**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**



**ĐỒ ÁN**

**HỌC PHẦN TRÍ TUỆ NHÂN TẠO**

**Đề tài**

**BÀI TOÁN TÌM ĐƯỜNG ĐI TRÊN BẢN ĐỒ**

**Nhóm sinh viên thực hiện:**

**1. Thạch Quang Huy B2207523**

**2. Nguyễn Trọng Nguyên B2207547**

**3. Trần Chí Hiếu B2207520**

#### Cần Thơ, 29/10/2024

**NHẬN XÉT CỦA GIẢNG VIÊN**

---------------------------------------------------------------------------------

---------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------

*Cần Thơ, ngày tháng năm*

(Ký và ghi rõ họ tên)

**MỤC LỤC**

**PHÂN CÔNG CÔNG VIỆC**

**Thạch Quang Huy: thiết kế html, css, thuật toán leo đồi.**

**Nguyễn Trọng Nguyên: thuật toán háu ăn, ucs.**

**Trần Chí Hiếu: thuật toán A\*, powerpont.**

**PHẦN NỘI DUNG**

## Mô tả bài toán

Bài toán tìm đường đi trên bản đồ là một bài toán thực tiễn trong nhiều lĩnh vực như giao thông, logistics, trò chơi điện tử, và định vị. Mục tiêu của bài toán là tìm ra con đường ngắn nhất hoặc tối ưu giữa hai điểm trên bản đồ, đồng thời đảm bảo tiêu chí về chi phí (ví dụ như khoảng cách, thời gian, hoặc năng lượng) và tuân thủ các giới hạn về môi trường (chướng ngại vật, địa hình, v.v.). Bài toán này được giải quyết bằng cách mô hình hóa bản đồ như một đồ thị, trong đó mỗi điểm (đỉnh) là một vị trí trên bản đồ, và các cạnh là các đường đi giữa các vị trí với chi phí nhất định

## Phân tích bài toán:

Đầu vào: Một bản đồ, được biểu diễn dưới dạng một ma trận hoặc danh sách các đỉnh và cạnh trong đồ thị, bao gồm:

* + - 1. Điểm bắt đầu và kết thúc.
      2. Các giá trị heuristic (H) có sẵn của các đỉnh đó. Thường là hàm Manhattan, Euclidean,...
      3. Khoảng từ đỉnh này sang đỉnh khác (G).

Đầu ra: Đường đi tối ưu từ điểm bắt đầu đến điểm kết thúc, bao gồm:

* + - 1. Đường đi: Danh sách các điểm từ điểm xuất phát đến điểm đích.
      2. Tổng chi phí: Tổng chi phí di chuyển của đường đi (G).
      3. Thứ tự duyệt: Thứ tự các đỉnh được duyệt khi tìm kiếm đường đi.

## 3. Cơ sở lý thuyết (các giải thuật được sử dụng trong đồ án)

Các giải thuật được sử dụng trong bài toán này bao gồm:

* + - 1. Thuật toán A\*
         1. *Là một trong những giải thuật tìm kiếm heuristic mạnh mẽ nhất. A* sử dụng một hàm heuristic để ước lượng chi phí còn lại từ điểm hiện tại đến đích. Nó kết hợp ưu điểm của giải thuật Dijkstra (tìm kiếm đường ngắn nhất) với tìm kiếm theo heuristic (Greedy Search) để tìm kiếm một cách hiệu quả.
         2. Hàm đánh giá: f(n) = g(n) + h(n)
         3. g(n): Chi phí từ điểm bắt đầu đến điểm hiện tại.
         4. h(n): Chi phí ước tính từ điểm hiện tại đến điểm đích (thường sử dụng khoảng cách Euclidean hoặc Manhattan).
      2. Leo đồi
         1. Là một phương pháp heuristic, tập trung vào việc tìm kiếm một cách tham lam theo hướng giảm dần hàm chi phí. Tuy nhiên, nó có thể mắc kẹt ở đỉnh cục bộ và không đảm bảo tìm được lời giải tối ưu.
         2. Hàm đánh giá: f(n) = h(n).
         3. Nhược điểm: Dễ bị mắc kẹt tại các đỉnh cục bộ.
      3. USC (Uniform Cost Search)
         1. Là một phiên bản của Dijkstra, trong đó luôn chọn đường có chi phí thấp nhất mà chưa được mở rộng. UCS đảm bảo tìm được đường ngắn nhất nếu tất cả các cạnh có chi phí dương.
         2. Hàm đánh giá: f(n) = g(n) (chỉ dựa vào chi phí từ điểm bắt đầu).
      4. Háu ăn
         1. Giải thuật này chỉ tập trung vào việc ước lượng chi phí từ điểm hiện tại đến đích, mà không quan tâm đến chi phí đã đi qua.
         2. Hàm đánh giá: f(n) = h(n) (chỉ dựa vào hàm heuristic).
         3. Ưu điểm: Chạy nhanh, nhưng có thể không tìm ra đường ngắn nhất.

**4. Thiết kế và cài đặt**

1. Thiết kế:
   1. Có 2 cách để truyền dữ liệu đầu vào:
      1. Nhập số đỉnh, số cung, chọn thuật toán, nhập điểm bắt đầu, kết thúc, nhập heuristic cho các đỉnh, nhập cung và g của cung đó.
      2. Chọn một file có định dạng: số đỉnh số cung.

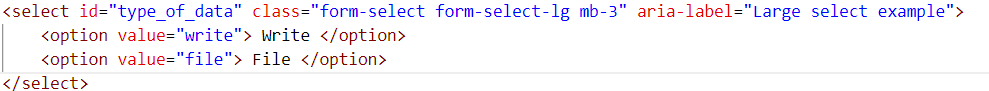
start: đỉnh end: đỉnh

H(đỉnh) = giá trị

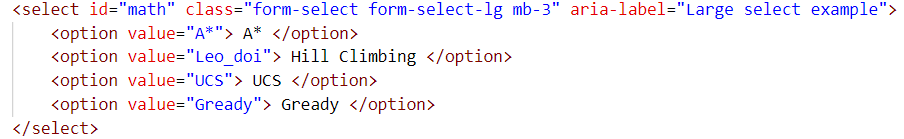
đỉnh đỉnh giá trị g

* 1. Đầu ra là một bảng gồm stt các bước, đỉnh đang xét, danh sách cái đỉnh trong open và close, đường đi từ đỉnh start đến đỉnh end, chi phí của đường đi đó.

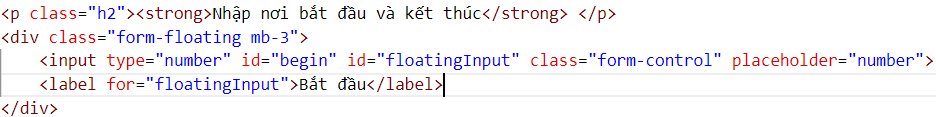
1. Cài đặt: JavaScipt cho A\*.
   1. Cài đặt nút chọn phương thức nhập dữ liệu

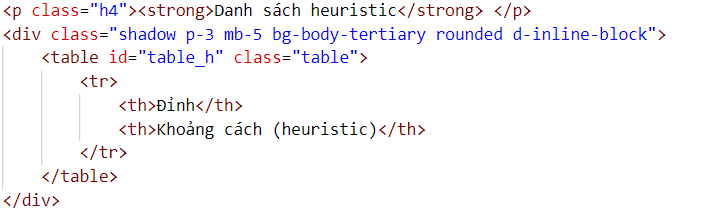


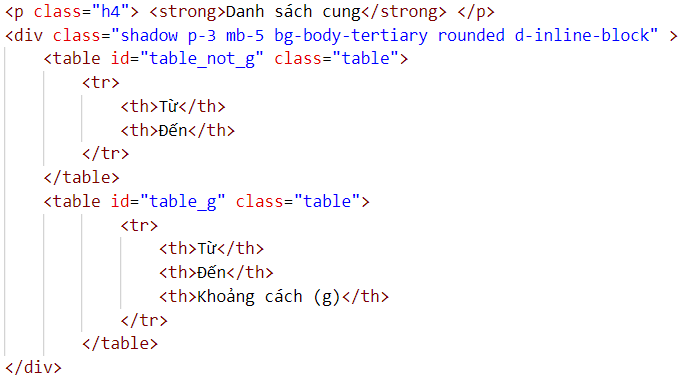
* 1. Cài đặt nút chọn thuật toán



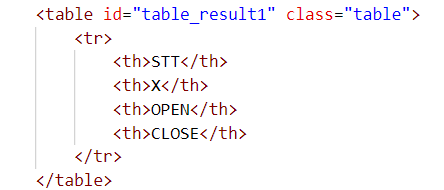
* 1. Cài đặt danh sách các cung, các đỉnh, danh sách các đỉnh và Heuristic, các cung và chi phí.

****

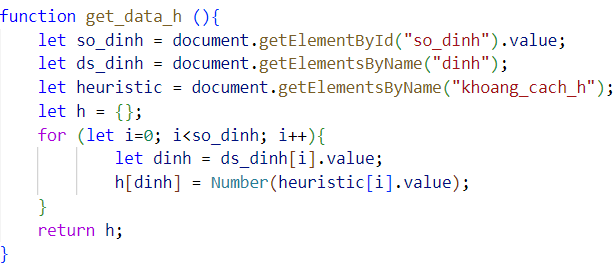
****

****

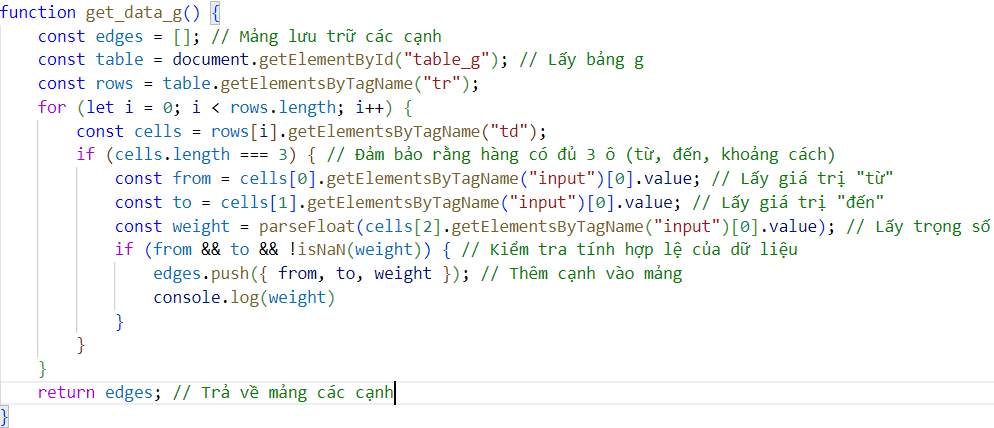
* 1. **Bảng kết quả**

****

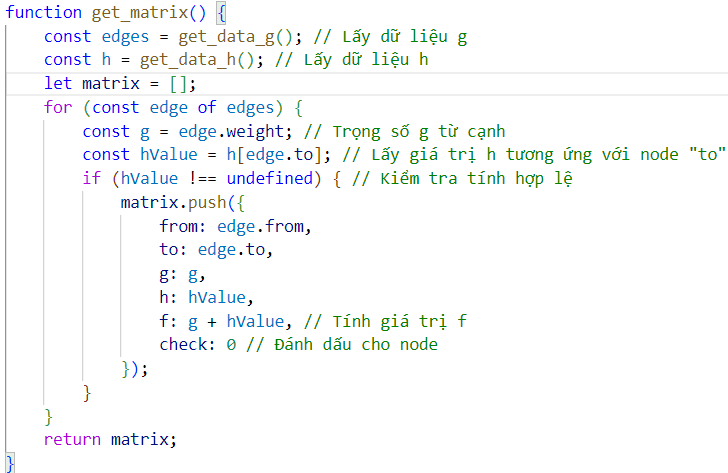
* 1. **Hàm lấy H**

****

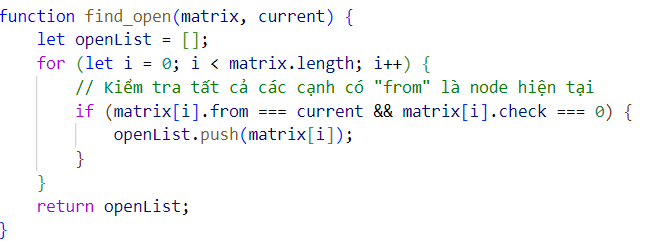
* 1. **Hàm lấy G**

****

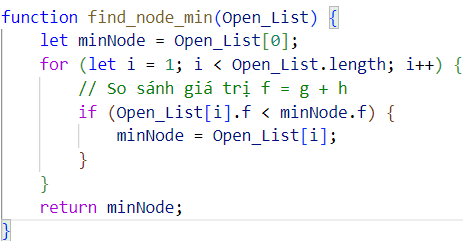
* 1. **Hàm cài đặt ma trận**

****

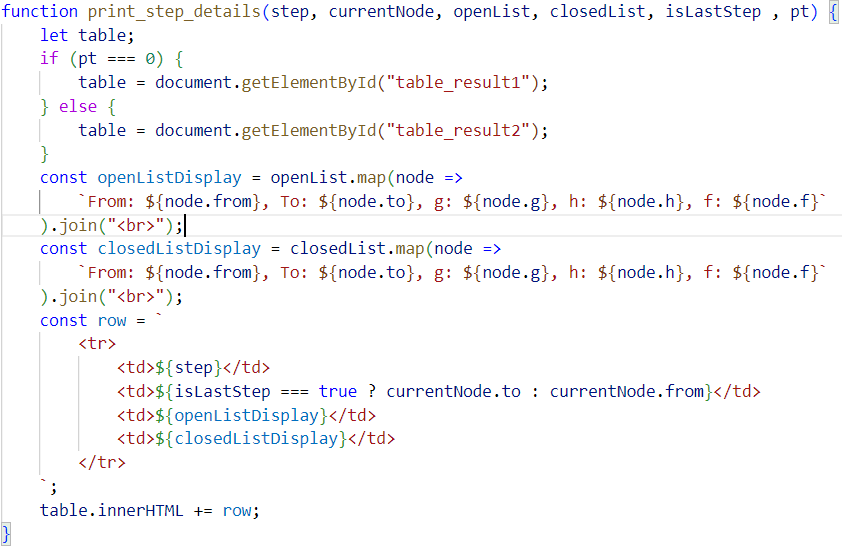
* 1. **Hàm trả về tất cả các nút có đỉnh current == from**

****

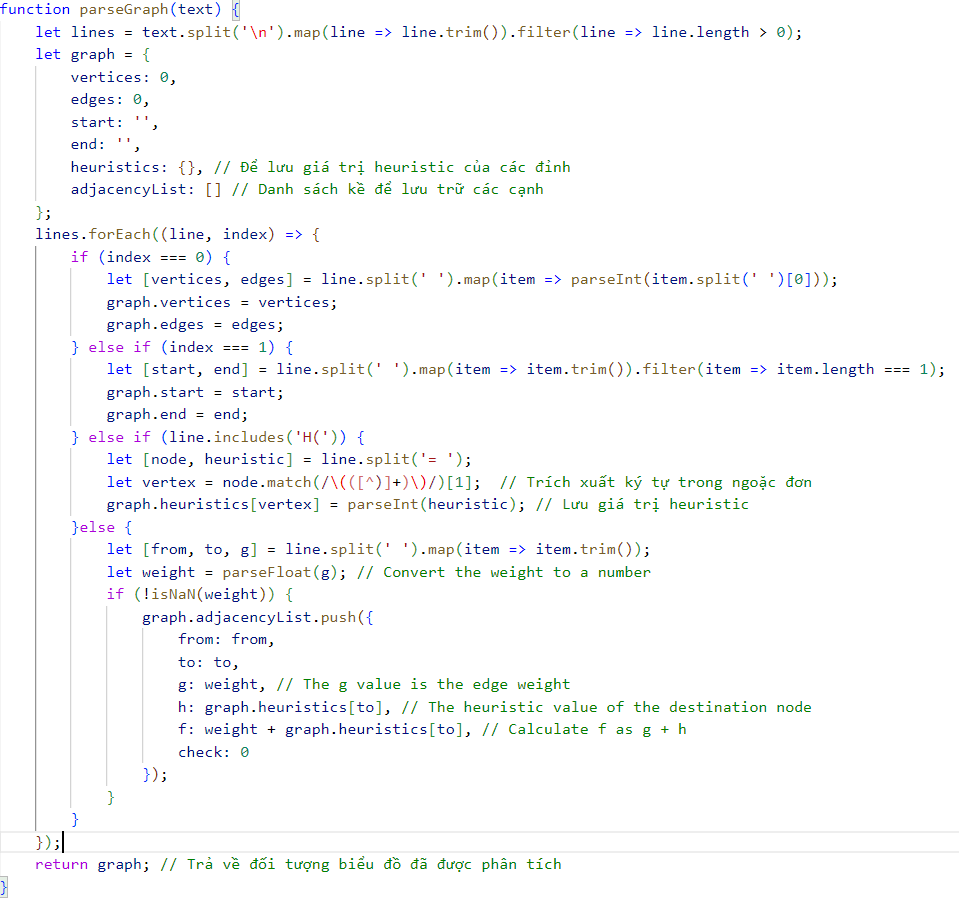
* 1. **Hàm trả về nút có f nhỏ nhất**

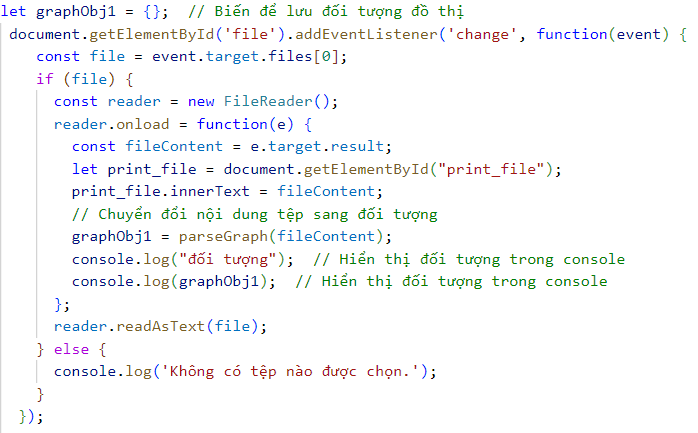
****

* 1. **Hàm in kết quả duyệt**

****

* 1. **Hàm chuyển đổi file thành đối tượng Graph**

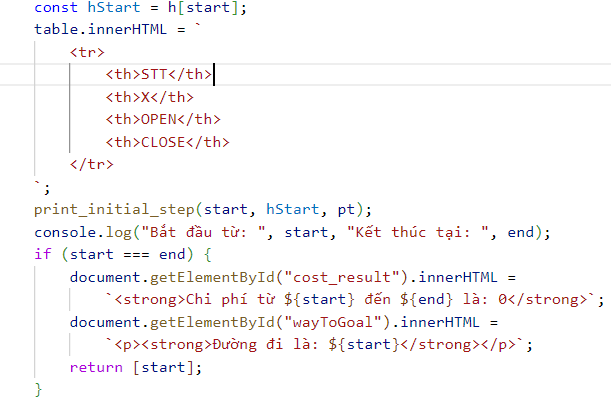


****

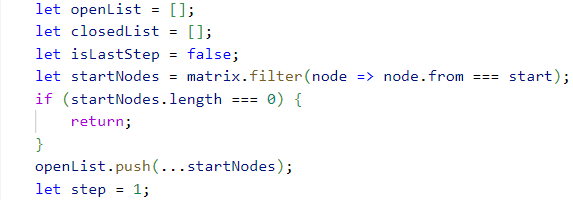
* 1. **Hàm A\***
     1. Khởi tạo các giá trị cần thiết tùy vào phương thức nhập

****

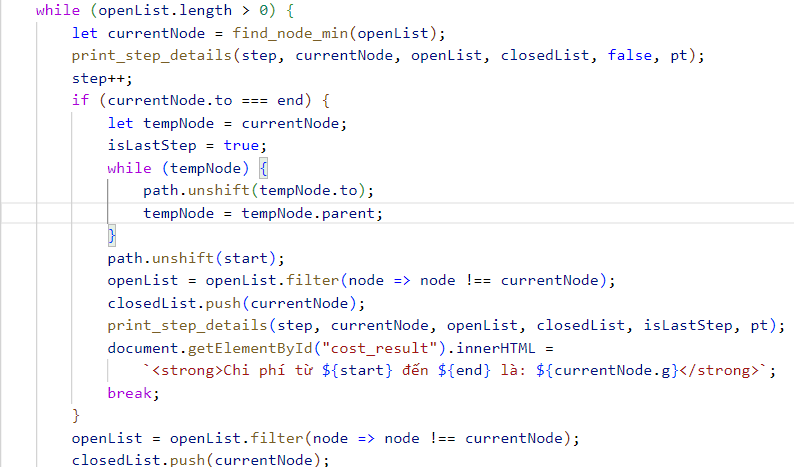
* 1. **Khởi tạo nút ban đầu và in nút khởi tạo.**

****

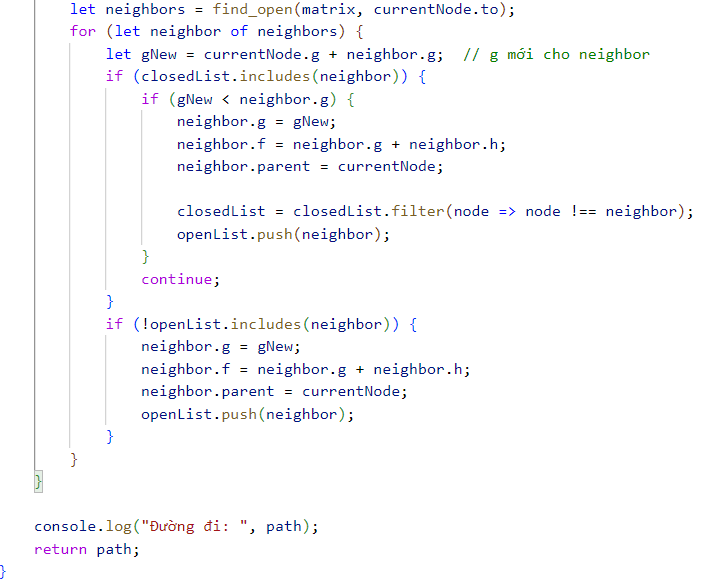
* 1. **Khởi tạo openList, closeList, thêm đỉnh có from == start vào openList**

****

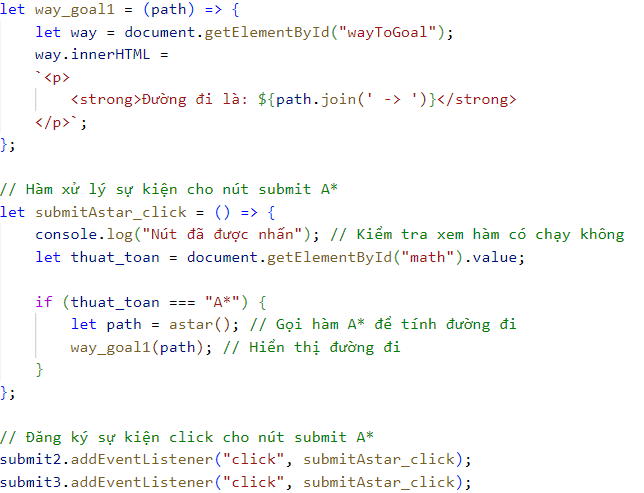
* 1. **Vòng lặp duyệt**

****

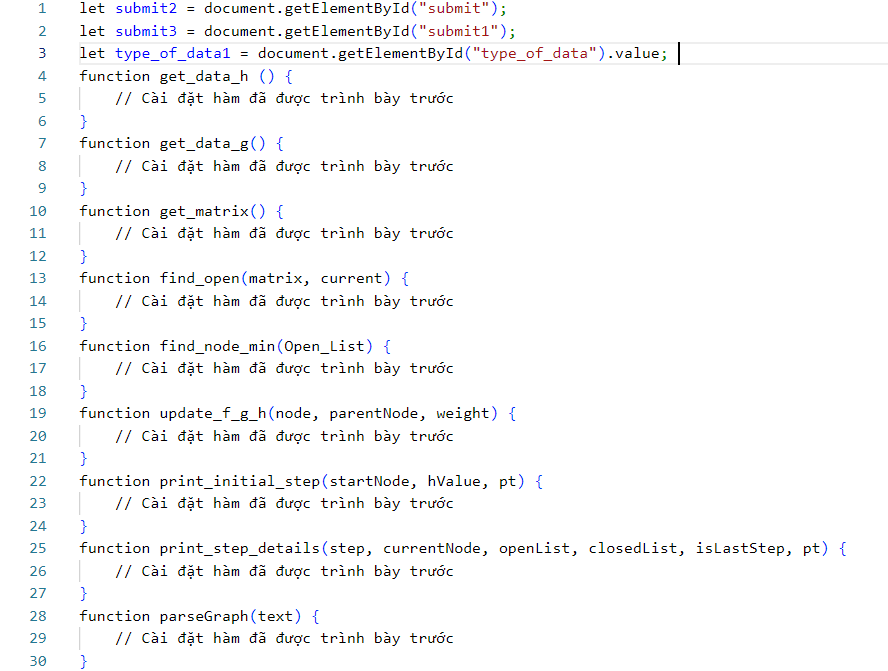
* 1. **Duyệt các nút lân cận và cập nhật G H F, openList và closeList và in đường đi chi phí đường đi.**

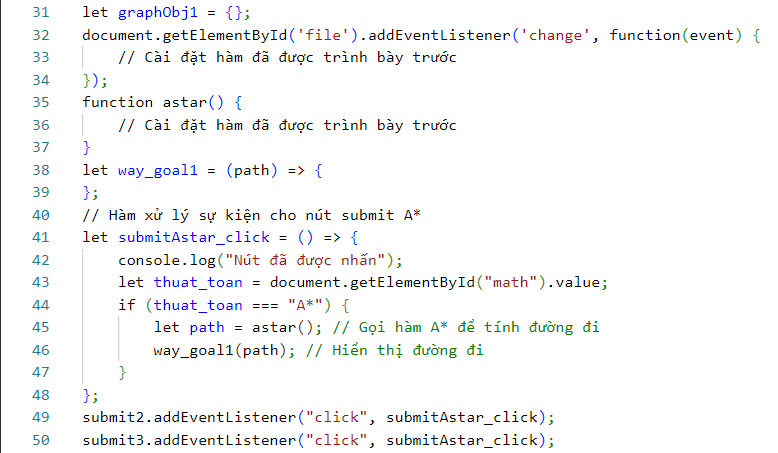
****

* 1. **Các hàm hiển thị đường đi và lắng nghe khi nhất nút submit**

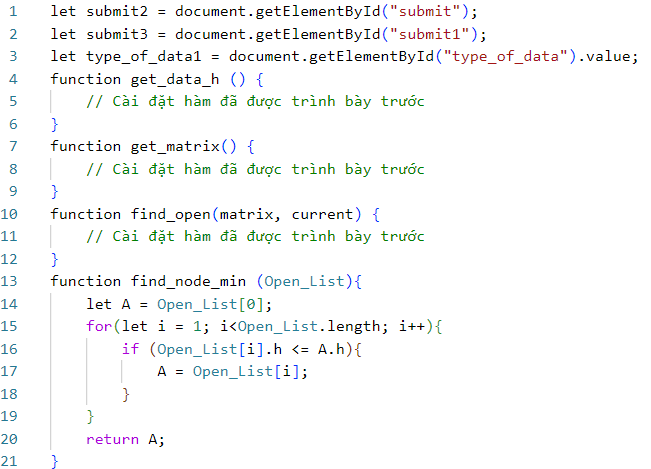
****

* 1. **Để cài chạy chương trình .**

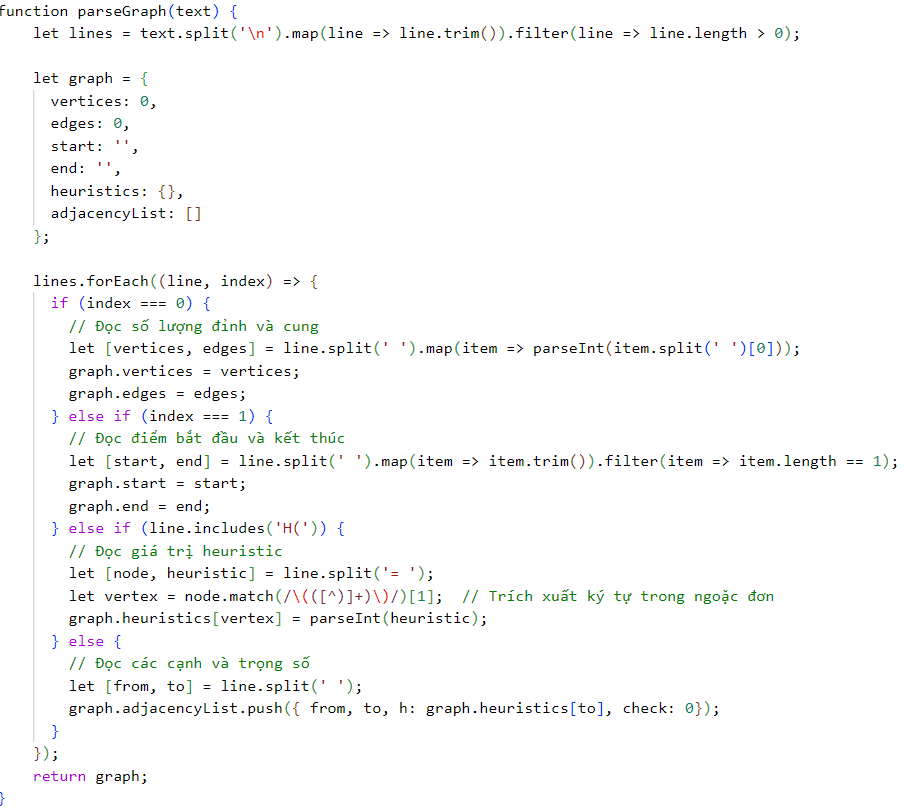
****

****

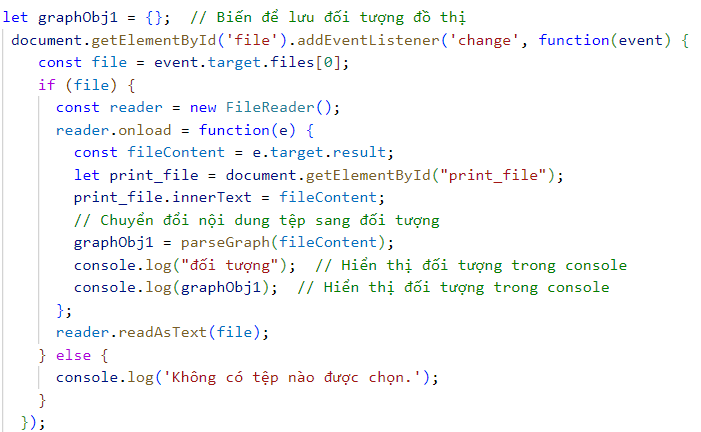
1. **Cài đặt hàm cho giải thuật leo đồi.**
   1. **Cài đặt các hàm tương tự như A\***

****

* 1. **Hàm đọc file**

****

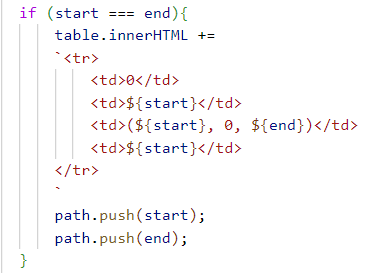
* 1. **Hàm chuyển file đã đọc thành đối tượng.**

****

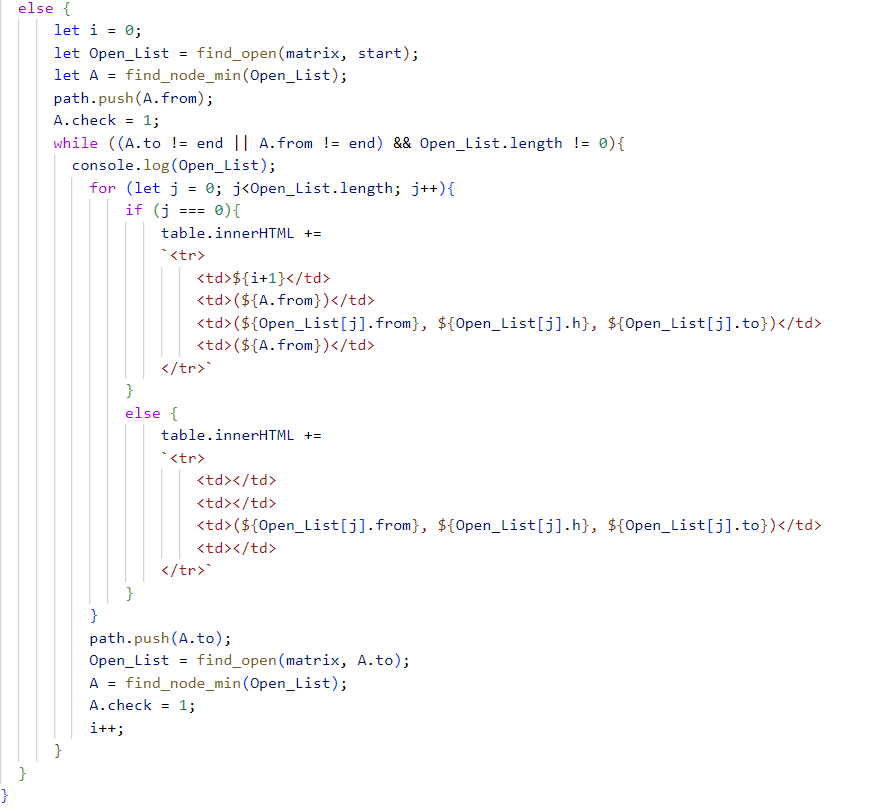
* 1. **Khởi tạo các giá trị cần thiết cho thuật toán thựa vào phương thức nhập.**

****

* 1. **Dùng if kiểm tra điều kiện dừng**

****

* 1. **Nếu chưa đến đỉnh đích thì thực thi thuật toán leo đồi.**

****

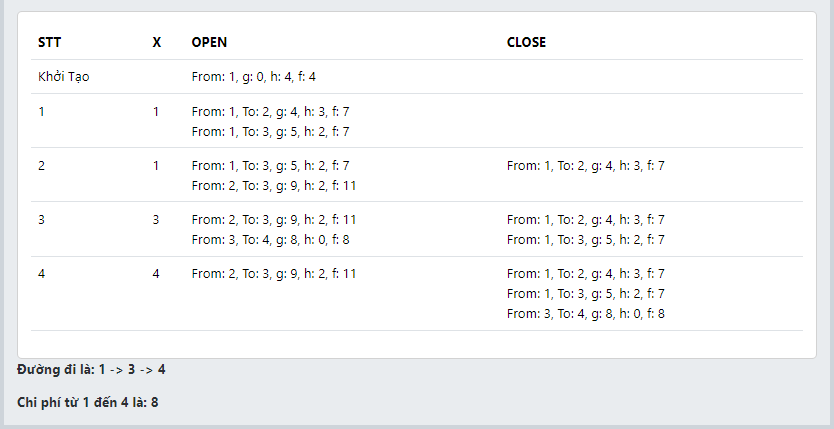
* 1. **Tạo nút submit, gọi hàm duyệt leo đồi, in đường đi.**

****

**PHẦN KẾT LUẬN**

## 1. Kết quả đạt được :

* Có thể chọn một trong nhiều thuật toán để áp dụng vào dữ liệu đầu vào, hiển thị được đường đi, chi phí.
* Thực hiện các giải thuật trên nền giao diện web giúp người dùng nhập bản đồ dưới dạng thủ công hoặc file theo định dạng có sẳn.
* Về kết quả của thuật toán A\*:
  + Với input là phương thức nhập từ bàn phím: số đỉnh: 4 và số cung:4.
  + Đỉnh bắt đầu là 1 và kết thúc là 4.
  + Các đỉnh và heuristic của các đỉnh như sau: 1-4, 2-3, 3-2, 4-0.
  + Các danh sách cung và giá trị g như sau: 1-2-4, 2-3-5, 1-3-5, 3-4-3.
  + Kết quả đạt được:



## 2. Hướng phát triển :

1. Có thể tích hợp bản đồ thực tế để phát triển sau này.
2. Cải thiện tốc độ xử lý của các thuật toán đối với dữ liệu đầu vào tương đối lớn, có thể tối ưu hóa bằng cách sử dụng các kỹ thuật như pruning, heuristic tốt hơn hoặc sử dụng dữ liệu đã được tối ưu hóa trước đó.
3. Phát triển thêm: So sánh và đánh giá hiệu suất: của các giải thuật với nhau.
4. Hill Climbing với random restarts: Sử dụng kỹ thuật khởi động lại ngẫu nhiên để tránh bị kẹt ở các đỉnh cục bộ, giúp thuật toán có khả năng tìm ra lời giải tối ưu hơn.
5. UCS phân tán: Nghiên cứu cách phân tán thuật toán UCS, chia nhỏ bài toán trên các nút mạng để xử lý song song, giảm thời gian tính toán.

# TÀI LIỆU THAM KHẢO

[1]. A\* Search Algorithm,” **GeeksforGeeks.**

[2]. Trí Tuệ Nhân Tạo, Khoa Khoa Học Máy Tính, Trường Đại Học Cần Thơ

[3]. Algorithms của Robert Sedgewick & Kevin Wayne