**BÀI TẬP ĐIỀU KIỆN MÔN TOÁN RỜI RẠC 2 - 2021**

**Câu 1:**

void DFS(int u){

int i, j, top;

top = 1;

stack[top] = u;

exet[u] = 0;

while(top > 0){

int s = stack[top];

cout << s << " ";

top--;

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

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

exet[i] = 0;

top++;

stack[top] = i;

break;

}

}

}

cout << endl;

}

**Câu 2**:

void BFS(int u){

int i, j, dau, cuoi;

dau = cuoi = 1;

queue[cuoi] = u;

exet[u] = 0;

while(dau <= cuoi){

int s = queue[dau];

dau++;

cout << s <<" ";

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

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

cuoi++;

queue[cuoi] = i;

exet[i] = 0;

}

}

}

}

**Câu 3:**

void TPLT\_DFS(int a[][]){

solt = 0;

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

if(chuaxet[i] == 1){

DFS(u);

solt++;

}

}

return solt;

}

**Câu 4;**

int TPLT\_BFS(int a[][]){

int solt = 1;

BFS(1);

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

if(exet(i) == 1){

BFS(i);

solt++;

}

}

return solt;

}

**Câu 5;**

void T\_DFS(int a[][]){

stack <int>s;

s.push(u);

chuaxet[u] = 1;

while(!s.empty()){

u = s.top();

s.pop():

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

s.push(v);

dau[c] = u;

cuoi[c] = v;

c++;

chuaxet[v] = 1;

break;

}

}

}

if(c < n - 1){

cout <<"Do thi khong lien thong" << endl;

}

else{

cout << "\nCay khung T: ";

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

if(dau[i] < cuoi[i]){

cout << "(" << dau[i] << "," << cuoi[i] <<")";

}

else{

cout << "(" << cuoi[i] << "," << dau[i] <<")";

}

}

}

**Câu 6:**

void T\_BFS(int u){

queue <int> q;

q.push(u);

chuaxet[u] = 1;

while(!q.empty()){

u = q.front();

q.pop();

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

if(a[u][v] == 1 && chuaxet[v] == 0){

q.push(v);

dau[c] = u;

cuoi[c] = v;

c++;

chuaxet[v] = 1;

}

}

}

if(c < n - 1){

cout << "Do thi khong lien thong" << endl;

}

else{

cout << "\nCay khung T: ";

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

if(dau[i] < cuoi[i]){

cout <<"("<<dau[i]<<","<<cuoi[i]<<")";

}

else{

cout << "(" << cuoi[i] << "," << dau[i] << ")";

}

}

}

}

**Câu 7;**

void EULER(int a[][]){

stack <int> s;

s.push(u);

int t = 1;

while(!s.empty()){

int v = s.top();

int x = 1;

while(x <= n && a[v][x] == 0)

x++;

if(x <= n){

s.push(x);

a[v][x] = 0;

a[x][v] = 0;

}

else{

CE[t] = v;

t++;

s.pop();

}

}

for(int i = t - 1; i > 0; i--){

cout << CE[i] <<" ";

}

}

**Câu 8:**

void DIJKSTRA(int u){

int u, v;

cout <<"Nhap duong di tu dinh s= ";

cin >> s;

cout <<"Den diem t= ";

cin >> t;

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

d[v] = a[s][v];

p[v] = s;

vs[v] = 0;

}

p[s] = 0;

d[s] = 0;

vs[s] = 1;

cout << "Duong di ngan nhat tu dinh" << s << "den diem" << t << ":" << s << " ";

while(!vs[t]){

int min = 2000;

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

if((!vs[v]) &&(d[v] < min)){

u = v;

min = d[v];

}

}

cout << u <<" ";

vs[u] = 1;

if(!vs[t]){

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

if((!vs[v]) &&((d[u] + a[u][v]) < d[v]){

d[v] = d[u] + a[u][v];

p[v] = u;

}

}

}

}

cout << "\nDo dai duong di ngan nhat= " << d[t];

}

**Câu 9:**

void FLOYD(int a[][]){

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

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

d[i][j] = a[i][j];

if(d[i][j] == max)

s[i][j] = 0;

else s[i][j] = j;

}

}

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

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

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

if(d[i][j] != max && d[i][j] > (d[i][k] + d[k][j])){

d[i][j] = d[i][k] + d[k][j];

s[i][j] = s[i][k];

}

}

}

}

}

**Câu 10;**

void PRIM (int a[][], int u){

int k, top, min, l, t, u;

int s[100];

sc = 0; w = 0; u = 1;

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

chuaxet[i] = 1;

}

top = 1;

s[top] = u;

chuaxet[u] = 0;

while(sc < n - 1){

min = MAX;

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

t = s[i];

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

if(chuaxet[j] && min > a[i][j]){

min = a[i][j];

k = t;

i = j;

}

}

}

sc++;

w += min;

T[sc][l] = k;

T[sc][2] = l;

chuaxet[l] = 0;

a[k][l] = MAX;

a[i][k] = MAX;

top++;

s[top] = 1;

}

}

**Câu 11;**

void Heap(int first, int last){

int j, k, t1, t2, t3;

j = first;

while(j <= (last / 2)){

if((2 \* j) < last && w[2 \* j + 1] < w[2 \* j]){

k = 2 \* j + 1;

}

else k = 2 \* j;

if(w[k] < w[j]){

t1 = dau[j];

t2 = cuoi[j];

t3 = w[j];

dau[j] = dau[k];

cuoi[j] = cuoi[k];

w[j] = w[k];

dau[k] = t1;

cuoi[k] = t2;

w[k] = t3;

j = k;

}

else j = last;

}

}

int Find(int i){

int tro = i;

while(f[tro] > 0) tro = f[tro];

return tro;

}

void Union(int i, int j){

int x = f[i] + f[j];

if(f[i] > f[j]){

f[i] = j;

f[j] = x;

}

else{

f[j] = i;

f[i] = x;

}

}

void KRUSKAL(int a[][]){

int i, last, u, v, r1, r2, nCanh, nDinh;

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

f[i] = -1;

for(int i = m / 2; i > 0; i++){

Heap(i,m);

}

last = m;

nCanh = 0;

nDinh = 0;

min1 = 0;

connect = TRUE;

while(nDinh < n - 1 && nCanh < m){

nCanh++;

u = dau[1];

v = cuoi[1];

r1 = Find(u);

r2 = Find(v);

if(r1 != r2){

nDinh++;

Union(r1, r2);

daut[nDinh] = u;

cuoit[nDinh] = v;

min1 += w[1];

}

dau[1] = dau[last];

cuoi[1] = cuoi[last];

w[1] = w[last];

last--;

Heap(1,last);

}

if(nDinh != n - 1)

connect = FALSE;

}

**Câu 12;**

#include <bits/stdc++.h>

#define V 10

using namespace std;

bool readFile(int graph[V][V], int &n){

ifstream read("DT.INP");

if(read.is\_open()){

read >> n;

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

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

read >> graph[i][j];

}

}

read.close();

return true;

}

return false;

}

bool writeFile(int graph[V][V], int n, int value){

ofstream write("DT.OUT");

if(write.is\_open()){

write << value << endl;

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

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

write << graph[i][j] <<" ";

}

write << endl;

}

return true;

}

return false;

}

bool BFS(int rGraph[V][V], int n, int value){

bool visited[V];

memse(visited, 0, sizeof(visited));

queue <int> q;

q.push(s);

visited[s] = true;

parent[s] = -1;

while(!q.empty()){

int u == q.front();

q.pop();

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

if(visited[v] == false && rGraph[u][v] > 0){

q.push(v);

parent[v] = u;

visited[v] = true;

}

}

}

return (visited[t] == true);

}

int Ford\_Fulkerson(int graph[V][V], int s, int t){

int n = t, u, v;

int rGraph[V][V];

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

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

rGraph[u][v] = graph[u][v];

}

}

int parent[V];

int max\_flow = 0;

while(BFS(rGraph, s, t, n, parent)){

int path\_flow = INT\_MAX;

for(v = t; v != s; v = parent[v]){

u = parent[v];

rGraph[u][v] -= path\_flow;

rGraph[v][u] += path\_flow;

}

max\_flow += path\_flow;

}

return max\_flow;

}

int main()

{

int n, graph[V][V];

if(!readFile(graph, n)){

cout << "Unable to read file.";

}

int val = Ford\_Fulkerson(graph, 1, n);

if(!writeFile(graph, n, val)){

cout << "Unable to write file.";

}

return 0;

}