**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 8**

***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 |  |  |  |  |  |  | o |  |  |  |  |  |  |  |  |
| 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 | 1/10/2023 |  |  |  |  |  |  |  |  |  |
| Week | | 01 | 02 | 03 | 04 | 05 | 06 |  |  |  |  |  |  |  |  |  |
| Note | |  |  |  |  |  |  |  |  |  | o – Begin  o – Complete 50%  o – Complete 100% | | | | | |

## Định nghĩa về duyệt đồ thị theo chiều rộng, chiều xâu

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

1. **Duyệt Đồ Thị Theo Chiều Rộng (BFS):**
   * BFS bắt đầu từ một đỉnh gốc và truy cập tất cả các đỉnh kề với đỉnh gốc trước.
   * Sau khi duyệt hết tất cả các đỉnh kề, BFS tiếp tục đến các đỉnh kề của các đỉnh vừa được truy cập và tiếp tục quá trình này cho đến khi tất cả các đỉnh đã được duyệt.
   * BFS sử dụng hàng đợi (queue) để lưu trữ các đỉnh cần duyệt. Đỉnh gốc được đưa vào hàng đợi, sau đó các đỉnh kề với đỉnh gốc được thêm vào hàng đợi, và quá trình này diễn ra cho đến khi hàng đợi trống.

Ưu điểm của BFS:

* + BFS cho phép tìm kiếm đường đi ngắn nhất từ một đỉnh đến một đỉnh khác trong đồ thị không có trọng số (hoặc trọng số đồng nhất).

Một số đặc điểm quan trọng về BFS:

Hàng Đợi (Queue): BFS sử dụng hàng đợi để theo dõi các đỉnh cần duyệt. Đỉnh gốc được đưa vào hàng đợi, sau đó các đỉnh kề với đỉnh gốc được thêm vào hàng đợi. Đỉnh nào được thêm vào hàng đợi trước đó sẽ được thăm trước.

Duyệt Theo Tầng: Khi thực hiện BFS, các đỉnh được thăm theo từng "tầng" của đồ thị. Tầng 0 bao gồm đỉnh gốc, tầng 1 bao gồm các đỉnh kề với đỉnh gốc, tầng 2 bao gồm các đỉnh kề với các đỉnh tầng 1, và cứ tiếp tục như vậy cho đến khi không còn đỉnh nào để thăm hoặc không có đỉnh mới để thêm vào hàng đợi.

Tìm Kiếm Đường Đi Ngắn Nhất (Nếu Có): BFS thường được sử dụng để tìm kiếm đường đi ngắn nhất giữa hai đỉnh trong đồ thị không có trọng số hoặc có trọng số đồng nhất. Vì BFS thăm các đỉnh theo từng tầng, nếu có đường đi từ đỉnh gốc đến đích, thì đường đi đó sẽ được tìm thấy trong số các đường đi có thể.

Tránh Lặp Lại: Để tránh việc thăm lại các đỉnh đã được thăm, một danh sách (hoặc tập hợp) các đỉnh đã thăm thường được sử dụng. Trước khi thêm một đỉnh vào hàng đợi, bạn nên kiểm tra xem đỉnh đó đã được thăm chưa.

Độ Phức Tạp: Độ phức tạp thời gian của BFS là O(V + E), trong đó V là số đỉnh và E là số cạnh trong đồ thị. Điều này bởi vì BFS thăm mỗi đỉnh và cạnh đúng một lần.

1. **Duyệt Đồ Thị Theo Chiều Sâu (DFS):**
   * DFS bắt đầu từ một đỉnh gốc và tiếp tục duyệt qua các đỉnh kề của đỉnh gốc trước khi tiếp tục vào các đỉnh kề của các đỉnh vừa được duyệt.
   * Quá trình này được thực hiện đệ quy hoặc thông qua việc sử dụng ngăn xếp (stack) để lưu trữ các đỉnh cần duyệt.

Ưu điểm của DFS:

* + DFS có khả năng tìm kiếm theo chiều sâu vào đồ thị và thường được sử dụng trong việc kiểm tra tính liên thông của đồ thị hoặc tìm kiếm các đường đi trong đồ thị.

Một số đặc điểm quan trọng về DFS:

Đệ Quy Hoặc Sử Dụng Stack: DFS thường được triển khai bằng cách sử dụng đệ quy hoặc stack. Trong cả hai trường hợp, bạn theo dõi các đỉnh cần duyệt và đẩy chúng vào stack để xử lý sau.

Duyệt Sâu Trước: DFS sẽ tiếp tục đi "sâu" vào đồ thị trước khi quay lại các nhánh khác. Điều này có nghĩa rằng nó thường duyệt xong một nhánh trước khi qua nhánh khác. Kết quả là, nó có thể không tìm kiếm đường đi ngắn nhất như BFS.

Tránh Lặp Lại: Để tránh việc thăm lại các đỉnh đã được thăm, một danh sách (hoặc tập hợp) các đỉnh đã thăm thường được sử dụng. Trước khi thêm một đỉnh vào stack hoặc tiếp tục đi sâu vào một đỉnh mới, bạn nên kiểm tra xem đỉnh đó đã được thăm chưa.

Sử Dụng Đệ Quy: DFS thường được triển khai bằng cách sử dụng đệ quy. Mỗi khi bạn thăm một đỉnh kề, bạn gọi lại hàm DFS với đỉnh đó làm đỉnh gốc. Điều này tạo ra một chu trình đệ quy và giúp duyệt sâu vào đồ thị.

Độ Phức Tạp: Độ phức tạp thời gian của DFS là O(V + E), tương tự như BFS, với V là số đỉnh và E là số cạnh trong đồ thị.

## Thiết kế

1. **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] << " --> ";

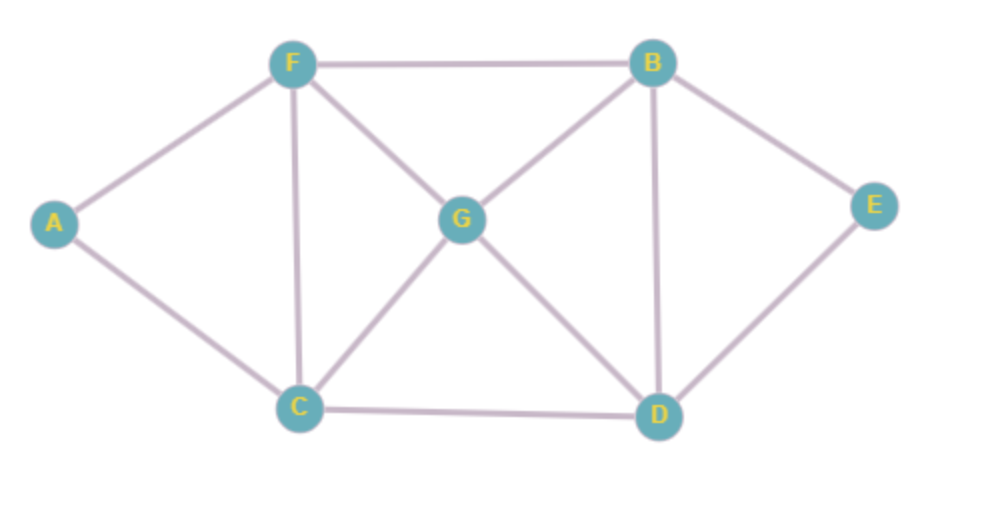
}

cout << CE[1] << endl;

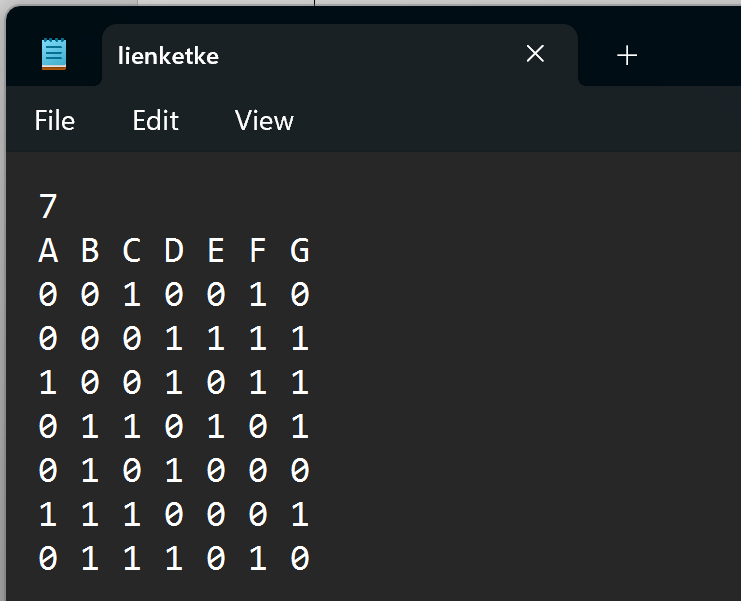
}

Khi đó, kết quả chạy:

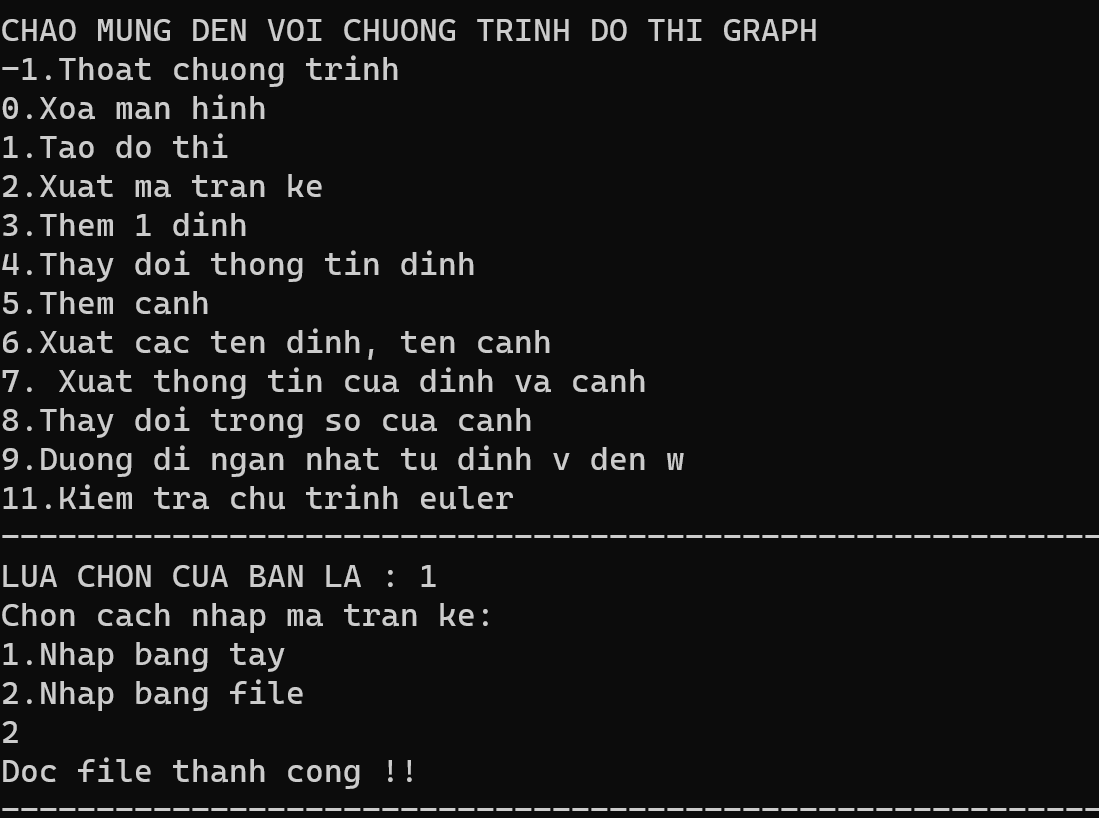
Giả sử ta có đồ thị:

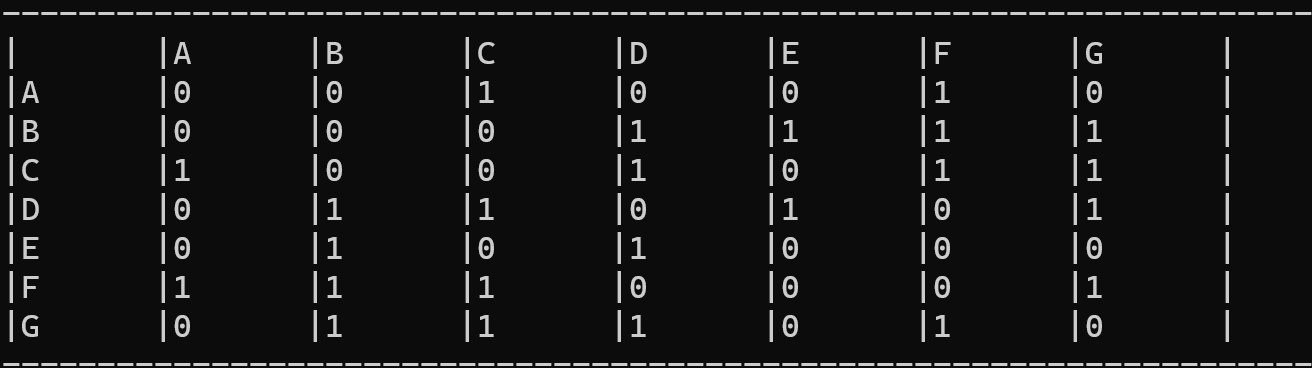


File liên kết kề sẽ được lưu như sau:

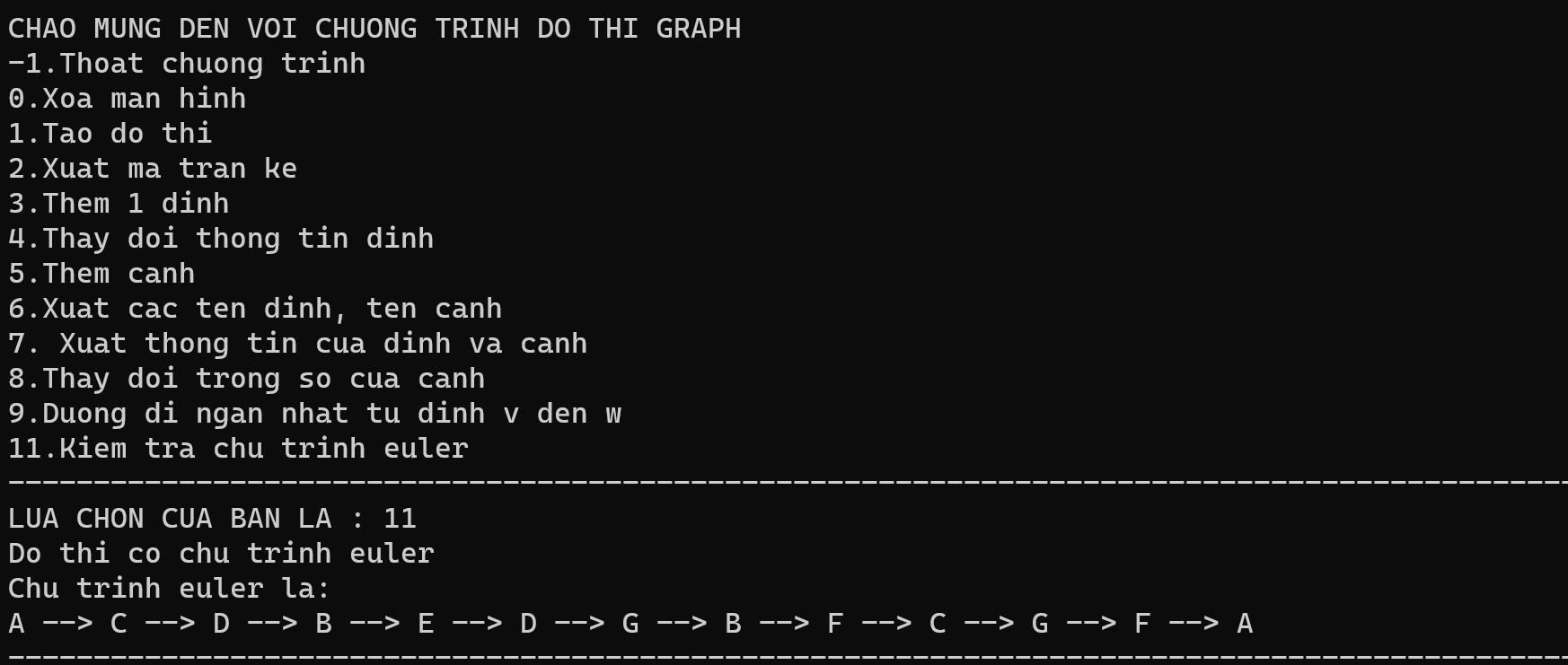


Đọc file từ chương trình:



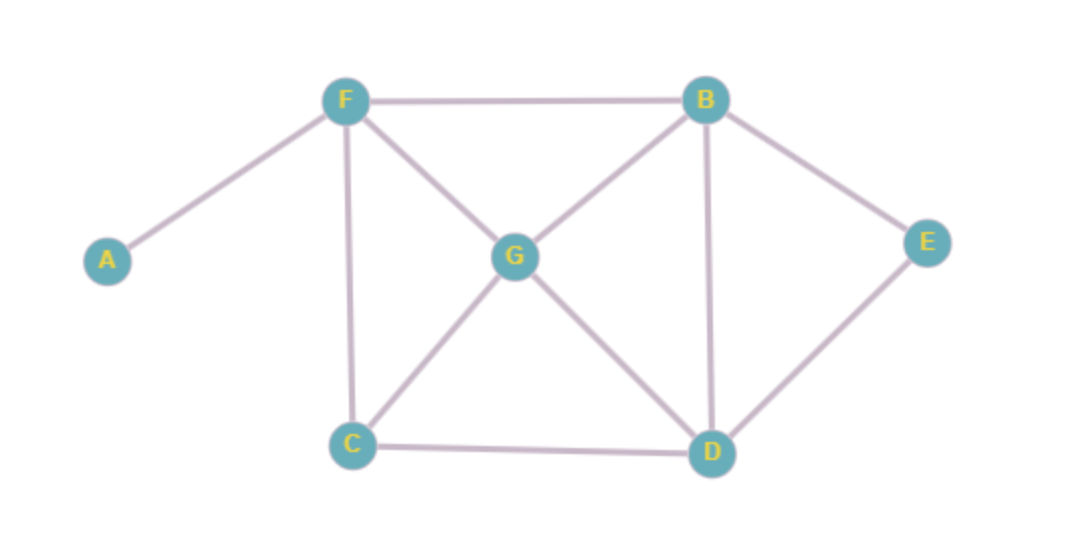


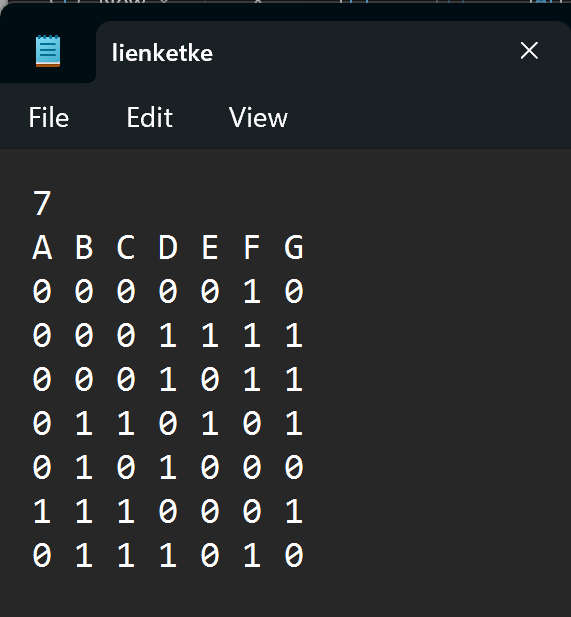
Chọn chức năng số 11, kiểm tra chu trình euler.



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:

