi=u;

cout<<v<<' '<<u<<endl;

}else{

T[neT].dau=u;

T[neT].cuoi=v;

cout<<u<<' '<<v<<endl;

}

treedfs(v);

}

}

};

int main(){

Graph g;

g.readdata();

g.init();

cout<<"DFS tree"<<endl;

g.treedfs(g.gets());

}

## Programming practice & homework 5.1: Xây dựng cây bao trùm DFS ngăn xếp và BFS hàng đợi

#include<bits/stdc++.h>

using namespace std;

#define max 100

struct edge{

int start,end;

};

class graph{

int x[max][max],visited[max], n, eDFS=0,eBFS=0;

edge DFSTr[max],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;

}

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[eDFS].start=u;

DFSTr[eDFS].end=i;

}else{

DFSTr[eDFS].end=u;

DFSTr[eDFS].start=i;

}

st.push(u);

st.push(i);

eDFS++;

break;

}

}

}

}

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[eBFS].start=u;

BFSTr[eBFS].end=i;

}else{

BFSTr[eBFS].end=u;

BFSTr[eBFS].start=i;

}

eBFS++;

}

}

}

}

void KetQua(){

cout<<"DFS tree"<<endl;

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

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

}

cout<<"BFS tree"<<endl;

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

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

}

}

};

int main(){

graph g;

g.readdata();

g.init();

g.DFStree(g.root);

g.init();

g.BFStree(g.root);

g.KetQua();

}

## Programming practice & homework 5.2: Xây dựng cây bao trùm bé nhất theo Kruskal

#include<bits/stdc++.h>

using namespace std;

#define max 100

struct edge{

int s,e,w;

};

class graph{

int a[max][max],n,ne,aT[max][max],neT,visited[max],dH;

edge G[max],K[max];

public:

void readdata(){

cin>>n;

ne=0;

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

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

cin>>a[i][j];

if(a[i][j]!=0 && i<j){

ne++;

G[ne].s=i;

G[ne].e=j;

G[ne].w=a[i][j];

}

}

}

void init(){

for(int i=1;i<=n;++i) visited[i]=0;

}

void dfsTree(int u){// Duyet tren cay

visited[u]=1;

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

if(aT[u][v]&&!visited[v]) dfsTree(v);

}

void bubblesort(){

for(int i=ne;i>=1;i--){

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

if(G[j].w>G[j+1].w) swap(G[j],G[j+1]);

}

}

}

void Kruskal(){

//Khoi tao cay trong

neT=0;

dH=0;

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

for(int j=1;j<=n;++j) aT[i][j]=0;

}

// Sap xep theo trong so tang dan danh sach canh

bubblesort();

//Xet canh do thi theo trong so tang dan

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

init(); dfsTree(G[i].s);

if(!visited[G[i].e]){

neT++;

K[neT].s=G[i].s;

K[neT].e=G[i].e;

K[neT].w=G[i].w;

dH+=K[neT].w;

aT[G[i].s][G[i].e]=aT[G[i].e][G[i].s]=G[i].w;

}

}

if(neT!=n-1){

cout<<"Do thi khong lien thong";

}else{

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

for(int i=1;i<=neT;++i) cout<<K[i].s<<" "<<K[i].e<<endl;

}

}

};

int main(){

graph g;

g.readdata();

g.Kruskal();

}