**Báo cáo Đồ Án môn học Lý thuyết Đồ thị**

**Nhóm:** P16 – Lynx

**Thành viên 1:** Nguyễn Thanh Trúc Đào **MSSV:** 23880217

**Thành viên 2:** Nguyễn Nhất Anh Hy **MSSV:** 23880238

**Thành viên 3:** Âu Thành Luân **MSSV:** 23880250

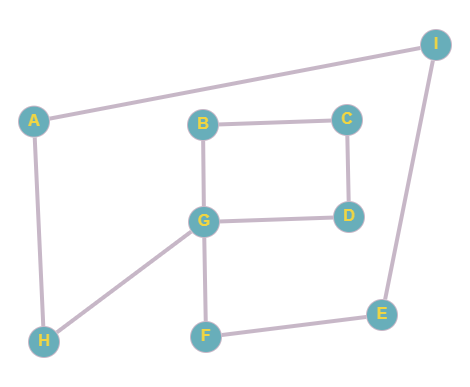
**Thành viên 4:** Trần Nguyễn Tuấn Vỹ **MSSV:** 23880291

**Nội dung báo cáo**

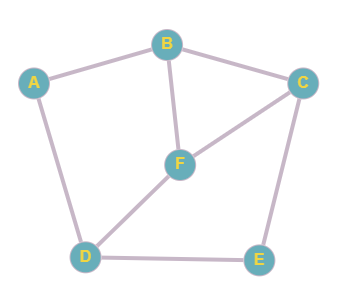
**Phần 1: Bài tập Lý thuyết**

**Câu 1:**

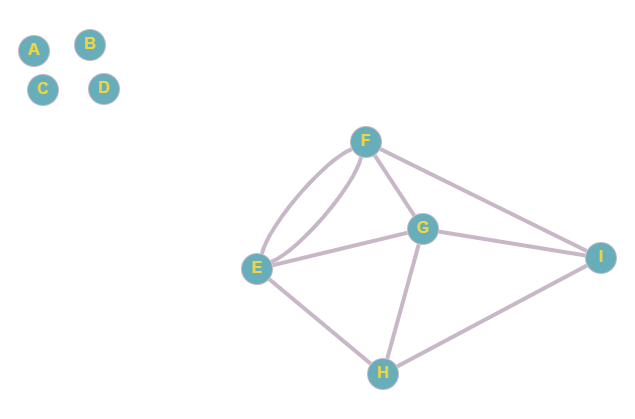
a)



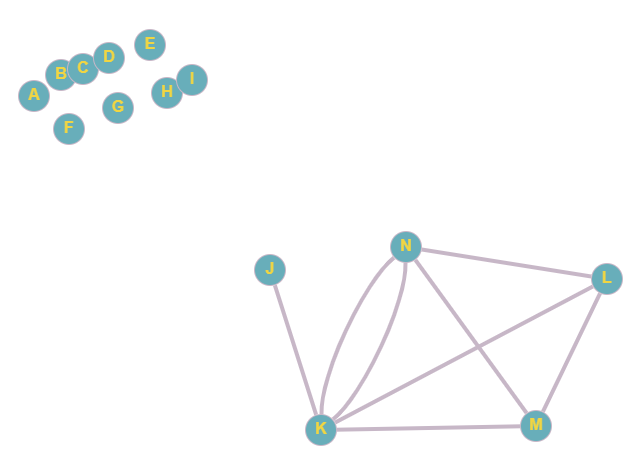
b)



c)



d)



**Câu 2:**

Các tệp tin kết quả: Graph\_1a.txt, Graph\_1b.txt, Graph\_1c.txt, Graph\_1d.txt nằm trong folder **DOC**.

**Phần 2: Hướng dẫn sử dụng chương trình cài đặt**

***1. Mô tả các hàm chính của chương trình:***

Chương trình gồm 4 lớp chính:

* **List:** cho phép thao tác với danh sách liên kết (DSLK) để tạo ra danh sách kề tương ứng với input đầu vào. Đồng thời, lớp List cũng chứa thông số về bậc của mỗi đỉnh giúp việc kiểm tra các thuật toán (có liên quan đến bậc của 1 đỉnh) dễ dàng hơn mà không cần duyệt lại toàn bộ DSLK.
* **AdjacencyList:** lưu thông tin về toàn bộ đồ thị bao gồm số đỉnh của đồ thị và dữ liệu là mảng với mỗi thành viên là lớp List của mỗi đỉnh.
* **Euler:** dùng để kiểm tra một đồ thị có là đồ thị Euler/ nửa Euler hay không và in ra đường đi/ chu trình tương ứng.
* **Hamilton:** dùng để kiểm tra một đồ thị có là đồ thị Hamilton / nửa Hamilton hay không và in ra đường đi/ chu trình tương ứng.

Các hàm – thuật toán chính:

* **DFS\_visit (DFS – Deep First Search):** dùng để kiểm tra một đồ thị có liên thông hay không. Tuy nhiên, hàm isConnected() để kiểm tra tính “Liên thông” của mỗi lớp Euler và Hamilton dùng trong đồ án có đôi chút khác biệt, cụ thể:
  + Một đồ thị có đỉnh cô lập vẫn có thể được xem là “liên thông” trong đồ thị Euler/nửa Euler (nếu các đỉnh không cô lập còn lại liên thông hay nói cách khác là chúng chỉ có 1 thành phần liên thông)
  + Một đồ thị có đỉnh cô lập không được xem là “liên thông” trong đồ thị Hamilton / nửa Hamilton (do không thể có đường đi đến các đỉnh cô lập)
* **Euler\_Print** và **Euler\_Rec** (**Thuật toán Fleury** – dùng để in đường đi / chu trình Euler): gồm các bước:
  + Kiểm tra đồ thị có là là đồ thị Euler/ nửa Euler hay không:
    - Kiểm tra đồ thị có liên thông hay không.
    - Nếu đồ thị liên thông thì kiểm tra đồ thị có 0 hoặc 2 đỉnh lẻ và trong trường hợp có 2 đỉnh lẻ (có đường đi Euler), đỉnh bắt đầu có phải là đỉnh lẻ hay không.
  + Nếu đồ thị đã là đồ thị Euler/ nửa Euler: Chạy thuật toán ở đỉnh bắt đầu (u):
    - **B1:** Xét các cạnh liền kề u: ưu tiên chọn cạnh {u, v} (v là đỉnh kề với u) không là cạnh cầu (Bridge)
    - **B2:** In ra cạnh {u, v}. Xóa cạnh {u, v} trong DSLK (trong DSLK của u và v sau khi xóa, vị trí của u và v sẽ trở thành -1).
    - **B3:** Chạy thuật toán với đỉnh v.
    - Thực hiện cho đến khi không còn cạnh nào trong đồ thị.
    - **Lưu ý:** Kiểm tra một cạnh {u, v} có là cạnh cầu hay không bằng cách (Hàm **isBridge**): Kiểm tra số đỉnh liên thông với u (dùng DFS), sau đó xóa cạnh {u, v} và kiểm tra lại số cạnh liên thông với u. Nếu số cạnh sau khi xóa cạnh {u, v} nhỏ hơn thì cạnh {u, v} là cạnh cầu.
* **Hamilton\_run** và **Hamilton\_Rec** (**Backtracking** – dùng để kiểm tra/ vét cạn các đường đi trong đồ thị Hamilton): gồm các bước:
  + Kiểm tra đồ thị có liên thông hay không.
  + Nếu đồ thị liên thông, chạy thuật toán quay lui:

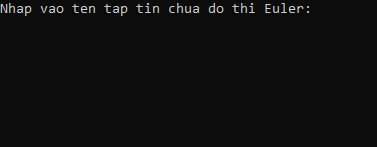
Xem hình minh họa:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a[0] | a[1] | … | a[i-1] | a[i] | … | a[nVer – 1] |

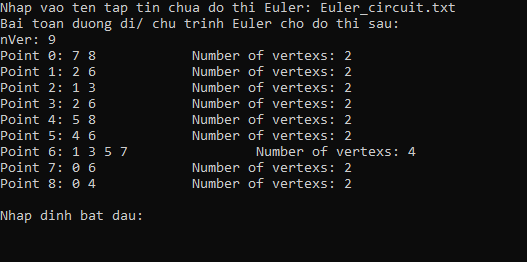
* + - Khởi tạo mảng có số phần tử (tạm gọi là nVer) là số đỉnh của đồ thị, mảng này dùng để lưu trữ các đỉnh có đường đi với nhau, khi đủ nVer phần tử trong mảng thì sẽ tạo thành đường đi/ chu trình Hamilton.
    - Phần tử đầu tiên là đỉnh bắt đầu.
    - Mỗi phần tử a[i] sẽ liền kề với a[i-1] và mọi phần tử a[j] (j từ 0 đến (i – 1)) sẽ không trùng với a[i] (đảm bảo mọi đỉnh chỉ xuất hiện 1 lần trên đường đi). Sau khi thêm 1 đỉnh vào mảng thì xóa cạnh tương ứng giữa 2 đỉnh đó trong DSLK.
    - Thực hiện thuật toán đến khi kiểm tra hết phần tử trong DSLK của từng đỉnh. Nếu có trường hợp đủ nVer đỉnh, sẽ in ra đường đi/ chu trình Hamilton (là chu trình nếu đỉnh a[nVer – 1] là đỉnh kề với đỉnh a[0]).

**2. Cách chạy chương trình:**

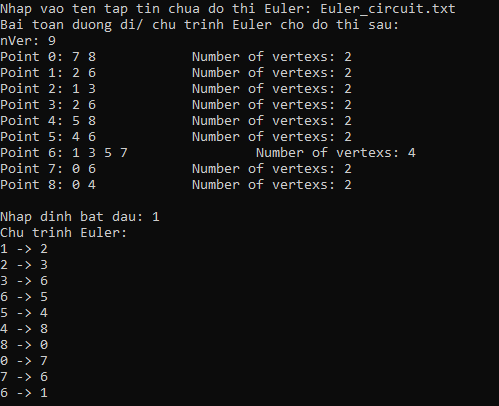
**B1:** Người dùng nhập vào đường dẫn/ tên tập tin (nếu nằm cùng thư mục) chứa đồ thị cần kiểm tra đường đi/chu trình Euler/Hamilton (xem ảnh minh họa bên dưới):



**B2:** Chương trình in ra thông tin về đồ thị và DSLK của từng đỉnh. Người dùng nhập đỉnh bắt đầu để tìm đường đi/chu trình Euler/Hamilton (xem ảnh minh họa bên dưới):



Sau khi người dùng nhập đỉnh bắt đầu, chương trình sẽ in ra đường đi/chu trình (nếu có).



***3. Lưu ý cho giáo viên:***

- Về các vấn đề liên quan đến chu trình/ đường đi Hamilton:

* Các định lý như Dirac hay Ore, … chỉ là điều kiện đủ để một đồ thị là đồ thị Hamilton. Do đó, nếu một đồ thị không thỏa các định lý vẫn có thể là một đồ thị Hamilton/ nửa Hamilton.
* Đồng thời, nếu đồ thị thỏa các định lý trên, cũng chỉ giúp khẳng định một đồ thị là đồ thị Hamilton nhưng không thể cho được thuật toán in ra chu trình Hamilton.
* Do đó, trong đồ án này, nhóm lựa chọn thuật toán quay lui (backtracking) để kiểm tra và in ra (nếu có) tất cả các đường đi/ chu trình có thể có trong một đồ thị. Tuy nhiên, một nhược điểm dễ thấy của thuật toán này là thuật toán chỉ hoạt động được trên đồ thị có số lượng đỉnh và cạnh nhỏ.

**4. Đánh giá mức độ hoàn thành:**

- Đồ án đã hoàn thành các câu hỏi được giao.

- Về mã nguồn, đã cài đặt được các thuật toán như đã trình bày ở trên. Các kỹ thuật hướng đối tượng chỉ sử dụng ở mức cơ bản, vừa đủ và cần thiết.

- Ở các câu hỏi về đồ thị Hamilton, lời giải vẫn tồn tại các vấn đề như đã nói ở phần 3.