**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC TÂY ĐÔ**

**KHOA KỸ THUẬT – CÔNG NGHỆ**

A logo with a green and pink leaf

Description automatically generated

**Trí tuệ - Sáng tạo - Năng động – Đổi mới**

**NIÊN LUẬN 1**

**TÌM CHU TRÌNH EULER**

**CHUYÊN NGÀNH**

**CÔNG NGHỆ THÔNG TIN**

**SINH VIÊN THỰC HIỆN**

**QUÁCH TRƯỜNG PHÚC - 227060168**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC TÂY ĐÔ**

**KHOA KỸ THUẬT – CÔNG NGHỆ**

A logo with a green and pink leaf

Description automatically generated

**Trí tuệ - Sáng tạo - Năng động – Đổi mới**

**NIÊN LUẬN 1**

**TÌM CHU TRÌNH EULER**

**CHUYÊN NGÀNH**

**CÔNG NGHỆ THÔNG TIN**

**SINH VIÊN THỰC HIỆN**

**QUÁCH TRƯỜNG PHÚC - 227060168**

**GIÁO VIÊN HƯỚNG DẪN**

**THS. NGUYỄN CHÍ CƯỜNG**

**MỤC LỤC**

[MỤC LỤC HÌNH ẢNH iii](#_Toc185175248)

[LỜI CẢM ƠN iv](#_Toc185175249)

[LỜI CAM ĐOAN v](#_Toc185175250)

[TÓM TẮT vi](#_Toc185175251)

[ABSTRACT vii](#_Toc185175252)

[ĐÁNH GIÁ KẾT QUẢ THỰC HIỆN viii](#_Toc185175253)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN ix](#_Toc185175254)

[CHƯƠNG I. TỔNG QUAN 1](#_Toc185175255)

[1.1. Mô tả bài toán 1](#_Toc185175256)

[1.2. Mục tiêu cần đạt được 1](#_Toc185175257)

[1.3. Phạm vi nghiên cứu 1](#_Toc185175258)

[1.3.1. Thuật toán và lý thuyết đồ thị 1](#_Toc185175259)

[1.3.2. Ứng dụng thực tế 1](#_Toc185175260)

[1.3.3. Môi trường triển khai 2](#_Toc185175261)

[1.3.4. Phạm vi dữ liệu 2](#_Toc185175262)

[1.4. Hướng giải quyết 2](#_Toc185175263)

[1.5. Phương pháp nghiên cứu 2](#_Toc185175264)

[1.6. Kế hoạch thực hiện 3](#_Toc185175265)

[CHƯƠNG II. CƠ SỞ LÝ THUYẾT 4](#_Toc185175266)

[2.1. Định nghĩa đồ thị 4](#_Toc185175267)

[2.2. Các thuật ngữ cơ bản 5](#_Toc185175268)

[2.2.1. Đỉnh cô lập, đỉnh treo, bậc đỉnh 5](#_Toc185175269)

[2.2.2. Đường đi, chu trình, đồ thị liên thông: 7](#_Toc185175270)

[2.3. Biểu diễn đồ thị 9](#_Toc185175271)

[2.3.1. Khái niệm ma trận kề 10](#_Toc185175272)

[2.3.2. Quy tắc 10](#_Toc185175273)

[2.4. Tìm kiếm trên đồ thị 11](#_Toc185175274)

[2.4.1. Depth First Search – DFS là gì? 11](#_Toc185175275)

[2.4.2. Breadth First Search – BFS là gì? 14](#_Toc185175276)

[2.4.3. Sử dụng DFS để kiểm tra tính liên thông của đồ thị 14](#_Toc185175277)

[2.5. Chu Trình Euler 16](#_Toc185175278)

[2.5.1. Tổng quan 16](#_Toc185175279)

[2.5.2. Khái niệm 16](#_Toc185175280)

[2.5.3. Giải thuật Fleury 17](#_Toc185175281)

[2.6. Môi trường sử dụng 21](#_Toc185175282)

[2.7. Công cụ hỗ trợ 21](#_Toc185175283)

[2.7.1. Visual Studio Code (VS Code) 21](#_Toc185175284)

[2.7.2. GitHub 22](#_Toc185175285)

[2.7.3. diagrams.net 22](#_Toc185175286)

[2.8. Ngôn ngữ sử dụng 23](#_Toc185175287)

[2.8.1. HTML 23](#_Toc185175288)

[2.8.2. CSS 24](#_Toc185175289)

[2.8.3. JavaScript 25](#_Toc185175290)

[2.9. Thư viện sử dụng 26](#_Toc185175291)

[2.9.1. jQuery 26](#_Toc185175292)

[2.9.2. Bootstrap 26](#_Toc185175293)

[2.9.3. p5 28](#_Toc185175294)

[CHƯƠNG III. KẾT QUẢ 29](#_Toc185175295)

[3.1. Vẽ bằng dữ liệu tải lên của tệp (txt) 29](#_Toc185175296)

[3.2. Vẽ thủ công 31](#_Toc185175297)

[3.3. Vẽ ngẫu nhiên 35](#_Toc185175298)

[KẾT LUẬN – ĐÁNH GIÁ 37](#_Toc185175299)

[TÀI LIỆU THAM KHẢO 38](#_Toc185175300)

**MỤC LỤC HÌNH ẢNH**

[Hình 2. 1 Đồ thị trong thực tế 4](#_Toc185005247)

[Hình 2. 2 Ví dụ về đơn đồ thị vô hướng 4](#_Toc185005248)

[Hình 2. 3 Ví dụ về đa đồ thị vô hướng 5](#_Toc185005249)

[Hình 2. 5 Ví dụ về bậc định, đỉnh cô lập, đỉnh treo 6](#_Toc185005251)

[Hình 2. 7 Ví dụ về đường đi ,chu trình của đồ thị 7](#_Toc185005253)

[Hình 2. 8 Ví dụ về đồ thị liên thông 8](#_Toc185005254)

[Hình 2. 9 Ví dụ về các đỉnh cắt, cạnh cầu 8](#_Toc185005255)

[Hình 2. 10 Ví dụ về đồ thị liên thông mạnh, yếu 9](#_Toc185005256)

[Hình 2. 11 Đồ thị G 10](#_Toc185005257)

[Hình 2. 12 Kết quả ma trận kề của đồ thị G 11](#_Toc185005258)

[Hình 2. 13 Đồ thị ban đầu 13](#_Toc185005259)

[Hình 2. 14 Bước đầu tiên 13](#_Toc185005260)

[Hình 2. 15 Bước thứ hai 14](#_Toc185005261)

[Hình 2. 16 Bước thứ ba 14](#_Toc185005262)

[Hình 2. 17 Bước thứ bốn 14](#_Toc185005263)

[Hình 2. 18 Kết thúc 15](#_Toc185005264)

[Hình 2. 19 Lưu đồ thuật toán DFS 16](#_Toc185005265)

[Hình 2. 20 Bài toán 7 cây cầu 17](#_Toc185005266)

[Hình 2. 21 Đồ thị G1, G2, G3 18](#_Toc185005267)

[Hình 2. 22 Bước đầu tiên 19](#_Toc185005268)

[Hình 2. 23 Bước thứ hai 19](#_Toc185005269)

[Hình 2. 24 Bước thứ ba 20](#_Toc185005270)

[Hình 2. 25 Bước thứ tư 20](#_Toc185005271)

[Hình 2. 26 Bước thứ năm 20](#_Toc185005272)

[Hình 2. 27 Bước thứ sáu 21](#_Toc185005273)

[Hình 2. 28 Lưu đồ giải thuật Fleury 21](#_Toc185005274)

[Hình 3. 1 Giao diện chính 30](#_Toc185005275)

[Hình 3. 2 Giao diện chức năng này 30](#_Toc185005276)

[Hình 3. 3 Chức năng chọn tệp tin 30](#_Toc185005277)

[Hình 3. 4 Chức năng làm mới tệp 31](#_Toc185005278)

[Hình 3. 5 Trở về trang chủ 31](#_Toc185005279)

[Hình 3. 6 Kết quả sau khi tải lên 32](#_Toc185005280)

[Hình 3. 7 Giao diện chính của chức năng này 33](#_Toc185005281)

[Hình 3. 8 Nhận số đỉnh 33](#_Toc185005282)

[Hình 3. 9 Tạo ra các ô nhập dữ liệu 33](#_Toc185005283)

[Hình 3. 10 Nhập đỉnh kết nối vào các ô nhập liệu 34](#_Toc185005284)

[Hình 3. 11 Ví dụ khi nhập dữ liệu 34](#_Toc185005285)

[Hình 3. 12 Thêm đỉnh muốn kết nối 34](#_Toc185005286)

[Hình 3. 13 Sau khi nhấn “Thêm đỉnh” 35](#_Toc185005287)

[Hình 3. 14 Vẽ ra đồ thị mong muốn 35](#_Toc185005288)

[Hình 3. 15 Kết quả 36](#_Toc185005289)

[Hình 3. 16 Giao diện chính 37](#_Toc185005290)

[Hình 3. 17 Tiến hành tạo đồ thị ngẫu nhiên 37](#_Toc185005291)

[Hình 3. 18 Kết quả khi được tạo ngẫu nhiên 38](#_Toc185005292)

**LỜI CẢM ƠN**

Lời đầu tiên, tôi xin gửi lời cảm ơn chân thành tới Thầy Nguyễn Chí Cường, người đã tận tình hướng dẫn và chỉ bảo tôi trong suốt quá trình thực hiện bài niên luận này. Thầy đã giúp tôi vượt qua nhiều khó khăn và cung cấp những kiến thức quý báu, góp phần quan trọng vào việc hoàn thành nghiên cứu này. Bên cạnh đó, tôi cũng xin bày tỏ lòng biết ơn sâu sắc tới các thầy cô trong Khoa Kỹ Thuật Công Nghệ, Trường Đại học Tây Đô, vì đã tạo điều kiện thuận lợi trong quá trình học tập và nghiên cứu. Cảm ơn gia đình, cha mẹ và những người thân yêu đã luôn động viên, khích lệ tôi cả về tinh thần lẫn vật chất trong suốt chặng đường học tập. Cuối cùng, tôi cũng xin cảm ơn bạn bè đã luôn hỗ trợ và đóng góp ý kiến quý báu, giúp tôi hoàn thiện bài niên luận này. Những lời động viên và sự giúp đỡ của mọi người chính là nguồn động lực to lớn giúp tôi hoàn thành tốt công việc nghiên cứu. Xin chân thành cảm ơn!

**LỜI CAM ĐOAN**

Tôi xin cam đoan rằng Niên luận này là do chính tôi thực hiện, không sao chép dưới bất kỳ hình thức nào hay thuê hoặc nhờ người khác thực hiện.

Dữ liệu và kết quả phân tích trong Niên luận đảm bảo tính chính xác, khách quan và trung thực, không có bất kỳ sự ngụy tạo và điều chỉnh kết quả nghiên cứu bằng sự chủ quan của tác giả.

*Cần Thơ, ngày…..tháng……năm 20…*

Sinh viên thực hiện

*(Ký tên và ghi rõ họ và tên)*

**Quách Trường Phúc**

**TÓM TẮT**

Chu trình Euler là một vấn đề lâu đời trong lý thuyết đồ thị, có ứng dụng rộng rãi trong các lĩnh vực như mạng lưới, tối ưu hóa, phân tích dữ liệu, và nhiều bài toán thực tiễn khác. Bài niên luận này tập trung vào việc tìm kiếm chu trình Euler trong đồ thị vô hướng, một chu trình đi qua mỗi cạnh của đồ thị đúng một lần và quay lại điểm xuất phát.

Việc xác định chu trình Euler có ý nghĩa quan trọng trong các bài toán liên quan đến vận hành mạng, thiết kế tuyến đường, hay các bài toán liên quan đến phân chia tài nguyên. Bối cảnh nghiên cứu xuất phát từ các ứng dụng trong thực tế, nơi các hệ thống mạng và tuyến đường yêu cầu tối ưu hóa việc đi qua tất cả các kết nối mà không trùng lặp.

Ví dụ, bài toán tìm chu trình Euler đã được áp dụng trong các bài toán như giải quyết vấn đề "Người bán hàng đi qua tất cả các thành phố mà không đi qua một thành phố nào hai lần".

Trong nghiên cứu này, mục tiêu là tìm ra phương pháp hiệu quả để phát hiện và kiểm tra điều kiện tồn tại của chu trình Euler trong đồ thị, đồng thời ứng dụng các thuật toán tìm kiếm chu trình Euler. Kết quả nghiên cứu đã xác định các điều kiện tồn tại chu trình Euler, bao gồm việc kiểm tra bậc của các đỉnh trong đồ thị.

Đối với đồ thị vô hướng, một chu trình Euler tồn tại nếu và chỉ nếu tất cả các đỉnh có bậc chẵn. Từ đó, các thuật toán tìm chu trình Euler đã được triển khai và thử nghiệm, cho thấy khả năng áp dụng hiệu quả trong các bài toán thực tế. Các kết quả đạt được không chỉ giúp khẳng định lý thuyết, mà còn mở ra các ứng dụng trong việc tối ưu hóa và giải quyết các vấn đề mạng phức tạp. Nghiên cứu này sẽ đóng góp vào việc phát triển các thuật toán tìm chu trình Euler, từ đó có thể áp dụng chúng trong các hệ thống đồ thị phức tạp, đồng thời nâng cao khả năng giải quyết bài toán trong các lĩnh vực khoa học và công nghệ.

**ABSTRACT**

Euler's Circuit is a classic problem in graph theory, with wide applications in fields such as network optimization, data analysis, and many practical problems. This research focuses on finding Euler's Circuit in undirected graphs, a circuit that traverses each edge of the graph exactly once and returns to the starting point.

Determining Euler's Circuit is crucial in problems related to network operation, route design, and resource allocation. The research context stems from real-world applications, where network systems and routes require optimization to traverse all connections without duplication.

For example, the Euler's Circuit problem has been applied to solve the "Traveling Salesman Problem", where a salesman needs to visit all cities without visiting any city twice.

In this study, the objective is to develop an efficient method to detect and verify the existence of Euler's Circuit in graphs, and to apply algorithms for finding Euler's Circuit. The research results have identified the conditions for the existence of Euler's Circuit, including checking the degree of vertices in the graph.

For undirected graphs, an Euler's Circuit exists if and only if all vertices have even degrees. Subsequently, algorithms for finding Euler's Circuit have been implemented and tested, demonstrating their effective application in practical problems. The results not only confirm the theory but also open up applications in optimizing and solving complex network problems. This research will contribute to the development of Euler's Circuit algorithms, enabling their application in complex graph systems and enhancing problem-solving capabilities in scientific and technological fields.

# ĐÁNH GIÁ KẾT QUẢ THỰC HIỆN

**NIÊN LUẬN, TIỂU LUẬN, KHOÁ LUẬN**

(Học kỳ: I, Năm 3)

**TÊN ĐỀ TÀI: TÌM CHU TRÌNH EULER**

**GIÁO VIÊN HƯỚNG DẪN:**

|  |  |  |
| --- | --- | --- |
| STT | HỌ VÀ TÊN | MSCB |
| 1 | Nguyễn Chí Cường |  |

**SINH VIÊN THỰC HIỆN:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | MSSV | THƯỞNG  *(Tối đa 1,0 điểm)* | ĐIỂM |
| 1 | Quách Trường Phúc | 227060168 |  |  |

|  |  |
| --- | --- |
| **I. HÌNH THỨC** *(Tối đa 1,5 điểm)* |  |
| **II. NỘI DUNG** *(Tối đa 5 điểm)* |  |
| **Ứng dụng** *(tối đa 3 điểm)* |  |
| * Giới thiệu chương trình (0,5 điểm) |  |
| **III. CHƯƠNG TRÌNH DEMO** *(Tối đa 3.5 điểm)* |  |
| **Giao diện thân thiện với người dùng** *(0.5 điểm)* |  |
| **Hướng dẫn sử dụng** *(0.5 điểm)* |  |
| **Kết quả thực hiện đúng với kết quả của phần ứng dụng**   * Giải thuật đúng, thực thi chính xác,… *(2.5 điểm)* |  |

**Ghi chú:**

1. *Điểm trong khung “sinh viên thực hiện” là điểm kết quả cuối cùng của từng sinh viên trong quá trình thực hiện* Niên luận…/tiểu luận/khoá luận*.*
2. *Nếu sinh viên demo chương trình và trả lời vấn đáp không đạt yêu cầu của giáo viên hướng dẫn thì sinh viên sẽ nhận điểm F cho học phần này.*

Cần Thơ, ngày .......... tháng năm 2024

**GIÁO VIÊN CHẤM**

# NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

🖎🕮✍

.....................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

|  |
| --- |
| *Cần Thơ, ngày tháng năm 2024* |
| **Giáo viên hướng dẫn** |
| **………………..** |

# CHƯƠNG I. TỔNG QUAN

1. **Mô tả bài toán**

Bài toán tập trung vào việc xác định và tìm chu trình Euler hoặc nửa Euler trên đồ thị thông qua các chức năng được triển khai trên nền tảng web. Đồ thị sẽ được xây dựng và xử lý bằng ba phương pháp chính: tạo ngẫu nhiên, vẽ thủ công, và nhập từ file. Sau khi đồ thị được tạo, hệ thống sẽ xác định loại đồ thị (Euler, nửa Euler, hoặc không là Euler) và sau đó sẽ tìm chu trình hoặc đường đi Euler của đồ thị đó.

1. **Mục tiêu cần đạt được**

* Nắm vững các lý thuyết liên quan đến đồ thị, chu trình Euler và các khái niệm cơ bản về lý thuyết đồ thị, bao gồm cả định nghĩa, tính chất và ứng dụng của chu trình Euler.Triển khai thuật toán DFS, Fleury thành công.
* Triển khai thuật toán DFS (Depth-First Search) và Fleury thành công, bao gồm việc hiểu rõ cách thức hoạt động của các thuật toán này, cách áp dụng chúng vào các vấn đề thực tế và tối ưu hóa hiệu suất của chương trình.Chương trình phải cho ra kết quả chính xác.
* Đảm bảo chương trình hoạt động hiệu quả, ít lỗi nhất có thể, bao gồm việc kiểm tra và xử lý các trường hợp ngoại lệ, tối ưu hóa mã nguồn và đảm bảo chương trình có thể chạy trên nhiều nền tảng khác nhau.
* Chương trình phải cho ra kết quả chính xác, bao gồm việc kiểm tra và xác minh kết quả đầu ra, đảm bảo chương trình có thể xử lý các dữ liệu đầu vào khác nhau và cung cấp kết quả chính xác trong mọi trường hợp.

1. **Phạm vi nghiên cứu**
   1. **Thuật toán và lý thuyết đồ thị**

Phạm vi nghiên cứu bao gồm các khái niệm cơ bản và lý thuyết nền tảng về đồ thị, như chu trình Euler, đường đi Euler, bậc đỉnh, tính liên thông, điểm khớp, và các định lý liên quan. Nghiên cứu tập trung vào thuật toán tìm kiếm sâu (DFS) và thuật toán Fleury để xác định và xây dựng chu trình Euler hoặc đường đi Euler trên đồ thị vô hướng.

* 1. **Ứng dụng thực tế**

Đề tài nghiên cứu tìm hiểu các ứng dụng của lý thuyết đồ thị và thuật toán chu trình Euler trong các lĩnh vực như mạng lưới giao thông, tối ưu hóa tuyến đường, thiết kế mạch điện, và phân tích dữ liệu. Việc ứng dụng còn mở rộng đến các công cụ trực quan hóa giúp minh họa rõ ràng cấu trúc và kết quả phân tích của đồ thị.

* 1. **Môi trường triển khai**

Chương trình được xây dựng và triển khai trên nền tảng web, sử dụng các công nghệ như JavaScript, HTML, CSS, và thư viện hỗ trợ như p5.js để trực quan hóa đồ thị. Quá trình phát triển và thử nghiệm được thực hiện trên các trình duyệt hiện đại như Google Chrome, Mozilla Firefox, và Microsoft Edge, đảm bảo khả năng tương thích và hiệu suất ổn định.

* 1. **Phạm vi dữ liệu**

Dữ liệu đầu vào được giới hạn ở các ma trận kề, biểu diễn đồ thị vô hướng. Kích thước đồ thị được thiết kế phù hợp với khả năng xử lý của chương trình, đảm bảo hiệu suất tối ưu khi thao tác trên các đồ thị có số đỉnh từ nhỏ đến trung bình. Dữ liệu được nhập từ ba nguồn: tạo ngẫu nhiên, vẽ thủ công, và nhập từ file định dạng văn bản theo cấu trúc ma trận kề.

1. **Hướng giải quyết**

* Xây dựng giao diện: Thiết kế giao diện trực quan với các chức năng hỗ trợ người dùng tạo đồ thị, bao gồm tạo đồ thị mới, nhập dữ liệu đồ thị, hiển thị đồ thị, chỉnh sửa đồ thị và lưu đồ thị. Giao diện phải dễ sử dụng và thân thiện với người dùng, cho phép họ tạo và chỉnh sửa đồ thị một cách dễ dàng.
* Thuật toán xử lý: Sử dụng các thuật toán kiểm tra bậc đỉnh và tìm chu trình Euler, nửa Euler dựa trên điều kiện lý thuyết. Thuật toán phải đảm bảo tính chính xác và hiệu quả khi tìm chu trình hoặc đường đi, và phải có khả năng xử lý các đồ thị phức tạp. Ngoài ra, thuật toán cũng phải có khả năng kiểm tra bậc đỉnh và tìm chu trình Euler, nửa Euler trong thời gian thực.
* Xuất kết quả: Hiển thị loại đồ thị (Euler, nửa Euler, không là Euler) và chu trình hoặc đường đi Euler. Kết quả phải được hiển thị một cách rõ ràng và dễ hiểu, cho phép người dùng dễ dàng phân tích và hiểu kết quả. Ngoài ra, kết quả cũng phải được lưu lại để người dùng có thể tham khảo sau này.

1. **Phương pháp nghiên cứu**

* Nghiên cứu tài liệu: Tìm hiểu và nghiên cứu các tài liệu liên quan đến lý thuyết đồ thị, chu trình Euler và các thuật toán liên quan.
* Phân tích yêu cầu: Phân tích yêu cầu bài toán và xác định các chức năng cần thiết cho hệ thống.
* Thiết kế hệ thống: Thiết kế giao diện và xác định cấu trúc dữ liệu phù hợp cho hệ thống.
* Phát triển ứng dụng: Xây dựng chức năng vẽ đồ thị và triển khai thuật toán kiểm tra và tìm chu trình hoặc đường đi Euler.
* Kiểm thử và tối ưu hóa: Kiểm tra độ chính xác của ứng dụng và tối ưu hiệu năng xử lý.
* Đánh giá kết quả: Đánh giá kết quả của hệ thống và xác định các điểm mạnh, điểm yếu cũng như những cải tiến cần thiết cho phiên bản tiếp theo. Phương pháp này sẽ giúp đảm bảo rằng hệ thống không chỉ hoạt động hiệu quả mà còn đáp ứng được nhu cầu của người dùng. Bên cạnh đó, việc thu thập phản hồi từ người dùng cũng sẽ được thực hiện để cải thiện trải nghiệm và tính năng của ứng dụng trong tương lai.

1. **Kế hoạch thực hiện**

* **Tuần 1 - 2: Nghiên cứu và thiết kế hệ thống**
* Phân tích yêu cầu bài toán và xác định các chức năng cần thiết cho hệ thống.
* Tìm hiểu và nghiên cứu tài liệu liên quan đến lý thuyết đồ thị, chu trình Euler và các thuật toán liên quan.
* Thiết kế giao diện dễ sử dụng, trình bày rõ ràng và thân thiện với người dùng, cho phép họ tạo và chỉnh sửa đồ thị một cách dễ dàng.
* Xác định cấu trúc dữ liệu phù hợp cho hệ thống, bao gồm cả cấu trúc dữ liệu cho đồ thị và cấu trúc dữ liệu cho thuật toán.
* **Tuần 3 - 4: Phát triển ứng dụng**
* Xây dựng chức năng vẽ đồ thị ngẫu nhiên, thủ công và từ file.
* Triển khai thuật toán DFS và Fleury kiểm tra và tìm chu trình hoặc đường đi Euler.
* **Tuần 5 - 6: Kiểm thử và tối ưu hóa**
* Kiểm tra độ chính xác của chương trình với nhiều đồ thị khác nhau.
* Tối ưu hiệu năng xử lý của mã nguồn, thuật toán.
* **Tuần 7 - 8: Hoàn thiện và báo cáo**
* Kiểm tra, tìm lỗi có thể xảy ra trong chương trình
* Hoàn thành chương trình
* Soạn thảo báo cáo, trình bày về chức năng, cấu trúc dữ liệu và thuật toán của hệ thống.

# CHƯƠNG II. CƠ SỞ LÝ THUYẾT

1. **Định nghĩa đồ thị**

A computer icons with a white background

Description automatically generated with medium confidenceĐồ thị là một cấu trúc toán học rời rạc, bao gồm hai thành phần chính là các đỉnh và các cạnh, dùng để mô tả mối liên kết giữa các đỉnh. Các cạnh có thể có hướng hoặc không có hướng, và dựa trên đặc điểm này cũng như cách kết nối các đỉnh, đồ thị được phân thành nhiều loại khác nhau.

*Nguồn: Nguyễn Hữu Danh, 2023.*

Hình 2. 1 Đồ thị trong thực tế

Một đơn đồ thị vô hướng G=(V,E) là một đồ thị mà tập V (tập đỉnh) không rỗng và mỗi cạnh trong tập E (tập cạnh) là một cặp không có thứ tự của các đỉnh phân biệt.

A diagram of a city

Description automatically generated

*Nguồn: Nguyễn Hữu Danh, 2023.*

Hình 2. 2 Ví dụ về đơn đồ thị vô hướng

Đa đồ thị vô hướng cũng có tập đỉnh V và tập cạnh E, nhưng cho phép các cạnh bội tức là nhiều cạnh có thể cùng kết nối một cặp đỉnh. Điều này khiến đa đồ thị có tính linh hoạt hơn, nhưng không phải đa đồ thị nào cũng là đơn đồ thị.

A diagram of a river

Description automatically generated

*Nguồn: Nguyễn Hữu Danh, 2023.*

Hình 2. 3 Ví dụ về đa đồ thị vô hướng

Một đơn đồ thị có hướng G = (V, E) gồm một tập khác rỗng V mà các phần tử của nó gọi là các đỉnh và một tập E mà các phần tử của nó gọi là các cung, đó là các cặp có thứ tự của các phần tử thuộc V.

A diagram of a chicago river

Description automatically generated

Nguồn: Nguyễn Hữu Danh, 2023.  
Hình 2. 4 Ví dụ về đồ thị có hướng

1. **Các thuật ngữ cơ bản**
2. **Đỉnh cô lập, đỉnh treo, bậc đỉnh**

* Hai đỉnh uu và vv trong đồ thị G=(V,E) được gọi là liền kề nếu có cạnh nối chúng, ký hiệu (u,v) ∈ E. Cạnh này cũng được gọi là cạnh liên thuộc với hai đỉnh u và v. Bậc của một đỉnh v, ký hiệu deg(v) là số cạnh liên thuộc với đỉnh đó. Trong trường hợp một cạnh là khuyên (nối đỉnh với chính nó), bậc của đỉnh sẽ tăng thêm hai đơn vị. Các đỉnh có bậc bằng một được gọi là đỉnh treo, còn các đỉnh không có cạnh liên thuộc được gọi là đỉnh cô lập. Đỉnh v gọi là đỉnh treo nếu deg(v) = 1 và gọi là đỉnh cô lập nếu deg(v) = 0.

A diagram of a diagram

Description automatically generated**Ví dụ:**

*Nguồn: Nguyễn Hữu Danh, 2023.*

Hình 2. 5 Ví dụ về bậc định, đỉnh cô lập, đỉnh treo

Ta có deg(v1) = 3, deg(v2) = 4, deg(v3) = 5, deg(v4) = 1, deg(v5) = 3, deg(v6) = 2, deg(v7) = 2. Đỉnh v8 là đỉnh cô lập và v4 là đỉnh treo.

**Mệnh đề** Cho đồ thị G = (V, E). Khi đó

A black symbol with a white background

Description automatically generated

**Hệ quả:** Số đỉnh bậc lẻ của một đồ thị là một số chẵn.

**Mệnh đề:** Trong một đơn đồ thị, luôn tồn tại hai đỉnh có cùng bậc.

* Trong đồ thị có hướng G, một đỉnh u được gọi là nối tới đỉnh v (hay v được gọi là được nối từ u) nếu (u,v) là một cung của G. Trong trường hợp này, u được gọi là đỉnh đầu, còn v là đỉnh cuối của cung. Bậc vào và bậc ra của một đỉnh v trong đồ thị có hướng được ký hiệu lần lượt là deg- (v) (t.ư. deg+(v)). Bậc vào là số lượng cung có đỉnh cuối là v, trong khi bậc ra là số lượng cung có đỉnh đầu là v.

**Ví dụ:**

A diagram of a diagram

Description automatically generated

Nguồn: Nguyễn Hữu Danh, 2023.  
Hình 2. 6 Ví dụ về đỉnh cô đơn, đỉnh treo

* deg- (v1 ) = 2, deg+(v1) = 3,
* deg- (v2 ) = 5, deg+(v2) = 1,
* deg- (v3 ) = 2, deg+(v3) = 4,
* deg- (v4 ) = 1, deg+(v4) = 3,
* deg- (v5 ) = 1, deg+(v5) = 0,
* deg- (v6 ) = 0, deg+(v6) = 0.

Đỉnh có bậc vào và bậc ra cùng bằng 0 gọi là đỉnh cô lập. Đỉnh có bậc vào bằng 1 và bậc ra bằng 0 gọi là đỉnh treo, cung có đỉnh cuối là đỉnh treo gọi là cung treo.

1. **Đường đi, chu trình, đồ thị liên thông:**

Đường đi trong đồ thị vô hướng là một dãy liên tiếp các đỉnh, bắt đầu từ đỉnh u và kết thúc tại đỉnh v, trong đó mỗi cặp đỉnh liên tiếp trong dãy đều có cạnh nối. Nếu điểm bắt đầu và điểm kết thúc trùng nhau, đường đi này được gọi là chu trình. Một đường đi hoặc chu trình được gọi là đơn nếu không có cạnh nào bị lặp lại.

A black line with black dots and a triangle

Description automatically generated **Ví dụ:**

*Nguồn: Nguyễn Hữu Danh, 2023.*

Hình 2. 7 Ví dụ về đường đi ,chu trình của đồ thị

* a, d, c, f, e là đường đi đơn độ dài 4.
* d, e, c, a không là đường đi vì (e,c) không phải là cạnh của đồ thị.
* Dãy b, c, f, e, b là chu trình độ dài 4.
* Đường đi a, b, e, d, a, b có độ dài 5 không phải là đường đi đơn vì cạnh (a,b) có mặt hai lần.

**Định nghĩa**: Đồ thị được gọi là liên thông nếu giữa mọi cặp đỉnh phân biệt đều tồn tại ít nhất một đường đi. Nếu không liên thông, đồ thị có thể chia thành các thành phần liên thông độc lập. Đỉnh cắt và cạnh cắt là các yếu tố quan trọng trong việc xác định cấu trúc liên thông của đồ thị. Đỉnh cắt là đỉnh mà khi bị loại bỏ cùng với các cạnh liên thuộc sẽ làm tăng số thành phần liên thông. Tương tự, cạnh cắt là cạnh mà khi loại bỏ sẽ làm đồ thị bị chia thành nhiều thành phần hơn.

**Ví dụ:**A diagram of a network

Description automatically generated

Nguồn: Nguyễn Hữu Danh, 2023

Hình 2. 8 Ví dụ về đồ thị liên thông

Ở đây đồ thị G là liên thông, nhưng đồ thị G’ không liên thông và có 3 thành phần liên thông.

A diagram of a network

Description automatically generated**Ví dụ:**

*Nguồn: Nguyễn Hữu Danh, 2023.*

Hình 2. 9 Ví dụ về các đỉnh cắt, cạnh cầu

Trong đồ thị trên, các đỉnh cắt là v, w, s và các cầu là (x,v), (w,s).

* Trong một đồ thị liên thông, giữa mọi cặp đỉnh phân biệt luôn tồn tại một đường đi sơ cấp nối chúng. Đối với mọi đơn đồ thị có n đỉnh (n≥2), nếu tổng bậc của hai đỉnh bất kỳ không nhỏ hơn n, thì đồ thị đó luôn liên thông. Ngoài ra, một đơn đồ thị trong đó bậc của mỗi đỉnh không nhỏ hơn một nửa số đỉnh cũng được đảm bảo là liên thông. Đặc biệt, nếu đồ thị có đúng hai đỉnh bậc lẻ, thì hai đỉnh này bắt buộc phải liên thông, tức là tồn tại ít nhất một đường đi nối giữa chúng.
* Trong một đồ thị liên thông G=(V, E), một đỉnh được gọi là điểm khớp nếu và chỉ nếu tồn tại hai đỉnh u và v sao cho mọi đường đi nối giữa u và v đều phải đi qua đỉnh này. Đối với một đơn đồ thị G với n đỉnh, m cạnh và k thành phần liên thông, các tính chất liên quan đến cấu trúc và điểm khớp đóng vai trò quan trọng trong việc phân tích sự kết nối và phân tách của đồ thị.

Khi đó:

A black text on a white background

Description automatically generated

Một đồ thị có hướng G được gọi là **liên thông mạnh** nếu với mọi cặp đỉnh phân biệt u và v, luôn tồn tại cả đường đi từ u đến v và từ v đến u. Nếu đồ thị G không đảm bảo liên thông mạnh nhưng đồ thị vô hướng nền của nó là liên thông, thì G được gọi là **liên thông yếu**. Ngoài ra, G được gọi là **liên thông một chiều** nếu với mọi cặp đỉnh phân biệt u và v, tồn tại ít nhất một đường đi giữa hai đỉnh, tức là từ u đến v hoặc từ v đến u.

**Ví dụ:**

A diagram of a diagram

Description automatically generated with medium confidence

Nguồn: Nguyễn Hữu Danh, 2023.  
Hình 2. 10 Ví dụ về đồ thị liên thông mạnh, yếu

Đồ thị G là liên thông mạnh nhưng đồ thị G’ là liên thông yếu (không có đường đi từ u tới x cũng như từ x tới u).

1. **Biểu diễn đồ thị**

Có rất nhiều cách khác nhau để biểu diễn một đồ thị, mỗi cách đều có những ưu điểm và ứng dụng riêng phù hợp với từng bài toán hoặc nhu cầu cụ thể. Một số phương pháp phổ biến để biểu diễn đồ thị bao gồm:

1. **Ma trận kề**: Đây là một bảng hai chiều, trong đó mỗi ô biểu diễn mối liên hệ giữa hai đỉnh của đồ thị. Nếu có cạnh nối hai đỉnh, ô tương ứng sẽ có giá trị khác 0 (thường là 1 trong đồ thị không trọng số).
2. **Ma trận trọng số**: Một dạng mở rộng của ma trận kề, trong đó các ô chứa trọng số của cạnh nối hai đỉnh. Phương pháp này thường được sử dụng cho các bài toán liên quan đến đồ thị có trọng số.
3. **Danh sách cạnh**: Liệt kê tất cả các cạnh trong đồ thị, mỗi cạnh được biểu diễn bằng cặp đỉnh mà nó kết nối. Cách này thích hợp để lưu trữ đồ thị thưa, khi số lượng cạnh ít hơn nhiều so với số lượng đỉnh.
4. **Danh sách kề**: Mỗi đỉnh trong đồ thị được liên kết với một danh sách các đỉnh mà nó kết nối. Phương pháp này thường tiết kiệm bộ nhớ hơn so với ma trận kề, đặc biệt khi đồ thị có ít cạnh.

Trong phạm vi nghiên cứu này, để đơn giản hóa và phù hợp với mục đích trình bày, ta chỉ tập trung vào việc biểu diễn đồ thị bằng **ma trận kề**, một phương pháp trực quan và dễ hiểu, thường được sử dụng rộng rãi trong lý thuyết đồ thị.

* 1. **Khái niệm ma trận kề**

Xét đồ thị G = (X, U) (Có hướng hay vô hướng)

Giả sử tập X gồm n đỉnh và được sắp xếp theo thứ tự X = {x1, x2, …, xn}, tập U gồm n cạnh và được sắp thứ tự U = {u1, u2, …, un}

* 1. **Quy tắc**

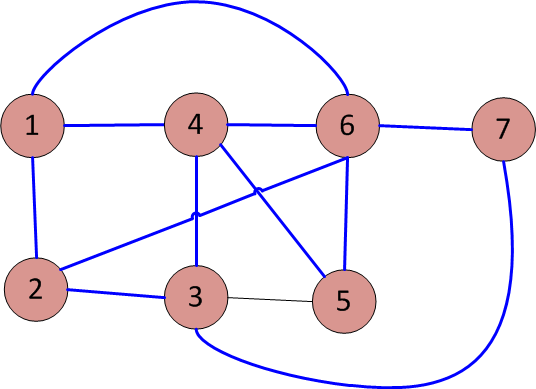
Ma trận kề của đồ thị G, ký hiệu B(G), là một ma trận nhị phân cấp n x n được định nghĩa như sau: B = (Bij) với:

* B = (Bij = 1) nếu có cạnh nối xi tới xj
* B = (Bij = 0) nếu không có cạnh nối xi tới xj

Nếu G là đồ thị vô hướng, ma trận liên thuộc của đồ thị G, ký hiệu A(G), là ma trận nhị phân cấp nxm được định nghĩa như sau: A = (Aij)

* A = (Aij) = 1 nếu có cạnh nối xi tới xj
* A = (Aij) = 0 nếu không có cạnh nối xi tới xj

**Ví dụ:**



*Nguồn:Wikipedia.*

Hình 2. 11 Đồ thị G

Gọi A là ma trận kề biểu diễn đồ thị G.

Từ đồ thị G, ta thấy:

1 và 2 có cạnh nối => A12 = A21 = 1A 12 = A 21 = 1

1 và 4 có cạnh nối => A14 = A41 = 1A 14 = A 41 = 1 AA

1 và 6 có cạnh nối => A16 = A61 = 1A 16 = A 61 = 1 AASDA

2 và 3 có cạnh nối => A23 = A32 = 1A 23 = A 32 = 1 SAD

2 và 6 có cạnh nối => A26 = A62 = 1A 26 = A 62 = 1

3 và 4 có cạnh nối => A34 = A43 = 1A 34 = A 43 = 1 A

3 và 5 có cạnh nối => A35 = A53 = 1A 35 = A 53 = 1

3 và 7 có cạnh nối => A37 = A73 = 1A 37 = A 73 = 1

4 và 6 có cạnh nối => A46 = A64 = 1A 46 = A 64 = 1

4 và 5 có cạnh nối => A45 = A54 = 1A 45 = A 54 = 1

5 và 6 có cạnh nối => A56 = A65 = 1A 56 = A 65 = 1 AA

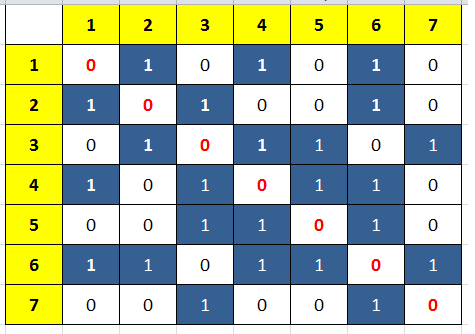
6 và 7 có cạnh nối => A67 = A76 = 1A 67 = A 76 = 1

Còn lại các cặp đỉnh không có cạnh với nhau => Aij = Aji = 0

**Kết quả của ma trận kề:**

*Nguồn:Wikipedia.*

Hình 2. 12 Kết quả ma trận kề của đồ thị G



1. **Tìm kiếm trên đồ thị**

Trong phạm vi nghiên cứu này, trọng tâm chính là thuật toán tìm kiếm theo chiều sâu (Depth First Search - DFS), được lựa chọn để giải quyết bài toán đã đặt ra. Mặc dù có đề cập đến thuật toán tìm kiếm theo chiều rộng (Breadth First Search - BFS) nhằm cung cấp cái nhìn toàn diện hơn về các phương pháp duyệt đồ thị, nội dung bài viết sẽ tập trung phân tích sâu vào cơ chế hoạt động, các bước triển khai, và ứng dụng cụ thể của DFS. Lựa chọn này xuất phát từ sự phù hợp của DFS trong việc xử lý các bài toán liên quan đến chu trình Euler, nơi cần duyệt theo chiều sâu để xác định cấu trúc liên thông và các liên kết phức tạp của đồ thị.

1. **Depth First Search – DFS là gì?**

* Tìm kiếm theo chiều sâu (Depth First Search - DFS) là một thuật toán quan trọng dùng để duyệt hoặc tìm kiếm trong các cấu trúc dữ liệu dạng đồ thị hoặc cây. Thuật toán khởi đầu từ một nút gốc (được chọn tùy ý trong trường hợp đồ thị) và tiếp tục duyệt xuống theo từng nhánh, đi càng sâu càng tốt trước khi quay lại để khám phá các nhánh chưa được duyệt.
* Kết quả của quá trình thực hiện DFS là một **cây bao trùm** (spanning tree), một dạng đồ thị đặc biệt không chứa chu trình, trong đó tất cả các đỉnh được kết nối với nhau thông qua các cạnh duy nhất. Cây bao trùm giúp xác định cấu trúc liên thông của đồ thị trong quá trình duyệt.
* Để triển khai DFS, cần sử dụng một cấu trúc dữ liệu ngăn xếp (stack). Ngăn xếp này sẽ lưu trữ các đỉnh tạm thời trong quá trình duyệt, với kích thước tối đa bằng tổng số đỉnh của đồ thị. Thuật toán hoạt động theo nguyên tắc LIFO (Last In, First Out), đảm bảo rằng các đỉnh mới được thêm vào ngăn xếp sẽ được xử lý trước khi quay lui để duyệt các đỉnh khác.
* Tóm lại, ta cần thực hiện các bước sau:
* Lấy một đỉnh bất kỳ trong đồ thì đưa vào ngăn xếp.
* Lấy top value của ngăn xếp để duyệt và thêm vào visited list.
* Tạo một list bao gồm các đỉnh liền kề của đỉnh đang xét, thêm những đỉnh không có trong visited list vào ngăn xếp.
* Tiếp tục lặp lại bước 2 và bước 3 đến khi ngăn xếp rỗng.

**Ví dụ:**

Hãy xem thuật toán Tìm kiếm theo chiều sâu hoạt động như thế nào với một ví dụ. Chúng ta dùng đồ thị vô hướng có 5 đỉnh.

A black background with white squares

Description automatically generated

Nguồn: viblo.asia

Hình 2. 13 Đồ thị ban đầu của ví dụ về DFS

A screenshot of a computer

Description automatically generatedChúng ta bắt đầu từ đỉnh 0, thuật toán DFS bắt đầu bằng cách đứa nó vào danh sách Visited và đưa tất cả các cạnh liền kề đỉnh đang xét vào ngăn xếp.

Nguồn: viblo.asia

Hình 2. 14 Bước đầu tiên của ví dụ về DFS

A screenshot of a video game

Description automatically generatedTiếp theo, chúng ta truy cập phần tử ở đầu ngăn xếp tức là 1 và đi đến các nút liền kề của nó. Vì 0 đã được truy cập, nên 2 là số được xét.

Nguồn: viblo.asia

Hình 2. 15 Bước thứ hai của ví dụ về DFS

A screenshot of a game

Description automatically generatedĐỉnh 2 có một đỉnh liền kề chưa được thăm là 4, vì vậy chúng ta thêm đỉnh đó vào vị trí đầu của ngăn xếp và duyệt nó.

Nguồn: viblo.asia  
Hình 2. 17 Bước thứ bốn

Nguồn: viblo.asia

Hình 2. 16 Bước thứ ba

A screenshot of a video game

Description automatically generated

Sau khi chúng ta duyệt phần tử 3 cuối cùng, nó không có bất kỳ nút liền kề nào chưa được duyệt, vì vậy chúng tôi đã hoàn thành tìm kiếm theo chiều sâu trong đồ thị trên.

A screenshot of a video game

Description automatically generated

Nguồn: viblo.asia

Hình 2. 18 Kết thúc

1. **Breadth First Search – BFS là gì?**

* Tìm kiếm theo chiều rộng (Breadth-First Search - BFS) là một thuật toán dùng để duyệt qua các cấu trúc dữ liệu như đồ thị hoặc cây. Phương pháp áp dụng cho cả cây và đồ thị với nguyên tắc cơ bản tương tự. Tuy nhiên, do đồ thị có thể chứa chu trình, nên để tránh duyệt lại các nút đã thăm, BFS sử dụng một mảng boolean để theo dõi trạng thái "đã truy cập" của từng đỉnh. Đây là cơ chế giúp đảm bảo mỗi nút chỉ được xử lý một lần trong quá trình duyệt. BFS sử dụng cấu trúc dữ liệu hàng đợi (queue), hoạt động theo nguyên tắc FIFO (First In, First Out), để đảm bảo khám phá các đỉnh theo từng lớp (level) từ gần đến xa. Một trong những ứng dụng quan trọng của BFS là tìm đường đi ngắn nhất trong các đồ thị không trọng số, nhờ khả năng duyệt đồng thời tất cả các cạnh từ một đỉnh đến các đỉnh kề.
* . Để triển khai thuật toán tìm kiếm theo chiều rộng (BFS) tiêu chuẩn, mỗi đỉnh của đồ thị được phân vào một trong hai trạng thái: đã thăm hoặc chưa thăm. Mục tiêu chính của thuật toán là đảm bảo rằng mỗi đỉnh chỉ được duyệt một lần, từ đó tránh việc lặp lại trong các chu trình.
* Thuật toán hoạt động như sau:
* Chọn một đỉnh bất kỳ làm điểm khởi đầu và thêm nó vào cuối hàng đợi.
* Lấy phần tử đầu tiên trong hàng đợi và đánh dấu đỉnh này là đã thăm bằng cách thêm vào danh sách đã duyệt.
* Tạo danh sách tất cả các đỉnh kề của đỉnh đang xét. Đối với mỗi đỉnh kề chưa được thăm, thêm chúng vào cuối hàng đợi.
* Lặp lại các bước 2 và 3 cho đến khi hàng đợi trống.

1. **Sử dụng DFS để kiểm tra tính liên thông của đồ thị**
   * **Ý tưởng:**

Thuật toán DFS duyệt qua tất cả các đỉnh và cạnh của đồ thị bằng cách bắt đầu từ một đỉnh ban đầu và tiếp tục đi sâu đến các đỉnh kề chưa được thăm. Nếu thuật toán có thể duyệt đến mọi đỉnh từ một đỉnh bất kỳ, đồ thị được coi là liên thông.

* + **Thực hiện:**

**Khởi tạo:**

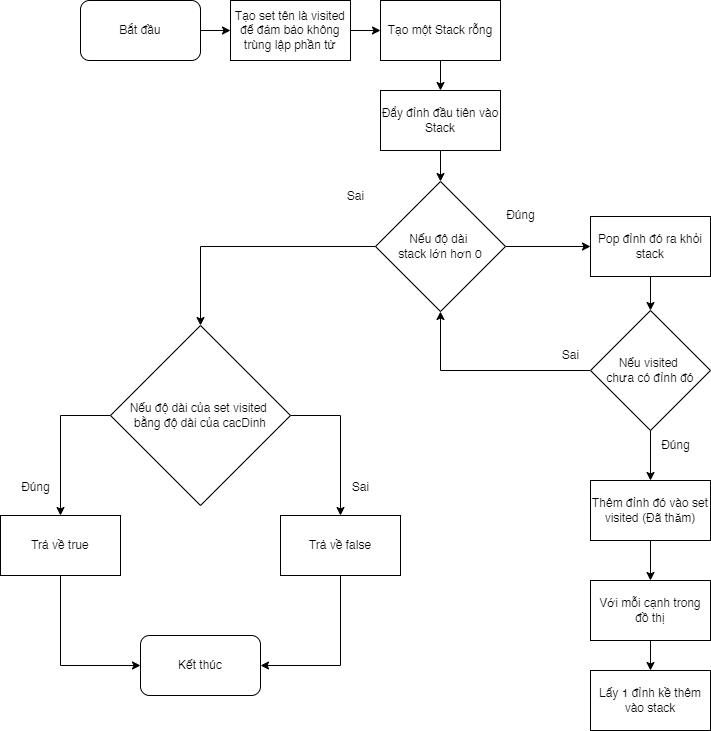
* Tạo một tập hợp rỗng daTham để lưu các đỉnh đã được duyệt.
* Tạo một ngăn xếp stack để quản lý quá trình duyệt (tuân theo nguyên tắc LIFO).

**Xét duyệt**

* Vòng lặp với điều kiện ngăn xếp không rỗng thì sẽ:
* Lấy đỉnh đầu tiên từ ngăn xếp.
* Nếu đỉnh này chưa được thăm, thêm nó vào tập daTham.
* Duyệt qua danh sách các cạnh của đồ thị, tìm tất cả các đỉnh kề với đỉnh vừa thêm mà chưa được thăm, và đẩy các đỉnh này vào ngăn xếp stack.

**Kết quả**

* Sau khi kết thúc quá trình duyệt, nếu số lượng đỉnh trong tập daTham bằng tổng số đỉnh của đồ thị, đồ thị là liên thông.
* Ngược lại, nếu có đỉnh nào không được thăm, đồ thị không liên thông.
  + **Lưu đồ giải thuật:**

**

Hình 2. 19 Lưu đồ thuật toán DFS

1. **Chu Trình Euler**
   1. **Tổng quan**

Những nguyên lý cơ bản của lý thuyết đồ thị chỉ mới được hình thành từthế kỷ XVIII, khởi nguồn từ công trình nghiên cứu của Leonhard Euler. Ông đã công bố một bài báo nổi tiếng giải quyết bài toán 7 cây cầu ở Königsberg, một vấn đề nổi bật trong lịch sử toán học.

Thành phố Königsberg, thuộc Đức (ngày nay là Kaliningrad, Nga), được chia thành bốn khu vực bởi các nhánh của sông Pregel, như hình minh họa. Năm 1736, thành phố có 7 cây cầu nối liền các khu vực này. Cư dân địa phương thắc mắc liệu có thể xuất phát từ một điểm trong thành phố, đi qua tất cả các cây cầu một lần duy nhất và quay lại điểm xuất phát hay không.

A diagram of a curved line

Description automatically generated with medium confidence

*Nguồn: Nguyễn Hữu Danh, 2023.*

Hình 2. 20 Bài toán 7 cây cầu

Euler đã thành công trong việc giải bài toán này bằng cách trừu tượng hóa bản đồ thành một đa đồ thị, trong đó các khu vực được biểu diễn bằng các đỉnh và các cây cầu là các cạnh nối. Bài toán thực tế này được tổng quát hóa thành một câu hỏi lý thuyết: Liệu có tồn tại một chu trình trong đa đồ thị đi qua tất cả các cạnh đúng một lần hay không?

Do **Leonhard Euler** là người đầu tiên đưa ra lời giải, loại chu trình đi qua tất cả các cạnh của đồ thị một lần duy nhất được đặt tên là chu trình Euler, như một sự tri ân dành cho nhà toán học vĩ đại trong lịch sử nhân loại.

* 1. **Khái niệm**

Chu trình đơn trong đồ thị G đi qua mỗi cạnh của đồ thị đúng một lần được gọi là chu trình Euler. Đường đi đơn trong G đi qua mỗi cạnh của nó đúng một lần được gọi là đường đi Euler. Đồ thị được gọi là đồ thị Euler nếu nó có chu trình Euler. Đồ thị có đường đi Euler được gọi là nửa Euler.

**Ví dụ**: Xét các đồ thị G1, G2, G3 sau:

A black and white image of a square with black dots and a square with black dots

Description automatically generated

Nguồn: Nguyễn Hữu Danh, 2023

Hình 2. 21 Đồ thị G1, G2, G3

Đồ thị G1 là đồ thị Euler vì nó có chu trình Euler a, e, c, d, e, b, a. Đồ thị G3 không có chu trình Euler nhưng chứa đường đi Euler a, c, d, e, b, d, a, b vì thế G3 là nửa Euler. G2 không có chu trình Euler cũng như đường đi Euler.

**Định lý:** Đồ thị (vô hướng) liên thông G là đồ thị Euler khi và chỉ khi mọi đỉnh của G đều có bậc chẵn.

**Bổ đề**: Nếu bậc của mỗi đỉnh của đồ thị G không nhỏ hơn 2 thì G chứa chu trình đơn.

**Hệ quả**: Đồ thị liên thông G là nửa Euler (mà không là Euler) khi và chỉ khi có đúng hai đỉnh bậc lẻ trong G.

* 1. **Giải thuật Fleury**

Thuật toán Fleury là một thuật toán mang tính lý thuyết nhưng lại khá đơn giản và được giới thiệu lần đầu tiên vào năm 1883. Thuật toán này áp dụng cho các đồ thị đã có sẵn thông tin về tính liên thông của tất cả các cạnh, đồng thời yêu cầu đồ thị phải có tối đa hai đỉnh bậc lẻ. Quá trình thực hiện thuật toán bắt đầu từ một đỉnh có bậc lẻ, hoặc nếu không có đỉnh bậc lẻ trong đồ thị, thuật toán sẽ lựa chọn ngẫu nhiên một đỉnh bất kỳ làm điểm xuất phát.

Thuật toán hoạt động theo cách lựa chọn cạnh tiếp theo trong quá trình di chuyển, với điều kiện rằng việc loại bỏ cạnh đó không làm đồ thị bị rời rạc, trừ trường hợp không còn cạnh nào thỏa mãn điều kiện này. Trong trường hợp duy nhất này, thuật toán sẽ chọn cạnh còn lại tại đỉnh hiện tại để tiếp tục. Sau khi chọn được cạnh, thuật toán sẽ di chuyển đến đỉnh còn lại của cạnh và loại bỏ cạnh đó khỏi đồ thị. Quá trình này sẽ tiếp tục cho đến khi không còn cạnh nào trong đồ thị.

Khi thuật toán hoàn thành, kết quả sẽ phụ thuộc vào số lượng đỉnh bậc lẻ trong đồ thị:

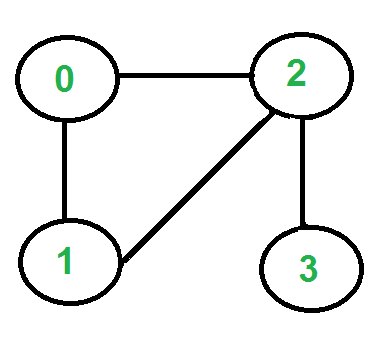
* Nếu đồ thị không có đỉnh bậc lẻ, dãy các cạnh được chọn sẽ tạo thành một chu trình Euler (tức là đồ thị có một chu trình đi qua tất cả các cạnh đúng một lần và quay lại điểm xuất phát).
* Nếu đồ thị có đúng hai đỉnh bậc lẻ, dãy các cạnh sẽ tạo thành một đường đi Euler (một đường đi đi qua tất cả các cạnh đúng một lần mà không quay lại điểm xuất phát).

Thuật toán Fleury đảm bảo rằng trong cả hai trường hợp trên, tất cả các cạnh của đồ thị sẽ được duyệt qua đúng một lần duy nhất, đồng thời quá trình lựa chọn các cạnh sẽ được thực hiện sao cho đồ thị không bị gián đoạn về tính liên thông, trừ khi không còn lựa chọn nào khác. Tuy nhiên, mặc dù thuật toán này mang lại kết quả chính xác, nhưng nó không được xem là tối ưu về mặt hiệu quả, đặc biệt khi số lượng cạnh trong đồ thị rất lớn.

**Tóm lại thuật toán hoạt động như sau:**

* Đảm bảo đồ thị không có đỉnh bậc 0 hoặc bậc lẻ.
* Nếu không có đỉnh bậc lẻ, thì bắt đầu duyệt ở bất kỳ đỉnh. Nếu có 2 đỉnh bậc lẻ, chọn một trong hai đỉnh để bắt đầu duyệt.
* Duyệt qua từng cạnh một. Phải luôn luôn ưu tiên chọn cạnh không phải là cầu, nếu không còn lựa chọn khác thì chọn cạnh cầu
* Dừng khi đã hết cạnh.

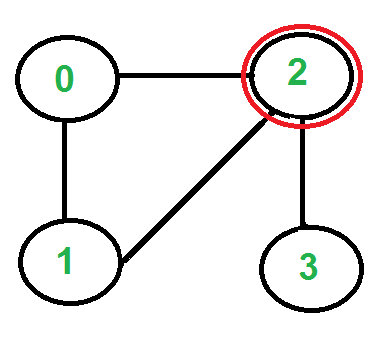
**Ví dụ:**



Nguồn: GeeksforGeeks

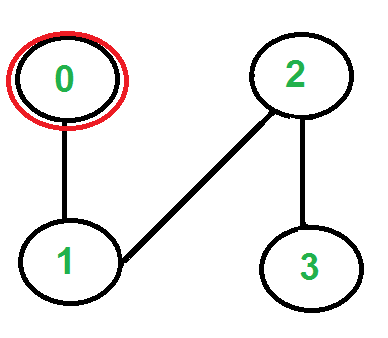
Hình 2. 22 Bước đầu tiên

* Ở đây có 2 đỉnh bậc lẻ là đỉnh “2” và đỉnh “3”, nên ta có thể bắt đầu ở một trong 2 đỉnh này. Ở ví dụ này sẽ chọn “2” để bắt đầu.



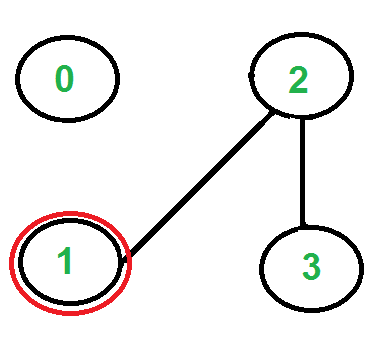
Nguồn: GeeksforGeeks  
Hình 2. 23 Bước thứ hai

* Từ đỉnh “2” ta có thể thấy có 3 cạnh đi ra từ nó. Đặc biệt là ở cạnh “2-3” chúng ta sẽ không chọn cạnh này vì cạnh này là cầu, một khi đi qua “3” thì sẽ không còn cạnh để quay lại. Nên ta có thể chọn 2 cạnh còn lại. Ở đây ta chọn “2-0”. Ta xoá cạnh này và di chuyển đến “0”. Chu trình hiện tại là: “2-0”.



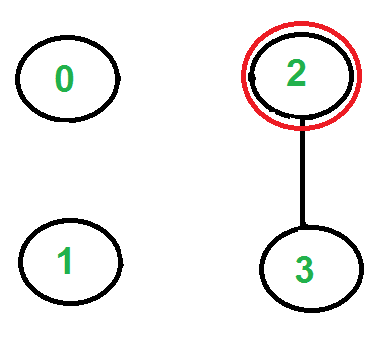
Nguồn: GeeksforGeeks  
Hình 2. 24 Bước thứ ba

* Ở đây chỉ còn 1 cạnh để chọn là “0-1” nên ta xoá cạnh này và di chuyển đến “1”. Chu trình hiện tại là: “0-2, 0-1”.



Nguồn: GeeksforGeeks  
Hình 2. 25 Bước thứ tư

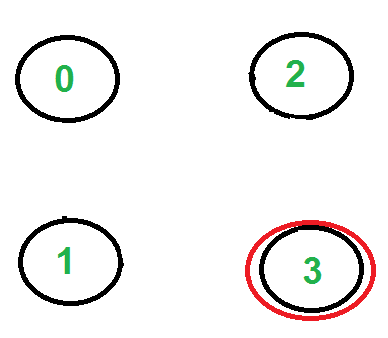
* Ở đây cũng chỉ có 1 cạnh để chọn là “1-2” nên ta làm điều tương tự là xoá cạnh này và di chuyển đến “2”. Chu trình hiện tại là: “0-2, 0-1, 1-2”.



Nguồn: GeeksforGeeks

Hình 2. 26 Bước thứ năm

* Tương tự như trên. Chọn cạnh “2-3”, xoá cạnh này, di chuyển đến “3”. Chu trình hiện tại là: “0-2, 0-1, 1-2, 2-3”.



Nguồn: GeeksforGeeks  
Hình 2. 27 Bước thứ sáu

* Đến đây thì không còn cạnh nào để đi nữa nên thuật toán sẽ dừng!
* Chu trình là: “0-2, 0-1, 1-2, 2-3”.

**Lưu đồ giải thuật**

**A diagram of a work flow

Description automatically generated**

Hình 2. 28 Lưu đồ giải thuật Fleury

1. **Môi trường sử dụng**

* Website tĩnh là một loại website mà nội dung của nó được lưu trữ sẵn trên máy chủ dưới dạng các tệp được gửi trực tiếp đến trình duyệt của người dùng mà không cần xử lý hay thay đổi nội dung dựa trên hành vi người dùng. Các website này chủ yếu được phát triển bằng ngôn ngữ HTML để tạo cấu trúc cơ bản, cùng với CSS để định hình giao diện và phong cách trình bày. Ngoài ra, hình ảnh, âm thanh hoặc video thường được thêm vào để tăng tính hấp dẫn và làm phong phú nội dung.
* Một đặc điểm nổi bật của website tĩnh là nội dung hiển thị giống nhau cho tất cả khách truy cập, bất kể thời gian hay ngữ cảnh truy cập. Điều này tương tự như việc cung cấp một tài liệu in ấn cho nhiều khách hàng: nội dung không thay đổi trừ khi được cập nhật thủ công. Ví dụ điển hình của website tĩnh bao gồm các trang quảng bá đơn giản như website tài liệu quảng cáo, website năm trang hoặc website thông tin công ty, nơi thông tin được cố định và không yêu cầu sự tương tác phức tạp từ phía người dùng.

1. **Công cụ hỗ trợ**
2. **Visual Studio Code (VS Code)**

* Visual Studio Code (VS Code) là một trình soạn thảo mã nguồn được phát triển bởi Microsoft, hỗ trợ trên các hệ điều hành Windows, macOS, và Linux. Được phát hành lần đầu vào ngày 29 tháng 4 năm 2015 tại hội nghị Build 2015, Visual Studio Code nhanh chóng trở thành công cụ phổ biến cho lập trình viên nhờ giao diện thân thiện, khả năng tùy chỉnh mạnh mẽ, và hỗ trợ đa ngôn ngữ lập trình.
* Công cụ này tích hợp khả năng gỡ lỗi (debugging) cho nhiều ngôn ngữ như JavaScript, TypeScript, Python, C++, và nhiều ngôn ngữ khác. Tính năng Syntax Highlighting giúp làm nổi bật cú pháp cho các ngôn ngữ phổ biến như HTML, CSS, JavaScript, Python, Ruby, và Rust, trong khi công nghệ IntelliSense hỗ trợ tự động hoàn thành thông minh, đưa ra gợi ý mã phù hợp cho các ngôn ngữ như TypeScript, HTML, CSS, và JSON. VS Code cũng cho phép sử dụng các đoạn mã mẫu (snippets) để tăng tốc độ lập trình. Bên cạnh đó, người dùng có thể tùy chỉnh giao diện và hành vi của phần mềm thông qua các file cấu hình JSON, từ thay đổi theme, gán phím tắt, đến cài đặt các tiện ích mở rộng phù hợp với nhu cầu lập trình. Kho tiện ích mở rộng của VS Code rất phong phú, với sự hỗ trợ của giao thức Language Server Protocol, giúp cải thiện phân tích mã nguồn thông qua linter và các công cụ tự động hóa. Một tính năng nổi bật khác là khả năng quản lý phiên bản (Version Control) tích hợp sẵn, cho phép lập trình viên dễ dàng theo dõi và kiểm soát các thay đổi trong dự án với các hệ thống như Git, SVN, và Perforce, tất cả đều được thực hiện trực tiếp từ giao diện của VS Code.

1. **GitHub**
   * GitHub là một dịch vụ cung cấp kho lưu trữ mã nguồn dựa trên hệ thống kiểm soát phiên bản Git, được phát triển với mục đích hỗ trợ cộng đồng lập trình viên trong việc quản lý, lưu trữ và chia sẻ mã nguồn. Ra mắt lần đầu vào tháng 4 năm 2008, GitHub nhanh chóng trở thành nền tảng phổ biến, cung cấp các giải pháp miễn phí cho các dự án mã nguồn mở và các tùy chọn trả phí cho doanh nghiệp hoặc cá nhân muốn bảo mật mã nguồn của mình.
   * GitHub là một nền tảng mạnh mẽ cho việc lưu trữ, quản lý mã nguồn và cộng tác trong phát triển phần mềm. Với khả năng tạo và quản lý các kho lưu trữ (repository) sử dụng Git, GitHub cho phép người dùng tải mã lên, theo dõi lịch sử phát triển, và quản lý các thay đổi của dự án một cách hiệu quả. Nền tảng này hỗ trợ cộng đồng lập trình viên thông qua các tính năng như thảo luận, xem xét mã nguồn, và đóng góp vào dự án thông qua pull request. Ngoài ra, GitHub còn cung cấp công cụ lưu trữ tài liệu và wiki trong các kho lưu trữ, giúp tổ chức thông tin tốt hơn. Tính năng theo dõi vấn đề (issue tracking) hỗ trợ quản lý các yêu cầu tính năng hoặc sửa lỗi trong dự án. Bên cạnh đó, GitHub cung cấp các dịch vụ bổ sung như **Gist**, một công cụ giống pastebin để chia sẻ các đoạn mã nhỏ, và **GitHub Pages**, giúp tạo và lưu trữ các trang web tĩnh trực tiếp từ kho lưu trữ công khai. Khả năng tích hợp và mở rộng của GitHub rất đa dạng, với sự hỗ trợ các công cụ kiểm tra tự động, linter, và hệ thống CI/CD thông qua plugin, cùng API mạnh mẽ để kết nối với các công cụ và hệ thống khác, làm cho GitHub trở thành một nền tảng toàn diện trong hệ sinh thái phát triển phần mềm.
2. **diagrams.net**
   * diagrams.net (trước đây là draw.io) là một phần mềm vẽ sơ đồ đa nền tảng được phát triển bằng HTML5 và JavaScript. Công cụ này hỗ trợ tạo ra nhiều loại sơ đồ như lưu đồ (flowcharts), khung giao diện (wireframes), sơ đồ UML, sơ đồ tổ chức, và sơ đồ mạng.
   * diagrams.net có thể chạy trực tuyến dưới dạng web app mà không yêu cầu đăng nhập hoặc đăng ký tài khoản, cho phép người dùng mở tệp từ ổ cứng và lưu trực tiếp về thiết bị. Đối với người dùng ngoại tuyến, diagrams.net cung cấp phiên bản ứng dụng dành cho các hệ điều hành Linux, macOS, và Windows, được xây dựng trên nền tảng Electron. Ngoài ra, công cụ hỗ trợ lưu trữ và xuất tệp dưới các định dạng phổ biến như PNG, JPEG, SVG, và PDF, giúp chia sẻ và sử dụng tài liệu một cách dễ dàng.
   * Theo đó phần mềm này còn tích hợp với các dịch vụ lưu trữ đám mây lớn như Dropbox, OneDrive, Google Drive, GitHub cho phép người dùng truy cập tệp từ bất kỳ thiết bị nào. Bên cạnh đó, công cụ hỗ trợ nhúng vào các nền tảng như NextCloud, MediaWiki, Notion, Atlassian Confluence, và Jira thông qua các plugin chuyên biệt, giúp tối ưu hóa khả năng cộng tác và quản lý sơ đồ trong các nhóm làm việc.
   * Theo các chuyên trang công nghệ như TechRadar và PCMag, diagrams.net được đánh giá là một giải pháp thay thế hiệu quả và tiết kiệm chi phí so với các công cụ phổ biến như Lucidchart, Microsoft Visio, và SmartDraw. Ưu điểm vượt trội của nó bao gồm giao diện thân thiện, dễ sử dụng, tính năng linh hoạt hỗ trợ đa dạng loại sơ đồ, và khả năng hoạt động mà không yêu cầu tài khoản, tăng cường tính riêng tư cho người dùng.
3. **Ngôn ngữ sử dụng**
4. **HTML**

**HTML** (viết tắt của **Hyper Text Markup Language**, hay **Ngôn ngữ Đánh dấu Siêu văn bản**) là một ngôn ngữ đánh dấu được thiết kế để tạo cấu trúc và trình bày nội dung cho các trang web trên World Wide Web. HTML là nền tảng cơ bản để xây dựng các tài liệu web và thường được kết hợp với các công nghệ khác như **CSS** (Cascading Style Sheets) để định dạng giao diện và **JavaScript** để thêm tính năng tương tác.

**Cấu Trúc và Chức Năng của HTML**

1. Cấu trúc ngữ nghĩa:

HTML mô tả cấu trúc của một trang web thông qua các phần tử HTML, được định nghĩa bằng các thẻ (tags) như:

* + <h1>: Tiêu đề chính.
  + <p>: Đoạn văn bản.
  + <a>: Liên kết.
  + <img>: Hình ảnh.

Các phần tử này giúp trình bày ngữ nghĩa của tài liệu, như tiêu đề, danh sách, trích dẫn, hoặc liên kết, và có thể bao gồm các phần tử khác như phần tử con.

1. Tính năng đa phương tiện:

HTML cho phép nhúng hình ảnh, video, âm thanh, và các đối tượng tương tác như biểu mẫu. Các trình duyệt web sử dụng mã HTML để hiển thị nội dung này thành các trang web đa phương tiện mà người dùng có thể tương tác.

1. Tương tác với trình duyệt:

Trình duyệt nhận tài liệu HTML từ máy chủ web hoặc từ kho lưu trữ cục bộ, sau đó render tài liệu thành giao diện trang web. Các thẻ HTML không hiển thị trực tiếp trên trang nhưng được trình duyệt sử dụng để xác định cấu trúc và nội dung.

1. Kết hợp với CSS và JavaScript:
   * CSS (Cascading Style Sheets): Quy định giao diện và bố cục của nội dung HTML. Kể từ năm 1997, W3C (World Wide Web Consortium) khuyến khích sử dụng CSS để thay thế HTML trong việc định dạng trực quan.
   * JavaScript: Được nhúng vào HTML để kiểm soát hành vi và thêm tính năng tương tác cho các trang web.
2. **CSS**

Trong lĩnh vực tin học, Cascading Style Sheets (CSS), hay còn gọi là ngôn ngữ định kiểu theo tầng, được sử dụng để mô tả cách trình bày của các tài liệu web được viết bằng HTML và XHTML. CSS là một công cụ mạnh mẽ cho phép lập trình viên tách biệt phần cấu trúc nội dung khỏi phần định dạng, mang lại sự linh hoạt và hiệu quả trong thiết kế web.

**Ứng dụng và phạm vi**

Ngoài việc được áp dụng phổ biến cho HTML và XHTML, CSS còn hỗ trợ các định dạng khác như XML, SVG (Scalable Vector Graphics), và XUL (XML User Interface Language). Điều này giúp CSS trở thành một chuẩn định dạng toàn diện cho các ngôn ngữ đánh dấu khác nhau.

**Ưu Điểm của CSS**

1. Tính linh hoạt và tái sử dụng:

CSS cho phép bạn áp dụng một kiểu dáng duy nhất cho nhiều trang web, giúp giảm thiểu công việc lặp lại và dễ dàng thực hiện các thay đổi toàn cục khi cần.

1. Cải thiện khả năng bảo trì**:**

Bằng cách tách biệt nội dung và định dạng, CSS giúp việc chỉnh sửa hoặc nâng cấp giao diện trang web trở nên đơn giản hơn mà không làm ảnh hưởng đến nội dung chính.

1. Hỗ trợ đa nền tảng và thiết bị:

CSS cung cấp các thuộc tính giúp tối ưu hóa hiển thị trên nhiều loại thiết bị, từ màn hình máy tính, điện thoại di động, đến máy in.

**Kết Hợp HTML với các công nghệ khác**

* HTML và CSS**:** HTML cung cấp cấu trúc cơ bản cho trang web, trong khi CSS định dạng giao diện, giúp trang web trở nên bắt mắt và dễ sử dụng hơn.
* SVG và XUL**:** CSS cung cấp khả năng định dạng các đối tượng đồ họa và giao diện người dùng dựa trên các ngôn ngữ đánh dấu này.

1. **JavaScript**

JavaScript là một ngôn ngữ lập trình dựa trên các ý niệm nguyên mẫu, được sử dụng rộng rãi cả trong phát triển web phía người dùng (client-side) lẫn phía máy chủ (server-side) thông qua Node.js. Ngôn ngữ này ban đầu được phát triển bởi **Brendan Eich** tại hãng Netscape Communications, với tên gọi đầu tiên là Mocha, sau đó đổi thành LiveScript, và cuối cùng được đặt tên là JavaScript nhằm tận dụng sự phổ biến của ngôn ngữ Java vào thời điểm đó.

Dù cú pháp của JavaScript tương tự với C, nó lại có nhiều điểm chung với Self hơn là Java. Các tệp mã nguồn JavaScript thường được lưu với phần mở rộng .js.

* **Ứng Dụng Của JavaScript**

**1. Trên Trình Duyệt**

JavaScript thường được sử dụng để tạo các trang web động và tăng cường trải nghiệm người dùng. Một số ứng dụng điển hình bao gồm:

* Tương tác với DOM (Document Object Model): Tạo hiệu ứng hình ảnh, thay đổi nội dung trang web, kiểm tra dữ liệu nhập vào của người dùng, và cập nhật dữ liệu tự động mà không cần tải lại trang.
* Công nghệ kết hợp**:** JavaScript kết hợp với HTML và CSS để xây dựng các công nghệ nổi bật như:
  + DHTML (Dynamic HTML): Tăng cường khả năng tương tác động.
  + Ajax: Gửi và nhận dữ liệu từ máy chủ mà không cần tải lại trang.
  + SPA (Single Page Applications): Xây dựng ứng dụng web tải nhanh và mượt mà.

**2. Bên Ngoài Trình Duyệt**

JavaScript không chỉ giới hạn trong trình duyệt mà còn được sử dụng trong nhiều môi trường khác như:

* Node.js (2009): Mang JavaScript từ trình duyệt đến môi trường máy chủ, cho phép xử lý backend nhanh chóng và hiệu quả.
* Adobe Acrobat: Hỗ trợ JavaScript để xử lý và tương tác trong tệp PDF.
* Hệ điều hành: Dashboard trên macOS 10.4 và Active Scripting của Microsoft đều sử dụng JavaScript để hỗ trợ các tính năng kịch bản động.
* Và nhiều môi trường khác nữa…

1. **Thư viện sử dụng**
   1. **jQuery**

jQuery là một thư viện JavaScript mã nguồn mở, được thiết kế để đơn giản hóa các thao tác với HTML DOM (Document Object Model), xử lý sự kiện, tạo hoạt ảnh CSS, và thực hiện các yêu cầu Ajax. Được phát hành theo giấy phép MIT License, jQuery miễn phí và có thể sử dụng tự do trong các dự án cá nhân lẫn thương mại.

* **Tính Phổ Biến và Ứng Dụng**
* Tính đến tháng 5 năm 2019, jQuery đã được triển khai trên 73% trong số 10 triệu trang web phổ biến nhất, cho thấy sự thống trị của nó trong hệ sinh thái web.
* Phân tích từ các công cụ theo dõi web chỉ ra rằng jQuery là thư viện JavaScript được sử dụng rộng rãi nhất trên thế giới, với tỷ lệ sử dụng cao hơn ít nhất 3 đến 4 lần so với bất kỳ thư viện JavaScript nào khác.
* **Tính năng chính của jQuery**

jQuery là một thư viện JavaScript mạnh mẽ, cung cấp cú pháp ngắn gọn và thân thiện, giúp lập trình viên dễ dàng chọn, sửa đổi, và thao tác các phần tử HTML. Thư viện này hỗ trợ quản lý sự kiện trên trang web một cách hiệu quả, đảm bảo tính đồng nhất trên các trình duyệt khác nhau. Ngoài ra, jQuery cho phép gửi và nhận dữ liệu từ máy chủ mà không cần tải lại trang, góp phần xây dựng các ứng dụng web động và mượt mà hơn. Không chỉ vậy, jQuery còn giúp lập trình viên dễ dàng thêm các hiệu ứng động vào trang web, từ các chuyển đổi đơn giản đến các hiệu ứng phức tạp, tạo nên trải nghiệm trực quan và hấp dẫn. Quan trọng hơn, jQuery giúp giải quyết vấn đề không tương thích giữa các trình duyệt, mang lại trải nghiệm nhất quán cho người dùng trên mọi nền tảng.

* 1. **Bootstrap**

Bootstrap là một framework CSS miễn phí và mã nguồn mở chuyên dụng cho phát triển web front-end, giúp đơn giản hóa quá trình tạo và thiết kế trang web. Framework này cung cấp một tập hợp các mẫu thiết kế kiểu chữ, biểu mẫu, nút, thanh điều hướng, và các thành phần giao diện người dùng khác bằng ngôn ngữ HTML, CSS, và JavaScript.

* **Tính năng chính Của Bootstrap:**
* Bootstrap cung cấp các định nghĩa kiểu mẫu có sẵn cho toàn bộ các thành phần HTML như văn bản, bảng biểu, biểu mẫu, nút nhấn, thanh điều hướng, v.v. Điều này giúp các nhà phát triển tiết kiệm thời gian và công sức trong việc thiết kế và định dạng trang web.
* Bootstrap hỗ trợ CSS Flexbox và Grid System, cho phép tạo các bố cục linh hoạt và responsive. Người dùng có thể dễ dàng tạo các thiết kế phù hợp với nhiều kích thước màn hình từ nhỏ (dưới 576px) đến lớn hơn 1200px. Điều này bao gồm định nghĩa các vùng chứa với chiều rộng cố định hoặc chiều rộng linh hoạt, giúp nội dung tự động thay đổi kích thước theo từng thiết bị.
* Bootstrap đi kèm với một số thành phần JavaScript có sẵn, bao gồm hộp thoại, tooltip, modal, thanh tiến trình, trình đơn thả xuống điều hướng, và băng chuyền. Các thành phần này không yêu cầu thư viện như jQuery, mà chỉ cần gọi các plugin của Bootstrap trực tiếp từ các tệp JavaScript được thêm vào trang web.
* Bootstrap cung cấp phiên bản biên dịch sẵn với tệp CSS và JavaScript để dễ dàng thêm vào bất kỳ dự án nào. Tuy nhiên, framework này cũng hỗ trợ người dùng triển khai các tùy chỉnh sâu hơn thông qua Sass (công cụ tiền xử lý CSS) để thay đổi kiểu dáng, áp dụng chủ đề, và loại bỏ các thành phần không cần thiết.
* Bootstrap giúp duy trì giao diện người dùng thống nhất trên mọi trình duyệt và thiết bị. Điều này đảm bảo rằng người dùng sẽ có trải nghiệm nhất quán, từ các thiết bị di động đến máy tính để bàn.
* **Một vài thành phần của Bootstrap**
* Bootstrap cung cấp 5 loại container với các kích thước cố định khác nhau tùy thuộc vào kích thước của màn hình. Từ nhỏ hơn 576px, 576–768px, 768–992px, 992–1200px, đến lớn hơn 1200px.
* Với hệ thống Grid System của Bootstrap, người dùng có thể tạo các cột và hàng linh hoạt bằng cách sử dụng các lớp CSS. Các lớp này giúp sắp xếp và căn chỉnh nội dung một cách dễ dàng, bao gồm dòng lưới (row) và cột (col).
* Bootstrap cung cấp các mẫu thiết kế sẵn như các thanh điều hướng, nút nhấn, mẫu form, hộp thoại, và các kiểu biểu mẫu tương tác. Người dùng có thể tùy chỉnh hoặc sử dụng trực tiếp các mẫu này vào dự án của mình.
* **Ứng dụng thực tiễn**
* Tạo trang web thông tin: Các trang web đơn giản như blog, portfolio cá nhân, hoặc trang quảng bá công ty thường sử dụng Bootstrap để tạo giao diện đẹp mắt, dễ điều hướng.
* Ứng dụng doanh nghiệp: Bootstrap cũng được sử dụng để xây dựng các ứng dụng web doanh nghiệp phức tạp hơn, chẳng hạn như quản lý nội dung (CMS), quản lý tác vụ (task management), và quản lý quan hệ khách hàng (CRM).
* Ứng dụng di động: Bootstrap có khả năng tạo thiết kế responsive, hỗ trợ tối ưu hóa cho thiết bị di động, giúp các trang web hiển thị tốt trên thiết bị nhỏ như điện thoại thông minh, máy tính bảng.
  1. **p5**

p5.js là một thư viện JavaScript thân thiện, được thiết kế để giúp người dùng học lập trình và tạo nghệ thuật trực tuyến. Thư viện này miễn phí và mã nguồn mở, được phát triển bởi một cộng đồng toàn diện và hỗ trợ, với mục tiêu mang lại một công cụ dễ tiếp cận cho mọi đối tượng, từ nghệ sĩ, nhà thiết kế, người mới bắt đầu cho đến các giáo viên.

* **Mục đích và đặc điểm**
* Được phát triển để giúp người học dễ dàng khám phá các khái niệm lập trình thông qua việc tạo ra các tác phẩm nghệ thuật trực quan, giúp kết nối lập trình và nghệ thuật một cách trực quan và dễ hiểu.
* Có một cộng đồng đông đảo, chào đón tất cả mọi người, không phân biệt nền tảng hay kinh nghiệm. Thư viện này thúc đẩy việc học tập và sáng tạo, mở rộng khả năng cho các nghệ sĩ, nhà thiết kế và lập trình viên.
* Cú pháp đơn giản và dễ làm quen, đặc biệt phù hợp cho người mới bắt đầu. Nó cung cấp một nền tảng tuyệt vời để các nhà sáng tạo khám phá và thử nghiệm các ý tưởng nghệ thuật bằng cách lập trình.
* **Ứng dụng**
* Tạo nghệ thuật số: p5.js cho phép người dùng tạo ra các tác phẩm nghệ thuật động, bao gồm vẽ đồ họa 2D, hiệu ứng hình ảnh và âm thanh tương tác.
* Giáo dục lập trình: Với mục tiêu là công cụ giáo dục, p5.js được sử dụng rộng rãi trong các lớp học lập trình, giúp học sinh và sinh viên hiểu được cách lập trình thông qua việc tạo ra những sản phẩm nghệ thuật thú vị.
* Tương tác và hoạt hình: p5.js cũng hỗ trợ việc tạo ra các tương tác và hoạt ảnh, cho phép người dùng tạo ra các trải nghiệm phong phú và thú vị trên web.
* **Miễn phí**

p5.js là một dự án mã nguồn mở, với tất cả tài nguyên và công cụ đều có sẵn miễn phí. Điều này khuyến khích cộng đồng đóng góp và cải thiện thư viện, tạo ra một công cụ mạnh mẽ và linh hoạt hơn.

**CHƯƠNG III. KẾT QUẢ**

A screenshot of a computer

Description automatically generated

Hình 3. 1 Giao diện chính

Chương trình bao gồm ba chức năng chính, đó là:

1. **Vẽ bằng dữ liệu tải lên của tệp (txt)**

**A screenshot of a computer

Description automatically generated**

Hình 3. 2 Giao diện chức năng này

Chức năng này cho phép người dùng tải lên một tệp văn bản (.txt) chứa dữ liệu đồ thị và vẽ đồ thị tương ứng. Cách sử dụng như sau:

* **Tải lên tệp:**
* Người dùng bấm vào nút **Tải lên** nằm ở giữa màn hình để chọn tệp **.txt** từ thiết bị của mình.

A screenshot of a computer

Description automatically generated

Hình 3. 3 Chức năng chọn tệp tin

* Định dạng của tệp phải tuân thủ quy tắc sau:
  + **Dòng đầu tiên**: Chứa **số đỉnh** của đồ thị (ví dụ: 4, nghĩa là đồ thị có 4 đỉnh).
  + **Các dòng tiếp theo**: Chứa các giá trị nhị phân đại diện cho ma trận kề của đồ thị, mỗi dòng tương ứng với các cạnh kết nối của đỉnh với các đỉnh còn lại.

Ví dụ về tệp có định dạng hợp lệ:

4

1 1 1 0

1 0 1 1

1 1 1 0

0 1 0 0

* **Chọn lại tệp:**

Nếu muốn chọn lại tệp khác, người dùng có thể bấm vào nút *Tải lại* dưới màn hình.

A screenshot of a computer

Description automatically generated

Hình 3. 4 Chức năng làm mới tệp

* **Về trang chủ:**

Để trở về trang chủ, người dùng chỉ cần bấm vào nút **Trang chủ** ở góc trên bên trái của màn hình.

*A yellow and white rectangle

Description automatically generated*

Hình 3. 5 Trở về trang chủ

Sau khi đã chọn tệp mong muốn thì chương trình sẽ bắt đầu thực hiện công việc và cho ra kết quả mong muốn.

* **Kết quả cuối cùng:**

A screenshot of a cell phone

Description automatically generated

Hình 3. 6 Kết quả sau khi tải lên

1. **Vẽ thủ công**

A screenshot of a computer

Description automatically generated

Hình 3. 7 Giao diện chính của chức năng này

Chức năng này cho phép người dùng tự tay vẽ một đồ thị bằng cách nhập dữ liệu kết nối giữa các đỉnh. Dưới đây là cách thực hiện từng bước:

* **Nhập số đỉnh:**
* Người dùng bắt đầu bằng cách nhập vào số lượng đỉnh mong muốn trong ô nhập liệu tương ứng.
* Đây là bước cần thiết để xác định quy mô đồ thị.

A screenshot of a computer

Description automatically generated

Hình 3. 8 Nhận số đỉnh

* **Tạo ô nhập liệu**
* Sau khi nhập số đỉnh, nhấn nút "Tạo".
* Chương trình sẽ hiển thị các ô nhập liệu tương ứng với số đỉnh đã nhập, cho phép người dùng thiết lập các kết nối giữa các đỉnh.

A screenshot of a computer

Description automatically generated

Hình 3. 9 Tạo ra các ô nhập dữ liệu

* **Nhập đỉnh kết nối**
* Với các ô nhập liệu đã được tạo, người dùng điền thông tin về các đỉnh mà mình muốn kết nối.
* Ví dụ: Nếu muốn kết nối đỉnh A với đỉnh B, nhập các giá trị tương ứng vào ô nhập liệu.

A screenshot of a computer

Description automatically generated

Hình 3. 10 Nhập đỉnh kết nối vào các ô nhập liệu

A screenshot of a computer

Description automatically generated

Hình 3. 11 Ví dụ khi nhập dữ liệu

* **Thêm đỉnh kết nối (Tuỳ chọn)**
* Nếu muốn thêm nhiều kết nối hơn, người dùng có thể nhấn nút "Thêm đỉnh" để tạo ra thêm các ô nhập liệu.
* Tính năng này hữu ích trong trường hợp người dùng cần một đồ thị nhiều cạnh hơn nữa.

A screenshot of a quiz

Description automatically generated

Hình 3. 12 Thêm đỉnh muốn kết nối

A screenshot of a test

Description automatically generated

Hình 3. 13 Sau khi nhấn “Thêm đỉnh”

* **Vẽ đồ thị**
* Khi đã hoàn tất việc nhập liệu và kết nối các đỉnh, người dùng nhấn nút "Vẽ".
* Hệ thống sẽ xử lý dữ liệu và hiển thị đồ thị tương ứng với các kết nối đã được chỉ định.

A screenshot of a quiz

Description automatically generated

Hình 3. 14 Vẽ ra đồ thị mong muốn

* **Kết quả cuối cùng:**

A screenshot of a cell phone

Description automatically generated

Hình 3. 15 Kết quả

1. **Vẽ ngẫu nhiên**

A screenshot of a computer

Description automatically generated

Hình 3. 16 Giao diện chính

Chức năng này được thiết kế để hỗ trợ người dùng tạo ra các đồ thị ngẫu nhiên một cách nhanh chóng và tiện lợi. Với thao tác đơn giản, người dùng chỉ cần nhập số đỉnh mong muốn và nhấn vào nút "Vẽ", hệ thống sẽ tự động vẽ ra một đồ thị ngẫu nhiên, bao gồm các đỉnh và cạnh được kết nối ngẫu nhiên theo quy luật được lập trình sẵn.

A screenshot of a computer

Description automatically generated

Hình 3. 17 Tiến hành tạo đồ thị ngẫu nhiên

* **Kết quả**

A screenshot of a cell phone

Description automatically generated

Hình 3. 18 Kết quả khi được tạo ngẫu nhiên

# KẾT LUẬN – ĐÁNH GIÁ

* **Kết quả đạt được**

Bài niên luận này đã thành công trong việc giải quyết bài toán tìm chu trình Euler trên một đồ thị vô hướng. Các chức năng chính được thiết kế và triển khai hoạt động ổn định, đảm bảo tính chính xác cao và ít khả năng phát sinh lỗi trong quá trình sử dụng. Giao diện hệ thống được xây dựng trực quan, dễ thao tác, và kết quả đầu ra được xử lý chính xác, đáp ứng tốt yêu cầu của người dùng.

* **Thu hoạch chuyên môn – Kinh nghiệm**

Thông qua quá trình thực hiện, đã giúp nắm vững các khái niệm cơ bản về đồ thị, đặc biệt là các điều kiện tồn tại chu trình Euler. Ngoài ra, việc triển khai thuật toán DFS và Fleury giúp hiểu sâu hơn về cách tối ưu hóa xử lý dữ liệu đồ thị. Kỹ năng lập trình và sử dụng các công cụ hỗ trợ như Visual Studio Code, GitHub, và thư viện JavaScript cũng được cải thiện đáng kể.

* **Ưu điểm, Hạn chế - Nguyên nhân**
* **Ưu điểm**

Chương trình được thiết kế với tính ổn định cao, đảm bảo hoạt động mượt mà trong hầu hết các trường hợp sử dụng. Giao diện trực quan, rõ ràng, giúp người dùng dễ dàng thao tác và hiểu cách hoạt động của hệ thống. Đặc biệt, kết quả đầu ra chính xác, đáp ứng yêu cầu về tính toán và phân tích, là một điểm mạnh quan trọng. Ngoài ra, chương trình tích hợp ba chức năng chính trong một bài toán gồm tạo đồ thị ngẫu nhiên, vẽ đồ thị thủ công, và nhập từ file, mang lại sự linh hoạt và đa dạng trong việc sử dụng.

* **Hạn chế**

Tuy nhiên, chương trình vẫn tồn tại một số hạn chế cần cải thiện. Khi xử lý các đồ thị có kích thước quá lớn, hiệu suất có thể giảm đáng kể, làm chậm quá trình thực thi. Đối với dữ liệu nhập thủ công, người dùng phải tuân thủ các quy định định dạng cụ thể, điều này đôi lúc gây bất tiện và làm tăng nguy cơ nhập sai. Một hạn chế kỹ thuật khác là thuật toán chưa được tối ưu hóa hoàn toàn; việc sử dụng nhiều vòng lặp có thể dẫn đến tình trạng quá tải hoặc treo chương trình trong các trường hợp đặc biệt.

* **Hướng phát triển**

Một trong những hướng phát triển là bổ sung thêm các cách biểu diễn đồ thị, chẳng hạn như danh sách cạnh hoặc danh sách kề, giúp linh hoạt hơn trong việc nhập liệu và xử lý dữ liệu. Hơn nữa, việc phát triển chức năng hỗ trợ vẽ đồ thị có hướng sẽ mở rộng phạm vi ứng dụng của chương trình. Cuối cùng, tối ưu hóa các thuật toán hiện tại bằng cách giảm bớt vòng lặp không cần thiết hoặc áp dụng các thuật toán nâng cao sẽ cải thiện đáng kể hiệu suất, đặc biệt khi xử lý các đồ thị lớn và phức tạp.

# TÀI LIỆU THAM KHẢO

**Tài liệu tiếng việt:**

1. Nguyễn Hữu Danh. Toán Rời Rạc 2. Chương 1, 2, 3 và 4. Khoa Kỹ Thuật – Công Nghệ, Đại Học Tây Đô, 2023.
2. Nguyễn Chí Cường. Cấu trúc dữ liệu & Giải thuật. Khoa Kỹ thuật – Công nghệ, Đại Học Tây Đô, 2023.
3. Nguyễn Chí Cường. Lập trình căn bản. Khoa Kỹ thuật – Công nghệ, Đại Học Tây Đô, 2023.
4. Lâm Tấn Phương. Giáo trình Lý thuyết Thiết kế và lập trình Web, Khoa Kỹ thuật – Công nghệ, Đại Học Tây Đô, 2024.

**Tài liệu từ Internet:**

1. GeeksForGeeks. Fleury’s Algorithm for printing Eulerian Path or Circuit. https://www.geeksforgeeks.org/fleurys-algorithm-for-printing-eulerian-path/ (Tháng 12/2024)
2. Mozilla. JavaScript reference. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference (Tháng 12/2024)
3. Bootstrap. Bootstrap Documents. getbootstrap.com/docs/5.3 (Tháng 12/2024)
4. OCLPhase2. Fleury Algorithm. youtube.com/watch?v=HXNyr2N3euc (Tháng 12/2024)
5. Thái Thanh Hải. Data Structure & Algorithm - Graph Algorithms - Depth First Search (DFS). viblo.asia/p/data-structure-algorithm-graph-algorithms-depth-first-search-dfs-qPoL7zyXJvk (Tháng 12/2024)
6. Thái Thanh Hải. Data Structure & Algorithm - Graph Algorithms - Breadth First Search (BFS). viblo.asia/p/data-structure-algorithm-graph-algorithms-breadth-first-search-bfs-gwd43kMM4X9 (Tháng 12/2024)
7. Viblo. Biểu diễn đồ thị trên máy tính. viblo.asia/p/bieu-dien-do-thi-tren-may-tinh-XL6lABrD5ek (Tháng 12/2024)
8. Viblo. Đồ thị Euler và Chu trình Euler. viblo.asia/p/do-thi-euler-va-chu-trinh-euler-WAyK8Bj6lxX (Tháng 12/2024)
9. Wikipedia. Đường đi Euler. vi.wikipedia.org/wiki/Đường\_đi\_Euler (Tháng 12/2024)
10. Wikipedia. Ma trận kề. vi.wikipedia.org/wiki/Ma \_trận\_kề (Tháng 12/2024)