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

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**🕯✡🕮🕮✡🕯**



**BÁO CÁO ĐỒ ÁN CÔNG NGHỆ THÔNG TIN**

**TUẦN 5**

***ĐỀ TÀI:* GRAPH**

**GVHD**: ThS. Nguyễn Quang Ngọc

**Sinh viên thực hiện**:

|  |  |
| --- | --- |
| **HỌ VÀ TÊN** | **MÃ SỐ SINH VIÊN** |
| Nguyễn Minh Quang | 20143481 |
| Đỗ Minh Cường | 21110147 |

## Tiến độ nhiệm vụ

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | Đường đi ngắn nhất từ đỉnh v đến đỉnh w |  |  |  |  |  | o |  |  |  |  |  |  |  |  |  |
| 11 | Kiểm tra có chu trình Euler |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 | Cây khung bé nhất |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Day | | 03/09/2023 | 10/09/2023 | 17/09/2023 | 24/09/2023 | 24/09/2023 |  |  |  |  |  |  |  |  |  |  |
| Week | | 01 | 02 | 03 | 04 | 05 |  |  |  |  |  |  |  |  |  |  |
| Note | |  |  |  |  |  |  |  |  |  | o – Begin  o – Complete 50%  o – Complete 100% | | | | | |

## Định nghĩa về đường đi ngắn nhất từ đỉnh d tới đỉnh w:

Có 2 thuật toán để xác định đường đi ngắn nhất từ đỉnh d tới đỉnh w:

1.Thuật toán Dijkstra:

Có thể giải quyết bài toán tìm đường đi ngắn nhất trên đồ thị vô hướng lẫn có hướng miễn là trọng số **không âm**.

Ý tưởng cơ bản của thuật toán như sau:

* Bước 1: Từ đỉnh gốc, khởi tạo khoảng cách tới chính nó là 00, khởi tạo khoảng cách nhỏ nhất ban đầu tới các đỉnh khác là +∞+∞. Ta được danh sách các khoảng cách tới các đỉnh.
* Bước 2: Chọn đỉnh a có khoảng cách nhỏ nhất trong danh sách này và ghi nhận. Các lần sau sẽ không xét tới đỉnh này nữa.
* Bước 3: Lần lượt xét các đỉnh kề b của đỉnh a. Nếu khoảng cách từ đỉnh gốc tới đỉnh b nhỏ hơn khoảng cách hiện tại đang được ghi nhận thì cập nhật giá trị và đỉnh kề a vào khoảng cách hiện tại của b.
* Bước 4: Sau khi xét tất cả đỉnh kề b của đỉnh a. Lúc này ta được danh sách khoảng cách tới các điểm đã được cập nhật. Quay lại Bước 2 với danh sách này. Thuật toán kết thúc khi chọn được khoảng cách nhỏ nhất từ tất cả các điểm.

1. Thuật toán Bellman Ford:

Thuật toán Bellman-Ford là một thuật toán sử dụng để tìm đường đi ngắn nhất trong đồ thị có hướng và có thể chứa các cạnh có trọng số âm. Thuật toán này được đặt theo tên của nhà toán học Richard Bellman và Lester Ford, Jr., người đã độc lập phát triển thuật toán này vào những năm 1950.

Dưới đây là mô tả cơ bản về thuật toán Bellman-Ford:

1. Khởi tạo một mảng kết quả (result) và đặt giá trị của tất cả các đỉnh trong đồ thị là vô cùng (infinity), ngoại trừ đỉnh nguồn (source) được đặt là 0.
2. Lặp lại các bước sau cho V-1 lần, trong đó V là số lượng đỉnh trong đồ thị:
3. a. Duyệt qua tất cả các cạnh trong đồ thị. Với mỗi cạnh (u, v) có trọng số w, kiểm tra nếu result[u] + w nhỏ hơn result[v], thì cập nhật result[v] thành result[u] + w.
4. Kiểm tra xem có chu trình âm trong đồ thị hay không. Để làm điều này, lặp lại qua tất cả các cạnh trong đồ thị và kiểm tra nếu result[u] + w nhỏ hơn result[v], trong đó (u, v) là một cạnh có trọng số w, thì đồ thị chứa chu trình âm.
5. Nếu không có chu trình âm, thuật toán đã hoàn thành và kết quả là mảng result chứa các giá trị ngắn nhất từ đỉnh nguồn tới mọi đỉnh khác trong đồ thị.

Thuật toán Bellman-Ford có độ phức tạp thời gian là O(V\*E), trong đó V là số lượng đỉnh và E là số lượng cạnh trong đồ thị. Nó cũng có thể xử lý đồ thị chứa chu trình âm, nhưng chỉ khi không có chu trình âm từ đỉnh nguồn đến bất kỳ đỉnh nào khác.

## So sánh các thuật toán:

|  |  |  |
| --- | --- | --- |
|  | **Thuật toán Dijikstra** | **Thuật toán Bellman-Ford** |
| **Mục đích** | Tìm đường đi ngắn nhất từ đỉnh nguồn đến các đỉnh còn lại | Tìm đường đi ngắn nhất từ đỉnh nguồn đến các đỉnh còn lại |
| **Thời gian code** | Lâu | Nhanh |
| **Tìm được chu trình âm** | Không | Có |
| **Độ phức tập thời gian** | O(m x logn) | O(m x n) |

## Thiết kế

1. **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;

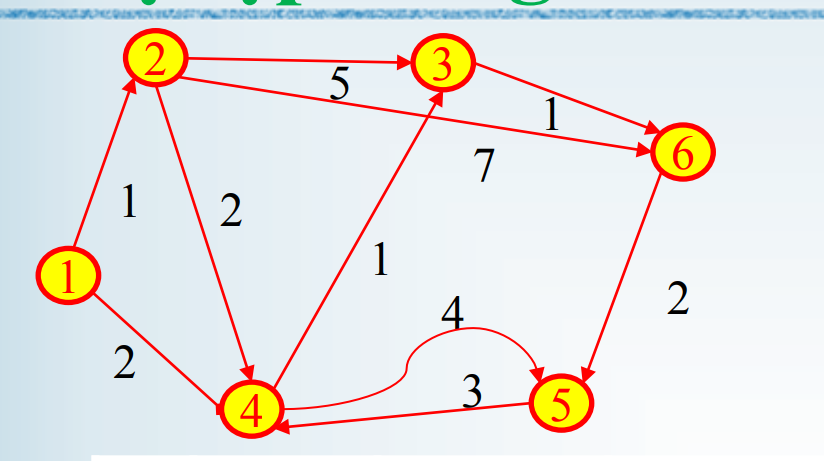
}

}

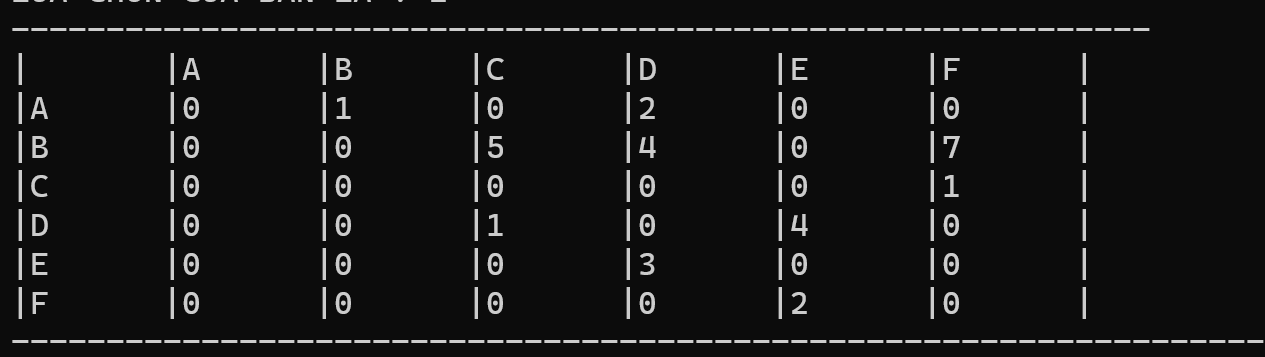
return min\_index;

}

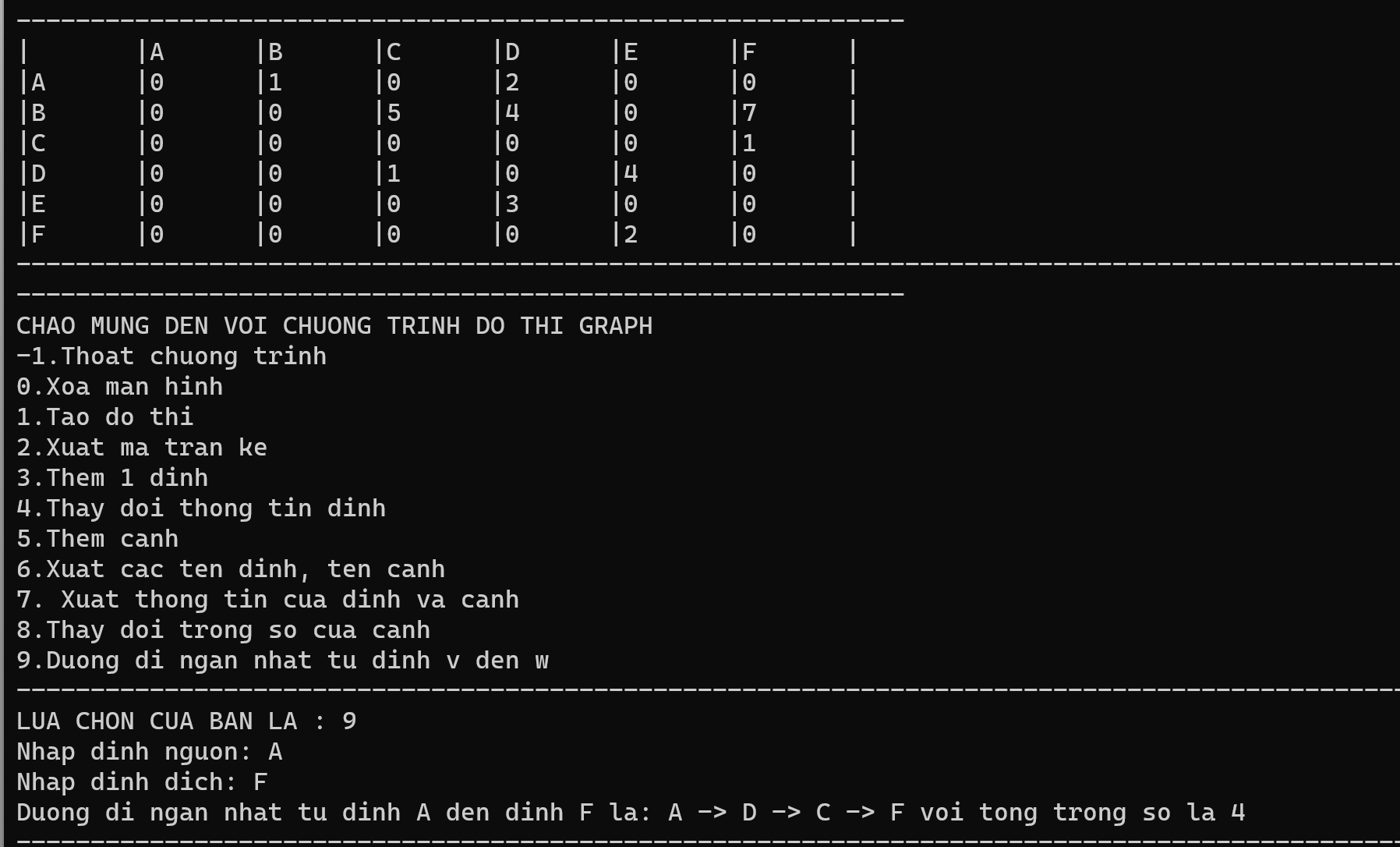
Giả sử cho đồ thị như hình:



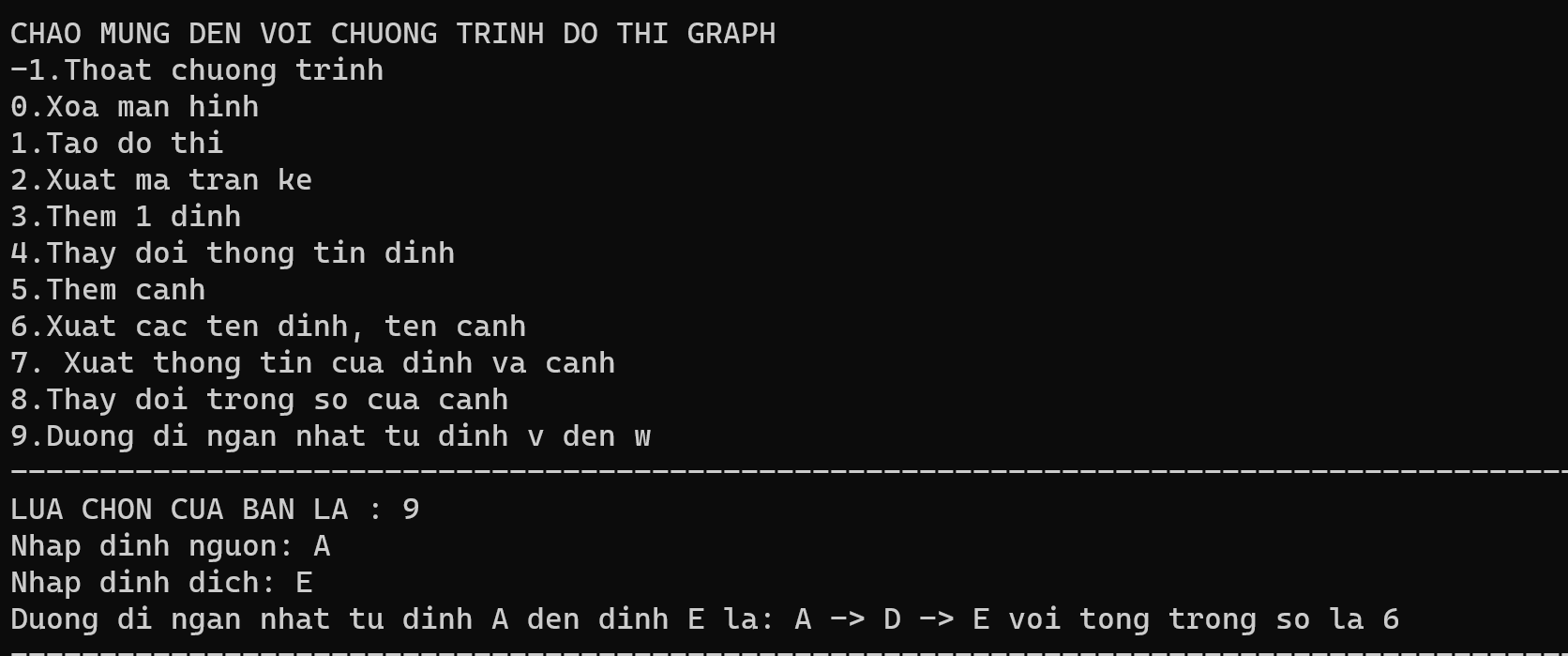
Khi nhập vào chương trình:



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:

