 **🙠🙟🕮🙝🙢**

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**~~~~~~\*~~~~~~**



**BÀI TẬP ĐIỀU KIỆN**

**Môn: Toán rời rạc 2**

|  |  |  |
| --- | --- | --- |
| ***Tên sinh viên*** | **:** | **TRỊNH VINH TUẤN ĐẠT** |
| ***Mã sinh viên*** | **:** | **B21DCCN031** |
| ***Giảng viên bộ môn***  ***Nhóm*** | **:**  **:** | **Vũ Văn Thỏa**  **06** |

**HÀ NỘI – 2023**



**BÀI TẬP ĐIỀU KIỆN**

**Bài 1:** **Viết hàm có tên là DFS(int u) trên C/C++ mô tả thuật toán duyệt theo chiều sâu các đỉnh của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận kề a[ ] [ ].**

**Bài Làm:**

int a[100][100], vs[100]= {0} , n, u;

void DFS(int u)

{

cout<<u<<" ";

vs[u] = 1;

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

{

if(a[u][i] == 1 && vs[i] == 0)

{

DFS(i);

}

}

}

**Bài 2:** **Viết hàm có tên là BFS( int u) trên C/C++ mô tả thuật toán duyệt theo chiều rộng các đỉnh của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận kề a[ ] [ ].**

**Bài Làm:**

int a[100][100], n, u, vs[100], q[100];

void BFS(int u)

{

int v, dq = 1, cq = 0;

cq++; q[cq] = u; vs[u] = 1;

while (dq <= cq)

{

v = q[dq]; dq++;

cout << v << ” ”;

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

if (vs[i]==0 && a[v][i]==1)

{

cq++; q[cq] = i;

vs[i] = 1;

}

}

}

**Bài 3: Viết hàm có tên là int TPLT\_DFS(int a[ ] [ ]) trên C/C++ tìm số thành phần liên thông của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận kề a[ ] [ ] bằng cách sử dụng hàm DFS(int u) đã biết mô tả thuật toán duyệt theo chiều sâu các đỉnh của đồ thị G.**

**Bài Làm:**

int a[100][100],lt[100], n;

void DFS(int u)

{

vs[u] = true;

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

{

if(a[u][i] == 1 && vs[i] == 0)

{

DFS(i);

}

}

}

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

{

int u, k= 0;

for (u= 1; u<= n; u++) lt[u]= 0;

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

if (lt[u] == 0)

{

k++;

DFS(u);

}

Return k;

}

**Bài 4:** **Viết hàm có tên là int TPLT\_BFS(int a[ ] [ ]) trên C/C++ tìm số thành phần liên thông của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận kề a[ ] [ ] bằng cách sử dụng hàm BFS(int u) đã biết mô tả thuật toán duyệt theo chiều rộng các đỉnh của đồ thị G.**

**Bài Làm:**

int a[100][100], lt[100], n, dq, cq, q[100];

void BFS(int u)

{

int v, dq = 1, cq = 0;

cq++; q[cq] = u; vs[u] = 1;

while (dq <= cq)

{

v = q[dq]; dq++;

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

if (vs[i]==0 && a[v][i]==1)

{

cq++; q[cq] = i;

vs[i] = 1;

}

}

}

int TPLT\_BFS(int a[][100])

{

int u, int k= 0;

for (u= 1; u<= n; u++) lt[u]= 0;

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

if (lt[u] == 0)

{

k++;

Bfs(u);

}

return k;

}

**Bài 5:** **Viết hàm có tên là T\_DFS(int a[ ] [ ]) trên C/C++ tìm cây khung T[ ] của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận kề a[ ] [ ] bằng cách sử dụng hàm DFS(int u) đã biết mô tả thuật toán duyệt theo chiều sâu các đỉnh của đồ thị G.**

**Bài Làm :**

int a[100][100], n, u, vs[100], e[100];

void DFS (int u)

{

vs[u]= 1;

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

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

{

e[v]= u;

DFS(v);

}

}

void T\_DFS(int a[][100])

{

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

vs[v] = 0;

DFS(u);

int dem= 0;

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

if (vs[v] ==1)

dem++;

if (dem == n)

{

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

if (e[v] != 0)

cout << v << " " << e[v] << endl;

}

else

cout << "Khong co Cay khung";

}

Bài 6 : **Viết hàm có tên là T\_BFS(int a[ ] [ ]) trên C/C++ tìm cây khung T[ ] của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận kề a[ ] [ ] bằng cách sử dụng hàm BFS(int u) đã biết mô tả thuật toán duyệt theo chiều rộng các đỉnh của đồ thị G.**

**Bài Làm :**

int a[100][100], n, u, vs[100], e[100], q[100];

void BFS(int u)

{

int v, dq= 1, cq= 0;

vs[u]= 1; e[u]= 0;

cq++; q[cq]= u;

while (dq <= cq)

{

int i = q[dq];

dq++;

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

if (vs[v]==0 && a[i][v]==1)

{

e[v]= u;

vs[v]= 1;

cq++; q[cq]= v;

}

}

}

void T\_BFS(int a[][])

{

int v;

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

vs[v] = 0;

BFS(u);

int dem= 0;

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

if (vs[v] ==1) dem++;

if (dem == n)

{

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

if (e[v] != 0)

cout << v << " " << e[v] << endl;

}

else

cout << "Khong co Cay khung";

}

**Bài 7** : **Viết hàm có tên là EULER(int a[ ] [ ]) trên C/C++ tìm chu trình/đường đi Euler CE[ ] của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận kề a[ ] [ ], biết rằng G là đồ thị Euler/nửa Euler**.

**Bài Làm :**

int n ,u ;

void EULER(int a[][100])

{

int top=0, v;

top++; s[top]= u; k= 0;

while (top > 0)

{

int v= s[top];

int ok= 1;

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

if (a[v][x] ==1)

{

top++; s[top]= x; ok= 1;

a[v][x]= 0; break;

}

if (ok== 1)

{

k++, ce[k]= v; top--;

}

}

for (v= k; v> 0; v--)

cout << ce[v] << “ ”;

}

**Bài 8** : **Viết hàm có tên là DIJKSTRA(int u) trên C/C++ tìm đường đi ngắn nhất d[v] xuất phát từ đỉnh u đến các đỉnh v của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận trọng số a[ ] [ ].**

**Bài Làm :**

int n, u, a[100][100], d[100], e[100], vs[100];

void DIJKSTRA(int u) {

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

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

e[v] = u;

}

d[u] = 0, e[u] = 0, vs[u] = 1;

while(1) {

int s = 0, Min = 32767, v;

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

if(vs[v] == 0 && d[v] < Min) {

s = v;

Min = d[v];

}

}

if(s == 0) {

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

if(v != u) {

cout << "d[" << v << "]=" << d[v] << " ";

cout << " ";

int tmp = v;

cout << tmp;

while (e[tmp] != 0) {

cout << "<-" << e[tmp];

tmp = e[tmp];

}

cout << endl;

}

}

return;

}

vs[s] = 1;

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

if(vs[v] == 0 && d[v] > d[s] + a[s][v]) {

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

e[v] = s;

}

}

}

}

**Bài 9** **: Viết hàm có tên là FLOYD(int a[ ] [ ]) trên C/C++ tìm đường đi ngắn nhất d[ ] [ ] giữa các cặp đỉnh của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận trọng số a[ ] [ ].**

**Bài Làm :**

int n, a[100][100], d[100][100], e[100][100];

int FLOYD(int a[][100]){

int i, j, k;

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

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

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

e[i][j] = i;

}

}

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

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

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

if(d[i][j] > d[i][k] + d[k][j]) {

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

e[i][j] = k;

}

}

}

}

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

if(d[i][i] < 0) return 0;

}

*//in ra đường đi ngắn nhất d[][] giữa các cặp đỉnh*

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

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

if(i!=j) {

cout << d[i][j] << " ";

if(e[i][j] != i) {

cout << j << "<-" << e[i][j] << "<-" << i;

}

else {

cout << j << "<-" << i;

}

cout << endl;

}

}

}

return 1;

}

**Bài 10** : **Viết hàm có tên là PRIM(int a[ ] [ ], int u) trên C/C++ tìm cây khung T[ ] nhỏ nhất bắt đầu tại đỉnh u của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận trọng số a[ ] [ ] bằng cách sử dụng thuật toán Prim.**

**Bài Làm :**

int n, a[100][100], d[100], e[100], vs[100];

void PRIM(int s)

{

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

{

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

e[v] = s;

}

vs[s] = 1;

d[s] = 0;

e[s] = 0;

int wt = 0, dem = 1;

while (dem < n)

{

int u = 0;

int MIN = 10001;

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

{

if (vs[v] == 0 && d[v] < MIN)

{

MIN = d[v];

u = v;

}

}

if (u == 0)

{

cout << "Khong co cay khung";

return;

}

vs[u] = 1;

wt = wt + a[u][e[u]];

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

{

if (vs[v] == 0 && d[v] > a[u][v])

{

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

e[v] = u;

}

}

dem++;

}

cout << wt << endl;

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

{

if (e[v] != 0)

{

cout << v << " " << e[v] << endl;

}

}

return;

}

**Bài 11** : **Viết hàm có tên là KRUSKAL(int a[ ] [ ]) trên C/C++ tìm cây khung T[ ] nhỏ nhất của đồ thị G = <V, E> được biểu diễn dưới dạng ma trận trọng số a[ ] [ ] bằng cách sử dụng thuật toán Kruskal**.

**Bài Làm :**

int n, m, d[10000], c[10000], ts[10000];

int t[100], vs[100];

void KRUSKAL(int a[100][100])

{

for (int i = 1; i <= m - 1; i++)

{

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

{

if (ts[i] > ts[j])

{

int tg = ts[i];

ts[i] = ts[j];

ts[j] = tg;

tg = d[i];

d[i] = d[j];

d[j] = tg;

tg = c[i];

c[i] = c[j];

c[j] = tg;

}

}

}

int wt = 0, k = 0;

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

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

{

if (!(vs[d[i]] != 0 && vs[d[i]] == vs[c[i]]))

{

k++;

t[k] = i;

wt = wt + ts[i];

if (k == n - 1)

{

cout << wt << endl;

for (int j = 1; j <= k; j++)

{

cout << d[t[j]] << " " << c[t[j]] << endl;

}

return;

}

int u = d[i], v = c[i];

if (vs[u] == 0 && vs[v] == 0)

{

vs[u] = k; vs[v] = k;

}

else if (vs[u] == 0 && vs[v] != 0) vs[u] = vs[v];

else if (vs[u] != 0 && vs[v] == 0) vs[v] = vs[u];

else if (vs[u] < vs[v])

{

int tg = vs[v];

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

{

if (vs[j] == tg) vs[j] = vs[u];

}

}

else if (vs[v] < vs[u])

{

int tg = vs[u];

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

{

if (vs[j] == tg) vs[j] = vs[v];

}

}

}

}

}

**Bài 12 :** **Viết chương trình hoàn chỉnh tìm luồng cực đại f[][] trên mạng G = <V, E> được biểu diễn dưới dạng ma trận trọng số c[ ] [ ] với đỉnh phát s và đỉnh thu t bằng cách sử dụng thuật toán đường tăng luồng dựa trên Ford-Fulkerson :**

**Yêu cầu :**

**(1) Nhập ma trận trọng số biểu diễn G từ tệp DT.INP ; với s= 1 ; t= n ;**

**(2) Tìm luồng cực đại f ;**

**(3) Xuất kết quả ra tệp DT.OUT :**

**- Dòng đầu ghi Val(f) ;**

**- N dòng sau ghi f[i][j] ;**

**Bài Làm :**

#include <bits/stdc++.h>

using namespace std;

int Stop = 0, q[100], d[100], vs[100], e[100], fl[100][100];

int c[100][100], n, valf;

*//res là hàm cập nhật trọng số nhỏ nhất và tăng luồng fl*

void res(int t) {

int phi = d[t];

valf += phi;

while(e[t] != 0) {

if(e[t] > 0) {

fl[e[t]][t] += phi;

}

else if(e[t] < 0) {

int h = abs(e[t]);

fl[t][h] -= phi;

}

t = abs(e[t]);

}

}

*//Hàm tìm đường tăng luồng bằng BFS*

void FindPath(int s, int t){

int cq, dq, u, v;

for (u = 1; u <= n; u++) vs[u] = 0;

cq = 1; dq = 1; q[cq] = s; vs[s] = 1; e[s] = 0; d[s] = 10000;

while (dq <= cq){

u = q[dq]; dq++;

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

if (vs[v]== 0) {

if (c[u][v] > 0 && fl[u][v] < c[u][v]) {

e[v] = u;

d[v] = (d[u]< c[u][v] - fl[u][v])?d[u]: c[u][v] - fl[u][v];

cq++; q[cq] = v; vs[v] = 1;

if (v == t) {

res(t);

return;

}

}

if (c[v][u] > 0 && fl[v][u] > 0) {

e[v] = -u;

d[v] = (d[u]< fl[v][u])?d[u]: fl[v][u];

cq++; q[cq] = v; vs[v] = 1;

if (v == t) {

res(t);

return;

}

}

}

}

Stop = 1;

}

*//Thuật toán Max\_Flow*

void Max\_Flow() {

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

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

fl[i][j] = 0;

}

}

Stop = 0;

while(!Stop) {

FindPath(1, n);

}

}

int main () {

freopen("DT.INP", "r", stdin);

freopen("DT.OUT", "w", stdout);

cin >> n;

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

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

cin >> c[i][j];

}

}

Max\_Flow();

*//xuất ra kết quả*

cout << "Gia tri luong cuc dai: Val(f) = " << valf;

cout << endl;

cout << "Luong cuc dai f: " << endl;

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

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

cout << fl[i][j] << " ";

}

cout << endl;

}

}

/\*

Kết quả chạy thuật toán:

DT.INP:

6

0 5 5 0 0 0

0 0 0 6 3 0

0 0 0 3 1 0

0 0 0 0 0 6

0 0 0 0 0 6

0 0 0 0 0 0

DT.OUT:

Gia tri luong cuc dai: Val(f) = 9

Luong cuc dai f:

0 5 4 0 0 0

0 0 0 3 2 0

0 0 0 3 1 0

0 0 0 0 0 6

0 0 0 0 0 3

0 0 0 0 0 0

\*/