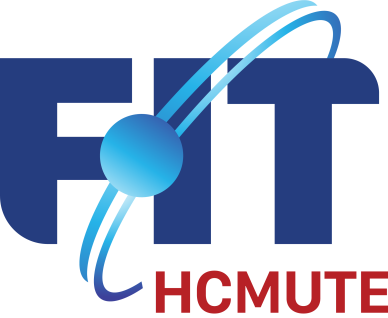
**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

🞠◊🞠◊🞠

****

**BÁO CÁO ĐỒ ÁN**

**ĐỒ ÁN CÔNG NGHỆ THÔNG TIN**

**ĐỀ TÀI**

**GRAPH**

Giảng viên hướng dẫn: ThS.Nguyễn Quang Ngọc

Nhóm: 03

TP. Hồ Chí Minh, tháng 11 năm 2023

**ĐH SƯ PHẠM KỸ THUẬT TPHCM XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**KHOA CNTT Độc lập - Tự do - Hạnh phúc**

**\*\*\*\*\*\*\* \*\*\*\*\*\*\***

**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Họ và tên Sinh viên : Đỗ Minh Cường MSSV: 21110147

Nguyễn Minh Quang MSSV: 20143481

Ngành: Công nghệ Thông tin

Tên đề tài: GRAPH

Họ và tên Giáo viên hướng dẫn: ThS Nguyễn Quang Ngọc

**Nhận xét**

1. Về nội dung đề tài & khối lượng thực hiện:

1. Ưu điểm

1. Khuyết điểm

1. Đánh giá loại:
2. Điểm: Tp.Hồ Chí Minh, ngày tháng 11 năm 2023

Giáo viên hướng dẫn

*(Ký & ghi rõ họ tên*

MỤC LỤC

Contents

[PHẦN I: MỞ ĐẦU 3](#_Toc151737082)

[1. Lý do chọn đề tài 3](#_Toc151737083)

[2. Mục đích của đề tài, đối tượng và phạm vi nghiên cứu 3](#_Toc151737084)

[3. Kết quả dự kiến đạt được 2](#_Toc151737085)

[4. Phân công nhiệm vụ: 3](#_Toc151737087)

[PHẦN II: THIẾT KẾ 7](#_Toc151737088)

[1. Tạo đồ thị 7](#_Toc151737089)

[2. Xuất đồ thị 11](#_Toc151737090)

[3. Thêm một đỉnh vào đồ thị đã có 14](#_Toc151737091)

[4. Thêm một cạnh: 15](#_Toc151737092)

[5. Xuất các tên đỉnh, tên cạnh: 17](#_Toc151737093)

[6. Xuất thông tin 1 đỉnh, 1 cạnh: 18](#_Toc151737094)

[7. Thay đổi trọng số của cạnh: 20](#_Toc151737095)

[8.Đường đi ngắn nhất từ đỉnh v đến đỉnh w: 21](#_Toc151737096)

[9.Kiểm tra chu trình Euler: 25](#_Toc151737097)

[10.Cây khung bé nhất: 31](#_Toc151737098)

[11.Duyệt đồ thị theo chiều rộng: 34](#_Toc151737099)

[12.Duyệt đồ thị theo chiều sâu: 37](#_Toc151737100)

[PHẦN III: TỔNG KẾT 40](#_Toc151737101)

[1. Đánh giá Sinh viên: 40](#_Toc151737102)

[2. Khó khăn: 40](#_Toc151737103)

[3. Ưu điểm: 40](#_Toc151737104)

[4. Nhược điểm: 40](#_Toc151737105)

[5. Ý tưởng phát triển: 40](#_Toc151737106)

# **PHẦN I: MỞ ĐẦU**

1. **Lý do chọn đề tài**

Lựa chọn đề tài GRAPH được đánh giá cao vì một số lý do quan trọng. Đầu tiên, đồ thị là một khái niệm cơ bản và rất quan trọng trong lĩnh vực khoa học máy tính và toán học. Việc hiểu rõ về đồ thị mang lại cơ hội tìm hiểu sâu sắc về cấu trúc dữ liệu và thuật toán, đồng thời mở ra nhiều khả năng trong việc giải quyết các vấn đề phức tạp.

Thứ hai, đồ thị được sử dụng rộng rãi trong nhiều ứng dụng thực tế như mô hình hóa mạng lưới xã hội, quản lý tuyến đường giao thông, hoặc phân tích dữ liệu mạng. Việc áp dụng kiến thức về đồ thị có thể giúp mở rộng khả năng giải quyết vấn đề và tạo ra những ứng dụng thực tế có giá trị.

Thứ ba, nắm vững kiến thức về đồ thị là quan trọng trong nhiều lĩnh vực nghiên cứu, bao gồm cả trí tuệ nhân tạo, thuật toán tối ưu, và quy hoạch mạng. Việc tìm hiểu và ứng dụng đồ thị sẽ mở ra cánh cửa cho việc tham gia vào các dự án nghiên cứu tiên tiến và đóng góp vào sự phát triển của lĩnh vực này.

Cuối cùng, chọn đề tài này cũng tạo cơ hội để kết nối với cộng đồng nghiên cứu về đồ thị thông qua việc tham gia vào các dự án mã nguồn mở và chia sẻ kiến thức. Điều này không chỉ mở rộng mạng lưới liên kết mà còn góp phần vào sự phát triển bền vững của cộng đồng nghiên cứu về đồ thị. Tóm lại, việc tìm hiểu về đồ thị và ứng dụng nó không chỉ mang lại kiến thức chuyên sâu mà còn là hành trình đưa kiến thức vào thực tế và góp phần vào sự phát triển của cộng đồng.

1. **Mục đích của đề tài, đối tượng và phạm vi nghiên cứu**

Mục đích của đề tài về đồ thị sử dụng ngôn ngữ lập trình C++ là áp dụng kiến thức về lý thuyết đồ thị để xây dựng một ứng dụng giúp giải quyết các vấn đề liên quan đến đồ thị. Đối tượng của đề tài là một ứng dụng được xây dựng và thiết kế bởi một nhóm gồm 2 sinh viên. Mục đích chính của ứng dụng là cung cấp các công cụ hiệu quả để thực hiện các phép tính và phân tích trên đồ thị.

Phạm vi nghiên cứu bao gồm sử dụng nhiều nguồn thông tin như Google và YouTube để tìm hiểu về lý thuyết đồ thị, từ cơ bản đến chuyên sâu. Ngoài ra, phạm vi còn bao gồm việc áp dụng kiến thức từ môn học và các kiến thức liên quan để phát triển ứng dụng và giải quyết các vấn đề cụ thể liên quan đến đồ thị trong lập trình C++.

Đánh giá và phân tích chung có thể tập trung vào sự hiệu quả và linh hoạt của ứng dụng trong việc xử lý đồ thị, đồng thời cũng có thể đề cập đến các thách thức và giải pháp tối ưu hóa khi sử dụng ngôn ngữ lập trình C++.

**3. Kết quả dự kiến đạt được**

Xây dựng một ứng dụng có khả năng giải quyết các vấn đề đồ thị cơ bản và nâng cao, đồng thời đáp ứng được nhu cầu tính toán của đối tượng sinh viên trong các môn học liên quan đến đồ thị và lập trình C++. Đánh giá tổng chung có thể chỉ ra sự hiệu quả và tính ứng dụng của ứng dụng trong thực tế.

## **4. Phân công nhiệm vụ:**

|  |  |  |
| --- | --- | --- |
| **Tên sinh viên** | **Tỉ lệ đóng góp** | **Nhiệm vụ** |
| **Nguyễn Minh Quang** | **100%** | Xuất thông tin 1 đỉnh, 1 cạnh  Xuất ma trận kề, ma trận liên kết,  Duyệt đồ thị theo chiều rộng, chiều xâu,  Đường đi ngắn nhất từ đỉnh v đến đỉnh w  Kiểm tra có chu trình Euler,  Cây khung bé nhất.  Dòng chảy lớn nhất(50%) |
| **Đỗ Minh Cường** | **100%** | Tạo một đồ thị,  Thêm một đỉnh vào đồ thị đã có,  Thay đổi thông tin của một đỉnh,  Thêm cạnh,  Thay đổi trọng số của cạnh,  Xuất các tên đỉnh, tên cạnh,  Dòng chảy lớn nhất(50%) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tiến trình hoàn thành môn Đồ án CNTT | | | | | | | | | | | | | | | | |
|  | Các phương thức | TKB | | | | | | | | | | | | | Cường | Quang |
| 1 | Tạo 1 đồ thị | o | o |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Thêm một đỉnh vào đồ thị đã có |  |  | o |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Thay đổi thông tin của một đỉnh |  |  | o |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Thêm cạnh |  |  |  | o |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Thay đổi trọng số của cạnh |  |  |  |  | o |  |  |  |  |  |  |  |  |  |  |
| 6 | Xuất các tên đỉnh, tên cạnh |  |  |  | o |  |  |  |  |  |  |  |  |  |  |  |
| 7 | Xuất thông tin 1 đỉnh, 1 cạnh | o | o |  |  | o |  |  |  |  |  |  |  |  |  |  |
| 8 | Xuất ma trận kề, ma trận liên kết |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Duyệt đồ thị theo chiều rộng, chiều xâu |  |  |  |  |  |  | o |  |  |  |  |  |  |  |  |
| 10 | Đường đi ngắn nhất từ đỉnh v đến đỉnh w |  |  |  |  |  | o |  |  |  |  |  |  |  |  |  |
| 11 | Kiểm tra có chu trình Euler |  |  |  |  |  |  |  | o |  |  |  |  |  |  |  |
| 12 | Cây khung bé nhất |  |  |  |  |  |  |  |  | o |  |  |  |  |  |  |
| 13 | Tìm dòng chảy lớn nhất (NetworkModel) |  |  |  |  |  |  |  |  |  | o | o | o |  |  |  |
| Day | | 03/09/2023 | 10/09/2023 | 17/09/2023 | 24/09/2023 | 24/09/2023 | 30/09/2023 | 7/10/2023 | 14/10/2023 | 21/10/2023 | 28/10/2023 | 4/11/2023 | 11/11/2023 |  |  |  |
| Week | | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 |  |  |  |
| Note | |  |  |  |  |  |  |  |  |  | o – Begin  o – Complete 50%  o – Complete 100% | | | | | |

**PHẦN II: THIẾT KẾ**

## **1. Tạo đồ thị**

Khai báo thư viện <iostream> và <fstream> trong IDE Dev-C++. Trong đó thư viện <fstream> để nhập dữ liệu đầu vào bằng file thay vì người sử dụng tự nhập để dễ dàng, tiện lợi hơn trong quá trình sử dụng ứng dụng.

#include <iostream>

#include <fstream>

Tạo hàm createGraph()

void createGraph() {

do {

cout << "Chon cach nhap ma tran ke: " << endl;

cout << "1.Nhap bang tay" << endl;

cout << "2.Nhap bang file" << endl;

cin >> chooseGraph;

}while(chooseGraph !=1 && chooseGraph !=2);

if(chooseGraph==1) {

input(a,n);

}

else {

inputFile(a,n);

}

}

Trong hàm createGraph() sẽ có 2 lựa chọn để người dụng có thể chọn. Lựa chọn đầu tiên là nhập bằng tay, lựa chọn thứ hai sẽ import từ file có sẵn trong máy tính.

Với lựa chọn đầu tiên là nhập bằng tay, tạo ra hàm input().

void input(int a[][100],int &n) {

do {

cout << "Nhap so luong dinh cua do thi : ";

cin >>n;

}while(n<=0||n>100);

cout << "Nhap " << n << " ten cac dinh cua do thi theo thu tu : ";

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

{

cin >> vertex[i];

}

//nhap ma tran

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

{

cout << "Nhap tinh chat ke cua cac dinh voi dinh " << vertex[i] << " co(1), khong(0) : ";

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

{

cin >> a[i][j];

}

}

}

Với lựa chọn thứ hai là import từ file bên ngoài, tạo ra hàm inputFile ().

void inputFile(int a[][100],int &n) {

ifstream iFile;

iFile.open("lienketke.txt");

if(iFile.is\_open()) {

iFile >>n;

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

iFile >> vertex[i];

}

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

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

iFile >> a[i][j];

}

}

iFile.close();

cout << "Doc file thanh cong !!\n";

}

else {

cout << "Loi, khong mo duoc file !!\n";

}

}

void input(int a[][100],int &n) {

do {

cout << "Nhap so luong dinh cua do thi : ";

cin >>n;

}while(n<=0||n>100);

cout << "Nhap " << n << " ten cac dinh cua do thi theo thu tu : ";

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

{

cin >> vertex[i];

}

//nhap ma tran

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

{

cout << "Nhap tinh chat ke cua cac dinh voi dinh " << vertex[i] << " co(1), khong(0) : ";

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

{

cin >> a[i][j];

}

}

}

Kết quả khi chạy chương trình :

A screenshot of a computer

Description automatically generated

Khi chọn số 1:

A black background with white text

Description automatically generated

Chọn tiếp số 1 để nhập tay:

A screenshot of a computer

Description automatically generated

Khi đã nhập xong dữ liệu. Đồ thị sẽ được lưu và chương trình sẽ tiếp tục hỏi lại chức năng bạn muốn sử dụng.

**2. Xuất đồ thị**

Sau khi đã nhập xong, tạo hàm xuất ra đồ thị ma trận kề là output().

void output(int a[][100],int n) {

for(int i=0;i<n;i++) cout << "----------";

cout << endl;

cout << "|";

cout << "\t";

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

cout << "|" << vertex[i] << "\t";

}

cout << "|" << endl;

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

cout << "|" << vertex[i] << "\t";

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

cout << "|" << a[i][j] << "\t";

}

cout << "|" << endl;

}

for(int i=0;i<n;i++) cout << "----------";

}

Tạo hàm Begin() để khởi tạo chương trình.

int Begin() {

cout << "------------------------------------------------------------------------------------------------------------------------\n";

cout << "CHAO MUNG DEN VOI CHUONG TRINH DO THI GRAPH\n";

cout << "-1.Thoat chuong trinh\n";

cout << "0.Xoa man hinh\n";

cout << "1.Tao do thi\n";

cout << "2.Xuat ma tran ke\n";

cout << "------------------------------------------------------------------------------------------------------------------------\n";

cout << "LUA CHON CUA BAN LA : ";

cin >> chooseBegin;

return chooseBegin;

}

Cuối cùng liên kết các hàm đó vào hàm main().

int main() {

chooseBegin = Begin();

switch(chooseBegin) {

case -1 : return 0;

case 0 : system("cls");

main();

break;

case 1 : createGraph();

main();

break;

case 2 : output(a,n);

main();

break;

default : cout << "Vui long chon so hop le !!!\n";

main();

}

return 0;

}

Nếu chọn 2 để xuất ma trận kề:

A black and white screen with white text

Description automatically generated

Dữ liệu file lienketke.txt:

A screenshot of a computer

Description automatically generated

Khi muốn nhập đồ thị khác bằng file:

A black background with white text

Description automatically generated

Và kết quả khi xuất ma trận kề thay đổi:

A screenshot of a computer

Description automatically generated

## **3. Thêm một đỉnh vào đồ thị đã có**

Tạo hàm addvertex().

void addvertex(int a[][100],int &n) {

if(n<100) {

n++;

cout << "Nhap ten dinh moi: ";

cin >> vertex[n-1];

//nhap ma tran

cout << "Nhap tinh chat ke cua dinh " << vertex[n-1] << " voi dinh khac (co(1), khong(0)): \n";

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

{

cout << "Voi dinh " << vertex[i] << " :";

cin >> a[i][n-1];

a[n-1][i] = a[i][n-1];

}

}

Tạo hàm addvertex().

A black screen with white text

Description automatically generated

Khi chọn lựa số 3 và nhập các dữ liệu đầu vào :

A black screen with white text

Description automatically generated

Kết quả khi xuất ma trận.

## **4. Thêm một cạnh:**

Tạo hàm addEdge().

void addEdge(int a[][100], int n) {

char sourceVertex, destinationVertex;

int sourceIndex = -1, destinationIndex = -1;

cout << "Nhap ten dinh nguon: ";

cin >> sourceVertex;

cout << "Nhap ten dinh dich: ";

cin >> destinationVertex;

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

if (vertex[i] == sourceVertex) {

sourceIndex = i;

}

if (vertex[i] == destinationVertex) {

destinationIndex = i;

}

}

if (sourceIndex == -1 || destinationIndex == -1) {

cout << "Mot hoac ca hai dinh khong ton tai trong do thi!" << endl;

return;

}

a[sourceIndex][destinationIndex] = 1;

a[destinationIndex][sourceIndex] = 1;

cout << "Them canh thanh cong!" << endl;

}

Đồ thị gốc:

A screenshot of a computer

Description automatically generated

Nhập dữ liệu để thêm cạnh:

A screenshot of a computer

Description automatically generated

Kết quả khi xuất ma trận (cạnh A, D đã được thêm vào đồ thị):

A screenshot of a computer

Description automatically generated

## **5**. **Xuất các tên đỉnh, tên cạnh:**

Tạo hàm printVertexEdge():

void printVertexEdge(int a[][100], int n) {

cout << "Cac dinh la: ";

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

cout << vertex[i] << ", ";

}

cout << endl;

cout << "Cac canh la: ";

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

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

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

cout << vertex[i] << vertex[j] << ", ";

}

}

}

cout << endl;

}

Kết quả khi chạy chương trình:

A screenshot of a computer

Description automatically generated

## **6. Xuất thông tin 1 đỉnh, 1 cạnh:**

Tạo hàm displayVertexEdgeInfo()

void displayVertexEdgeInfo(int a[][100], int n)

{

char vertexName;

cout << "Nhap ten dinh can xuat thong tin: ";

cin >> vertexName;

int vertexIndex = -1;

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

if (vertex[i] == vertexName) {

vertexIndex = i;

break;

}

}

if (vertexIndex == -1) {

cout << "Dinh khong ton tai trong do thi!" << endl;

return;

}

cout << "Thong tin cua dinh " << vertexName << ":" << endl;

cout << "Cac dinh ke voi " << vertexName << ": ";

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

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

cout << vertex[i] << ", ";

}

}

cout << endl;

}

Đồ thị gốc:

A screenshot of a computer

Description automatically generated

Khi chạy lệnh:

A black screen with white text

Description automatically generated

## **7. Thay đổi trọng số của cạnh:**

Tạo hàm changeWeight():

void changeWeight(int a[][100], int n)

{

char vertexName1, vertexName2;

int weight;

cout << "Nhap ten dinh dau cua canh: ";

cin >> vertexName1;

cout << "Nhap ten dinh cuoi cua canh: ";

cin >> vertexName2;

cout << "Nhap gia tri moi cua trong so: ";

cin >> weight;

int vertexIndex1 = -1;

int vertexIndex2 = -1;

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

if (vertex[i] == vertexName1) {

vertexIndex1 = i;

}

if (vertex[i] == vertexName2) {

vertexIndex2 = i;

}

}

if (vertexIndex1 == -1 || vertexIndex2 == -1) {

cout << "Mot trong hai dinh khong ton tai trong do thi!" << endl;

return;

}

a[vertexIndex1][vertexIndex2] = weight;

a[vertexIndex2][vertexIndex1] = weight;

cout << "Da thay doi trong so cua canh " << vertexName1 << "-" << vertexName2 << " thanh " << weight << endl;

}

Đồ thị gốc:

A screenshot of a computer

Description automatically generated

Kết quả khi chạy lệnh:

A black background with white text

Description automatically generated

A black and white screen with white text

Description automatically generated

## **8.Đường đi ngắn nhất từ đỉnh v đến đỉnh w:**

Tạo hàm dijkstra ().

void dijkstra(int a[][100], int n) {

char src, dest;

cout << "Nhap dinh nguon: ";

cin >> src;

cout << "Nhap dinh dich: ";

cin >> dest;

int dist[100];

bool sptSet[100];

int parent[100];

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

dist[i] = INT\_MAX;

sptSet[i] = false;

}

int srcIndex = -1, destIndex = -1;

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

if (vertex[i] == src) {

srcIndex = i;

}

if (vertex[i] == dest) {

destIndex = i;

}

}

if (srcIndex == -1 || destIndex == -1) {

cout << "Dinh nguon hoac dinh dich khong ton tai trong do thi!" << endl;

return;

}

dist[srcIndex] = 0;

parent[srcIndex] = -1;

for (int count = 0; count < n - 1; count++) {

int u = minDistance(dist, sptSet, n);

sptSet[u] = true;

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

if (!sptSet[v] && a[u][v] && dist[u] != INT\_MAX && dist[u] + a[u][v] < dist[v]) {

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

parent[v] = u;

}

}

}

if (dist[destIndex] != INT\_MAX) {

cout << "Duong di ngan nhat tu dinh " << src << " den dinh " << dest << " la: " << src;;

printPath(parent, destIndex);

cout << " voi tong trong so la " << dist[destIndex] << endl;

} else {

cout << "Khong co duong di tu dinh " << src << " den dinh " << dest << endl;

}

}

Hàm xuất ra đường đi chi tiết và hàm tìm khoảng cách nhỏ nhất.

void printPath(int parent[], int j) {

if (parent[j] == -1)

return;

printPath(parent, parent[j]);

cout << " -> " << vertex[j];

}

int minDistance(int dist[], bool sptSet[], int n) {

int min = INT\_MAX, min\_index;

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

if (!sptSet[v] && dist[v] <= min) {

min = dist[v];

min\_index = v;

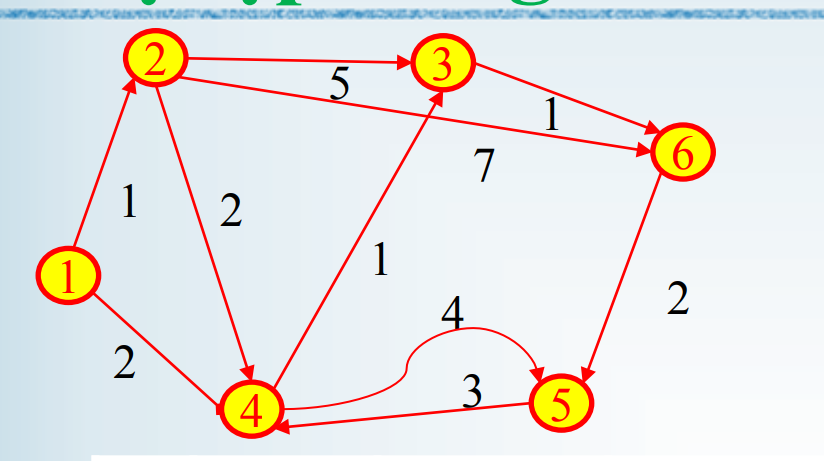
}

}

return min\_index;

}

Giả sử cho đồ thị như hình:

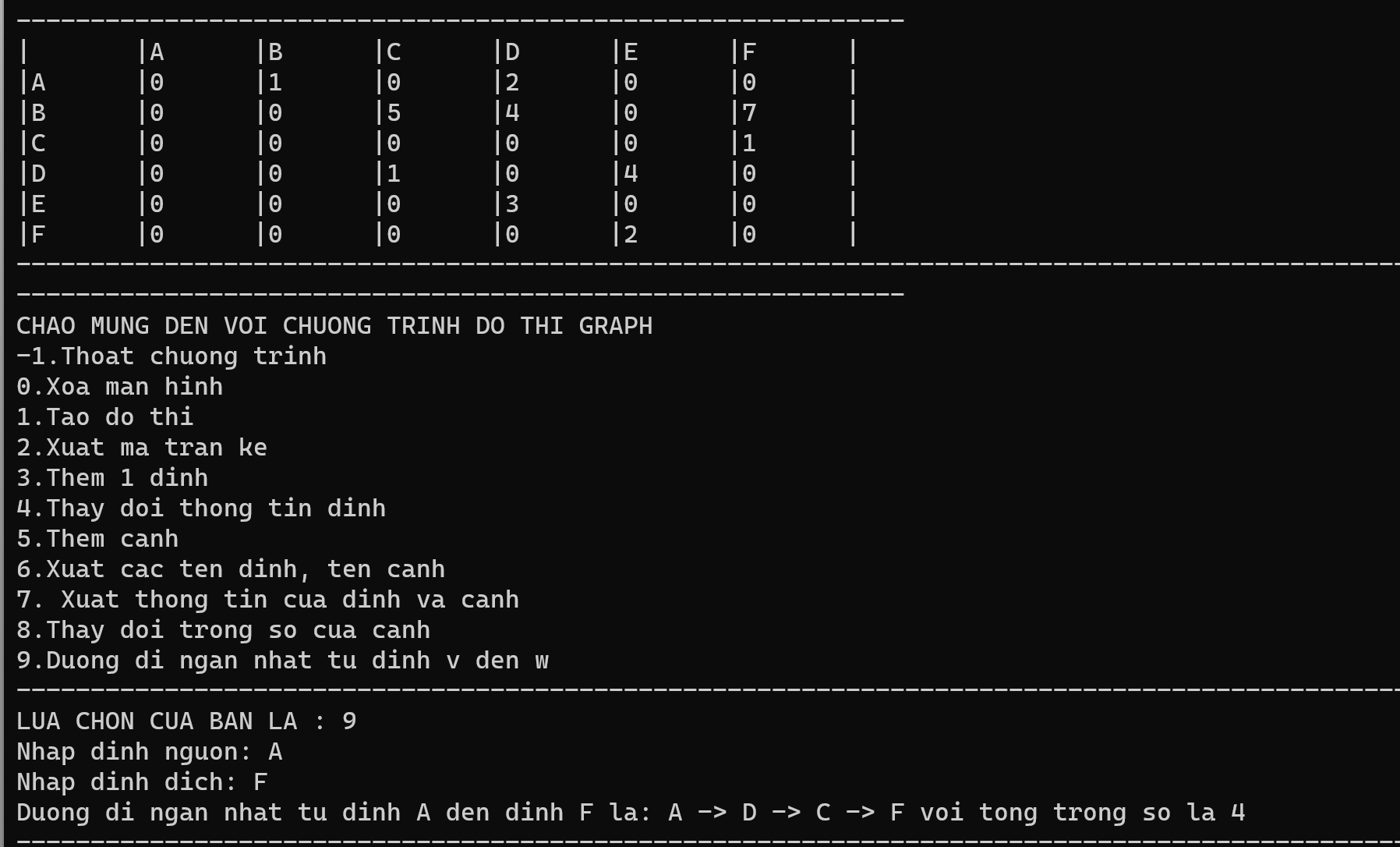


Khi nhập vào chương trình:

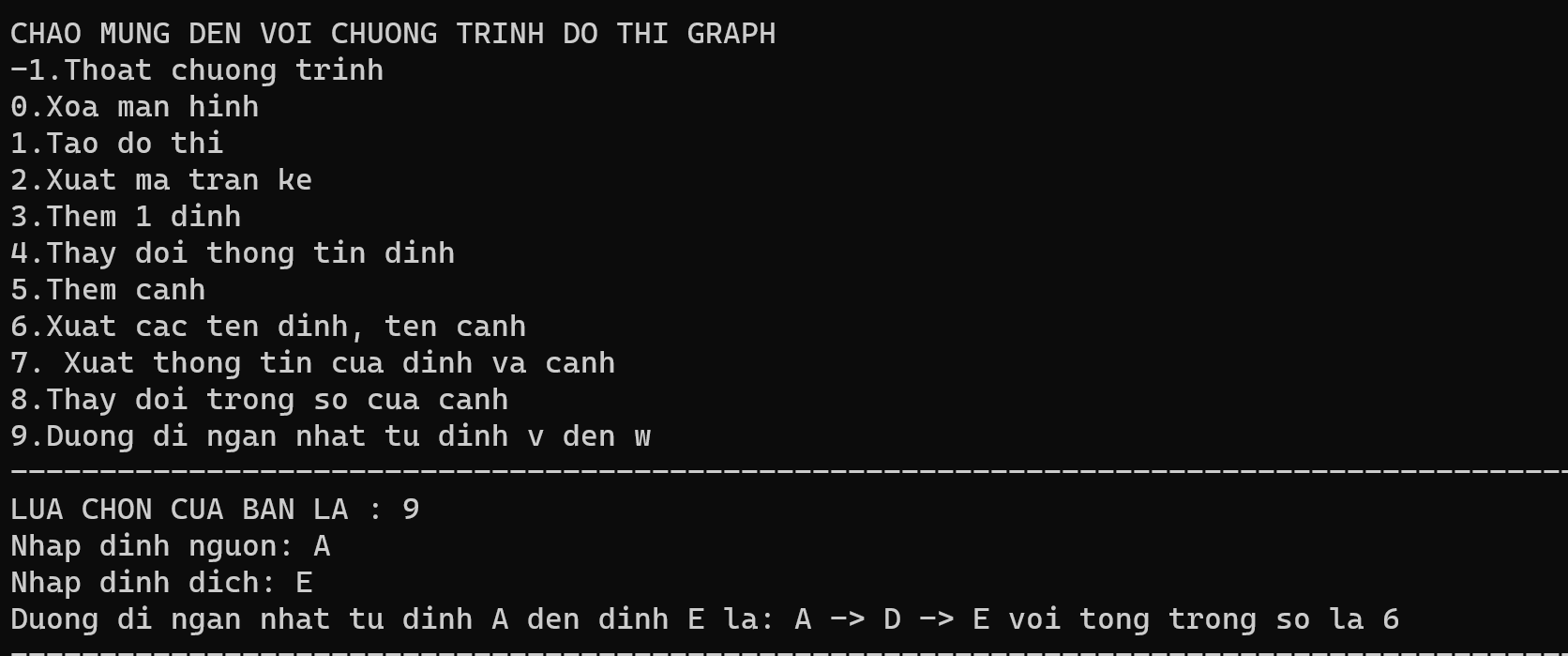
A black and white screen with white text

Description automatically generated

Chọn chức năng tìm đường đi ngắn nhất giữa 2 đỉnh được nhập(chức năng số 9). Có kết quả :



Tiếp tục thử với đỉnh nguồn là A và đỉnh đích là E:



Đáp án khi giải lập bảng tính nhãn bằng kết quả chạy chương trình:

A blue grid with numbers and lines

Description automatically generated

## **9.Kiểm tra chu trình Euler:**

Tạo hàm euler().

void euler(int a[][100], int n) {

int B[n];

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

B[i]=0;

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

B[i] +=a[i][j];

}

}

int kq = kteuler(B,n);

if(kq) {

cout << "Do thi co chu trinh euler\n";

timChuTrinhEuler(a,n);

}

else cout << "Do thi khong co chu trinh euler\n";

}

Hàm nhỏ để kiểm tra xem có euler.

int kteuler(int B[],int n) {

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

if(B[i]%2==1) return false;

}

return true;

}

Viết tiếp hàm timChuTrinhEuler() nếu đồ thị đã cho có chu trình Euler.

void timChuTrinhEuler(int a[][100], int n) {

int aCopy[100][100];

memcpy(aCopy, a, sizeof(aCopy));

char CE[MAX];

char stack[MAX];

int top = 1;

stack[top] = vertex[0]; // Initialize the stack with the starting vertex

int dCE = 0;

do {

char v = stack[top];

int x = 0;

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

if (vertex[i] == v) {

x = i;

break;

}

}

int hasNextEdge = 0;

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

if (aCopy[x][w] == 1) {

top++;

stack[top] = vertex[w];

aCopy[x][w] = 0;

aCopy[w][x] = 0;

hasNextEdge = 1;

break;

}

}

if (!hasNextEdge) {

dCE++;

CE[dCE] = v;

top--;

}

} while (top != 0);

cout << "Chu trinh euler la:\n";

for (int x = dCE; x > 1; x--) {

cout << CE[x] << " --> ";

}

cout << CE[1] << endl;

}

Khi đó, kết quả chạy:

Giả sử ta có đồ thị:

A diagram of a network

Description automatically generated

File liên kết kề sẽ được lưu như sau:

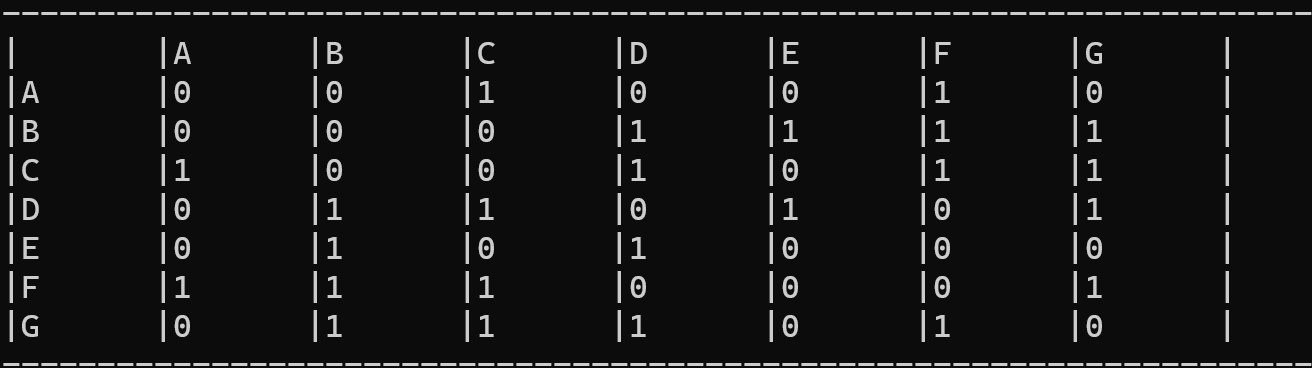
A screenshot of a computer

Description automatically generated

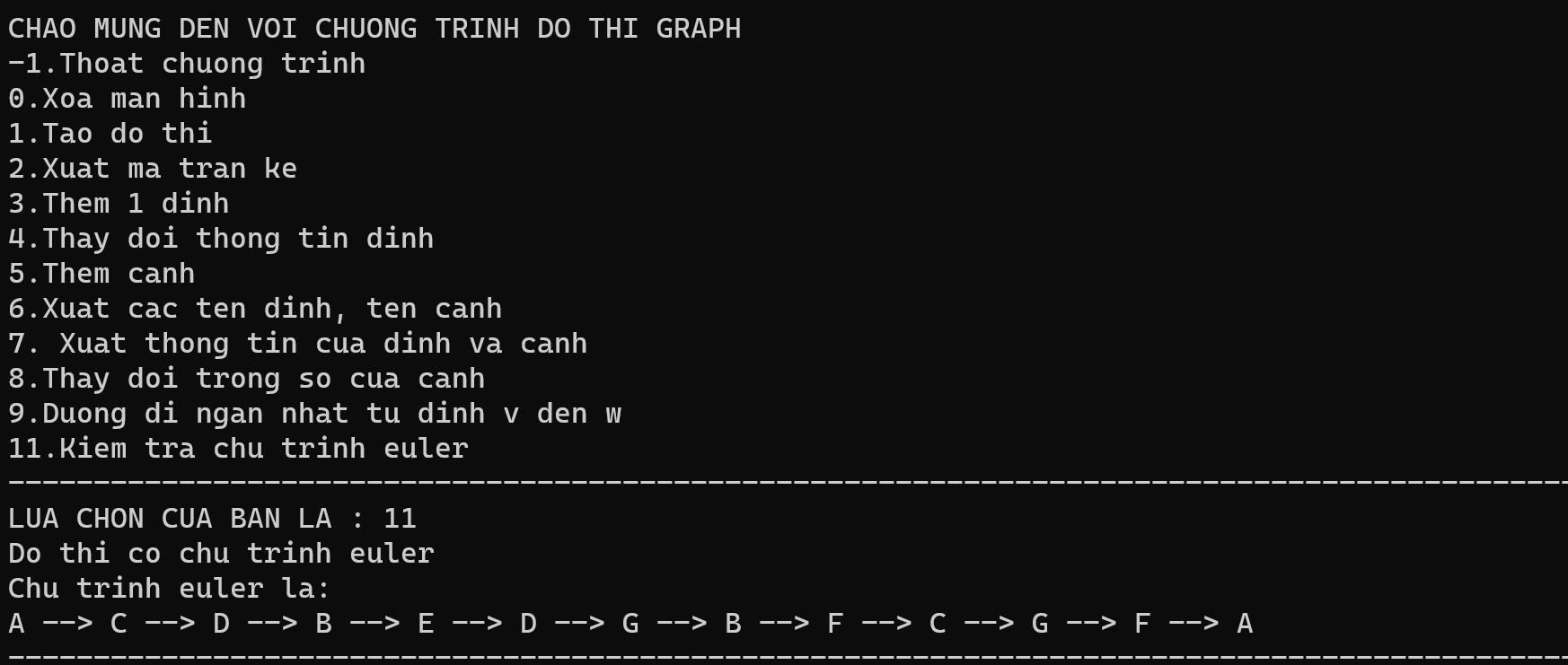
Đọc file từ chương trình:

A screenshot of a computer

Description automatically generated

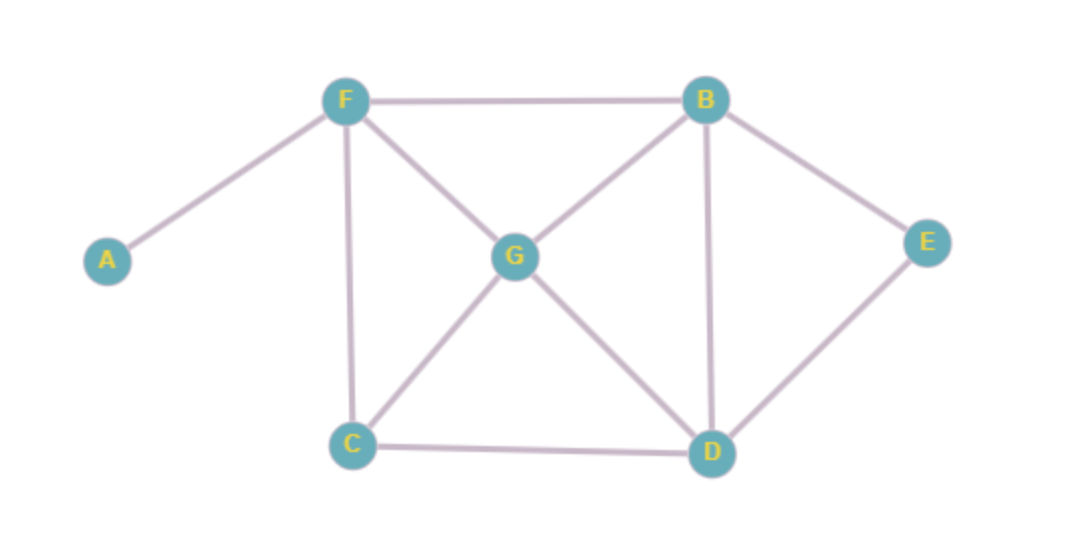


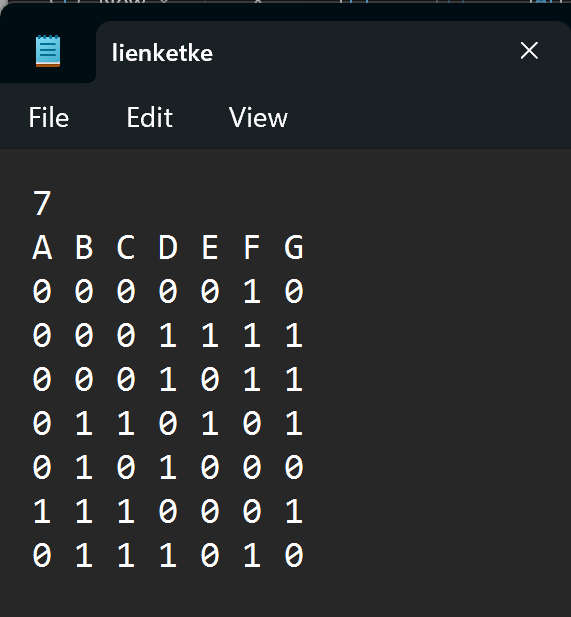
Chọn chức năng số 11, kiểm tra chu trình euler.



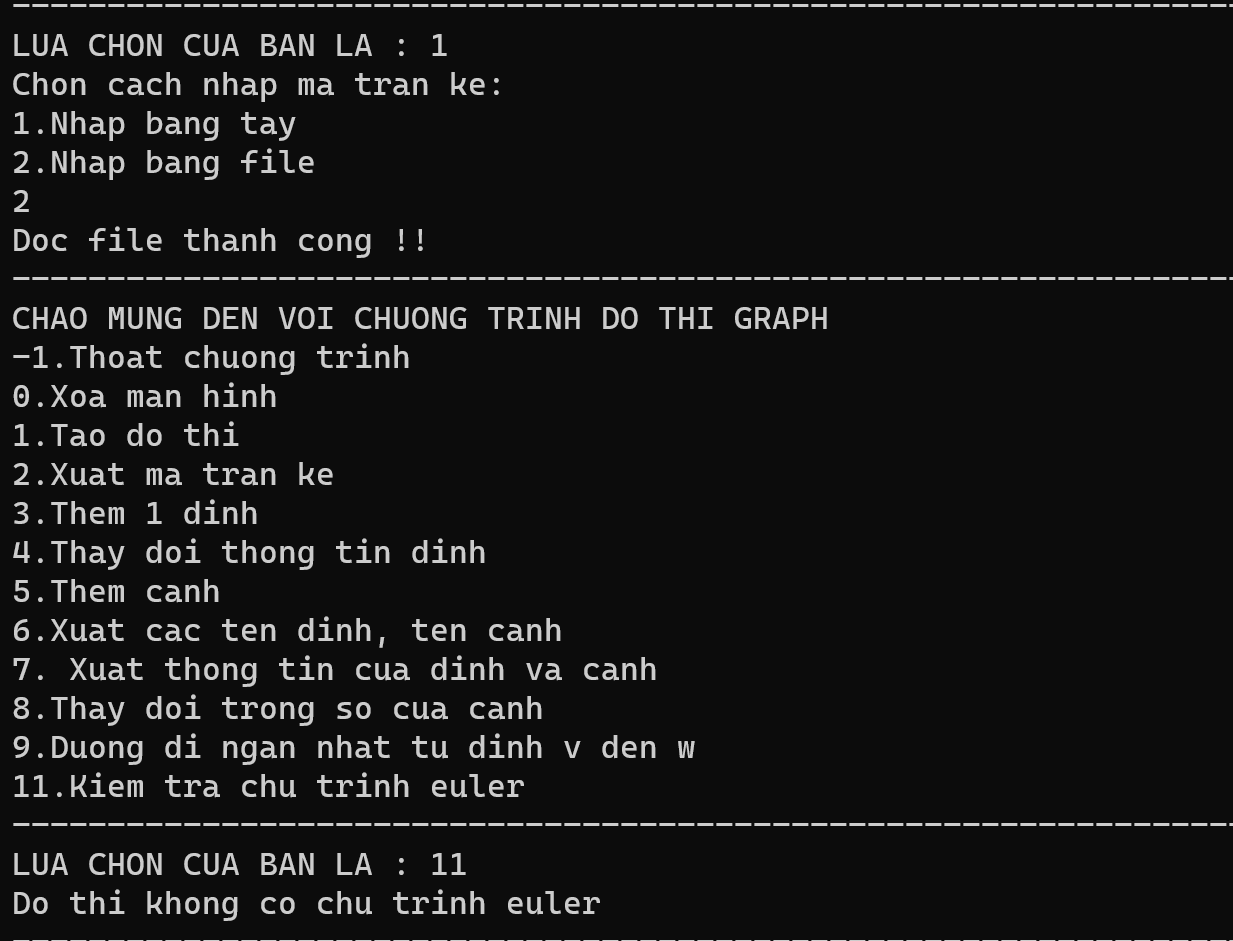
Kết quả sẽ hiện ra chu trình.

Thử lại với đồ thị không có chu trình euler:





Kết quả khi đã đọc file và chạy lại chức năng:



**10.Cây khung bé nhất:**

Tạo hàm primMST() được viết dựa trên thuật toán Prim.

void primMST(int a[][100], int n) {

int parent[100];

int key[100];

bool mstSet[100];

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

key[i] = MAX;

mstSet[i] = false;

}

key[0] = 0;

parent[0] = -1;

for (int count = 0; count < n - 1; count++) {

int u = minKey(key, mstSet, n);

mstSet[u] = true;

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

if (a[u][v] && !mstSet[v] && a[u][v] < key[v]) {

parent[v] = u;

key[v] = a[u][v];

}

}

}

printMST(a, parent, n);

}

Viết hàm minKey() tìm ra đỉnh có trọng số nhỏ nhất trong danh sách các đỉnh chưa được thêm vào cây khung bé nhât.

int minKey(int key[], bool mstSet[], int n) {

int min = MAX, minIndex;

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

if (!mstSet[v] && key[v] < min) {

min = key[v];

minIndex = v;

}

}

return minIndex;

}

Tạo hàm printMST() in ra cây khung bé nhất.

void printMST(int a[][100], int parent[], int n) {

cout << "Cay khung be nhat:\n";

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

cout << "Canh: " << vertex[parent[i]] << " - " << vertex[i] << " co trong so: " << a[i][parent[i]] << endl;

}

}

Tạo 1 đồ thị như hình vẽ để thử nghiệm:

A diagram of a network

Description automatically generated

Lưu vào file lienketke:s

A screenshot of a computer

Description automatically generated

Khi chạy lưu file và xem đồ thị trong chương trình:

A screenshot of a computer

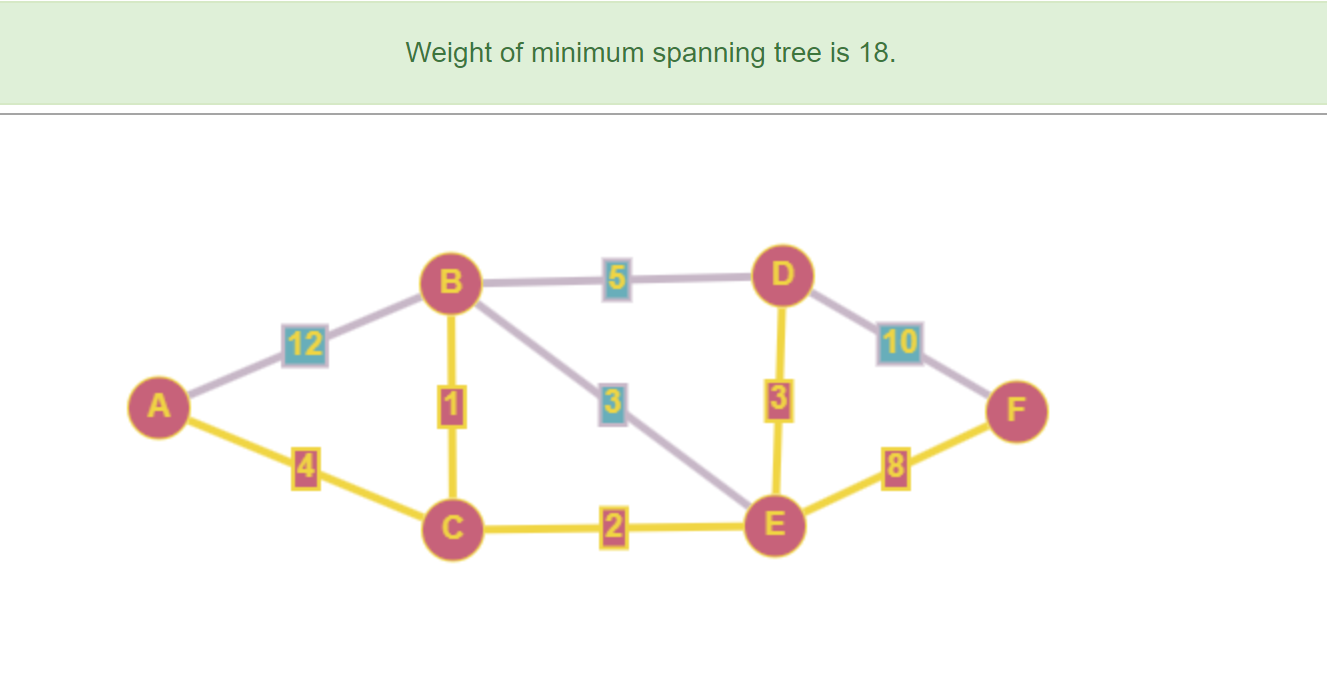
Description automatically generated

Chọn số 13, tìm cây khung bé nhất:

A screenshot of a computer

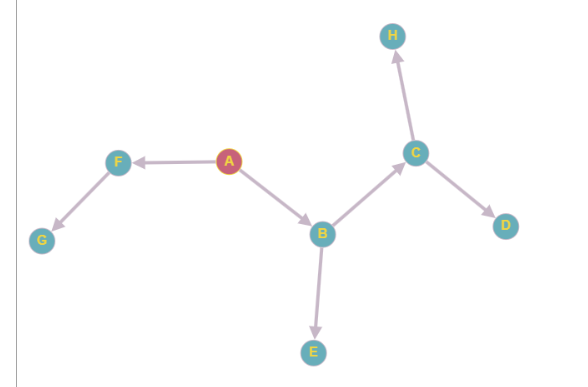
Description automatically generated

Kết quả ra cây khung bé nhất và tổng trọng số là 18. Kết quả chính xác theo trang web <https://graphonline.ru/en/>

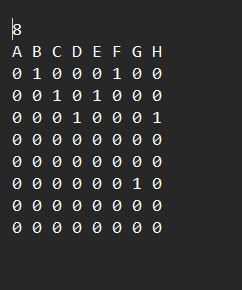


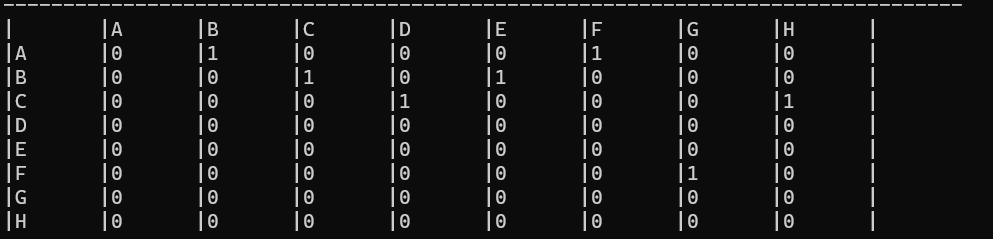
## **11.Duyệt đồ thị theo chiều rộng:**

Ta có đồ thị sau:

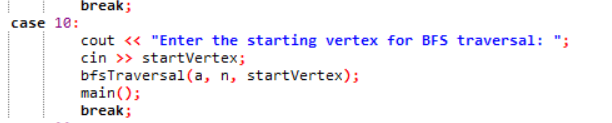


Nhập đồ thị bằng file:





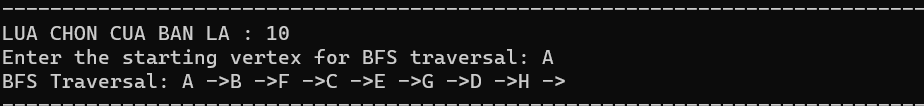
Tạo chức năng số 10: Duyệt đồ thị theo chiều rộng



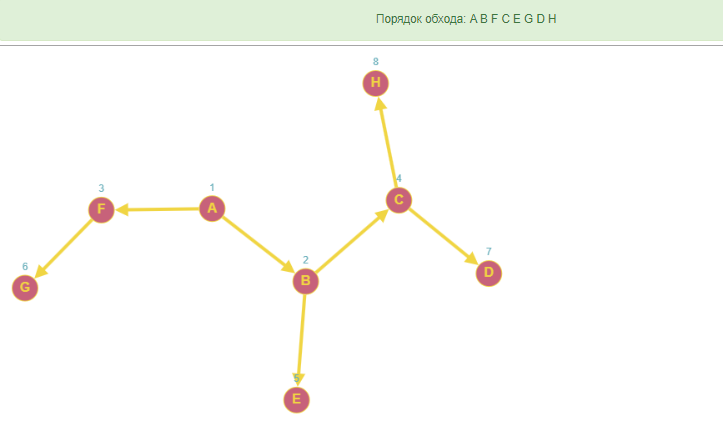
Tạo hàm “bfsTraversal()”

|  |
| --- |
| void bfsTraversal(int a[][100], int n, char startVertex)  {  int vertexIndex = -1;  for (int i = 0; i < n; i++) {  if (vertex[i] == startVertex) {  vertexIndex = i;  break;  }  }  bool visited[100] = {false};  queue<int> q;  cout << "BFS Traversal: ";  visited[vertexIndex] = true;  q.push(vertexIndex);  while (!q.empty())  {  int v = q.front();  q.pop();  cout << vertex[v] << " ->";  for (int i = 0; i < n; i++)  {  if (a[v][i] != 0 && !visited[i])  {  visited[i] = true;  q.push(i);  }  }  }  cout << endl;  } |

Kết quả khi chạy chương trình là:

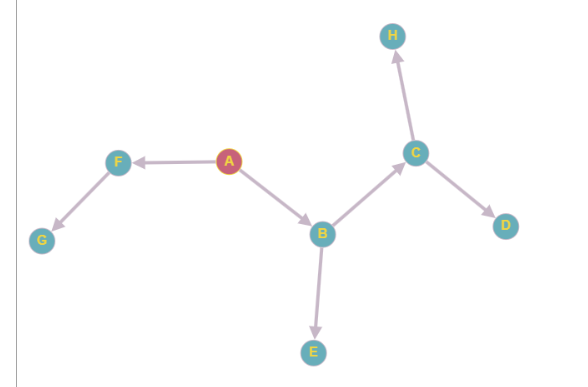


Trùng khớp so với thuật toán bfs tại trang graphonline.ru

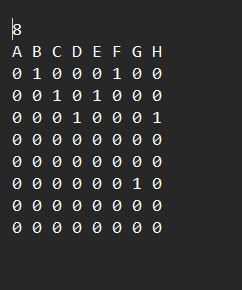


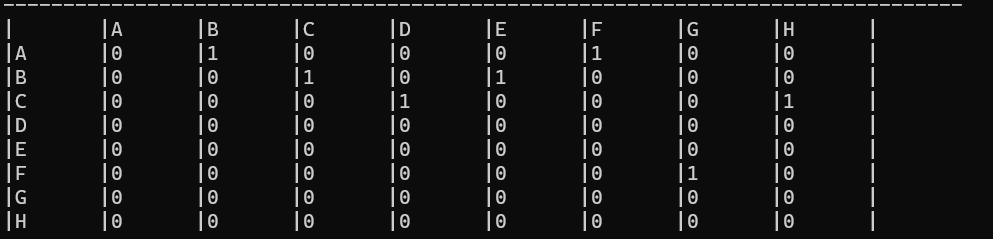
## **12.Duyệt đồ thị theo chiều sâu:**

Ta có đồ thị sau:

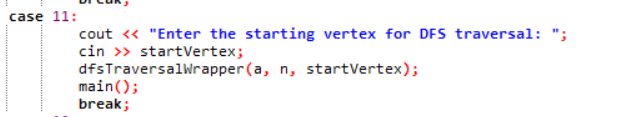


Nhập đồ thị bằng file:





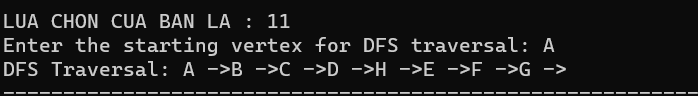
Tạo chức năng số 11: Duyệt đồ thị theo chiều sâu



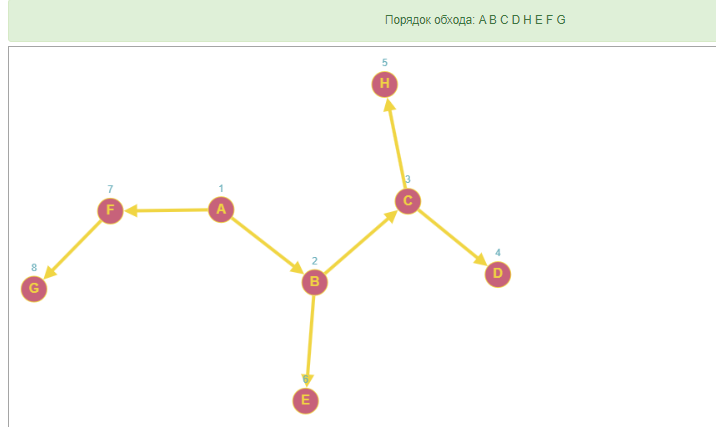
Tạo hàm “dfsTraversal()”

|  |
| --- |
| void dfsTraversal(int a[][100], int n, int v, bool visited[])  {  visited[v] = true;  cout << vertex[v] << " ->";  for (int i = 0; i < n; i++)  {  if (a[v][i] != 0 && !visited[i])  {  dfsTraversal(a, n, i, visited);  }  }  }  void dfsTraversalWrapper(int a[][100], int n, char startVertex)  {  int vertexIndex = -1;  for (int i = 0; i < n; i++) {  if (vertex[i] == startVertex) {  vertexIndex = i;  break;  }  }  bool visited[100] = {false};  cout << "DFS Traversal: ";  dfsTraversal(a, n, vertexIndex, visited);  cout << endl;  } |

Kết quả khi chạy chương trình là:



Trùng khớp so với thuật toán bfs tại trang graphonline.ru



# **PHẦN III: TỔNG KẾT**

## **1. Đánh giá Sinh viên:**

* Hầu hết các yêu cầu đã được đáp ứng.
* Giao diện người dùng được thiết kế đơn giản để sử dụng dễ dàng.
* Mã nguồn khá gọn và có thể tái sử dụng.
* Ứng dụng không hỗ trợ nhập hình ảnh vào đồ thị.

## **2. Khó khăn:**

* Học công nghệ mới là một thách thức, làm chậm tiến độ dự án.
* Lập trình đa luồng là một khía cạnh khó khăn do kiến thức và kinh nghiệm còn hạn chế.

## **3. Ưu điểm:**

* Mã nguồn có cấu trúc gọn gàng.
* Đáp ứng đủ yêu cầu của đề tài.
* Giao diện người dùng đơn giản, dễ sử dụng cho ứng dụng của đồ thị.
* Tính tái sử dụng, và khả năng bảo trì cao.

## **4. Nhược điểm:**

* Các luồng trong ứng dụng hoạt động nhưng có thể gây ra trải nghiệm người dùng không hoàn hảo.
* Ứng dụng còn xảy ra lỗi khi người dùng nhập đầu vào sai.
* Không hỗ trợ nhập hình ảnh vào đồ thị.

## **5. Ý tưởng phát triển:**

Thay vì chỉ hỗ trợ các tính năng cơ bản của graph, có thể cải thiện chúng để trở nên chuyên sâu và chuyên nghiệp hơn, giống như các phần mềm đồ thị chuyên nghiệp khác dùng thư viện Winbgim (graphics.h).