A black background with white lines

Description automatically generated**TRƯỜNG** **ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN-TRUYỀN THÔNG**

 **======\*\*\*======**

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: ỨNG DỤNG THUẬT TOÁN A\* TRONG TÌM ĐƯỜNG ĐI TRONG THỦ ĐÔ HÀ NỘI**

|  |  |
| --- | --- |
| **GVHD:**  **Lớp:**  **Khóa:**  **Nhóm:**  **Sinh viên :** | **Mai Thanh Hồng**  **2024IT6094010**  **18**  **12**  **Mai Văn Tiến – 2023607776**  **Phan Bá Khánh – 2023605083**  **Nguyễn Tường Nguyên - 2023603386**  **Phạm Công Chiến – 2023606132** |

***Hà Nội - 2025***

# 

# LỜI NÓI ĐẦU

Trong bối cảnh công nghệ thông tin (CNTT) ngày càng phát triển mạnh mẽ, vai trò của CNTT trong đời sống và sự phát triển kinh tế - xã hội là không thể phủ nhận. Đặc biệt, trí tuệ nhân tạo (AI) đã và đang trở thành một lĩnh vực cốt lõi của cuộc cách mạng công nghiệp 4.0 và sẽ tiếp tục giữ vai trò then chốt trong cuộc cách mạng công nghiệp lần thứ 5. AI giúp con người tiết kiệm thời gian, tối ưu hóa chi phí và nâng cao hiệu suất trong nhiều lĩnh vực.

Một trong những ứng dụng tiêu biểu của AI là trong lĩnh vực tìm đường và điều hướng thông minh. Trong đó, thuật toán A\* (A-star) nổi bật nhờ khả năng tìm kiếm đường đi hiệu quả với độ chính xác cao. Thuật toán này đã và đang được ứng dụng rộng rãi trong giao thông, vận tải, logistics và các hệ thống điều khiển tự động.

Với định hướng ứng dụng thực tiễn, nhóm chúng em đã chọn nghiên cứu và áp dụng thuật toán A\* để tìm đường đi ngắn nhất trên bản đồ, cụ thể là trong phạm vi thành phố Hà Nội. Nhóm đặt mục tiêu xây dựng một hệ thống mô phỏng tìm đường hiệu quả, từ đó rút ra những bài học kinh nghiệm và hướng phát triển trong các môi trường phức tạp hơn.

Chúng em hy vọng rằng đề tài này sẽ góp phần làm rõ cách thức ứng dụng thuật toán A\* trong thực tiễn, đồng thời mang lại cái nhìn tổng quan về quy trình xây dựng một hệ thống tìm đường hiệu quả trên nền tảng bản đồ số.Do kiến thức còn hạn chế, đề tài của nhóm không tránh khỏi những thiếu sót. Rất mong nhận được những góp ý quý báu từ cô và các bạn để hoàn thiện và phát triển nghiên cứu tốt hơn trong tương lai.

Chúng em xin chân thành cảm ơn sự hướng dẫn tận tình của giảng viên bộ môn – cô Mai Thanh Hồng, Giảng viên Khoa CNTT Trường Đại Học Công Nghiệp Hà Nội – đã tận tân giúp đỡ, chỉ dạy để nhóm hoàn thành đề tài này.

**MỤC LỤC**

[LỜI NÓI ĐẦU 1](#_Toc200558774)

[DANH MỤC HÌNH ẢNH 5](#_Toc200558775)

[DANH MỤC CÁC TỪ VIẾT TẮT 6](#_Toc200558776)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 7](#_Toc200558777)

[**1.1** **Tổng quan về trí tuệ nhân tạo** 7](#_Toc200558778)

[**1.1.1** **Khái niệm về trí tuệ nhân tạo** 7](#_Toc200558779)

[**1.1.2** **Phân loại** 9](#_Toc200558780)

[**1.1.3** **Lịch sử hình thành và phát triển** 10](#_Toc200558781)

[**1.1.4** **Những vấn đề chưa được giải quyết trong AI** 11](#_Toc200558782)

[**1.2** **Giới thiệu về đề tài** 12](#_Toc200558783)

[**1.3** **Các công nghệ được sử dụng** 12](#_Toc200558784)

[**1.3.1** **Giới thiệu ngôn ngữ lập trình Python** 12](#_Toc200558785)

[**1.3.2** **Các thư viện sử dụng trong đề tài** 13](#_Toc200558786)

[**1.3.3** **Giới thiệu hệ quản trị dữ liệu và Maptilper API** 15](#_Toc200558787)

[**1.3.4** **Các ngôn ngữ lập trình khác** 16](#_Toc200558799)

[CHƯƠNG 2: ÁP DỤNG THUẬT TOÁN VÀO TÌM ĐƯỜNG ĐI NGẮN NHẤT TRONG THỦ ĐÔ HÀ NỘI 19](#_Toc200558800)

[**2**.**1. Giới thiệu các thuật toán chung** 19](#_Toc200558801)

[**2.1.1. Tìm kiếm không có thông tin** 19](#_Toc200558802)

[**2.1.2. Tìm kiếm có thông tin** 21](#_Toc200558803)

[**2.2. Áp dụng thuật toán A\* vào tìm đường đi ngắn nhất trong thủ đô Hà Nội….** 25](#_Toc200558804)

[**2.2.1. Ý tưởng xây dựng** 25](#_Toc200558805)

[**2.2.2. Quy trình vận hành thuật toán A\* để tìm đường đi:** 26](#_Toc200558806)

[CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ 29](#_Toc200558807)

[**3.1. Quy trình thực nghiệm** 29](#_Toc200558808)

[**3.1.1.** **Mô tả thực nghiệm** 29](#_Toc200558809)

[**3.1.2.** **Thu thập dữ liệu** 30](#_Toc200558810)

[**3.1.3.** **Xây dựng dữ liệu** 31](#_Toc200558811)

[**3.1.4.** **Xử lý dữ liệu** 32](#_Toc200558812)

[**3.1.5.** **Xây dựng các chức năng khác** 35](#_Toc200558813)

[**3.2.** **Vận hành và đánh giá kết quả** 36](#_Toc200558814)

[**3.2.1.** **Vận hành** 36](#_Toc200558815)

[**3.2.2.** **Đánh giá thuật toán** 38](#_Toc200558816)

[KẾT LUẬN……. 40](#_Toc200558817)

[TÀI LIỆU THAM KHẢO ...43](#_Toc200558823)

# DANH MỤC HÌNH ẢNH

[Hình 1. Mã lệnh tạo cơ sở dữ liệu. 15](#_Toc200563903)

[Hình 2. Khởi tạo API key cho Maptiler 30](#_Toc200563904)

[Hình 3. Hàm lấy tọa độ từ địa chỉ bằng Maptiler Geocoding API 31](#_Toc200563905)

[Hình 4. Gọi API Maptiler Directions để lấy lộ trình 31](#_Toc200563906)

[Hình 5. Hàm tạo đồ thị từ dữ liệu đường đi trả về 32](#_Toc200563907)

[Hình 6. Hàm heuristic tính khoảng cách Euclid giữa hai điểm 33](#_Toc200563908)

[Hình 7. Cài đặt thuật toán A\* 33](#_Toc200563909)

[Hình 8. Hàm xử lý yêu cầu tìm đường trong frontend (Flask) 34](#_Toc200563910)

[Hình 9. Lưu thông tin tìm đường vào cơ sở dữ liệu SQLite 35](#_Toc200563911)

[Hình 10. Xóa lịch sử tìm đường từ cơ sở dữ liệu. 36](#_Toc200563912)

Hình 11. Thông tin đường đi người dùng nhập vào ............................................... 37

Hình 12. Hình ảnh đường đi hiện trên bản đồ ......................................................... 37

Hình 13. Nếu sử dụng GGMaps Javacript API thay vì Maptiler ........................... 38

Hình 13. Bảng tổng hợp lịch sử tìm kiếm đường đi ................................................ 39

# 

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| Từ viết tắt | Nội dung | Ý nghĩa |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| BFS | Breadth First Search | Tìm kiếm theo chiều rộng |
| DFS | Depth First Search | Tìm kiếm theo chiều sâu |
| A\* | A-Star | Tìm kiếm A\* |
| API | Application Programming Interface | Giao diện lập trình ứng dụng |

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## **Tổng quan về trí tuệ nhân tạo**

### **Khái niệm về trí tuệ nhân tạo**

Ngày nay, cụm từ trí tuệ nhân tạo không còn quá xa lạ và chúng ta có thể được bắt gặp rất nhiều trong các ứng dụng. Trí tuệ nhân tạo (tiếng Anh : Artificial Intelligence - hay còn được gọi tắt là AI) là một lĩnh vực của khoa học máy tính tập trung vào việc tạo ra các hệ thống có khả năng thực hiện các nhiệm vụ đòi hỏi trí thông minh của con người, chẳng hạn học hỏi, suy luận, giải quyết vấn đề, nhận thức, và thậm chí là cả sáng tạo. Có nhiều quan điểm khác nhau về trí tuệ nhân tạo và do vậy, có nhiều định nghĩa khác nhau về lĩnh vực khoa học này.

Mục đích của trí tuệ nhân tạo là xây dựng các thực thể thông minh. Tuy nhiên, do rất khó định nghĩa nào là thực thể thông minh nên cũng khó thông nhất được định nghĩa về trí tuệ nhân tạo (Ví dụ : So sánh giữa 2 người : nhà bác học Einstein và cầu thủ bóng đá Lionel Messi. Nếu xtes trên phương diện bóng đá thì có vẻ như Messi là ngườigiỏi hơn nhưng xét trên phương diện sự thông minh trong nghiên cứu khoa học thì nhà bác học Einstein lại hơn). Vì vậy, định nghĩa của trí tuệ nhân tạo ngày nay định định nghĩa thành bốn nhóm khác nhau:

* ***Hệ thống hành động (thông minh) như người***

Do con người là động vật có trí tuệ nên thuận theo cách tự nhiên sẽ lấy con người làm thước đo sự thông minh của máy tính. Theo định nghĩa, trí tuệ nhân tạo nhằm tạo ra các hệ thống có hành vi và hành độnh tương tự con người, đặc biệt trong những hoạt động liên quan đến trí tuệ. Hệ thống trí tuệ nhân tạo cần có những khả năng sau:

1. Xử lý ngôn ngữ tự nhiên: để có thể phân tích, hiểu câu hỏi và tổng hợp câu trả lời dựa trên một ngôn ngữ giao tiếp thông thường và phố biến nhất hiện nay là tiếng Anh.
2. Biểu diễn tri thức: phục vụ việc lưu giữ thức và thông tin trong hệ thống
3. Suy diễn: sử dụng những tri thức đã có để đưa ra câu trả lười
4. Học máy: sử dụng để thích nghi và học hỏi trong môi trường, hoàn cảnh khác nhau.

* ***Hệ thống suy nghĩ (thông minh) như người***

AI hành động một cách thông minh dựa trên quá trình phân tích dữ liệu tương tự quá trình suy nghĩ của con người, từ đây mô hình hóa và tạo ra những hệ thống có mô hình nhận thức, tư duy tương tự con người hay nói cách khác là mô phỏng bộnão con người. Tuy nhiên, suy nghĩ của con người luôn luôn linh hoạt và thay đổi bất ngờ, nên đây cũng chính là bước cản lớn nhất khi phát triển AI. *Hệ thống hành động một cách hợp lý*

Con người chúng ta là loài vật suy nghĩ và hành động bị chi phối bởi những tâm lý và cảm xúc khác nhau. Do vậy, không phải lúc nào con người cũng suy nghĩ và hành động theo hướng đạt kết quả tốt. Từ đây xuất hiện cách tiếp cận theo hướng xây dựng các hệ thống cho phép đạt tới kết quả tốt mà không cần học theo con người.

* ***Hệ thống suy nghĩa hợp lý***

Đây là cách tiếp cận tập trung vào việc xây dựng các tác tử (agent) có khả năng hành động hợp lý, tức là hành động đem lại kết quả tốt nhất hoặc kết quả kỳ vọng tốt nhất khi có yếu tố không chắc chắn. Cách tiếp cận này được coi là tổng quát và bao gồm các cách tiếp cận khác: vì hành động hợp lý có thể bao gồm suy diễn hoặc không, có thể dựa trên cách suy nghĩ giống người hoặc không, có thể gồm các kỹ thuật đùng để vượt qua phép thử Turing

Các hệ thống 1 và 2 liên quan đến việc lấy con người là thước đo mức độ thành công còn hệ thống 3 và 4 đánh giá dựa trên tiêu chuẩn của sự hợp lý (một hệ thống hợp lý nếu nó làm việc phù hợp với những gì hệ thống biết). Ngoài ra, hệ thống 2 và 4 liên quan đến các suy nghĩ và suy diễn còn hệ thống 1 và 3 liên quan đến các hành động.

### **Phân loại**

Dựa theo khả năng của máy, AI phân thành 4 loại :

1. **AI phản ứng** : có khả năng phản ứng với các kích thích. Nhược điểm của loại AI này là không thể học được và không có trí nhớ. Nó giống như các động vật cấp thấp có phản ứng được như không có trí nhớ như con người.
2. **AI có bộ nhớ hạn chế** : có khả năng học và ghi nhớ, có khả năng vận dụng các kinh nghiệm để đưa ra các quyết định cho hiện tại và tương lai. Đặc điểm là khi AI này học trong một lĩnh vực cụ thể nào đó nếu ta cung cấp đủ dữ liệu thì khả năng học của loại AI này là rất nhanh. Machine Learning và Deep Learning đều thuộc loại AI này.
3. **AI có tâm trí** : là loại AI có thể nắm bắt được cảm xúc, tâm trí của đối tượng mà nó giao tiếp, nó có thể tương tác với đối tượng về mặt xã hội. Ví dụ như đối tượng giao tiếp của AI là con người, khi chúng ta buồn thì chúng có thể nắm bắt được cảm xúc của chúng ta và thực hiện một số hành động để an ủi chúng ta.
4. **AI nhận thức** : là loại AI cao cấp nhất. Nó có khả năng tự nhận thức được cảm xúc nội tại, trạng thái bên trong bản thân nó. Nếu ta đạt được AI tự nhận thức thì ta sẽ đạt đến thiết bị trí tuệ nhân tạo giống hệt với con người và lúc đó sẽ xảy ra các hiện tượng như giống như con người cũng có tình yêu giữa các thiết bị AI hay thậm chí là giữa AI với con người.

### **Lịch sử hình thành và phát triển**

**Giai đoạn tiền khởi đầu (1943 – 1955)** : Ý tưởng về máy móc có thể suy nghĩ xuất hiện từ thời kỳ cổ đại. Năm 1943, McCulloch và Pitts giới thiệu mô hình mạng nơ-ron nhân tạo đầu tiên. Đến năm 1950, Alan Turing đề xuất bài kiểm tra Turing và khái niệm học máy. Năm 1956, tại hội nghị Dartmouth, thuật ngữ “trí tuệ nhân tạo” (AI) chính thức ra đời.

**Giai đoạn khởi đầu (1952 – 1969)** : Năm 1952, Arthur Samuel phát triển chương trình chơi cờ đam (checkers) có khả năng tự học từ kinh nghiệm, giúp chương trình có khả năng đánh bại cả người chơi nghiệp dư. Năm 1956, Jonh McCarthy phát triển ngôn ngữ lập trình Lisp, một trong những ngôn ngữ chính của AI trong nhiều năm. Trong khi đó, Newell và Simon phát triển hệ thống giải quyết vấn đề chung cư GPS. Minsky nghiên cứu “microworlds” và hệ thống giải toán SAIN năm 1963.

**Giai đoạn trầm lắng (1070 – 1980)** : AI gặp thách thức như ban đầu dựa vào tim kiếm các hành động, nhưng khi bài toán trở nên phức tạp, số lượng các khả năng cần xem xét trở nên quá lớn để xử lý và các thách thức khithiếu tri thức chuyên ngành, dẫn đến việc cắt giảm kinh phí từ các chính phủ như Mỹ và Anh dẫn đến “mùa đông AI’ đầu tiên (1974 – 1980)

**Hệ thống dựa trên tri thức (1969 – 1979)** : là giai đoạn chú trọng vào việc tích hợp tri thức chuyên ngành vào hệ thống AI, dẫn đến sự ra đời của các hệ chuyên gia như DENDRAL (1967) và MYCIN (1974). Năm 1972 Alain Colmerauer phát triển ngôn ngữ Prolog một ngôn giúp biểu diễn và suy tri thức dưới dạng các luật logic.

**Giai đoạn thương mai hóa (1980 – nay)** : Hệ chuyên gia được thương mại hóa vào những năm 1980. Mạng Nơ-ron và học sâu quay trở lại và rồi năm 1981, Nhật bản khởi động chương trình máy tính thế hệ thứ 5, kết thúc “mùa đông AI” đầu tiên.

**Trí tuệ nhân tạo chính thức trở thành ngành khoa học ( 1987 – nay):** AI trở thành lĩnh vực khoa học chính thức, với các phương pháp dựa trên thực nghiệm và thống kê như mô hình Markov và mạng nơ-ron nhiều lớp dùng trong ứng dụng nhận dạng tiếng nói và hình ảnh. AI phát triển dựa vào các ứng dụng thực tế với dữ liệu nhiều hơn, giúp xây dựng khả năng đưa ra các quyết định chính xác hơn.

**Cách tiếp cận dựa trên dữ liệu lớn (2001 – nay)** : Sự bùng nổ dữ liệu lớn và Internet thúc đẩy ngành AI phát triển. Các hệ thống như Google Translate, Siri hay Watson được xây dựng dựa trên xử lý lượng lớn dữ liệu khổng lồ từ bigdata, thay vì các quy tắc logic truyền thống.

1. **Những vấn đề chưa được giải quyết trong AI**

Mặc dù hiện nay AI đã đạt được nhiều tiến bộ để từng bước hiện đại hơn, tuy nhiên vẫn còn những vẫn đề mà AI chưa thể giải quyết được:

* Chưa trình tự sinh ra hàm heuristic
* Chưa có khả năng xử lý song song của con người
* Chưa có khả năng diễn giải một vấn đề theo nhiều phương pháp khác nhau như con người
* Chưa có khả năng học như con người
* Chưa có khả năng tự thích nghi với môi trường
* Chưa có tính đạo đức và quyết định đúng đắn trong mọi hoàn cảnh

## **Giới thiệu về đề tài**

Trong cuộc sống ngày nay, việc di chuyển giữa các địa điểm trong thành phố phục vụ cho các công việc cũng như đời sống trở nên hết sức cần thiết. Đặc biệt là trong khu vực thành phố Hà Nội bởi đây là khu vực trung tâm kinh tế của cả nước, cũng là nơi có tình trạng giao thông và cơ sở hạ tầng giao thông vô cùng phức tạp. Vì vậy, nhóm chúng em đã chọn nghiên cứu và xây dựng trang web dựa vào thuật toán A\* để có thể cung cấp gợi ý đường đi ngắn nhất giữa hai địa điểm bất kỳ trong thủ đô Hà Nội.Thuật toán A\* sẽ được mô phỏng trên các dạng đồ thị hoặc bản đồ để có thể tìm ra cung đường ngắn nhất nếu nó tồn tại. Với đề tài này, chúng em hi vọng sẽ cung cấp được cho mọi người về kiến thức ứng dụng để tìm ra đường đi ngắn nhất giữa hai địa điểm mà mọi người có nhu cầu tìm kiếm.

## **Các công nghệ được sử dụng**

### **Giới thiệu ngôn ngữ lập trình Python**

Python là một ngôn ngữ lập trình bậc cao, thông dịch, hướng đối tượng và đa mục đích, được tạo bởi Guido van Rossum vào năm 1991. Python được sử dụng rộng rãi trong các ứng dụng Web, phát triển phần mềm, khoa học dữ liệu mà máy học(ML). Python nổi bật nhờ cú pháp rõ ràng, dễ học, chạy trên nhiều nền tảng khác nhau và hệ thống thư viện lớn giúp nó trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới.

***Ưu điểm:***

* Cấu trúc đơn giản dễ dàng đọc và hiểu vì ngôn ngữ này có cú pháp cơ bản giống tiếng Anh
* Mã nguồn mở, miễn phí
* Là ngôn ngữ lập trình hướng đối tượng nên rất mạnh và mang lại nhiều lợi ích cho phát triển phần mềm.
* Tích hợp thư viện tiêu chuẩn lớn, chứa dòng mã có thể tái sử dụng cho hầu hết mọi tác vụ.
* Tương thích với nhiều nền tảng hỗ trợ lập trình, có thể sử dụng trên nhiều hệ điều hành khác nhau của máy tính.
* Ngôn ngữ thông dịch cao cấp
* Khả năng mở rộng và có thể nhúng

***Nhược điểm:***

* Tốc độ thực thi chậm
* Hạn chế của nhà phát triển
* Tiêu thụ bộ nhớ lớn
* Không thích hợp cho nhà phát triển trò chơi và thiết bị di động
* Phát hiện lỗi trong mã
* Hạn chế thiết kế và khó kiểm tra

### **Các thư viện sử dụng trong đề tài**

Python là ngôn ngữ lập trình nổi tiếng với hệ thống thư viện rộng lớn. Cho nên, trong đề tài của chúng em, chúng em đã sử dụng các thư viện để thiết kế trang web của chúng em:

* ***Sqlite3:*** đây là thư viện tích hợp trong Python để làm việc với cơ sở dữ liệu SQLite. Nó cung cấp một cách đơn giản và hiệu quả để tương tác với cơ sở dữ liệu SQLite để tạo, đọc, cập nhật và xóa bằng các đoạn code Python.
* ***Flask:*** là một micro – framework được viết bằng ngôn ngữ lập trình Python dùng để phát triển web. Đặc điểm chính của Flask nó nhẹ và linh hoạt, phù hợp các dự án nhỏ hoặc trung bình, nhưng vẫn đủ mạnh để mở rộng khi cần. Các thành phần quan trọng:

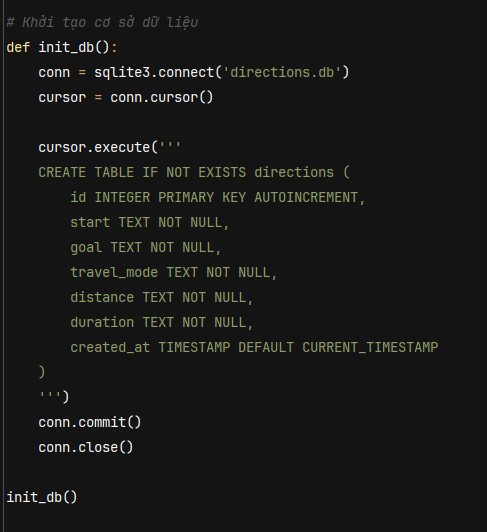
1. *Flask:* cung cấp lớp nền cho ứng dụng web.
2. *Request:* xử lý các yêu cầu giao thức HTTP như GET, POST, PUT, DELETE,... từ phía người dùng trang web.
3. *Jsonify:* Biến đổi các đối tượng Python (như dictinonary, list, str,...) thành JSON, một định dạng dữ liệu phổ biến khi giao tiếp giữa client và server (thường được sử dụng trong các API).
4. *Render\_template*: được dùng để kết hợp mã Python với HTML bằng cách sử dụng Jinja2 template engine (mặc định của Flask). Điều nay cho phép chúng em xây dựng các giao diện bằng cách truyền dữ liệu từ backend vào các file HTML. Các file HTML dùng render\_template được chúng em lưu trong thư mục templates.
5. *Ngoài ra route:* dùng để định nghĩa các URL mà người dùng có thể truy cập. Mỗi URL được gắn với một hàm Python (view function) dể xử lý yêu cầu từ người dùng.

* ***Googlemaps:*** thư viện này cung cấp các API để làm việc với dịch vụ Google Maps, cho phép ứng dụng truy xuất dữ liệu bản đồ như chỉ đường, thông tin địa lý,
* ***Heapq:*** chúng em đã sử dụng thư viện này để hỗ trợ cấu trúc dữ liệu heap (đống), một dạng hàng đợi ưu tiên. Heap hỗ trợ thao tác lấy giá trị lớn nhất hoặc nhỏ nhất một cách nhanh chóng. Thư viện này hỗ trợ rất hữu ích cho việc tìm phần tử ưu tiên cao nhất trong thuật toán A\* mà chúng em nghiên cứu trong đề tài này.
* ***Time:*** thư viện tiêu chuẩn của Python để làm việc với thời gian. Chúng em dùng thư viện này để đo việc thực thi của mã lệnh (ví dụ như time.sleep() – time.time())
* ***Datetime:*** là thư viện tích hợp trong Python để làm việc với ngày giờ. Cung cấp các phương thức để thao tác với ngày tháng, thời gian.

### **Giới thiệu hệ quản trị dữ liệu và Maptilper API**

SQLite là phần mềm quản trị cơ sở dữ liệu tương tự Mysql, PostgreSQL,... Đặc điểm của hệ quản trị này là không cần máy chủ, không cần cấu hình, nhanh, nhỏ gọn, có độ tin cậy cao và hỗ trợ hầu như đầy đủ các tính năng cần thiết trong chuẩn SQL92. Ngoài ra nó thích hợp với những dự án vừa và nhỏ với khối lượng yêu cầu truy cập dữ liệu đồng thời không cao.

Trong đề tài của chúnge em có sử dụng SQLite để lưu thông tin đường đã tìm kiếm cùng với độ dài quãng đường, thời gian di chuyển, phương thức di chuyển. Ngoài ra còn lưu được thời gian tìm kiếm và thực hiển các chức năng như sort hay là search.



Hình 1. Mã lệnh tạo cơ sở dữ liệu.

Tiếp theo chính là Maptilper API***:***

Trước tiên, chúng em xin giới thiệu khái niệm về API. API (Application Programming Interface) là giao diện lập trình ứng dụng, về cơ bản là một cầu nối trung gian giúp các hệ thống có thể tương tác với nhau. Ví dụ, khi người dùng ấn nút "like" trên Facebook, front-end sẽ gửi một gói dữ liệu thông qua API đến back-end để xử lý hành động đó, lưu vào cơ sở dữ liệu, từ đó phản ánh lên giao diện người dùng.

Hiện nay có rất nhiều API phổ biến, trong đó RESTful API – tương tác qua giao thức HTTP – là dạng được sử dụng nhiều nhất.

Maptiler API là một tập hợp các công cụ và dịch vụ được cung cấp bởi Maptiler, cho phép các nhà phát triển tích hợp bản đồ, dữ liệu địa lý và các chức năng liên quan vào ứng dụng hoặc trang web. Maptiler API hỗ trợ các tác vụ như hiển thị bản đồ, tìm kiếm địa điểm, định tuyến,... và là một giải pháp thay thế hiệu quả cho việc thu thập dữ liệu bản đồ thủ công – vốn tốn nhiều thời gian và công sức mà độ chính xác không được đảm bảo.

**Ưu điểm:**

* Dễ sử dụng: Có tài liệu hướng dẫn chi tiết, công khai, giúp quá trình tích hợp nhanh chóng.
* Tính chính xác: Dữ liệu bản đồ được cập nhật thường xuyên, đảm bảo độ tin cậy.
* Tùy biến linh hoạt: Cho phép tùy chỉnh giao diện và chức năng bản đồ phù hợp với nhu cầu người dùng.

**Nhược điểm:**

* Chi phí: Dù Maptiler có cung cấp gói miễn phí, nhưng khi sử dụng với lưu lượng truy cập lớn hoặc nhiều API có thể phát sinh chi phí.
* Giới hạn truy cập: Các gói miễn phí có giới hạn số lượt yêu cầu mỗi ngày, cần có chiến lược tối ưu hóa để tránh vượt quá hạn mức.

Việc tích hợp Maptiler API mang lại rất nhiều lợi ích cho đề tài của chúng em, giúp tiết kiệm thời gian và công sức so với việc xây dựng dữ liệu bản đồ từ đầu. Để truy cập API của Maptiler, người dùng cần có API Key – đây là mã định danh giúp nhà cung cấp quản lý việc truy cập và xác minh quyền sử dụng dịch vụ.

* + 1. **Các ngôn ngữ lập trình khác**

Với đề tài này, chúng em đã sử dụng thêm ngôn ngữ lập trình JavaScript và ngôn ngữ HTML. HTML (viết tắt của cụm từ: Hypertext Markup Language) tạm dịch là ngôn ngữ đánh dấu siêu văn bản. HTML được sử dụng để tạo và cấu trúc các thành phần trong trang web trên World Wide Web hoặc ứng dụng, phân chia các đoạn văn, heading, titles, blockquotes… và HTML không phải là ngôn ngữ lập trình.

JavaScript là ngôn ngữ lập trình phổ biến nhất hiện nay và là ngôn ngữ lập trình cao cấp. JavaScript là ngôn ngữ không thể thiếu trong thế giới Web, đóng vai trò quan trọng trong việc xây dựng và tối ưu trải nghiệm của người dùng. Đây là một ngôn ngữ tương đối dễ học và dễ dùng, chúng thường bị nhầm lẫn với Java vì một số điểm khá tương đồng.

***Ưu điểm:***

* Dễ học và dễ thực hiện
* Được sử dụng rộng rãi trên web
* Có thể chạy ngay trong trình duyệt ở phía máy khách
* Giảm nhu cầu trên máy chủ trang web

***Nhược điểm:***

* Đôi khi có thể được giải thích khác nhau bởi các trình duyệt khác nhau, điều này gây khó khăn cho việc viết mã trên nhiều trình duyệt.
* JavaScript là ngôn ngữ lập trình phổ biến nhất hiện nay, nên vì thế chúng có một cộng đồng hỗ trợ vô cùng rộng lớn, là công cụ vô cùng hữu ích để chạy một chương trình lên trang web và khả năng tương thích mạng lại sự trải nghiệm cho người dùng nhất quán trên mọi thiết bị.

# CHƯƠNG 2: ÁP DỤNG THUẬT TOÁN VÀO TÌM ĐƯỜNG ĐI NGẮN NHẤT TRONG THỦ ĐÔ HÀ NỘI

## **2**.**1. Giới thiệu các thuật toán chung**

Trong thuật toán tìm kiếm liên quan đến ngành trí tuệ nhân tạo, có rất nhiều thuật toán hay và phù hợp cho các bài toán. Nhóm chúng em sẽ tìm hiểu hai nhóm thuật toán tìm kiếm: Tìm kiếm không có thông tinvà tìm kiếm có thông tin.

### **2.1.1. Tìm kiếm không có thông tin**

Thuật toán tìm kiếm không có thông tin (Uninformed Search) hay còn gọi là tìm kiếm mù là loại thuật toán tìm kiếm không sử dụng bất kỳ thông tin chi tiết nào về bài toán ngoài thông tin cần thiết về định vị rõ trạng thái hiện tại và các hành động có thể thực hiện từ trạng thái đó. Một số thuật toán tìm kiếm mù phổ biến bao gồm: Tìm kiếm theo chiều sâu (DFS), tìm kiếm theo chiều rộng (BFS), sử dụng thuật toán Dijkstra (UCS).

* ***Tìm kiếm theo chiều rộng***

**Định nghĩa:** Thuật toán tìm kiếm theo chiều rộng (BFS – Breadth First Search) là thuật toán tìm kiếm tất cả các đỉnh ưu tiên chiều rộng của một đồ thị hoặc cấu trúc dữ liệu cây. Để tránh xử lý lại cùng mộ nút, chúng ta sẽ sử dụng mảng boolean đã truy cập, mảng này sẽ đánh dấu các đỉnh đã truy cập. BFS sử dụng cấu trúc dữ liệu hàng đợi (queue) (FIFO) để tìm đường đi ngắn nhất trong đồ thị.

**Mô tả cách thức hoạt động của thuật toán:**

1. Lấy một đỉnh bất kỳ trong đồ thị thêm vào cuối hàng đợi.
2. Lấy phần tử đầu tiên của hàng đợi và thêm nó vào danh sách đã duyệt.
3. Tạo một danh sach các đỉnh liền kề của đỉnh đang xét. Thêm những đỉnh không có trong danh sách đã duyệt vào cuối hàng đợi
4. Tiếp tục lặp lại những bước 2 và 3 cho đến khi hàng đợi trống.
5. Lưu ý: Đồ thị có thể chứa hai thành phần không liên kết khác nhua, vì vậy để đảm bảo rằng mọi đỉnh đều đã được ghé thăm, chúng ta cũng có thể chạy thuật toán BFS trên mọi đỉnh.

**Tính chất của thuật toán BFS:**

**Đầy đủ**: BFS là thuật toán đầy đủ, nếu có lời giải là tìm được đường đi (nếu đích là hữu hạn)

**Tối ưu**: Có (Nếu giá thành của mọi bước đi là bằng nhau, thuật toán tìm lời giải nông nhất). Nếu không bằng nhau thì chưa chắc.

**Ưu điểm**: Tìm được đường đi ngắn nhất trong đồ thị không có giá thành

**Nhược điểm**: Tốn nhiều bộ nhớ khi khi lưu tất cả các nút O(*bd+1*) (trong đó: b là độ rẽ nhánh, d là chiều sâu của nút đích)

🡺Bộ nhớ và thời gian đều lớn, do vậy không chạy được các bài toán có kích thước lớn.

* **Tìm kiếm theo chiều sâu**

**Định nghĩa:** Thuật toán tìm kiếm theo chiều sâu (DFS – Depth First Search) là thuật toán duyệt qua hoặc tìm kiếm cấu trúc dữ liệu dạng cây hoặc đồ thị. Thuật toán bắt đầu từ nút gốc (chọn một nút tùy ý làm nút gốc trong trường hợp đồ thị) và kiểm tra từng nhánh càng xa càng tốt trước khi quay lui. Thuật toán duyệt đỉnh dựa theo phương pháp LIFO.

**Mô tả cách thức hoạt động của thuật toán:**

1. Bắt đầu bằng cách đặt bất kỳ đỉnh nào của đồ thị trên cùng của ngăn xếp.
2. Lấy đỉnh trên cùng của ngăn xếp và thêm nó vào danh sách đã truy cập.
3. Tạo danh sách các nút liền kề của đỉnh đó.Thềm các nút không có trong danh sách đã truy cập vào đầu ngăn xếp
4. Tiếp tục lặp bước 2 và 3 cho đến khi ngăn xếp trống.

**Tính chất của thuật toán DFS:**

**Đầy đủ:** Không, trong trường hợp không gian trạng thái có độ sâu vô hạn

**Tối ưu:** Không

**Thời gian:** O(*bm*) rất lớn nếu m lớn hơn d(trong đó: b là độ rẽ nhánh, m là độ sâu của không gian trạng thái , d là độ số của nút đích) . Nếu có nhiều lời giải thì có thể nhanh hơn tìm kiếm theo chiều rộng

**Bộ nhớ:** O(*bm*) tốt hơn nhiều so với chiều rộng.

### **2.1.2. Tìm kiếm có thông tin**

Thuật toán tìm kiếm có thông tin (Informed Search) là một trong những phương pháp tìm kiếm trong trí tuệ nhân tạo sử dụng thông tin đặc biệt về bài toán để hỗ trợ quá trình ra quyết định. Thông tin đặc biẹt này thường được gọi là hàm heuristic. Đây là hàm ước lượng về khoảng cách hoặc chi phí từ trạng thái hiện tại đến trạng thái mục tiêu. Một số thuật toán nổi bật của tìm kiếm này là: tìm kiếm tham lam và tìm kiếm A\* (A-Star),...

* **Tìm kiếm tham lam**

**Định nghĩa:** Là thuật toán tìm kiếm đường đi dựa vào hàm ước lượng giá thành đường đi (hàm heuristic). “Tham lam” chính là chọn nút có vẻ tốt nhất để mở rộng, không quan tâm đến tương lai. Phương pháp của thuật toán này chính là mở rộng nút có giá thành đường đi tới đích là nhỏ nhất trước.

* f(n) = h(n) (trong đó: h(n) là hàm heuristic ước lượng giá thành đường đi từng nút n tới đích)

**Tính chất của tìm kiếm tham lam:**

**Đầy đủ**: Không (vì trong trường hợp bài toán có vô số nút mà có hàm h(n) đủ nhỏ thì không tìm được đích => vòng lặp)

**Tối ưu**: Không (vì thuật toán bị ảnh hưởng với hàm heuristic, vì vậy nếu hàm ước lượng quá cao sao với thực tế thì sẽ việc tìm kiếm sẽ bị ảnh hưởng).

**Thời gian:** O(*bm*) với b là độ rẽ nhánh, m là độ sâu của không gian trạng thái.

Nếu hàm ước lượng heuristic tốt, thuật oán có thể sẽ nhanh hơn rất nhiều.

**Bộ nhớ** : O(*bm*) lưu tất cả các nút trong bộ nhớ.

Nếu hàm heuristic tốt, số nút cần lưu giảm đi rất nhiều.

**Tìm kiếm A\***

**Định nghĩa**:Cùng với thuật toán tìm kiếm tham lam, tìm kiếm A\* cũng là thuật toán tìm kiếm nằm trong tìm kiếm có thông tin. Tìm kiếm A\* (A-Star) là thuật toán tìm kiếm trong đồ thị và tìm đường, được sử dụng trong nhiều lĩnh vực khoa học máy tinh do tính hoàn chỉnh, tối ưu và hiệu quả tối ưu của nó. Thuật toán A\* dựa trên thuật toán tìm kiếm theo chiều rộng (BFS) và đặc biệt khắc phục những nhược điểm của thuật toán tìm kiếm tham lam (chỉ phụ thuộc vào heuristic) để tìm đường đi ngắn nhất. Thuật toán A\* hoạt động dựa vào việc sử dụng đánh giá f(n), quyết định thứ tự duyệt các nút trong đồ thị, với n là một nút của đồ thị.

Phương pháp của bài toán là tìm nút n có f(n) = g(n) + h(n) nhỏ nhất để mở rộng:

g(n): giá thành đường đi từ điểm xuất phát tới nút n∉

h(n): hàm heuristic ước lượng giá thành đường đi từng nút n tới đích

f(n): ước lượng giá thành đường đi từ điểm xuất phát, qua n, tới đích

**Mô tả thuật toán tìm kiếm A\*:**

***Đầu vào***: bài toán tìm kiếm, hàm heuristic h

***Đầu ra***: đường tới nút đích

***Khởi tạo***: tập các nút biên (nút mở) O = S

***While (O ≠ ∅) do***:

Lấy nút n có f(n) là nhỏ nhất khỏi O

if n thuộc G return (Đường đi tới n)

***Với mọi m thuộc P(n):***

g(m) = g(n) + c (n, m)

f(m) = g(m) + h(m)

Thêm m vào O cùng với giá trị f(m)

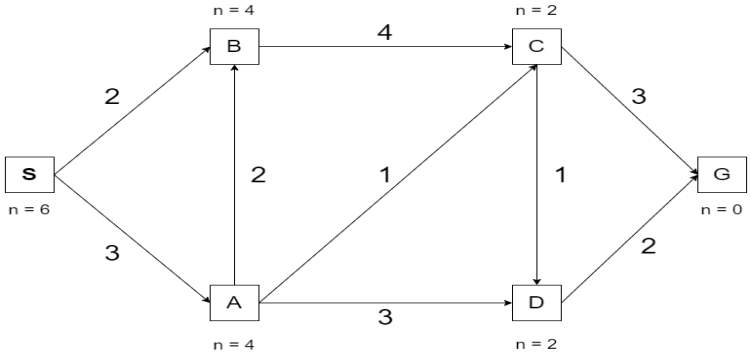
return không tìm được đường đi;

***Xử lý các nút lặp***: nếu nút có chi phí tốt hơn:

Đưa lại vào danh sách nếu đã mở rộng

Cập nhật lại nút cũ có giá thành kém hơn nếu đang trong tập biên

*Ví dụ: Tìm đường đi từ S đến G*



|  |  |  |
| --- | --- | --- |
| STT | Nút được mở rộng | Tập biên O |
| 0 |  | S(6) |
| 1 | S | C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps12.jpg |
| 2 | C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps13.jpg | C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps14.jpg |
| 3 | C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps15.jpg | C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps16.jpg |
| 4 | C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps17.jpg | C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps18.jpg |
| 5 | C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps19.jpg | C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps20.jpg |
| 6 | G | Đích |

Đường đi : G C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps21.jpg C C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps22.jpg A C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps23.jpg S

Giá thành: c = 7

**Ưu và nhược điểm của thuật toán tìm kiếm A\*:**

***Ưu điểm****:*

* **Hiệu quả**: A\* thường rất nhanh và tìm đường đi ngắn nhất nhờ sử dụng thông tin heuristic.
* **Đảm bảo tối ưu**: A\* đảm bản tìm được đường ngắn nhất nếu hàm heuristic h(n) không vượt quá chi phí thực tế điểm hiện tại đến điểm đích.

***Nhược điểm****:*

* **Tiêu tốn bộ nhớ**: A\* có thể tiêu tốn nhiều bộ nhớ vì lưu trữ nhiều nút trong danh sách mở.
* **Phụ thuộc vào hàm heuristic**: Hiệu quả của A\* phụ thuộc vào chất lượng của hàm heuristic. Nếu hàm heuristic không tốt, thuật toán có thể trở nên kém hiệu quả.

## **2.2. Áp dụng thuật toán A\* vào tìm đường đi ngắn nhất trong thủ đô Hà Nội**

### **2.2.1. Ý tưởng xây dựng**

Mục tiêu của đề tài này là ứng dụng thuật toán A\* để tìm đường đi ngắn nhất trong thành phố Hà Nội. Để thực hiện điều này, chúng ta cần mô hình hóa thành phố dưới dạng đồ thị (graph), trong đó các giao lộ được coi là các node và các đoạn đường đi giữa các đỉnh là các cạnh (edges).

**Xây dựng mô hình dưới dạng đồ thị:**

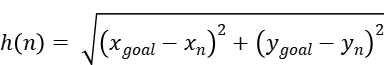
**Đỉnh (Nodes)**: Mỗi giao lộ hoặc địa điểm quan trọng trong thành phố sẽ được biểu diễn dưới dạng một đỉnh trong đồ thị.

**Cạnh (Edges)**: Các đoạn đường giữa hai giao lộ sẽ là cạnh nối giữa các đỉnh và mỗi cạnh sẽ có trọng số là khoảng cách giữa hai đỉnh hoặc thời gian dự kiến để di chuyển qua đoạn đường đó.

**Trọng số của cạnh**: Trọng số có thể được tính toán dựa trên khoảng cách thực tế (sử dụng dữ liệu bản đồ)

**Hàm ước lượng h(n)** : Để ước lượng khoảng cách giữa hai giao lộ, chúng ta có thể sử dụng khoảng cách Euclide giữa hai điểm hoặc áp dụng các thông tin giao thông thực tế để cải thiện độ chính xác.

**Xây dựng hàm heuristic h(n):** hàm heuristic h(n) là hàm ước lượng chi phí từ điểm n đến điểm đích. Vì vậy, nếu hàm ước lượng này càng chính xác, thì thuật toán tìm đường đi ngắn càng chính xác. Để ước lượng được hàng heuristic từ điểm bất kỳ đến đích, chúng em dựa nào công thức tính khoảng cách trong không gian Euclidean:



Trong đó: (x*goal*, y*goal*) là tọa độ của điểm đích.

(x*n*, y*n*) là tọa độ của đỉnh hiện tại.

### **2.2.2. Quy trình vận hành thuật toán A\* để tìm đường đi:**

Bước tiếp theo là triển khai thuật toán A\* trên mô hình đồ thị của thành phố Hà Nội. Dưới đây là quy trình chi tiết từng bước để tìm đường đi ngắn nhất giữa hai điểm bất kỳ:

1. ***Khởi tạo:***

* Tạo danh sách mở (open list) chứa điểm bắt đầu và danh sách đóng (closed list) rỗng.
* Gán chi phí g(start) = 0 và tính f(start) = g(start) + h(start) cho đỉnh bắt đầu, trong đó h(start) là ước lượng khoảng cách từ điểm bắt đầu đến đích.

1. ***Lặp lại cho đến khi tìm được đích hoặc danh sách rỗng:***

* Chọn đỉnh n có giá trị f(n) nhỏ nhất từ danh sách mở.
* Nếu n là đích, kết thúc và trả về đường đi ngắn nhất.
* Di chuyển n từ danh sách mở sang danh sách đóng (để đánh dấu rằng nó đã được khám phá)

1. ***Khám phá các đỉnh kề:***

* Đối với mỗi đỉnh m kề với n (đỉnh hiện tại):
* Nếu m đã có trong danh sách đóng, bỏ qua.
* Tính giá trị g(m) = g(n) = cost(n, m) là chi phí di chuyển từ n đến m
* Tính toán h(m) (heuristic ước lượng từ m đến đích)
* Tính toán giá trị tổng chi phí f(m) = g(m) + h(m).

1. ***Cập nhật danh sách mở:***

* Nếu m chưa có trong danh sách mở, thêm nó vào danh sách mở với giá trị f(m).
* Nếu m đã có trong danh sách mở nhưng giá trị g(m) mới tính toán nhỏ hơn giá trị trước đó, cập nhật lại thông tin của đỉnh m trong danh sách mở (gồm cả giá trị g(m), h(m) và f(m).

1. ***Lặp lại*:** Tiếp tục lặp lại quy trình cho đến khi đạt đến đích hoặc danh sách mở rộng.
2. ***Trả về kết quả*:** Khi thuật toán đạt đến đích, nó sẽ truy vết ngược từ đích về điểm xuất phát để xác định đường đi ngắn nhất.

# CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

## **3.1. Quy trình thực nghiệm**

### **Mô tả thực nghiệm**

**Mô tả vị trí*:***

Vị trí địa lý Hà Nội nằm ở phía Tây Bắc của trung tâm đồng bằng châu thổ sông Hồng, vị trí của thành phố Hà Nội có tọa độ 21.0278 ° vĩ độ Bắc và 105.8342 ° kinh độ Đông. Hà Nội chính là trung tâm kinh tế và chính trị của cả nước bao gồm. Vì vậy, giao thông Hà Nội có tầm quan trọng trong việc đáp ứng nhu cầu đi lại và vận chuyển hàng hóa với mạng lưới rất nhiều loại đường : như đường bộ, đường sắt, đường thủy, đường hàng không,...Cho đến cuối năm 2013, trên địa bàn thành phố Hà Nội khoảng 16.132 km đường bộ (***[tài liệu](https://vi.wikipedia.org/wiki/Giao_th%C3%B4ng_H%C3%A0_N%E1%BB%99i)****).* Con số này có lẽ tăng lên một cách chóng mặt khi Hà Nội đanng đẩy mạnh phát triển giao thông đường bộ trong thời gian gần đây để đảm bảo được nhu cầu đi lại cho người dân. Vì vậy, chúng em đã có hứng thú với đề tài này để có thể hỗ trợ mọi người trong việc di chuyển giữa các địa điểm.

**Xác định thiết kế trang** **web:**

Chúng em đã xác định và lên phương án để thiết kế một trang web hoàn chỉnh đến tay người dùng. Chúng em sẽ sử dụng một thuật toán là A\* để giúp chúng ta tìm được quãng đường ngắn nhất trên bản đồ khu vực Hà Nội. Để có thể áp dụng được thuật toán ta phải lập trình và đưa thuật toán này vào đề tài của chúng ta. Lập trình web sẽ chia thành hai thành phần cơ bản là phần Backend và Frontend.

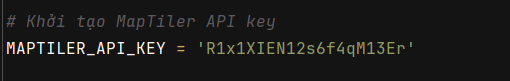
**Backend:** Chúng em xây dựng thuật toán A\* bằng ngôn ngữ lập trình Python để xử lý và tính toán đường đi tối ưu. Để có thể thực hiện các phép tính và trả về kết quả, hệ thống sử dụng Maptiler API để lấy dữ liệu bản đồ, đồng thời sử dụng SQLite để lưu trữ và hiển thị kết quả truy vấn đường đi.

**Frontend:** Chúng em sử dụng HTML để xây dựng giao diện trang web và JavaScript để phát triển các chức năng, đồng thời gửi các yêu cầu từ người dùng đến Backend và Maptiler API. Việc tích hợp Maptiler API là rất quan trọng vì nó giúp hiển thị bản đồ, địa điểm tìm kiếm, khoảng cách, thời gian di chuyển, và các thành phần liên quan khác một cách trực quan.

### **Thu thập dữ liệu**

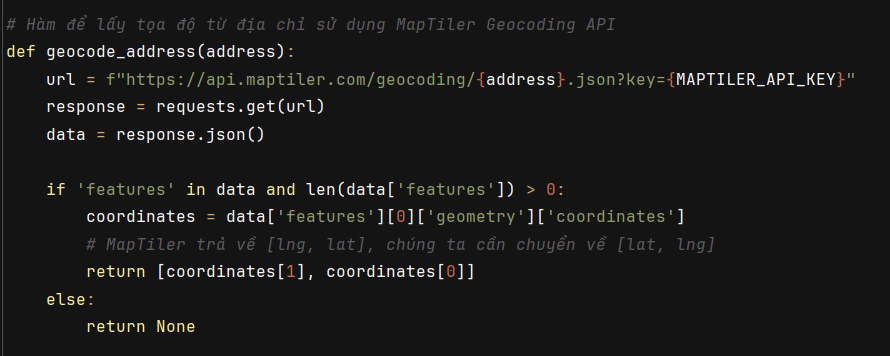
Trong quá trình thu thập dữ liệu, trang web sử dụng Maptiler API để lấy dữ liệu chỉ đường giữa các điểm cụ thể, bao gồm các thông tin chi tiết như tọa độ điểm xuấtphát và điểm đến, khoảng cách giữa các điểm, cũng như thời gian di chuyển dự kiến. Việc sử dụng API giúp đảm bảo rằng dữ liệu địa lý luôn chính xác, được cập nhật thường xuyên, và phù hợp với nhiều phương thức di chuyển khác nhau.

Bước đầu tiên là khởi tạo client của Maptiler API thông qua API key, từ đó cho phép hệ thống truy cập các dịch vụ của Maptiler để lấy dữ liệu chỉ đường hoặc các dữ liệu bản đồ liên quan khác phục vụ cho việc xử lý và hiển thị kết quả trên trang web.



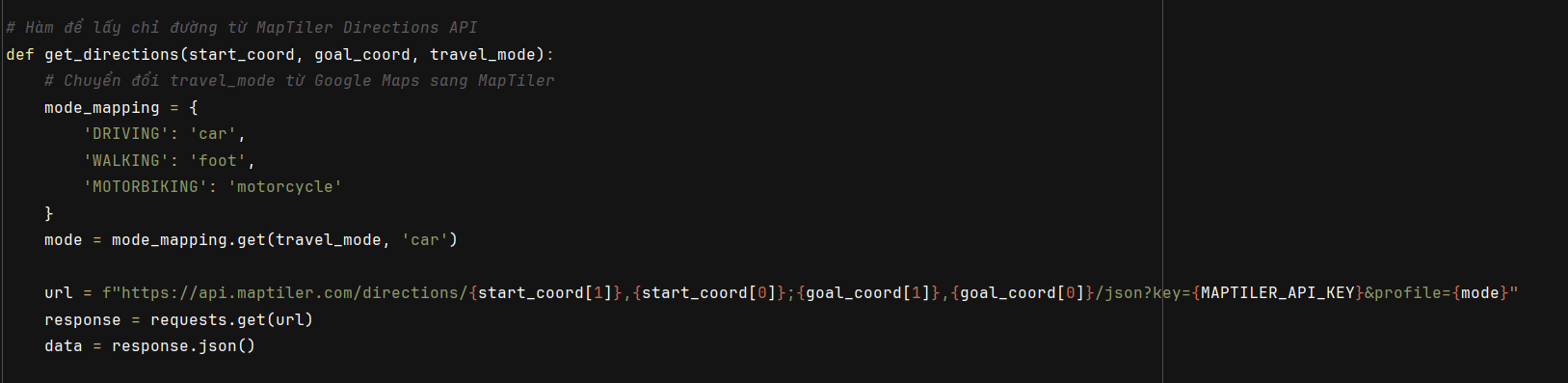
Hình 2. Khởi tạo API key cho Maptiler

Sau đó sử dụng API định vị địa chỉ (geocoding) để chuyển đổi địa chỉ thành tọa độ:



Hình 3. Hàm lấy tọa độ từ địa chỉ bằng Maptiler Geocoding API

Tiếp theo, ứng dụng sử dụng MapTiler Directions API để lấy lộ trình:



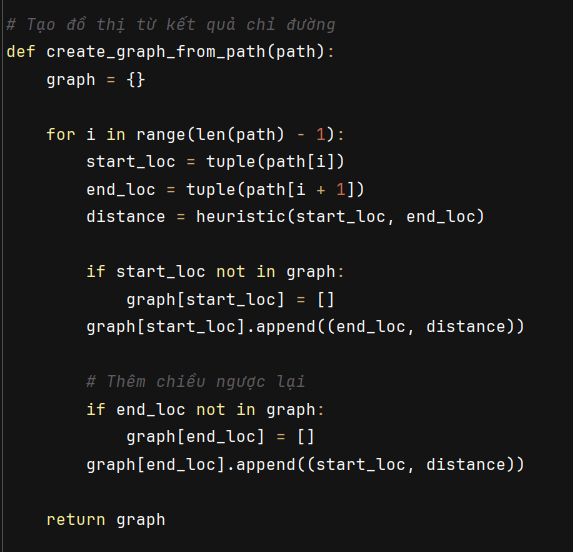
Hình 4. Gọi API Maptiler Directions để lấy lộ trình

Dữ liệu trả về dưới dạng JSON sẽ được xử lý để xây dựng đồ thị phục vụ cho thuật toán A\*. Gọi API MapTiler Directions, chuyển đổi tên phương tiện theo chuẩn MapTiler, lấy ra quãng đường, thời gian di chuyển và danh sách toạ độ lộ trình.

### **Xây dựng dữ liệu**

Sau khi dữ liệu được trả về từ MapTiler Directions API, ứng dụng sẽ phân tích danh sách các toạ độ trên lộ trình và xây dựng một đồ thị biểu diễn mạng lưới di chuyển:

* **Node** đại diện cho mỗi toạ độ (vị trí) trên tuyến đường.
* Cạnh đại diện cho đoạn đường giữa hai node liên kết trực tiếp, và chi phí chính là khoảng cách Euclid giữa hai node đó.

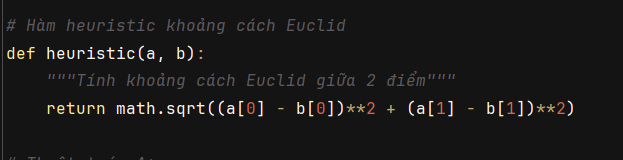


Hình 5. Hàm tạo đồ thị từ dữ liệu đường đi trả về

Mỗi lần API trả về dữ liệu chỉ đường, ứng dụng sẽ phân tích kết quả, tách rời các điểm bắt đầu, kết thúc và các bước di chuyển trung gian (steps) để xây dựng đồ thị này. Đồ thị (graph) sau khi xây dựng sẽ được sử dụng cho thuật toán A\*.

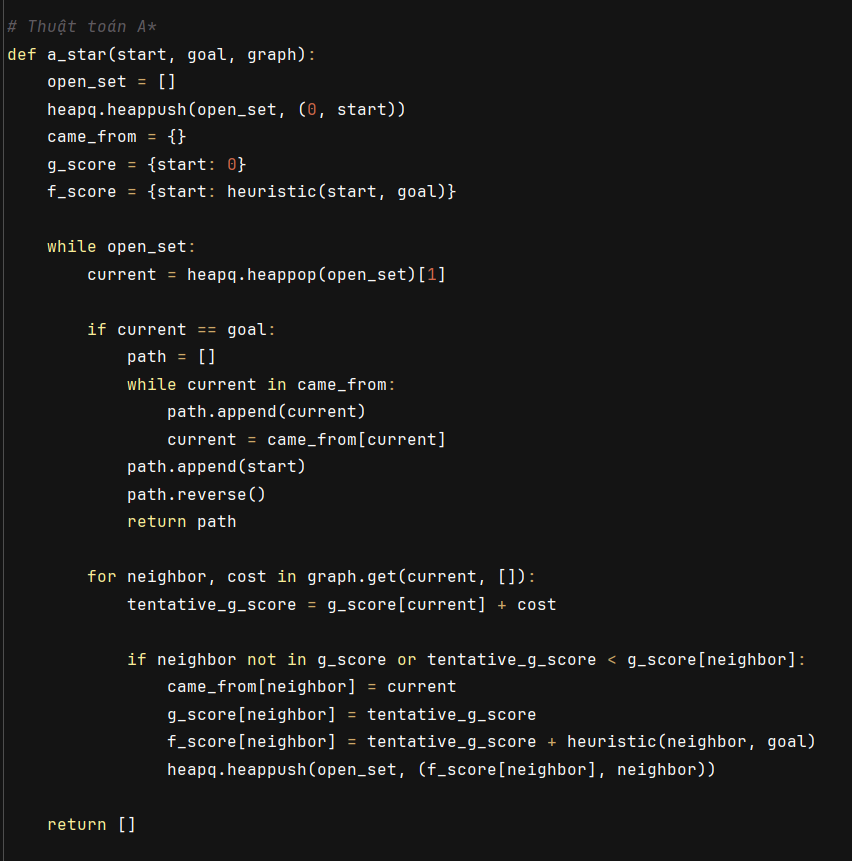
### **Xử lý dữ liệu**

Dữ liệu sau khi được thu thập và xây dựng đồ thị sẽ được sử dụng để tính toán lộ trình tối ưu thông qua việc sử dụng thuật toán A\*. Trước tiên để chạy được thuật toán trên đồ thị đó, thì ta phải tính được hàm heuristic thông qua việc sử dụng công thức tính toán khoảng cách Euclidean:



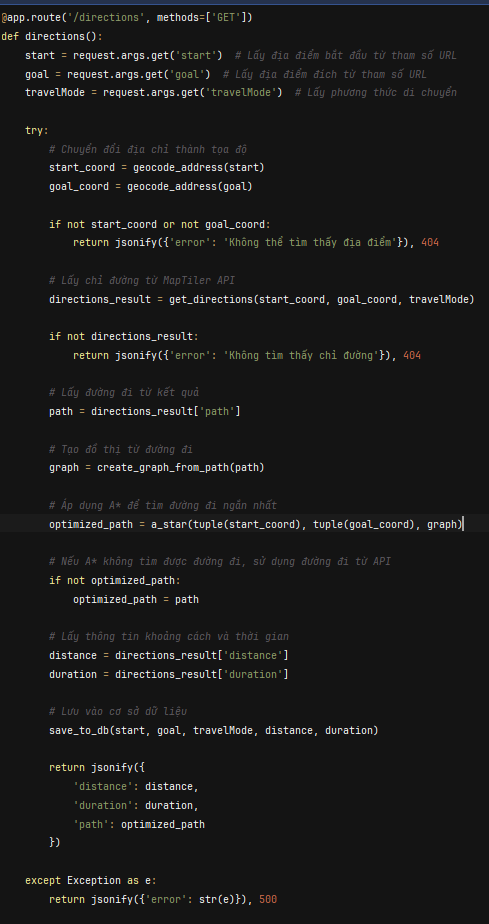
Hình 6. Hàm heuristic tính khoảng cách Euclid giữa hai điểm

Sau khi đã có hàm ước lượng heuristic, tiếp tục ta sẽ xây dựng hàm thuật toán A\* để tìm đường đi ngắn nhất thông qua dữ liệu đồ thị.



Hình 7. Cài đặt thuật toán A\*

Kết quả sau khi chạy thuật toán A\* trên đồ thị ta sẽ thu được lộ trình (path) tối ưu và chi tiết các điểm trung gian trên tuyến đường. Từ đó, ta sẽ sử dụng lộ trình để hiện thị các quãng đường cho người dùng.

Hệ thống cần một API chính để tiếp nhận dữ liệu từ frontend, gọi API MapTiler, xử lý thuật toán A\*, và trả kết quả về lại frontend.

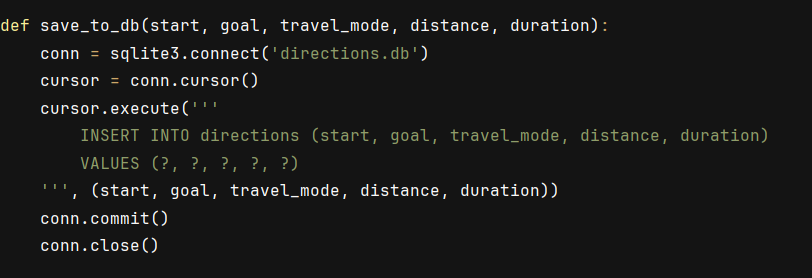
Hình 8. Hàm xử lý yêu cầu tìm đường trong frontend (Flask)

API này tiếp nhận ba tham số đầu vào (start, goal, travelMode), sau đó thực hiện các bước:

* Chuyển đổi địa chỉ sang tọa độ địa lý.
* Gọi MapTiler để lấy lộ trình và thông tin di chuyển.
* Xây dựng đồ thị từ tuyến đường.
* Áp dụng thuật toán A\* để tối ưu hoá lộ trình.
* Lưu kết quả vào cơ sở dữ liệu.
* Trả về cho frontend lộ trình chi tiết kèm khoảng cách và thời gian.

### **Xây dựng các chức năng khác**

Để lưu các thông tin dữ liệu vừa được tìm kiếm, chúng em đã thêm dòng lệnh sau:

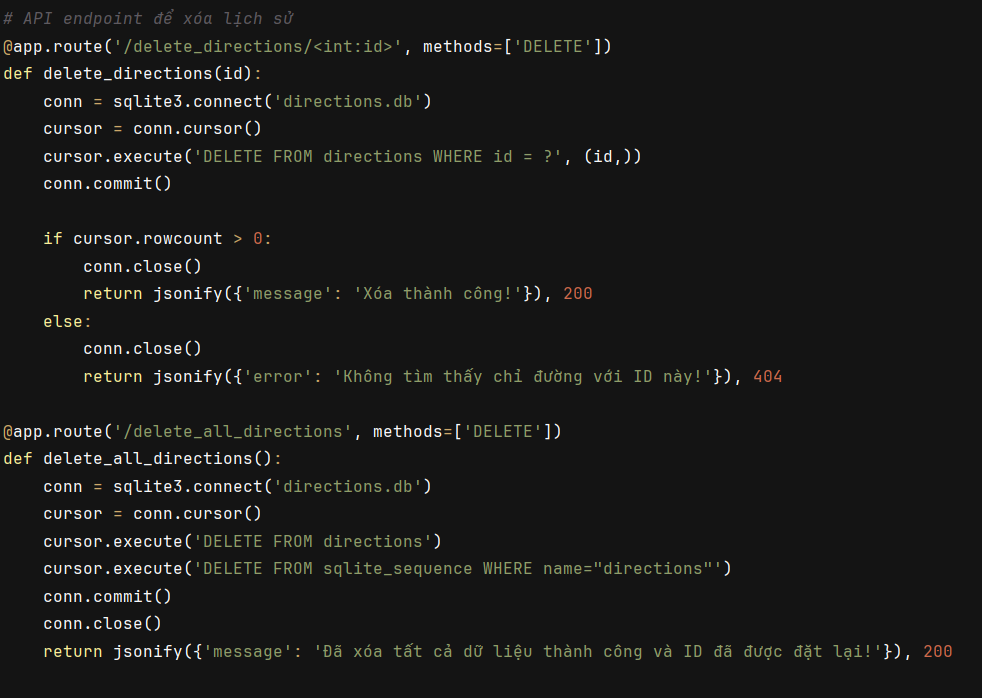


Hình 9. Lưu thông tin tìm đường vào cơ sở dữ liệu SQLite

Hàm này sẽ lưu thông tin như ID, điểm bắt đầu, điểm kết thúc, độ dài quãng đường và thời gian dự kiến sẽ di chuyển.

Để người dùng có thể nhập thông tin tìm kiếm và có thể trải nghiệm cũng như xem kết quả, chúng em đã sử dụng ngôn ngữ lập trình Python. Nó bao gồm việc khởi tọa bản đồ, tạo market (điểm bắt đầu, điểm kết thúc) và hiển thị đường đi trên bản đồ.

Hai API dưới đây giúp quản lý lịch sử chỉ đường đã lưu. Một API cho phép xóa từng chỉ đường dựa trên ID cụ thể, và một API khác hỗ trợ xóa toàn bộ lịch sử kèm việc thiết lập lại bộ đếm ID.



Hình 10. Xóa lịch sử tìm đường từ cơ sở dữ liệu.

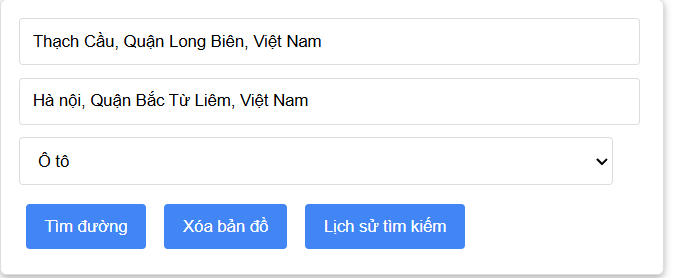
Mỗi hàm trên đều đóng một vai trò quan trọng trong việc xử lý tương tác bản đồ, tính toán và hiển thị tuyến đường, cùng với đó là việc xóa và cập nhật bản đồ dữ liệu dựa vào từ người dùng.

## **Vận hành và đánh giá kết quả**

### **Vận hành**

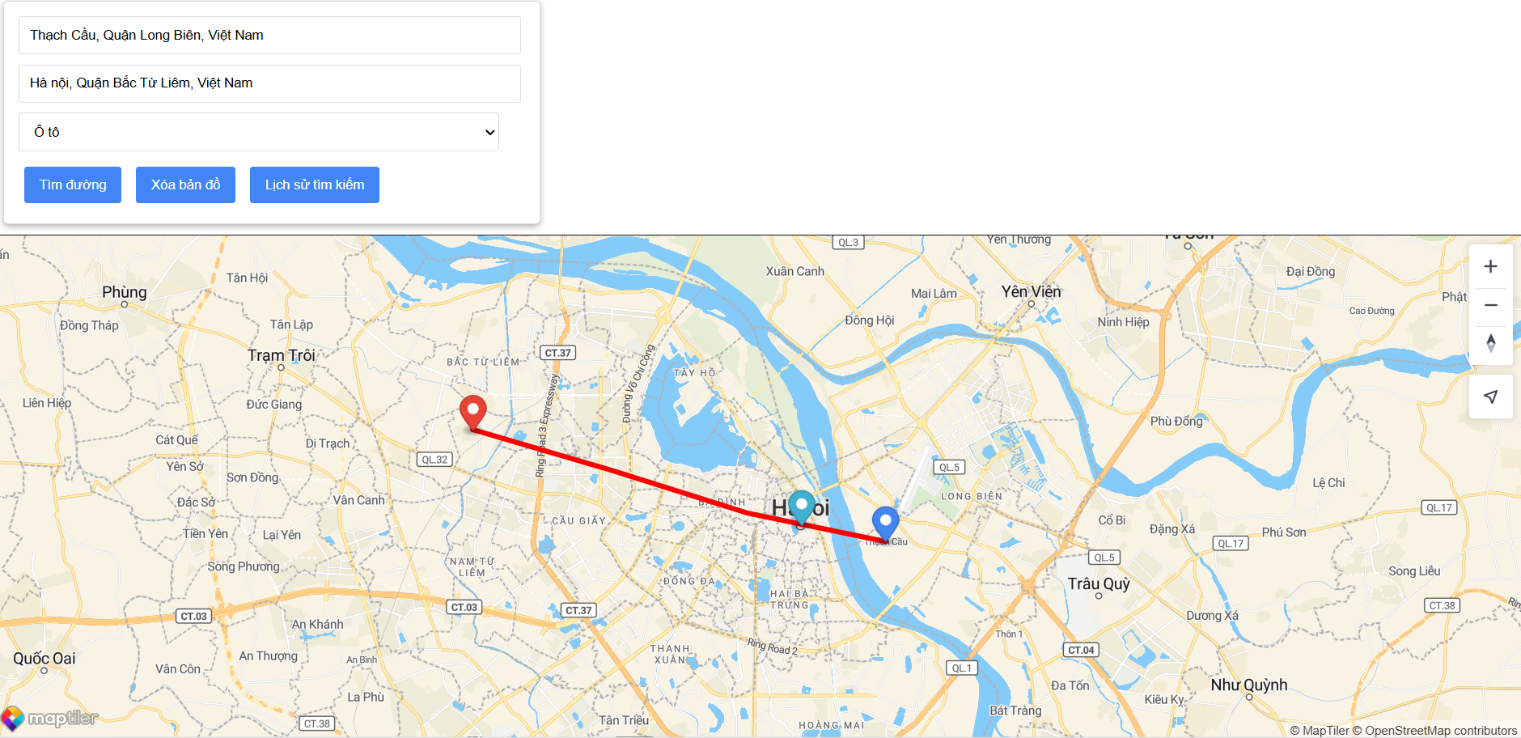
Sau khi đã cài đặt mã code xong, chúng em đã hoàn thành được chương trình và bắt đầu chạy thử. Trong việc demo này, chúng em sẽ chạy một ví dụ minh họa để đem đến góc nhìn chân thực nhất khi việc thiết kế và cài đặt hoàn tất.

Ví dụ: Em đang tìm kiếm đường đi từ địa điểm bắt đầu là ‘Thạch Cầu, Quận Long Biên’ và đến điểm đích là ‘Quận Bắc Từ Liêm’ và lựa chọn 1 trong 2 phương thức di chuyển là oto hoặc đi bộ.



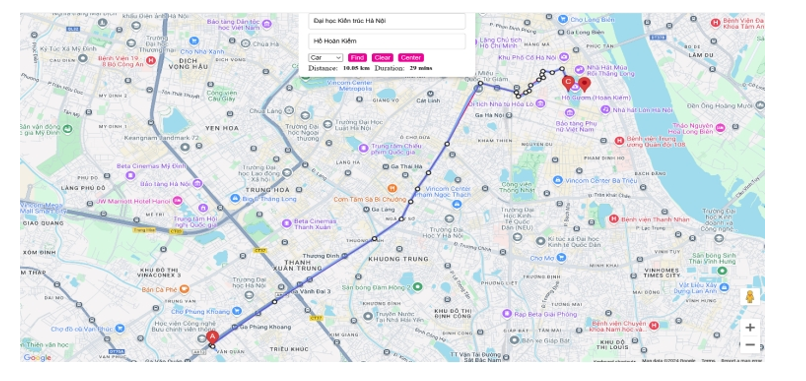
*Hình 11*

Sau đó chỉ cần nhấn Find, chương trình sẽ cho ra kết quả hiện trên màn hình:

**

*Hình 12*

Với Maptiler, đây là 1 công cụ miễn phí nên sẽ không thể tránh khỏi việc đường đi không thực sự hợp lý. Vì thế, nhóm chúng em đề xuất thêm 1 công cụ khác đó là GoogleMap Javacript API. Với các tính năng vượt trội hơi do được chính Google phát triển, do đó để sử dụng được nó chúng ta sẽ phải trả phí để sử dụng. Nó sẽ giúp người dùng tìm ra được đường đi ngắn nhất một cách hoàn toàn chính xác.

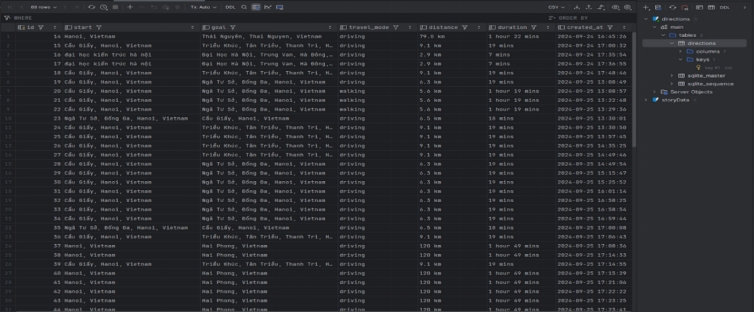


*Hình 13. Nếu sử dụng GGMaps Javacript API thay vì Maptiler*

Trên giao diện sau khi tìm kiếm đường đi và các thông tin khác như thời gian khoảng cách sẽ được hiển thị. Ngoài ra trên các đường đi đó còn xuất hiện các nút và đường kẻ màu đỏ thể hiện các đường nối giữa các nút. Đó chính là đường đi trên đồ thị dựa vào thuật toán A\*.

Ta có thể zoom lên để xem tên những cung đường đi qua, hay là bất kỳ chỗ nào mình muốn. Thêm nữa, ta còn có xem trực quan góc xem 360C:\Users\admin\AppData\Local\Temp\ksohtml7392\wps64.jpg bằng không gian 3 chiều khi ta click chuột vào hình người màu vàng (góc dưới phải giao diện). Ngoài ra ta có thể thay đổi giao diện giữa hai loại bản đồ được tích hợp là địa hình và vệ tinh.

Ngoài ra, ta còn nhận được bảng tổng hợp lịch sử tìm kiếm đường đi:



*Hình 14*

### **Đánh giá thuật toán**

Sau khi hoàn tất việc tích hợp thuật toán A\* trong ứng dụng tìm đường, nhóm tiến hành đánh giá hiệu quả hoạt động của thuật toán thông qua một số tiêu chí cụ thể:

* **Tính đúng đắn**: Thuật toán A\* luôn tìm ra được đường đi ngắn nhất giữa hai địa điểm nếu tồn tại. Điều này đảm bảo độ chính xác cao cho kết quả chỉ đường. Việc so sánh giữa kết quả do MapTiler cung cấp và kết quả do thuật toán A\* tính toán cũng cho thấy sự tương đồng lớn, đặc biệt là về mặt quãng đường và các điểm đi qua.
* **Tính tối ưu**: Do thuật toán sử dụng hàm ước lượng (heuristic) Euclidean nên lộ trình tìm được thường là lộ trình ngắn nhất về mặt không gian. Điều này giúp người dùng có được gợi ý hợp lý nhất trong việc lựa chọn tuyến đường.
* **Hiệu suất**: Với dữ liệu từ API đã được đơn giản hóa thành các đoạn đường liên tiếp, thuật toán A\* hoạt động rất nhanh do số lượng node là tương đối nhỏ. Quá trình tìm kiếm không gây chậm trễ hoặc tốn tài nguyên lớn.
* **Ưu điểm khi kết hợp với API**: Khi kết hợp với dữ liệu tuyến đường thực tế từ MapTiler API, thuật toán A\* có thể tính toán trên chính lộ trình đề xuất, từ đó cải thiện độ phù hợp với thực tế.

**KẾT LUẬN**

**Đánh giá**

Sau khi chạy và kiểm thử xong các chức năng cũng như xem kết quả của trang web tìm đường ngắn nhất dựa vào thuật toán A\*, nhóm em đã rút ra được những thứ mà nhóm em đã đạt và chưa đạt được.

**Đạt được:**

* Hiểu thêm được nhiều kiến thức về ngôn ngữ lập trình Python và hiểu được các để xây dựng lên một trang web.
* Là phần mềm dịch vụ cung cấp và phát triển với mục đích là thay thế bản đồ giấy thành bản đồ điện tử.
* Phần mềm cung cấp những hình ảnh 3D và với góc quay 360 độ, ảnh vệ tinh, bản đồ giao thông real time và hỗ trợ đường đi cho nhiều phương thức di chuyển trong giao thông như oto và đi bộ.
* Phần giao diện của trang web tương tác với người dùng, giúp người dùng có thể tìm kiếm thông tin đường đi giữa hai địa điểm cũng như có thể xem nhiều thông
* tin về đoạn đường tra cứu đó để đưa ra được những lựa chọn hợp lý.

**Chưa đạt được:**

* Chưa tạo được các lựa chọn đăng nhập tài khoản người dùng để có thể kiểm soát được lượng người truy cập trang web.
* Các phương thức di chuyển giữa các địa điểm còn hạn chế, thiếu xe máy, xe buýt, tàu điện,...
* Còn phụ thuộc vào Matiler API để lấy dữ liệu thông tin đường đi nên   
   không đảm bảo được tính chủ động, linh hoạt.
* Giao diện còn hơi đơn giản và chưa đẹp mắt về bố cụ trong giao diện.

**Ứng dụng**

Ngày nay công nghệ AI đang phát triển rất mạnh mẽ, nên chúng em luôn mong trang web này có thể ứng dụng vào nhiều ngành nghề cũng như linh vực đời sống hiện nay:

1. Ứng dụng trong giao thông và điều hướng: giúp cho người dùng có thể tìm được lộ trình tối ưu giữa hai địa điểm. Ngoài ra, còn cung cấp được dữ liệu thời gian thực về tình trạng giao thông, giúp người dùng có thể tránh kẹt xe.
2. Ứng dụng trong chuỗi cung ứng và vận tải: Các công ty cũng như hệ thống quản lý kho hàng, vận chuyển và phân phối sử dụng trang web để lập được kế hoạch cũng như lộ trình giao hàng sao cho tiết kiệm chi phí và thời gian nhất.
3. Trò chơi và ứng dụng giải trí: mang đến những trải nghiệm tuyệt vời cho người dùng thông qua việc tìm kiếm hay khám phá không gian 3D các địa điểm trong phạm vi bản đồ.
4. Ứng dụng trong công nghệ không người lái và robot: có thể thêm trang web này vào các bài tập training cho người máy hay robot có điều hướng và di chuyển cũng như có thể phân tích được biển báo giao thông.

**Hướng phát triển**

Với mục đích ngày càng phát triển và nâng cao trang web để đáp ứng nhu cầu của xã hội, phần mềm cần được tích hợp thêm nhiều tính năng giúp nâng cao hiệu suất và trải nghiệm của người dùng, chúng em có đề xuất một số hướng phát triển sau:

1. Như tích hợp giọng nói để trong ô tìm kiếm điểm đầu và điểm cuối cũng như chỉ đường giúp người dùng có thể đến các địa điểm đến mà không cần nhìn vào màn hình.
2. Trang web cần tích hợp thêm tính năng cảnh báo người dùng các biển báo cấm, giúp người dùng có thể yên tâm tin tưởng vào phần mềm, tránh trường hợp xấu có thể xảy ra
3. Để trang web thực sự có hiệu quả và thiết thực hơn nữa, chung em mong muốn được trang web trở thành một trang web public để có thể giúp ích được cho mọi người.
4. Tạo thêm các lựa chọn để đăng nhập tài khoản của người dùng để có thể kiểm soát được lượng người truy cập.
5. Nâng cao giao diện để dễ dàng thao tác và có những trải nghiệm tốt khi tìm kiếm trang web. Đòng thời tiếp thị để mang phần mềm đến với người dùng.

**TÀI LIỆU THAM KHẢO**

1. *Giáo trình trí tuệ nhân tạo – Từ Minh Phương*, NXB thông tin và truyền thông, 2020
2. *Tìm hiểu ngôn ngữ lập trình* – Google Search
3. [*Tìm hiểu Python* – Github/ TheAlgorithms](https://github.com/TheAlgorithms/Python)
4. [*Google Maps API* - Youtube : themesCode](https://www.youtube.com/watch?v=OGTG1l7yin4)