ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN NHẬP MÔN TRÍ TUỆ NHÂN TẠO

NHÓM 8

Giảng viên hướng dẫn : **PGS. TS. Trần Đình Khang**

Mã lớp học : **157487**

Danh sách thành viên :

SỐ THỨ TỰ	HỌ VÀ TÊN	MÃ SỐ SINH VIÊN
1	Nguyễn Lê Quân	20235199
2	Vũ Tiến Đạt	20235035
3	Vũ Nam Khánh	20235352
4	Nguyễn Trịnh Minh Hiếu	20235078
5	Nguyễn Hoàng Long	20235366

Hà Nội, tháng 05 năm 2025

MỤC LỤC

I. MỤC TIÊU 3

II. MÔ TẢ BÀI TOÁN 3

1. Dữ liệu đầu vào (Input) 3
2. Xử lý dữ liệu đầu vào 4
   a) Mục tiêu 4
   b) Cách thực hiện 4
3. Dữ liệu đầu ra (Output) 5

III. BIỂU DIỄN BÀI TOÁN 5

1. Biểu diễn các trạng thái của bài toán 5
2. Tập các hành động giữa các trạng thái 6

IV. THUẬT TOÁN SỬ DỤNG 6

1. Ý tưởng của thuật toán 6
2. Mô tả thuật toán 6
   a) Sự kiện xảy ra khi click vào một địa điểm trên bản đồ 6
   b) Cài đặt thuật toán (find_path) 7
   c) Độ phức tạp tính toán 8

V. CÔNG NGHỆ VÀ CÔNG CỤ ĐÃ DÙNG 8

VI. GIAO DIỆN VÀ CHỨC NĂNG CHÍNH 9

VII. KẾT QUẢ ĐẠT ĐƯỢC 13

VIII. HẠN CHẾ VÀ HƯỚNG PHÁT TRIỂN 13

IX. KẾT LUẬN 14

X. PHỤ LỤC: Tài liệu tham khảo 14

NỘI DUNG ĐỀ TÀI BÀI TẬP LỚN

Mô phỏng bản đồ phường Thành Công, quận Ba Đình, thành phố Hà Nội và tìm đường đi ngắn nhất giữa hai địa điểm bất kỳ trong phường

I. MỤC TIÊU

- Xây dựng một công cụ trực quan giúp tìm đường đi ngắn nhất giữa hai địa điểm trên bản đồ.
- Ứng dụng thuật toán A* để đảm bảo tìm được đường đi tối ưu.
- Cho phép người dùng tương tác với bản đồ bằng cách chọn địa điểm bắt đầu, địa điểm kết thúc và các địa điểm không được đi qua, nếu có.

II. MÔ TẢ BÀI TOÁN

1. Dữ liệu đầu vào (Input)

Dữ liệu đầu vào là bản đồ phường Thành Công từ trang **OpenStreetMap**, lưu trong tệp ảnh map.png tại thư mục res/.

Hình 2.1: *Ảnh bản đồ gốc của phường*

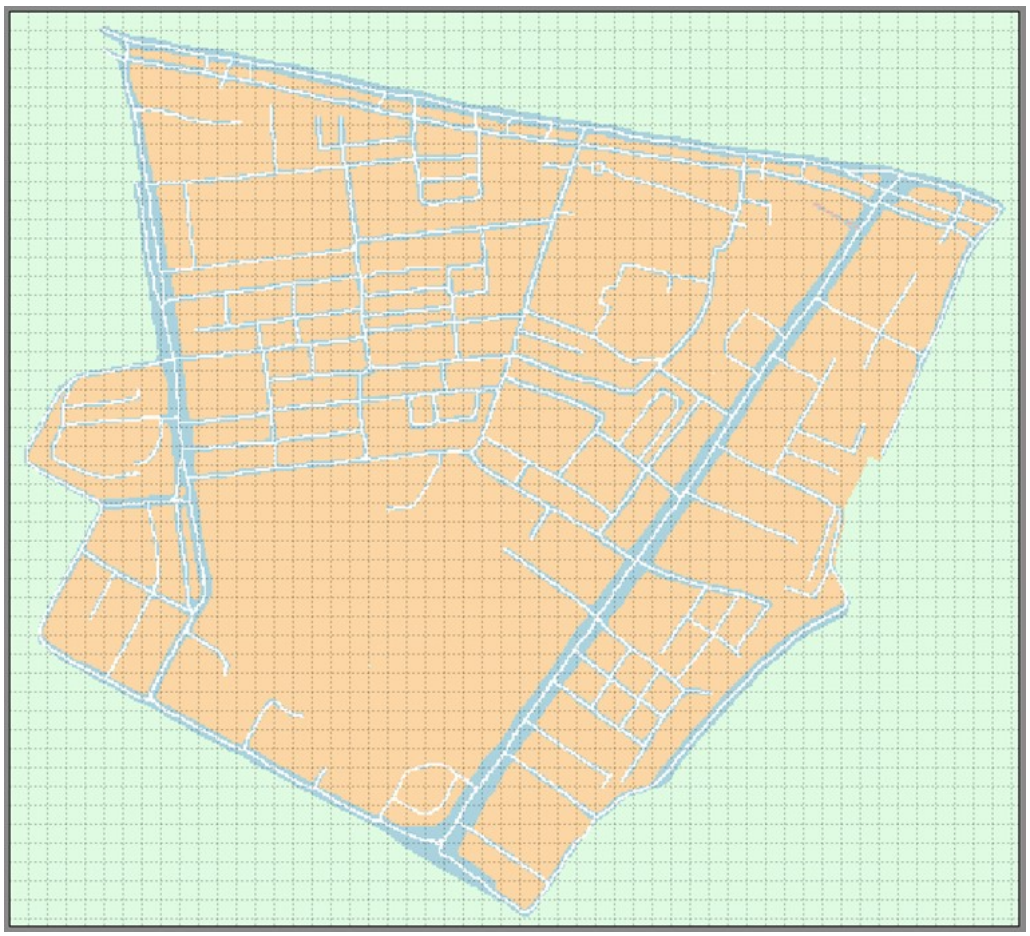
2. Xử lý dữ liệu đầu vào

a) Mục tiêu

Chuyển bản đồ từ dạng ảnh .png thành ma trận số nguyên trong tệp .csv để việc dùng thuật toán A* được thuận tiện hơn.

b) Cách thực hiện

Bước 1: Vẽ lại bản đồ bằng ứng dụng **Tiled**.



Hình 2.2: *Ảnh bản đồ đã vẽ lại*

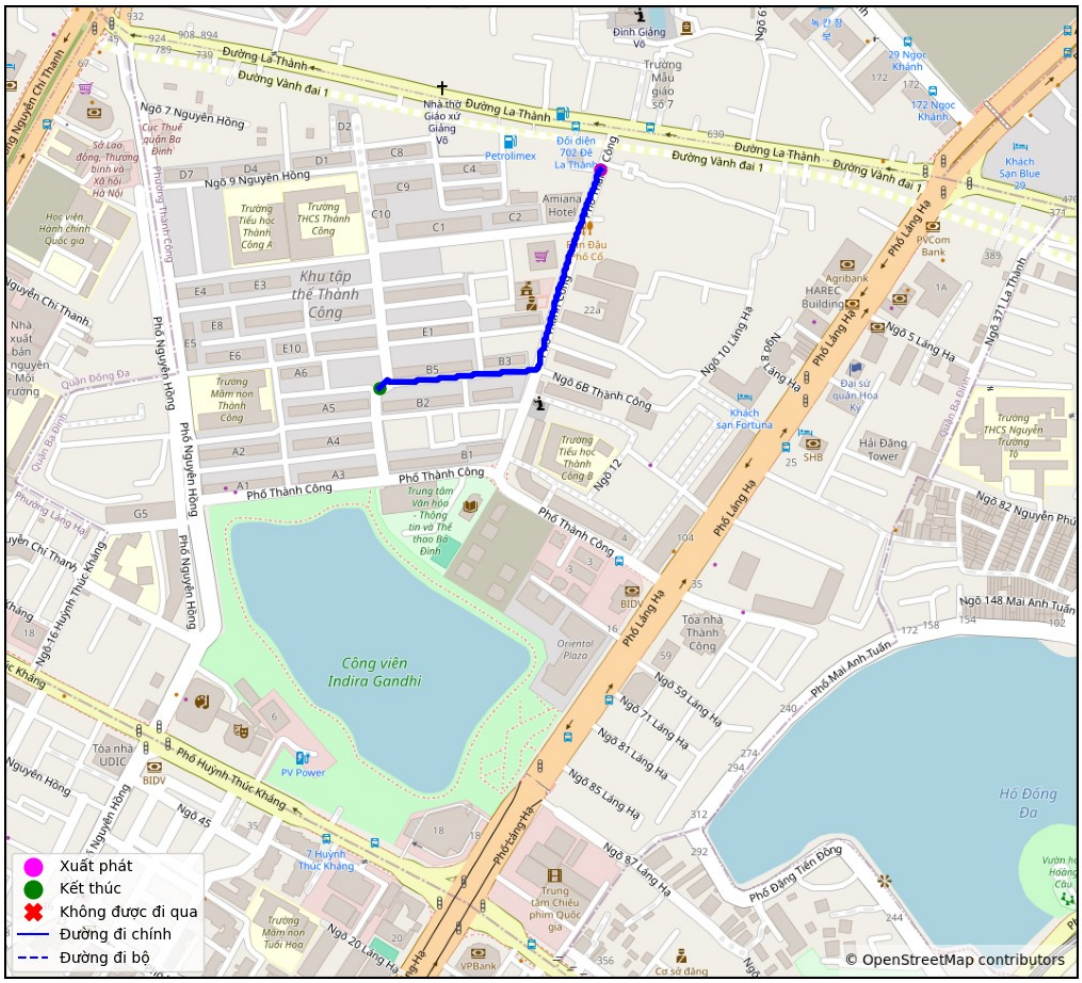
Bước 2: Xuất ra tệp .csv với mỗi ô có giá trị như sau:

- 0 nếu ô đó là đường đi.
- 1 nếu ô đó không là đường đi, tức đó là nhà hoặc hồ.
- 2 nếu ô đó nằm ngoài địa phận phường Thành Công.

Bước 3: Do ma trận bản đồ rất thưa (tỉ lệ số ô là đường trong bản đồ là khá thấp) nên thực hiện chuyển ma trận thành đồ thị để tăng tốc độ tính toán.

3. Dữ liệu đầu ra (Output)

- Đường đi ngắn nhất từ địa điểm xuất phát đến địa điểm kết thúc, nếu có, được hiển thị trực quan trên bản đồ.
- Ngược lại, thông báo không tìm thấy đường đi giữa hai địa điểm này.



Hình 2.3: *Đường đi từ địa điểm xuất phát đến địa điểm kết thúc*

III. BIỂU DIỄN BÀI TOÁN

1. Biểu diễn các trạng thái của bài toán

- Coi bản đồ là mặt phẳng Oxy, trong đó gốc tọa độ O là điểm trên cùng bên trái của bản đồ trục Ox hướng sang bên phải và trục Oy hướng xuống phía dưới.
- Các trạng thái **N** của bài toán được biểu diễn là các vị trí trên bản đồ.
- Mỗi vị trí có thể biểu diễn dưới dạng (x, y) với x, y là các số thực không âm chỉ tọa độ của vị trí lần lượt trên trục Ox và trục Oy.
- Các cặp tọa độ tương ứng với vị trí xuất phát và vị trí kết thúc sẽ lần lượt là trạng thái ban đầu $\mathbf{N_0}$ và trạng thái đích **ĐICH**.

2. **Tập các hành động giữa các trạng thái**

Tập các hành động **A** là toàn bộ các cạnh trong đồ thị nối hai đỉnh với nhau, khi di chuyển từ điểm này sang điểm khác thông qua một cạnh có trong đồ thị.

IV. THUẬT TOÁN SỬ DỤNG

Việc tìm kiếm đường đi được dựa trên thuật toán A* (A-star).

1. **Ý tưởng của thuật toán**

Hàm chi phí tổng f(n) được xác định bởi:

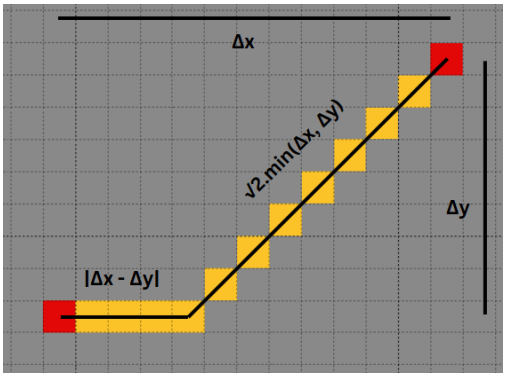
$$f(n) = g(n) + h(n),$$

trong đó:

- g(n) là chi phí thực tế từ điểm bắt đầu đến điểm n.
- h(n) là chi phí ước lượng từ điểm n đến đích.

Với bản đồ có thể di chuyển theo tám hướng khác nhau, hàm h(n) sẽ được tính bằng khoảng cách tám hướng *(octile distance)*, cụ thể như sau:

$$\boldsymbol{h(n) = |\Delta x - \Delta y| + \sqrt{2}\,\min(\Delta x, \Delta y)}$$



Hình 4.1: *Minh họa hàm h(n) theo khoảng cách tám hướng*

2. **Mô tả thuật toán**

a) Sự kiện xảy ra khi click vào một địa điểm trên bản đồ

- ❖ Nếu điểm click vào ngoài địa phận phường Thành Công thì báo lỗi.
- ❖ Nếu điểm click vào là đường đi thì sử dụng **Thuật toán A***.
- ❖ Nếu điểm click vào là nhà thì thực hiện hai bước sau:

Bước 1:

- Gọi đỉnh click vào là v; D(a, b) = Khoảng cách Euclid giữa a và b; V là tập các đỉnh.

- Tìm u = argmin(D(u, v)) (u ∈ V ≠ v).
- Duyệt tất cả các đỉnh trong đồ thị nên độ phức tạp tính toán là **O(N)** với N là số đỉnh của đồ thị.

<u>Bước 2:</u> Sử dụng **Thuật toán A*** với điểm u.

b) Cài đặt thuật toán (find_path)

Input:
start ← điểm bắt đầu
end ← điểm đích
restrict ← tập các điểm không thể đi qua
graph ← bản đồ dạng danh sách kề
heuristic ← hàm ước lượng chi phí (Octile)

Khởi tạo:
priority_queue ← hàng đợi chứa (**f_score**, **node**) # Cấu trúc heap
g_score[**start**] ← 0
came_from[**start**] ← None
Thêm (0, **start**) vào **priority_queue**
Trong khi **priority_queue** không rỗng:
    Lấy node có **f_score** nhỏ nhất: **current** ← **priority_queue**.pop()
        Nếu **current** ∈ **restrict**:
            Bỏ qua
        Nếu **current** == **end**:
            last_g_score ← **g_score**[**end**]
            Trả về đường đi được truy vết từ came_from

        Với mỗi **neighbor** kề **current**:
            new_cost ← **g_score**[**current**] + cost(**current** → **neighbor**)

            Nếu **neighbor** chưa có trong **g_score**
            hoặc g_new_cost < **g_score**[**neighbor**]:
                g_score[**neighbor**] ← new_cost
                f_score ← new_cost + heuristic(neighbor, end)
                Thêm (f_score, neighbor) vào **priority_queue**
                came_from[**neighbor**] ← **current**
Trả về **None** nếu không tìm thấy đường đi

c) Độ phức tạp tính toán

- Đồ thị G(N, M) có N đỉnh, M cạnh. Do đồ thị của bản đồ là đường đi rất thưa nên số cạnh M < 2N.
- Nếu click vào điểm không phải đường thì phải tìm node gần nhất là đường với độ phức tạp O(N).
- Mỗi lần thêm và lấy phần tử nhỏ nhất từ hàng đợi ưu tiên (heap) có độ phức tạp O(logN).
- Mỗi lần cập nhật cạnh là một lần thêm phần tử vào hàng đợi ưu tiên. Đồ thị có M cạnh nên độ phức tạp tính toán là:

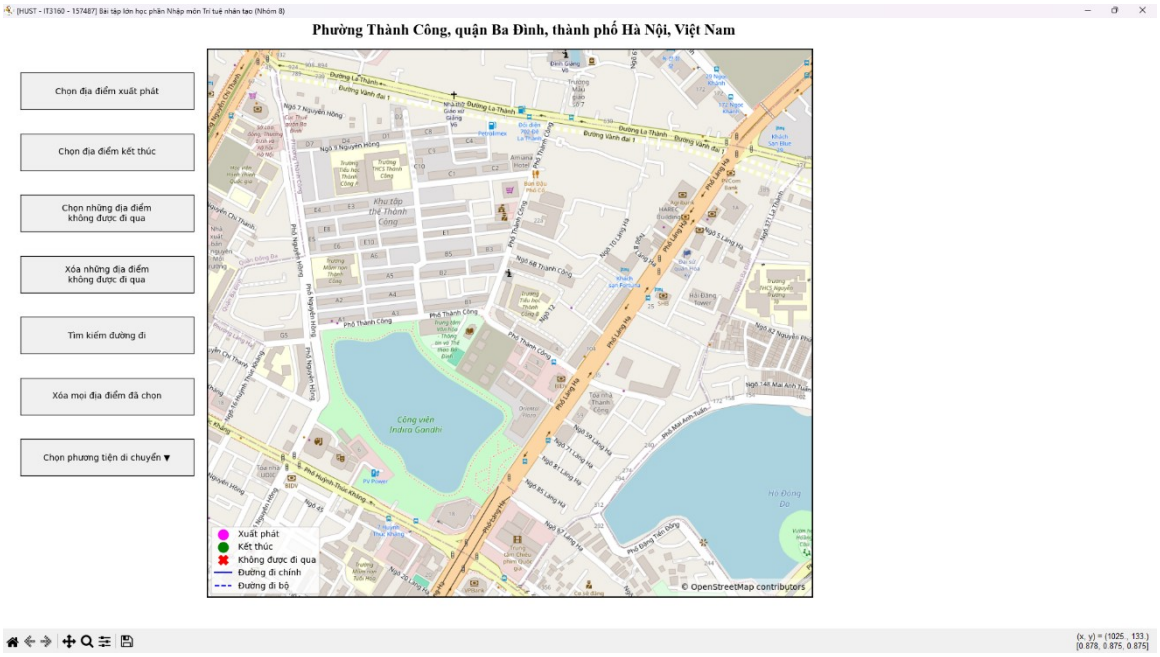
  O(N + M.logN) = O(M.logN) = O(2N.logN) = O(N.logN)

V. CÔNG NGHỆ VÀ CÔNG CỤ ĐÃ DÙNG

• **Ứng dụng vẽ map:** Tiled	
• **Ngôn ngữ lập trình:** Python	
• **Thư viện sử dụng:** matplotlib – hiển thị bản đồ và trực quan hóa numpy & pandas – xử lý dữ liệu bản đồ heapq – hàng đợi ưu tiên trong thuật toán A*	  
• **Môi trường phát triển:** Visual Studio Code	
• **Quản lý dự án:** GitHub	

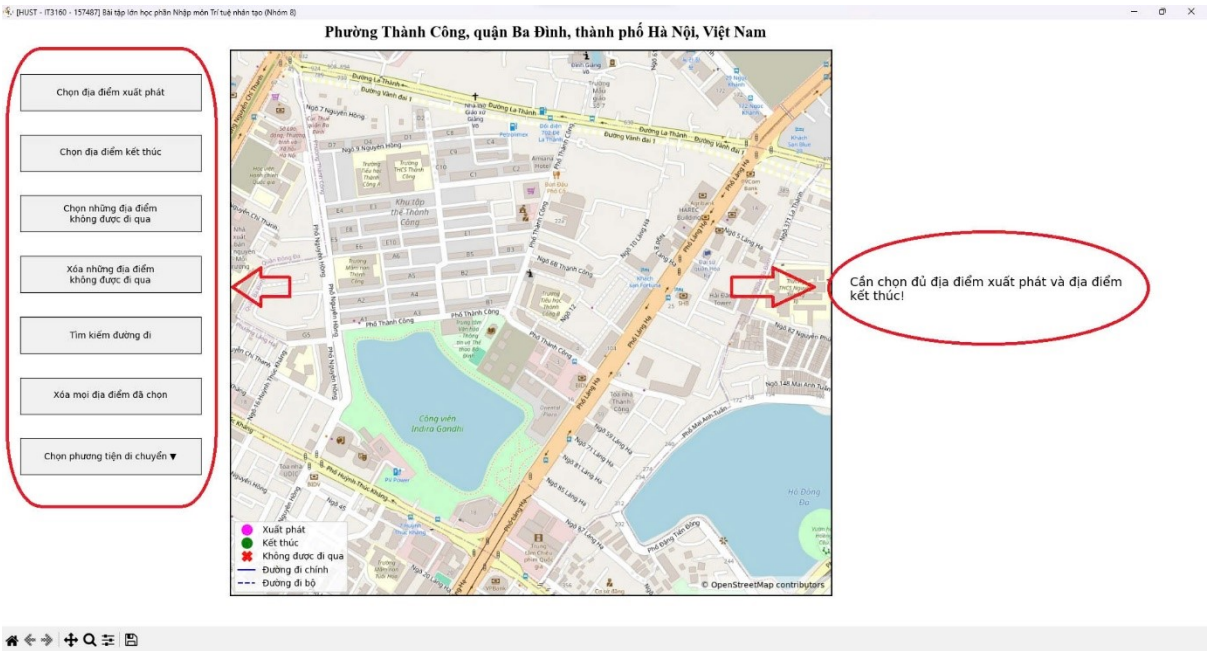
VI. GIAO DIỆN VÀ CHỨC NĂNG CHÍNH

- Hiển thị bản đồ phường ở giữa cửa sổ giao diện của matplotlib.



Hình 5.1: *Giao diện phần mềm*

- Bên trái màn hình là các nút để người dùng tùy chọn. Bên phải màn hình là các thông điệp của hệ thống báo cho người dùng.

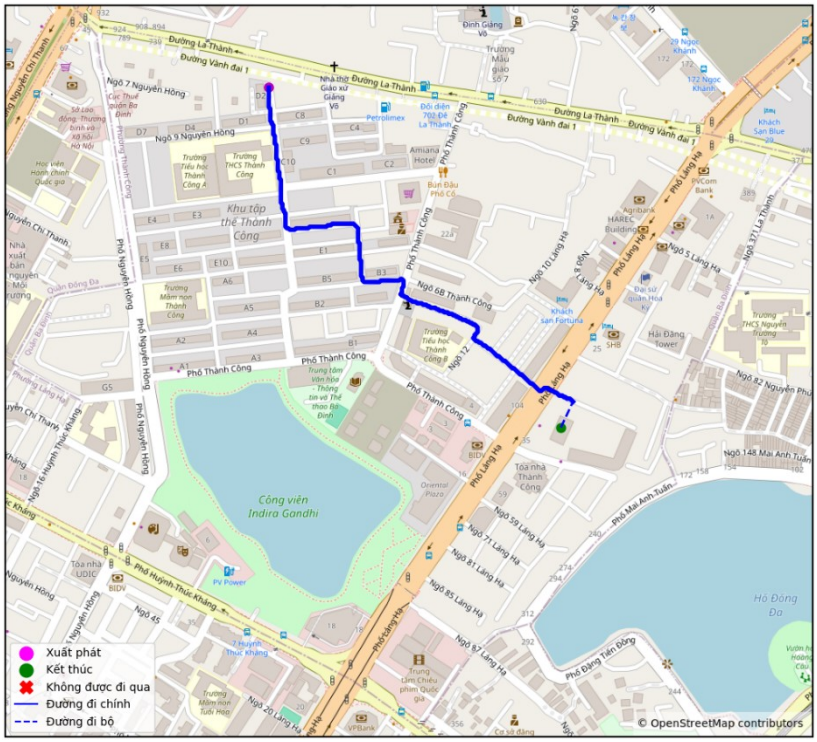


Hình 5.2: *Nút bấm và thông điệp hiển thị*

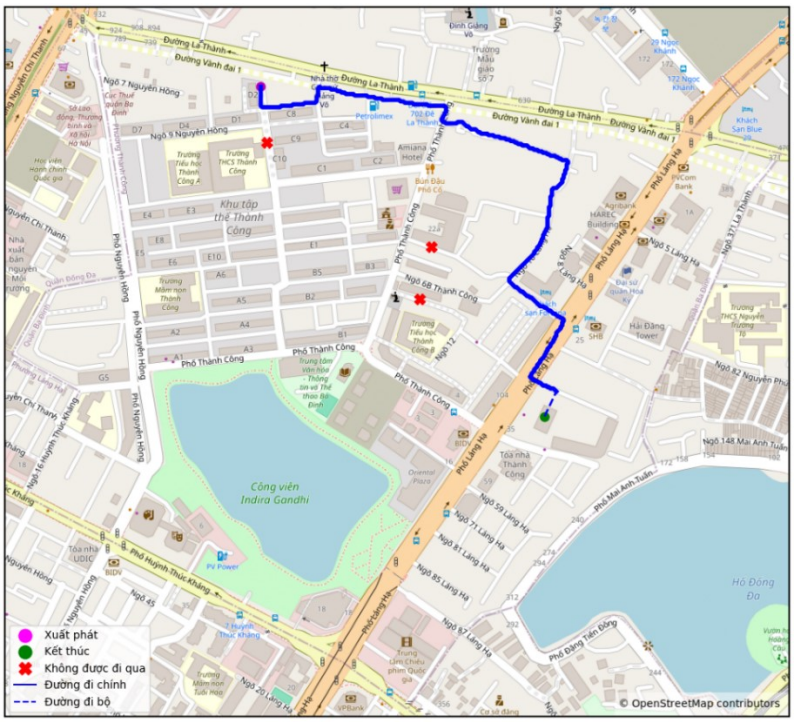
- Chọn điểm xuất phát, điểm kết thúc, các điểm không được đi qua

Hình 5.3: *Chọn các địa điểm trên bản đồ*

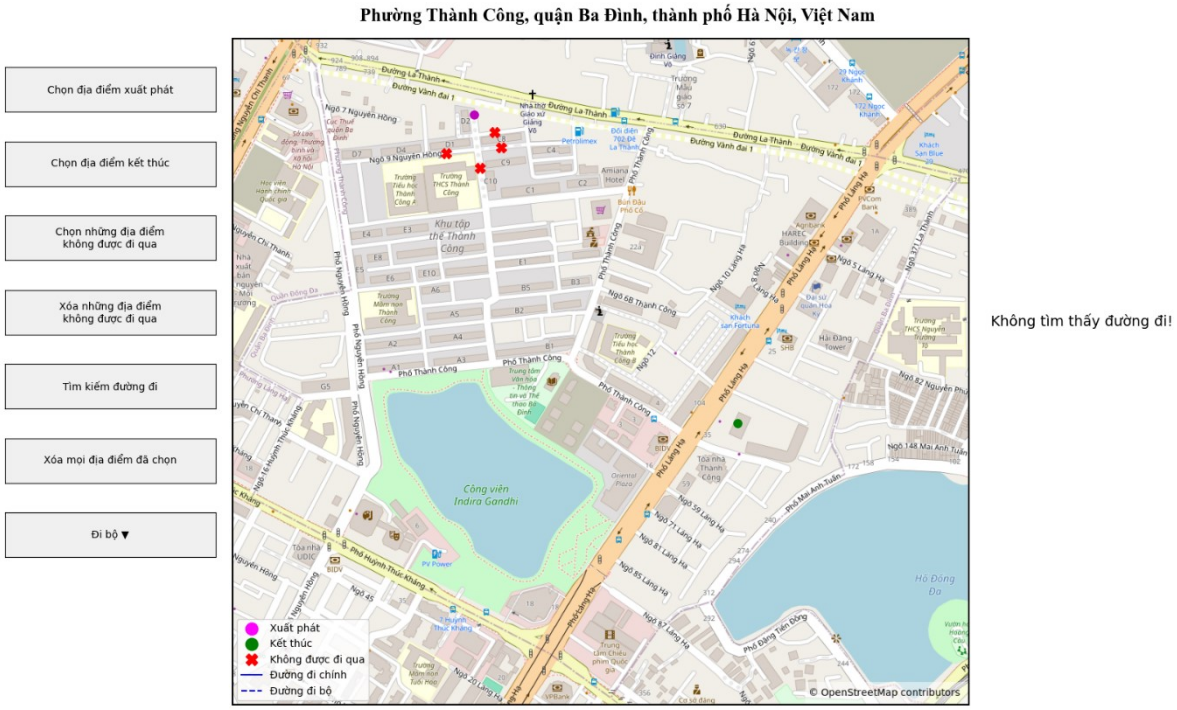
- Hiển thị đường đi ngắn nhất tìm được (nếu có). Khi điểm được chọn là nhà thì hiển thị đường đi bộ đến đường rồi mới hiển thị đường đi chính.



Hình 5.4: *Hiển thị đường đi khi không có điểm cấm*

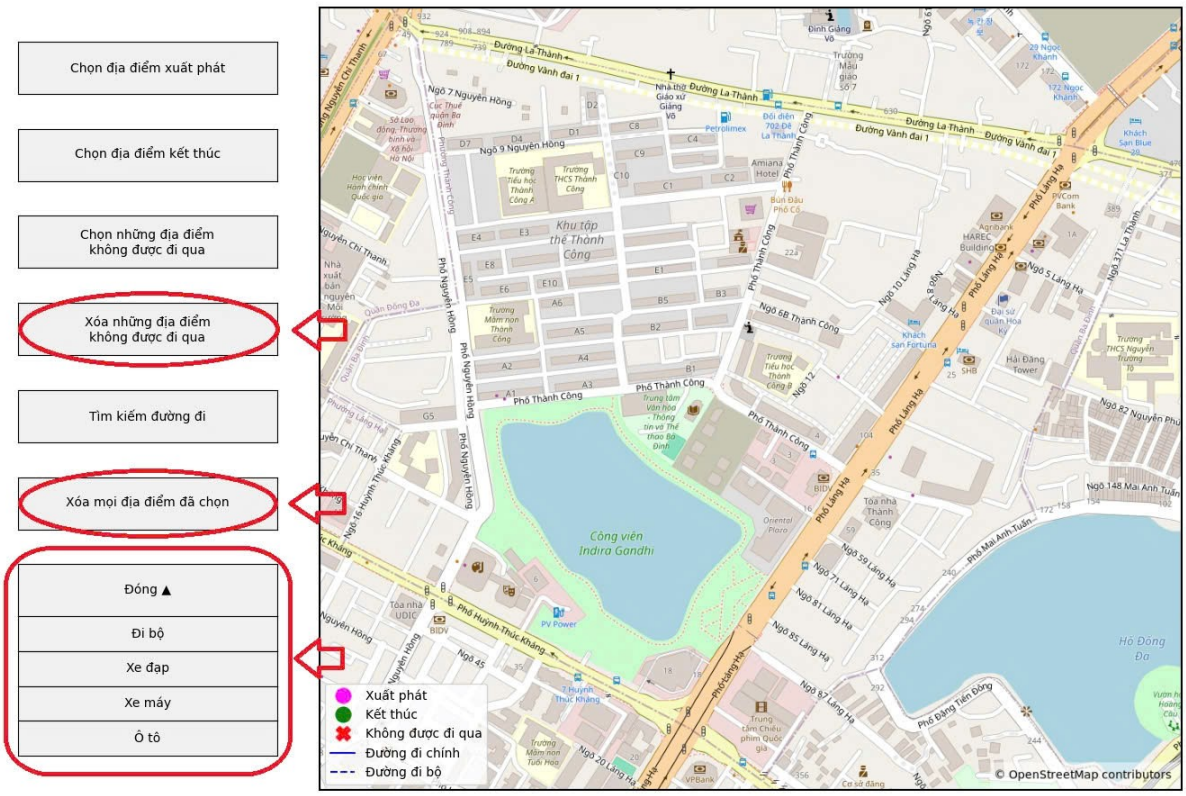


Hình 5.5: *Hiển thị đường đi khi cấm một số điểm*

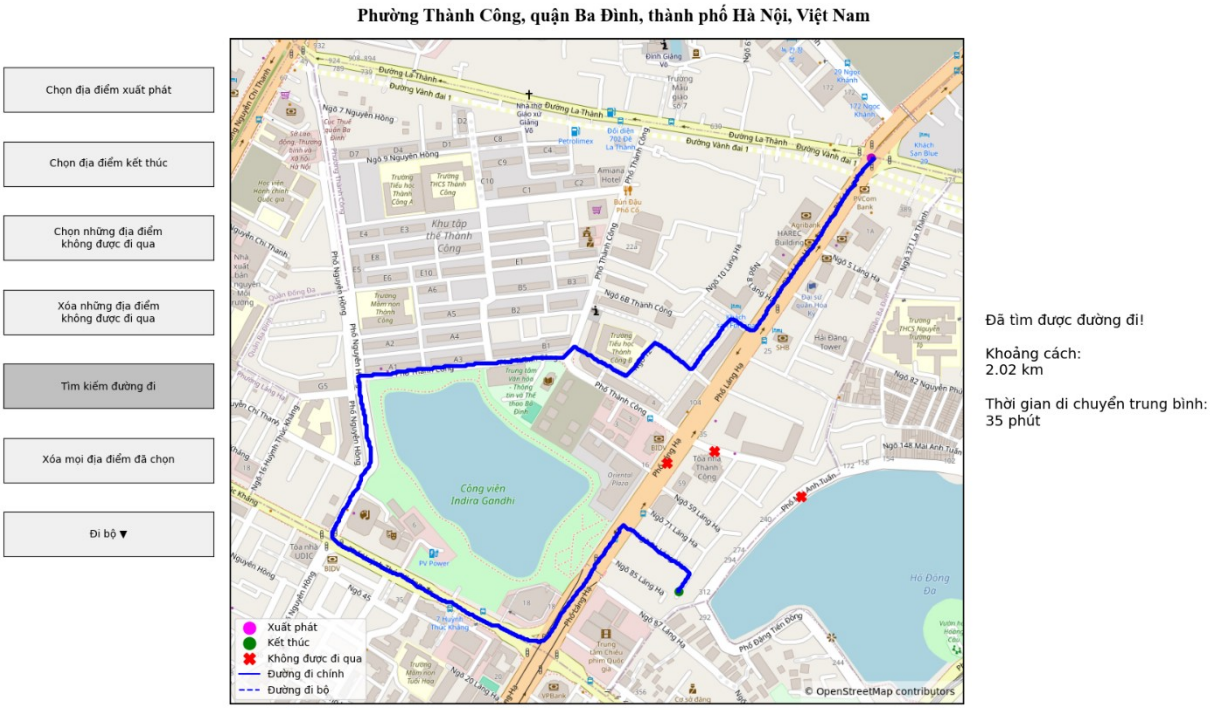


Hình 5.6: *Hệ thống thông báo không tìm thấy đường đi*

- Chức năng xóa các điểm cấm, xóa tất cả các lựa chọn, chọn phương tiện giao thông và hiển thị đường đi, thời gian.



Hình 5.7: *Các nút tiện ích*



Hình 5.8: *Hiển thị quãng đường và thời gian khi đi bộ*

Hình 5.9: *Hiển thị quãng đường và thời gian khi đi ô tô*

VII. KẾT QUẢ ĐẠT ĐƯỢC

- Xây dựng thành công một hệ thống tương tác đơn giản, trực quan, cho phép người dùng chọn điểm xuất phát, đích đến và khu vực cấm.
- Thuật toán A* được cài đặt chính xác, hoạt động ổn định và luôn tìm ra đường đi ngắn nhất (nếu tồn tại).
- Hệ thống xử lý tốt các tình huống đặc biệt như không tồn tại đường đi hoặc có vật cản trên bản đồ.

VIII. HẠN CHẾ VÀ HƯỚNG PHÁT TRIỂN

❖ **Hạn chế:** Bản đồ sử dụng hiện tại chỉ là mô hình đơn giản hóa, được tạo thủ công, chưa phản ánh dữ liệu bản đồ thực tế.

❖ **Hướng phát triển:**

- Tích hợp bản đồ thật thông qua API như **OpenStreetMap** hoặc **Google Maps** để tăng tính ứng dụng thực tế.
- Nâng cấp giao diện và trải nghiệm người dùng: bổ sung các yếu tố như lựa chọn phương tiện, hiển thị tắc đường, đèn giao thông,...
- Tối ưu hiệu năng thuật toán để xử lý bản đồ có quy mô lớn hơn.

IX. KẾT LUẬN

- Bài tập lớn lần này đã giúp nhóm hiểu rõ cách hoạt động của thuật toán A* và khả năng áp dụng vào bài toán tìm đường trong môi trường thực tế.
- Chương trình hoạt động ổn định, giao diện dễ sử dụng và đáp ứng đầy đủ các yêu cầu đặt ra ban đầu. Đây là một nền tảng tốt để tiếp tục phát triển thành ứng dụng có tính ứng dụng cao hơn.

X. PHỤ LỤC: Tài liệu tham khảo

1. Slide bài giảng Nhập môn trí tuệ nhân tạo
2. Python Documentation
3. Tài liệu thư viện sử dụng: matplotlib, heapq, numpy, pandas