TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

MÔN LÝ THUYẾT ĐỒ THỊ VÀ ỨNG DỤNG

BÁO CÁO ĐỒ ÁN

CHƯƠNG TRÌNH TÌM THÀNH PHẦN LIÊN THÔNG CỦA ĐỒ THỊ VÀ MINH HỌA THUẬT TOÁN DFS, BFS

Giảng viên hướng dẫn: *TS. Nguyễn Viết Hưng*

ThS. Nguyễn Phương Nam

Sinh viên thực hiện: *Nguyễn Thái Bình - 49.01.104.011*

Trần Lê Triều Dương - 49.01.104.026

Ngô Quang Khánh - 49.01.104.069

Nguyễn Hữu Minh Quân - 49.01.104.120

Nguyễn Ngọc Phú Tỷ - 49.01.104.172

Lớp: *2411COMP170102*

Thành phố Hồ Chí Minh, 2024

LỜI CẢM ƠN

Nhóm em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Viết Hưng và thầy Nguyễn Phương Nam, đã dành thời gian quý báu của mình để hỗ trợ tận tình, hướng dẫn cho nhóm chúng em trong suốt quá trình thực hiện đồ án cuối kỳ này. Nhóm chúng em rất trân trọng kiến thức mà hai thầy đã truyền tải cho chúng em trong suốt thời gian học tập vừa qua, chúng em tin rằng đây sẽ là bước đệm vững chắc để chúng em phát triển trong tương lai.

Trong quá trình thực hiện đồ án, mặc dù đã cố gắng hết sức, nhưng chúng em cũng không thể tránh khỏi những thiếu sót. Nhóm em rất mong nhận được sự góp ý quý báu từ thầy để có thể hoàn thiện hơn trong những nghiên cứu và công việc sau này. Chúng em chân thành cảm ơn!

Nhóm CB - Không Tên

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

CHƯƠNG 1: TỔNG QUAN

1.1. Giới thiệu đề tài

Chương trình tìm thành phần liên thông của đồ thị và minh họa thuật toán DFS, BFS được thiết kế nhằm minh họa trực quan các thuật toán cơ bản trên đồ thị vô hướng, bao gồm DFS (Duyệt theo chiều sâu), BFS (Duyệt theo chiều rộng) và tìm thành phần liên thông. Thông qua chương trình, ta có thể theo dõi chi tiết từng bước thực hiện của thuật toán, từ việc thăm đỉnh, duyệt qua các cạnh đến xác định các nhóm đỉnh liên thông trong đồ thị.

Với giao diện trực quan và sinh động của chương trình không chỉ giúp đơn giản hóa việc tiếp cận các thuật toán phức tạp mà còn làm nổi bật mối quan hệ giữa các thành phần của đồ thị. Đây không chỉ là công cụ hỗ trợ học tập mà còn là nền tảng để phát triển ứng dụng khác, đóng vai trò quan trọng trong việc giải quyết nhiều bài toán thực tế như tìm đường đi trong bản đồ, xác định đường đi ngắn nhất,…

1.2. Bảng chia việc

Thành viên	Công việc	Mức độ hoàn thành
Nguyễn Thái Bình	- Code CSS, JS hiển thị tổng quan. - Code chức năng hướng dẫn.	100%
Trần Lê Triều Dương	- Code CSS, JS hiển thị stack và danh sách kề. - Viết báo cáo.	100%
Ngô Quang Khánh	- Code thuật toán Tìm thành phần liên thông. - Code chức năng điều chỉnh thời gian duyệt. - Viết báo cáo.	100%
Nguyễn Hữu Minh Quân	- Code CSS, JS cho các button và giao diện tổng quan. - Code chức năng chuyển đổi hiện thị khung cấu trúc dữ liệu. - Code thuật toán BFS. - Viết báo cáo.	100%
Nguyễn Ngọc Phú Tỷ	- Thiết kế chương trình. - Code sườn chương trình, CSS, JS giao diện cơ bản. - Code thuật toán DFS. - Tạo github, drive lưu trữ. - Viết báo cáo.	100%

Bảng 1. Bảng phân chia việc

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Đồ thị và các thuật toán liên quan

Đồ thị là một cấu trúc dữ liệu bao gồm tập hợp các đỉnh (vertices) và các cạnh (edges) kết nối các đỉnh với nhau. Đồ thị có thể được biểu diễn dưới nhiều dạng, phổ biến nhất là danh sách kề (Adjacency List) và ma trận kề (Adjacency Matrix). Có hai loại đồ thị chính: đồ thị vô hướng (Undirected Graph), trong đó các cạnh không có hướng, và đồ thị có hướng (Directed Graph), trong đó các cạnh có hướng xác định. Ngoài ra, đồ thị còn có thể phân loại dựa trên trọng số (Weighted Graph) hoặc không trọng số (Unweighted Graph). Đồ thị là nền tảng quan trọng trong nhiều ứng dụng, như mạng máy tính, bản đồ giao thông, và trí tuệ nhân tạo.

DFS - Depth First Search (tìm kiếm theo chiều sâu) là một thuật toán duyệt hoặc tìm kiếm trong đồ thị bằng cách đi sâu vào từng nhánh trước khi quay lại và khám phá các nhánh khác. Thuật toán này sử dụng ngăn xếp (stack) trong cách triển khai đệ quy hoặc vòng lặp. DFS thường được sử dụng để kiểm tra tính liên thông của đồ thị, tìm chu trình, hoặc giải các bài toán như tìm đường đi giữa hai đỉnh.

BFS - Breadth First search (tìm kiếm theo chiều rộng) là một thuật toán duyệt đồ thị theo từng mức, nghĩa là các đỉnh lân cận của một đỉnh sẽ được duyệt hết trước khi chuyển sang các đỉnh ở mức tiếp theo. BFS thường được triển khai bằng cách sử dụng hàng đợi (queue) và rất hiệu quả để tìm đường đi ngắn nhất trong đồ thị không trọng số. BFS thường được ứng dụng trong các bài toán tìm kiếm mạng lưới, khám phá thành phần liên thông, hoặc kiểm tra tính hai phía của đồ thị.

Thành phần liên thông trong một đồ thị vô hướng là tập hợp các đỉnh sao cho có đường đi giữa bất kỳ hai đỉnh nào trong tập hợp, và tập hợp đó không thể mở rộng thêm bằng cách thêm vào một đỉnh khác. Để tìm tất cả các thành phần liên thông, người ta thường sử dụng DFS hoặc BFS. Thuật toán bắt đầu từ một đỉnh chưa được thăm, duyệt qua toàn bộ các đỉnh liên thông với nó và đánh dấu chúng thuộc cùng một thành phần. Sau đó, thuật toán lặp lại cho các đỉnh chưa được thăm đến khi tất cả các đỉnh trong đồ thị đều được đánh dấu. Đây là một bước quan trọng trong việc phân tích cấu trúc đồ thị và xác định tính kết nối.

2.2. Ngôn ngữ và công cụ phát triển

HTML (HyperText Markup Language) là ngôn ngữ đánh dấu tiêu chuẩn để tạo dựng cấu trúc của các trang web. Nó cho phép xác định các thành phần cơ bản như

tiêu đề, đoạn văn, danh sách, bảng, và các khu vực hiển thị khác. Mỗi trang web đều bắt đầu bằng HTML, đóng vai trò nền tảng để trình duyệt web có thể hiểu và hiển thị nội dung một cách chính xác. HTML có các thẻ đặc biệt giúp phân loại và tổ chức các phần tử trong trang, từ đó xây dựng nên cấu trúc cho giao diện và nội dung của trang web.

CSS (Cascading Style Sheets) là ngôn ngữ dùng để định dạng và tạo kiểu cho các thành phần HTML. CSS cho phép điều chỉnh màu sắc, kích thước, vị trí, và bố cục của các phần tử trên trang web. Bên cạnh đó, CSS cũng cung cấp khả năng tạo các hiệu ứng động, như chuyển động, thay đổi màu sắc hoặc kích thước khi người dùng tương tác, từ đó làm cho giao diện trang web trở nên sinh động và dễ sử dụng hơn. Việc phân tách CSS khỏi HTML giúp tối ưu hóa và dễ dàng bảo trì mã nguồn của trang web.

JavaScript là ngôn ngữ lập trình giúp làm cho các trang web trở nên tương tác và động. JavaScript có thể thay đổi nội dung của trang web mà không cần tải lại toàn bộ trang, cho phép người dùng có một trải nghiệm mượt mà và linh hoạt hơn. Ngôn ngữ này không chỉ xử lý dữ liệu và logic phức tạp, mà còn có thể thao tác với các phần tử HTML và CSS, thực hiện các tác vụ như gửi yêu cầu tới máy chủ, cập nhật nội dung trang, hay xử lý các sự kiện người dùng như nhấp chuột, di chuyển chuột, hay nhập liệu.

D3.js (Data-Driven Documents) là thư viện JavaScript được sử dụng để trực quan hóa dữ liệu. Thư viện này cho phép người dùng dễ dàng tạo ra các biểu đồ, đồ thị và các phần tử đồ họa khác trên nền tảng web thông qua SVG, HTML và CSS. D3.js hỗ trợ việc hiển thị dữ liệu dưới dạng đồ họa tương tác, giúp người dùng dễ dàng phân tích và tương tác với các bộ dữ liệu phức tạp. D3.js cung cấp nhiều tính năng mạnh mẽ, từ việc vẽ đồ thị đơn giản đến việc thực hiện các thao tác phức tạp với dữ liệu lớn.

SweetAlert2 là một thư viện JavaScript giúp tạo ra các hộp thoại thông báo đẹp mắt và dễ sử dụng. Thư viện này cho phép người phát triển web hiển thị các thông báo, cảnh báo hoặc thông tin quan trọng với thiết kế hiện đại và linh hoạt. Các hộp thoại của SweetAlert2 có thể dễ dàng tùy chỉnh về màu sắc, nội dung, và cách thức hiển thị, mang lại cho người dùng những trải nghiệm trực quan và dễ hiểu trong quá trình tương tác với trang web.

CHƯƠNG 3: THIẾT KẾ CHƯƠNG TRÌNH

3.1. Chương trình chính

Chương trình được phát triển bằng HTML, CSS và JavaScript, với mục đích minh họa giúp người dùng hình dung được các thuật toán trên đồ thị. Giao diện chính của chương trình có ba khung hiển thị. Đầu tiên là khung hiển thị đồ thị, nơi người dùng có thể thấy trực quan các đỉnh và cạnh của đồ thị. Nếu người dùng không tạo đồ thị riêng, chương trình sẽ tự động cung cấp một đồ thị mặc định. Đồ thị có thể di chuyển khi người dùng sử dụng con trỏ chuột để nắm kéo các đỉnh, giúp dễ dàng điều chỉnh vị trí của các đỉnh và vị trí các thành phần liên thông phù hợp với vị trí mà người dùng mong muốn.

Khung thứ hai hiển thị cấu trúc dữ liệu đang được sử dụng trong thuật toán. Khi người dùng chọn và chạy thuật toán, khung này sẽ cập nhật để hiển thị các thay đổi trong cấu trúc dữ liệu qua từng bước. Ví dụ, trong thuật toán DFS, người dùng sẽ thấy trạng thái của ngăn sếp (stack); trong BFS, khung sẽ hiển thị trạng thái của hàng đợi (queue), đối với chức năng tìm thành phần liên thông thì khung này sẽ liệt kê các thành phần liên thông đã được duyệt, giúp người dùng dễ dàng hiểu cách thuật toán hoạt động và cách thức xử lý của thuật toán.

Khung thứ ba là danh sách kề, nơi chương trình thể hiện cách đồ thị được lưu trữ dưới dạng danh sách kề. Khi thuật toán chạy, khung này sẽ cập nhật từng bước, giúp người dùng theo dõi các đỉnh và cạnh đang được duyệt.

Chương trình có thể hỗ trợ các thuật toán duyệt DFS, BFS và tìm các thành phần liên thông. Người dùng có thể điều chỉnh tốc độ duyệt để theo dõi quá trình một cách dễ dàng hơn. Ngoài ra, người dùng cũng có thể tạo đồ thị mới bằng cách nhập số đỉnh và số cạnh, và thêm các cạnh vào đồ thị qua giao diện. Nếu đồ thị đang được duyệt, các nút “chạy thuật toán” và “tạo đồ thị” sẽ bị vô hiệu hóa để tránh việc thay đổi đồ thị khi thuật toán đang chạy.

Chương trình còn có một nút “hướng dẫn” để người dùng tham khảo khi cần. Giúp người dùng dễ dàng nắm bắt cách sử dụng chương trình, đặc biệt là những người mới làm quen với chương trình.

3.2. Chức năng duyệt DFS

Chức năng duyệt đồ thị theo thuật toán tìm kiếm theo chiều sâu (DFS - Depth First Search) là một trong những tính năng quan trọng của chương trình. Khi người

dùng chọn thuật toán DFS, chương trình sẽ bắt đầu từ một đỉnh xuất phát và đi lần lượt qua các đỉnh kề chưa được thăm theo chiều sâu. Quá trình này tiếp tục cho đến khi không còn đỉnh nào có thể duyệt tiếp được từ đỉnh hiện tại. Đặc điểm của DFS là sử dụng một cấu trúc dữ liệu stack (ngăn xếp) để lưu trữ các đỉnh cần thăm, và khi không còn đỉnh con nào, thuật toán sẽ quay lại các đỉnh trước đó để tìm kiếm thêm, duyệt được cả đồ thị có nhiều hơn một thành phần liên thông.

Trong quá trình duyệt DFS, giao diện sẽ cập nhật thường xuyên, hiển thị trạng thái của stack và các đỉnh đang được thăm. Các đỉnh này sẽ được đánh dấu và làm nổi bật trên đồ thị để người dùng dễ dàng theo dõi tiến trình thuật toán. Đồng thời, khung minh họa cấu trúc dữ liệu cũng sẽ thay đổi, thể hiện sự thay đổi trong trạng thái của stack. Người dùng có thể điều chỉnh tốc độ duyệt để quan sát từng bước chi tiết hoặc để tăng tốc quá trình duyệt qua các đỉnh trong đồ thị.

3.3. Chức năng duyệt BFS

Duyệt đồ thị theo thuật toán tìm kiếm theo chiều rộng (BFS - Breadth First Search) là một thuật toán khác được tích hợp trong chương trình. Khác với DFS, BFS bắt đầu từ một đỉnh nguồn và duyệt qua các đỉnh kề theo chiều rộng. BFS sử dụng một cấu trúc dữ liệu hàng đợi (queue) để lưu trữ các đỉnh cần thăm tiếp theo. Khi một đỉnh được thăm, các đỉnh kề của nó sẽ được đưa vào hàng đợi và sẽ được thăm sau.

Khi thuật toán BFS được thực thi, khung hiển thị đồ thị sẽ cập nhật để người dùng thấy rõ các đỉnh đang được duyệt, từ đỉnh gốc cho đến các đỉnh ở các tầng tiếp theo. Khung cấu trúc dữ liệu cũng sẽ thay đổi để thể hiện trạng thái của hàng đợi, cho thấy các đỉnh đang chờ được duyệt. Giống như DFS, người dùng có thể điều chỉnh tốc độ để theo dõi thuật toán một cách dễ dàng hơn.

3.4. Chức năng tìm thành phần liên thông

Chức năng tìm thành phần liên thông cho phép xác định và phân nhóm các đỉnh của đồ thị thành các thành phần mà trong đó, mỗi đỉnh đều có thể đến được từ bất kỳ đỉnh nào khác trong cùng thành phần. Thuật toán này hữu ích trong việc phân tích đồ thị không có chu trình (đồ thị vô hướng) và giúp người dùng hiểu rõ hơn về cấu trúc của đồ thị. Chương trình sẽ sử dụng thuật toán DFS để duyệt qua đồ thị và xác định các thành phần liên thông. Mỗi thành phần liên thông sẽ được đánh dấu và hiển thị riêng biệt, giúp người dùng dễ dàng nhận diện các nhóm đỉnh liên kết với nhau.

Trong khi thuật toán tìm thành phần liên thông đang chạy, giao diện sẽ cập nhật và phân biệt các thành phần liên thông bằng cách liệt kê trên khung hiển thị cấu trúc dữ liệu thay vì hiển thị cấu trúc stack được sử dụng, giúp người dùng dễ dàng theo dõi. Mỗi khi một thành phần liên thông được phát hiện, các đỉnh trong thành phần đó sẽ được làm nổi bật, và người dùng có thể thấy rõ cách các đỉnh này được kết nối với nhau. Tính năng này hỗ trợ người dùng hiểu rõ hơn về cấu trúc của đồ thị và cách các đỉnh phân bố trong các nhóm liên thông.

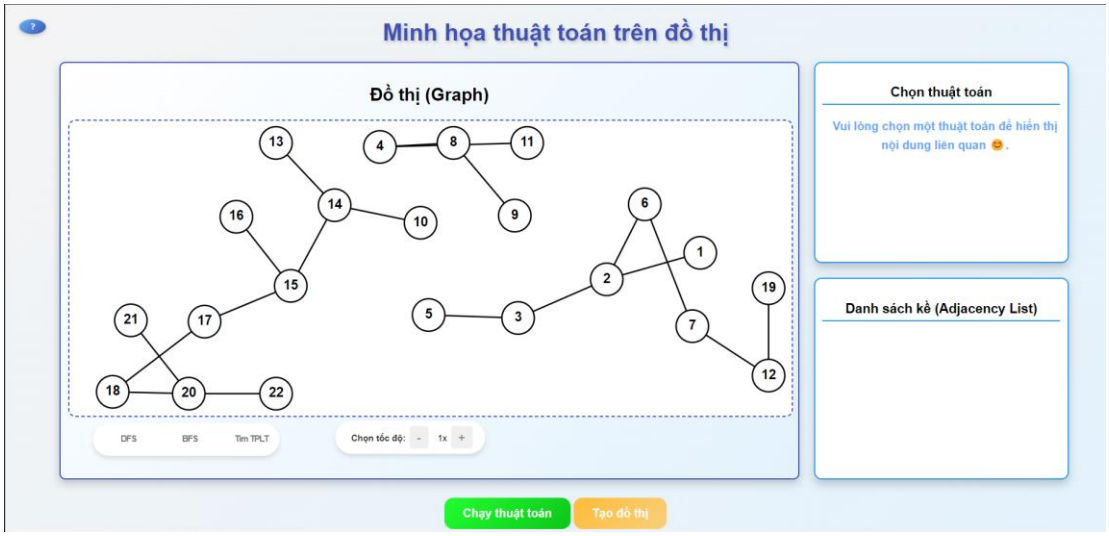
3.5. Chức năng tạo đồ thị

Chức năng tạo đồ thị cho phép người dùng tạo một đồ thị mới. Người dùng có thể nhập số lượng đỉnh và cạnh, sau đó nhập các cặp đỉnh nối với nhau để tạo các cạnh của đồ thị. Giao diện sẽ cung cấp một form nhập liệu đơn giản, nơi người dùng chỉ cần nhập các giá trị cần thiết. Sau khi nhập, đồ thị sẽ được vẽ trực quan và người dùng có thể thấy ngay các đỉnh và cạnh của đồ thị mới được tạo.

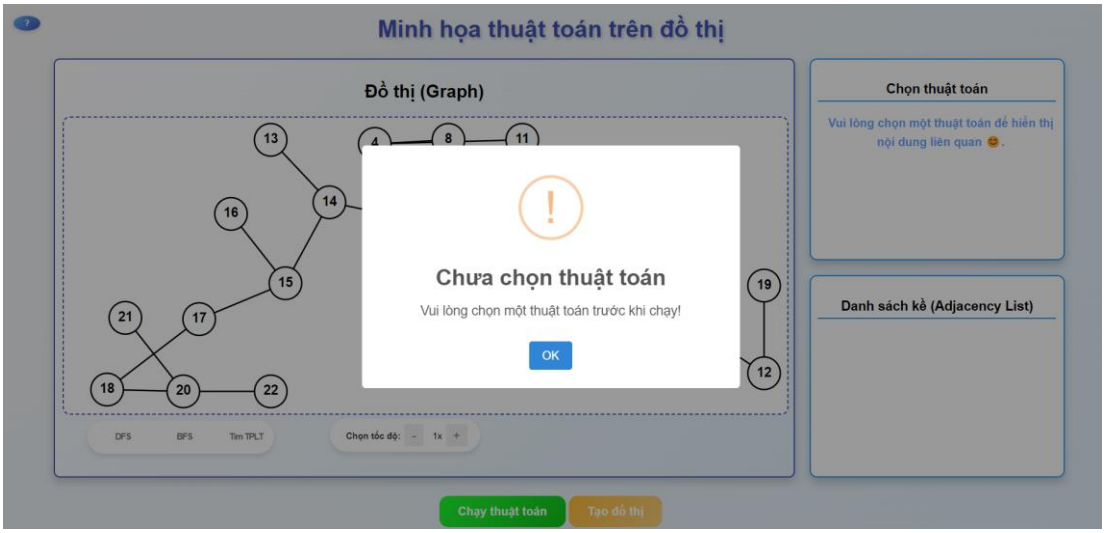
Đặc biệt, khi đồ thị đang được duyệt, các nút "chạy thuật toán" và "tạo đồ thị" sẽ bị vô hiệu hóa để tránh xung đột trong quá trình thực thi. Khi đồ thị mới được tạo thành công, chương trình sẽ tự động cập nhật các khung hiển thị, bao gồm khung đồ thị, khung cấu trúc dữ liệu và danh sách kề.

CHƯƠNG 4: SẢN PHẨM

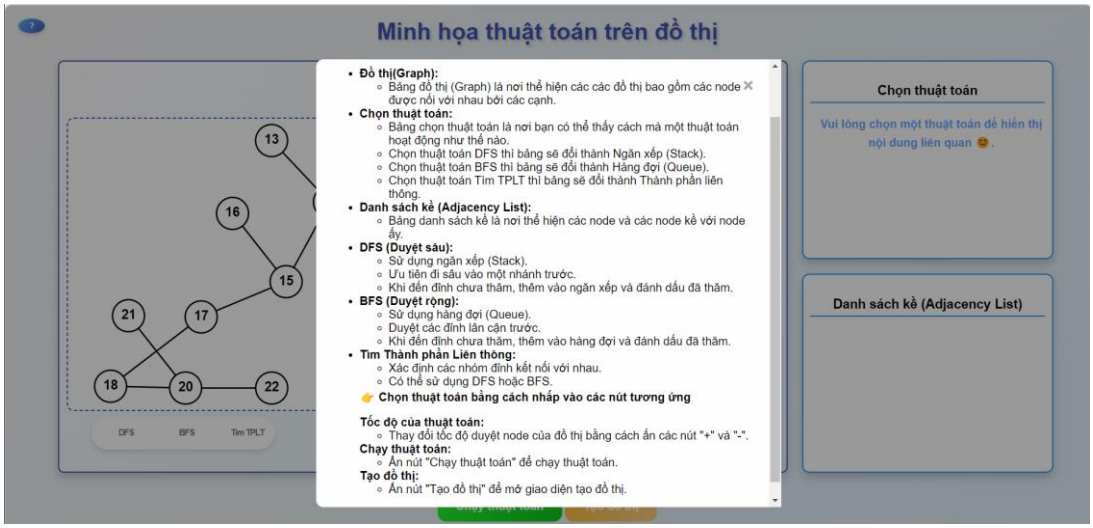
4.1. Giao diện chính



Hình 1. Giao diện chính



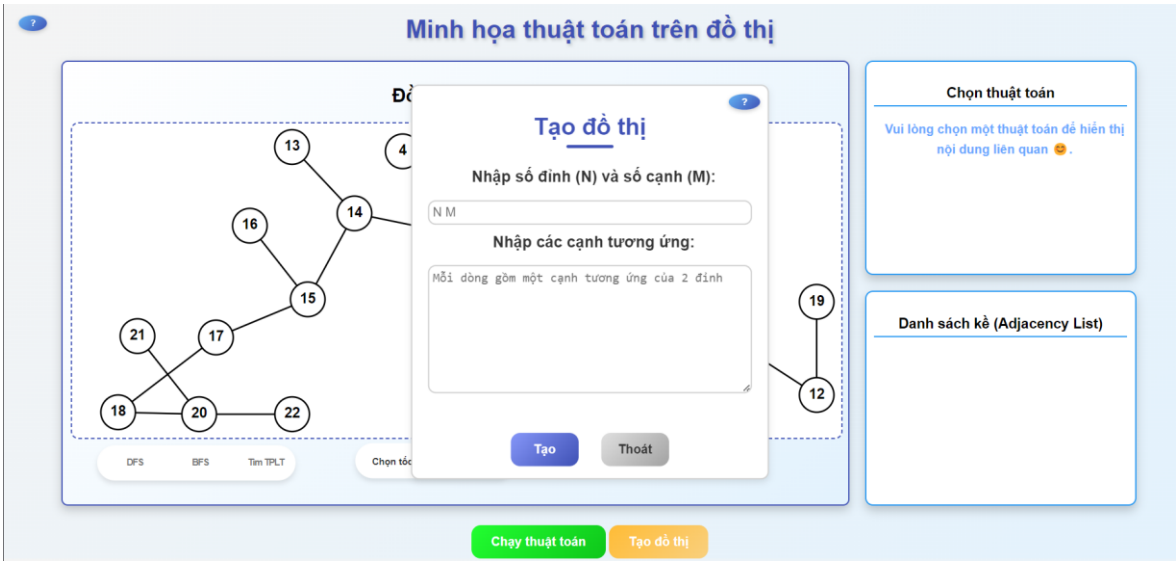
Hình 2. Giao diện thông báo khi chưa chọn thuật toán



Hình 3. Giao diện thông tin hướng dẫn chung

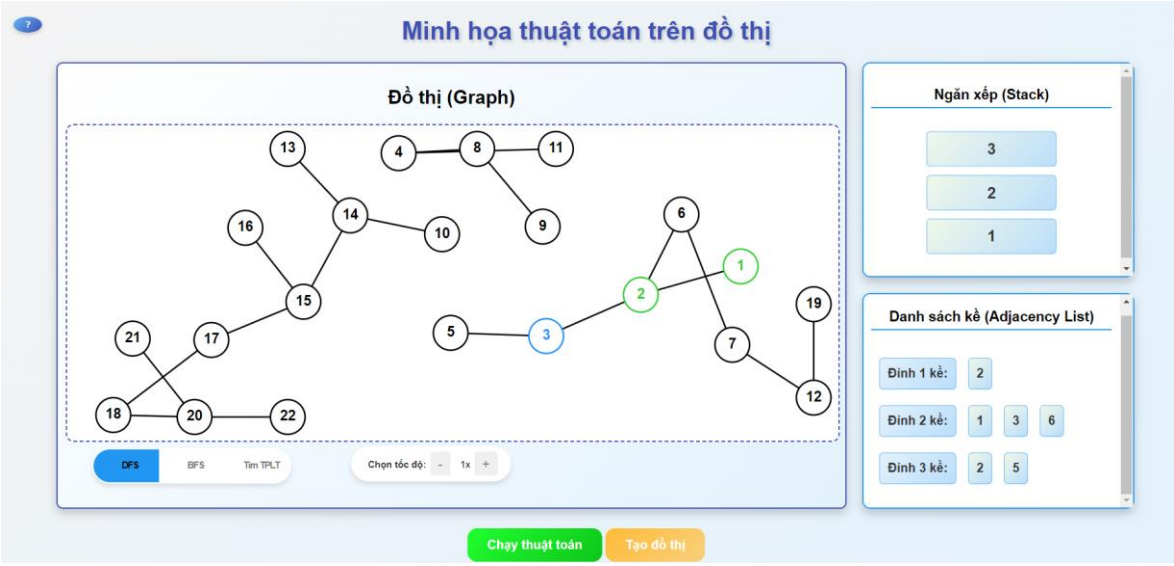
4.2. Giao diện tạo đồ thị

Hình 4. Giao diện hướng dẫn tạo đồ thị



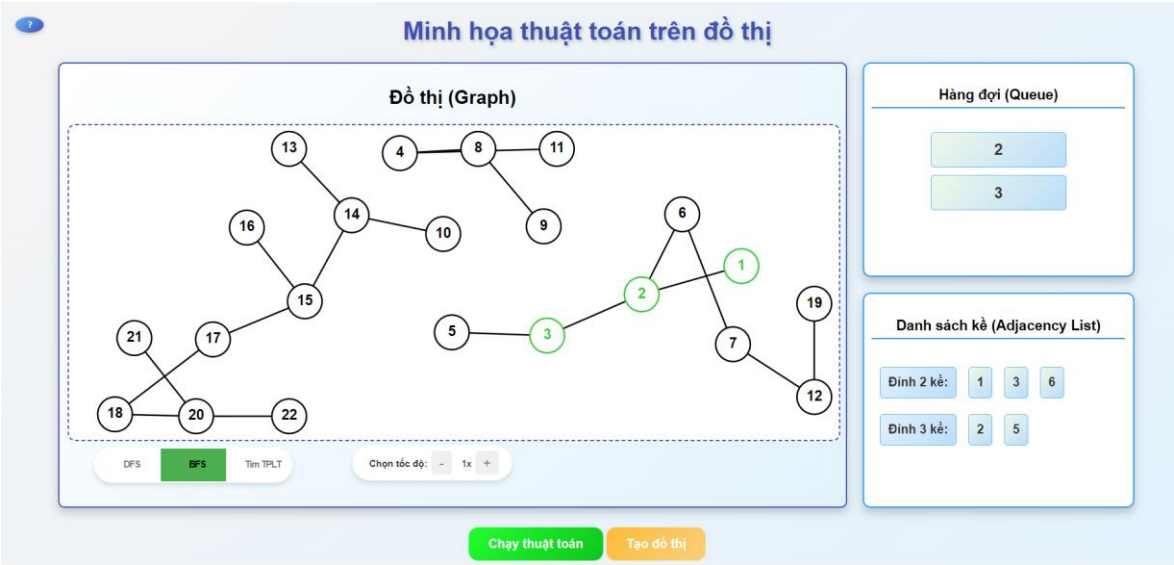
Hình 5. Giao diện tạo đồ thị

4.3. Giao diện duyệt DFS



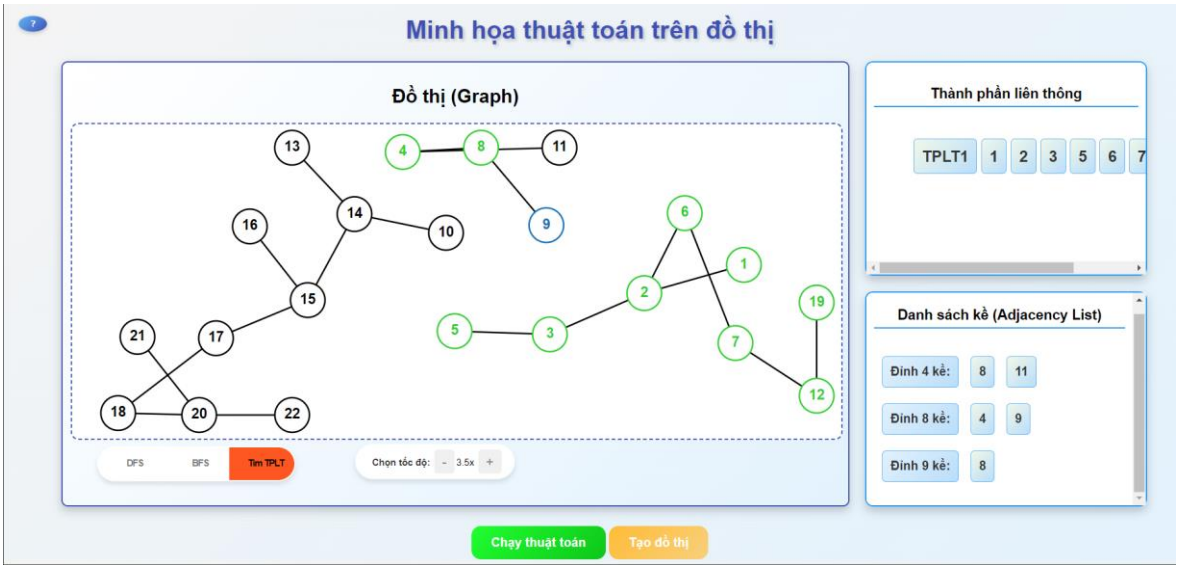
Hình 6. Giao diện chức năng duyệt DFS

4.4. Giao diện duyệt BFS



Hình 7. Giao diện chức năng duyệt BFS

4.5. Giao diện tìm thành phần liên thông



Hình 8. Giao diện chức năng tìm thành phần liên thông

CHƯƠNG 5: TỔNG KẾT

5.1. Kết quả đạt được

Chương trình đã thành công trong việc mô phỏng các thuật toán duyệt đồ thị cơ bản gồm DFS, BFS và tìm thành phần liên thông. Các thuật toán này hoạt động chính xác và cho phép người dùng theo dõi quá trình duyệt một cách trực quan, nhờ vào việc hiển thị trạng thái của các cấu trúc dữ liệu như stack (cho DFS) và queue (cho BFS). Các đỉnh và cạnh trong đồ thị được làm nổi bật rõ ràng, giúp người dùng dễ dàng quan sát và hiểu được các bước trong thuật toán.

Chương trình có chức năng tạo đồ thị được hoạt động ổn định, cho phép người dùng tạo đồ thị với số lượng đỉnh và cạnh tùy ý, đồng thời cung cấp đồ thị mặc định nếu người dùng không tạo đồ thị riêng. Khả năng di chuyển các đỉnh trong đồ thị cũng mang lại sự linh hoạt trong việc điều chỉnh bố cục đồ thị sao cho phù hợp với nhu cầu quan sát.

Chương trình hỗ trợ người dùng điều chỉnh tốc độ duyệt, giúp họ có thể quan sát từng bước trong quá trình thuật toán hoặc tăng tốc quá trình duyệt khi cần thiết. Các chức năng này hoạt động mượt mà, đáp ứng nhu cầu học tập và nghiên cứu của người dùng về lý thuyết và ứng dụng của các thuật toán đồ thị.

Tính năng hướng dẫn giúp người dùng dễ dàng làm quen với chương trình, đặc biệt là những người mới bắt đầu tìm hiểu về các thuật toán duyệt đồ thị. Chương trình còn đảm bảo tính ổn định và không xảy ra xung đột khi người dùng thực hiện các thao tác tạo đồ thị mới trong quá trình thuật toán đang chạy.

Tổng thể, chương trình đã hoàn thành mục tiêu minh họa các thuật toán đồ thị một cách trực quan, sinh động và dễ hiểu.

5.2. Hạn chế

Mặc dù chương trình đã hoàn thiện và thực hiện thành công việc mô phỏng các thuật toán duyệt đồ thị như BFS, DFS và tìm thành phần liên thông, nhưng hiện tại vẫn còn một số hạn chế cần phải cải thiện để nâng cao trải nghiệm người dùng và tính hiệu quả của công cụ. Một trong những hạn chế đáng chú ý là chương trình chưa hỗ trợ tính năng tạm dừng tại một vị trí cụ thể trong quá trình thuật toán đang chạy. Khi thực hiện các thuật toán duyệt đồ thị, việc tạm dừng tại một thời điểm nào đó để người dùng có thể kiểm tra trạng thái hoặc phân tích các bước duyệt sẽ là một tính năng hữu ích, nhưng hiện tại chưa được tính hợp ở chương trình này.

Bên cạnh đó, giao diện người dùng vẫn chưa thực sự tối ưu. Mặc dù chương trình có ba khung hiển thị chính, nhưng cách bố trí và sắp xếp các thành phần trong giao diện đôi khi chưa hợp lý và trực quan. Phần hiển thị các thành phần liên thông vẫn khó nhìn, chưa có hoàn thiện về giao diện. Các nút chức năng như "chạy thuật toán", "tạo đồ thị" hay các công cụ điều chỉnh tốc độ duyệt cần được thiết kế sao cho người dùng có thể dễ dàng nhận diện và sử dụng một cách logic hơn.

Ngoài ra, chương trình cũng thiếu phần lý thuyết giải thích ngắn gọn về các thuật toán mà nó mô phỏng. Việc tích hợp một phần lý thuyết giúp người dùng có thể dễ dàng tham khảo và nắm bắt nhanh các khái niệm cơ bản về thuật toán sẽ rất hữu ích. Mặc dù chương trình đã mô phỏng thuật toán, nhưng nếu không có phần lý thuyết đi kèm, người dùng sẽ gặp khó khăn trong việc hiểu rõ cách các thuật toán hoạt động và khó hình dung được thuật toán.

Một hạn chế khác là chương trình chưa hỗ trợ nhiều thuật toán. Hiện tại, chỉ có ba chức năng duyệt đồ thị cơ bản là DFS, BFS và tìm thành phần liên thông, nhưng chương trình vẫn thiếu các thuật toán khác như tìm đường đi ngắn nhất, thuật toán tìm cây khung nhỏ nhất, hay các thuật toán phân tích đồ thị phức tạp hơn.

Thêm vào đó, chức năng tạo đồ thị hiện tại chỉ cho phép người dùng tạo đồ thị từ đầu, nhưng không hỗ trợ chỉnh sửa đồ thị đã tạo, làm giảm đi độ linh hoạt của chương trình, bởi người dùng không thể sửa đổi hoặc cập nhật đồ thị hiện có sau khi đã tạo ra nó. Chức năng này cũng chưa hỗ trợ tạo được đồ thị có hướng và đồ thị có trọng số.

Cuối cùng, chương trình vẫn chưa minh họa rõ được trạng thái duyệt của các đỉnh trong quá trình thuật toán chạy. Mặc dù đỉnh đang được duyệt sẽ được làm nổi bật, nhưng sự hiển thị này vẫn còn chưa rõ ràng và trực quan.

5.3. Hướng phát triển

Trong tương lai, để nâng cao hiệu quả và khả năng sử dụng của chương trình, có thể hướng phát triển vào một số khía cạnh quan trọng. Đầu tiên, một trong những mục tiêu cần đạt được là cải thiện giao diện người dùng. Việc tối ưu hóa bố cục giao diện sẽ giúp người dùng dễ dàng sử dụng các tính năng của chương trình một cách trực quan hơn. Các nút chức năng cần được thiết kế sao cho rõ ràng và dễ nhận diện, giúp người dùng dễ dàng tìm kiếm và thao tác mà không gặp khó khăn. Ngoài ra, việc cải thiện các yếu tố hiển thị như màu sắc, kích thước và vị trí của các thành phần

trong giao diện sẽ giúp người dùng có trải nghiệm trực quan, sinh động và dễ dàng theo dõi các thuật toán.

Một hướng phát triển quan trọng khác là bổ sung tính năng tạm dừng và tiếp tục trong quá trình chạy thuật toán. Chức năng này sẽ giúp người dùng có thể tạm dừng thuật toán tại một điểm cụ thể để kiểm tra hoặc phân tích chi tiết quá trình duyệt đồ thị. Sau khi kiểm tra xong, người dùng có thể tiếp tục thuật toán từ điểm dừng mà không cần phải bắt đầu lại từ đầu, giúp tăng tính linh hoạt và sự tương tác của chương trình. Tính năng này cũng sẽ hỗ trợ việc giảng dạy và học tập, khi người dùng có thể dừng lại để giải thích các bước của thuật toán.

Chương trình cũng cần được mở rộng về mặt lý thuyết. Một phần lý thuyết ngắn gọn và dễ hiểu sẽ được tích hợp, giúp người dùng nhanh chóng nắm bắt các khái niệm cơ bản và ứng dụng của các thuật toán duyệt đồ thị. Không chỉ giúp người dùng hiểu rõ hơn về các thuật toán mà còn có thể tra cứu nhanh về lý thuyết nếu như người dùng cần.

Chức năng tạo đồ thị có hướng và có trọng số cũng là một tính năng rất cần được bổ sung. Người dùng có thể tạo các cạnh có hướng giữa các đỉnh và chỉ định trọng số cho mỗi cạnh, giúp mô phỏng chính xác hơn các bài toán thực tế như tìm đường đi ngắn nhất hay tối ưu hóa mạng lưới. Giao diện sẽ được cập nhật để cho phép chọn kiểu đồ thị (vô hướng, có hướng, có trọng số) và nhập trọng số khi tạo cạnh.

Để chương trình trở thành công cụ học tập mạnh mẽ hơn, việc bổ sung các thuật toán khác cũng sẽ là một bước tiến quan trọng. Các thuật toán như Dijkstra, Kruskal, Prim, Floyd-Warshall, và Bellman-Ford là những thuật toán cơ bản và quan trọng trong lý thuyết đồ thị mà người dùng có thể thử nghiệm. Tích hợp thêm các thuật toán này sẽ làm phong phú khả năng ứng dụng của chương trình và giúp người dùng không chỉ học được các thuật toán duyệt đồ thị mà còn nắm vững các thuật toán tìm đường đi, cây khung nhỏ nhất, hay các thuật toán về đồ thị có trọng số.

Chức năng chỉnh sửa đồ thị hiện tại chưa có, nhưng đây là một tính năng cần thiết để nâng cao tính linh hoạt của chương trình. Người dùng sẽ có thể chỉnh sửa, thêm hoặc xóa các đỉnh và cạnh của đồ thị đã tạo trước đó. Tính năng cho phép người dùng thử nghiệm với nhiều cấu trúc đồ thị khác nhau mà không phải tạo lại từ đầu mỗi lần. Hữu ích trong việc khám phá các đồ thị phức tạp hơn và thực hiện các bài toán thay đổi đồ thị theo yêu cầu.

Cuối cùng, để tăng cường sự minh họa và trực quan hóa quá trình duyệt, chương trình cần cải thiện khả năng hiển thị trạng thái của các đỉnh trong quá trình thuật toán. Cần có các hiệu ứng động khác khi thuật toán đang chạy, hiển thị trạng thái của đỉnh,... Giúp người dùng dễ dàng nhận diện các đỉnh và cạnh đang được duyệt để họ hiểu rõ hơn về cách thức hoạt động của thuật toán trong từng bước.

TÀI LIỆU THAM KHẢO

[1] Teller, Swizec. *Data Visualization with d3. js*. Packt Publishing, 2013.
[2] Nguyễn Văn Thức. (2016, August 21). "*Tìm hiểu về hộp thoại Sweet Alert 2"* trang ViBlo.

PHỤ LỤC

Link github: https://github.com/tynnp/GraphVisualizer

Link drive: https://drive.google.com/drive/folders/19G7hXeR3CMB-8Hmpvrcj16MNKRx9eB0H