TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN MÔN: TRÍ TUỆ NHÂN TẠO

Mã lớp: 20234IT6043001

ĐỀ TÀI:

ỨNG DỤNG THUẬT TOÁN A* VÀO BÀI TOÁN TÌM KIẾM ĐƯỜNG ĐI CHO ROBOT VẬN CHUYỂN

GVHD: TS.Trần Hùng Cường

Nhóm thực hiện: Nhóm 5

1. Nguyễn Đình Anh 2022603341 Lớp KHMT02 Khóa 17
2. Nguyễn Đức Khánh 2022607544 Lớp KHMT01 Khóa 17
3. Trần Văn Nhã 2022603089 Lớp KHMT02 Khóa 17

Hà Nội – Năm 2024

MỤC LỤC

LỜI MỞ ĐẦU

Trí tuệ nhân tạo (Artificial Intelligence - AI) hiện nay đang phát triển mạnh mẽ, và nổi lên như một minh chứng của cuộc cách mạng công nghiệp 4.0. Nó đã và đang trở thành xu hướng công nghệ tương lai mà các công ty, tập đoàn và các hãng công nghệ lớn trên thế giới luôn chạy đua với nhau để sáng tạo và phát triển. Ứng dụng của Trí tuệ nhân tạo là rộng lớn vô cùng, phải kể đến như xe tự hành của Google, Tesla; hệ thống nhận diện khuôn mặt, dự báo thời tiết,..

Trong thời đại công nghiệp hóa hiện đại hóa, việc áp dụng những thiết bị tự động hóa vào sản xuất đã trở thành một nhu cầu tất yếu đối với các doanh nghiệp sản xuất. Trong đó robot vận chuyển trong nhà máy là một trong những thiết bị tự động hóa được ứng dụng khá phổ biến. Và để hiểu rõ hơn về cách thức mà nó, cùng với các kiến thức xung quanh nó, chúng em đã quyết định chọn đề tài "**Ứng dụng thuật toán A* vào bài toán tìm đường đi tối ưu cho Robot vận chuyển**". Các vấn đề cụ thể trình bày trong cuốn báo cáo được chia thành 3 chương:

Chương 1: Tổng quan về trí tuệ nhân tạo

Chương 2: Tìm hiểu thuật toán A*

Chương 3: Ứng dụng thuật toán A* vào bài toán tìm đường đi cho robot vận chuyển

Trong quá trình tìm hiểu và thực hiện báo cáo bài tập lớn, mặc dù đã rất cố gắng hoàn thành báo cáo này, nhưng do lượng kiến thức và trình độ chuyên môn có hạn, nhóm chúng em không thể tránh khỏi những thiếu sót, vì vậy nhóm rất mong được nghe sự nhận xét và đánh giá của thầy dành cho nhóm.

Xin chân thành cảm ơn!

Nhóm sinh viên thực hiện

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

MỞ ĐẦU

1 Lý do chọn đề tài

- Trong lĩnh vực khoa học máy tính, bài toán tìm đường đi ngắn nhất là một trong những bài toán cơ bản, và được ứng dụng thực tế như bản đồ, trò chơi điện tử, và robot tự hành.
- Thuật toán A Star là sự kết hợp giữa tìm kiếm theo chiều sâu (DFS) và chiều rộng (BFS), sử dụng hàm heuristic để tối ưu hóa quá trình tìm kiếm, do đó hiệu quả hơn so với nhiều thuật toán khác.
- Với sự phát triển của công nghệ và nhu cầu ngày càng cao về các hệ thống tự động hóa và thông minh, việc tìm hiểu và nghiên cứu các thuật toán tìm kiếm hiệu quả là rất cần thiết.
- Theo một bài báo trên RTC (2024), việc ứng dụng robot vận chuyển trong nhà máy mang lại nhiều lợi ích, bao gồm giảm chi phí nhân công, tăng năng suất, nâng cao chất lượng sản phẩm, và giảm thiểu nguy cơ tai nạn lao động (RTC, 2024). [1]

Hình 1: Khả năng làm việc không ngừng nghỉ của Robot (Nguồn: RTC, 2024).

2 Mục tiêu nghiên cứu

2.1. Mục tiêu tổng quát

Mục tiêu tổng quát của đề tài là phát triển và áp dụng các thuật toán và phương pháp trí tuệ nhân tạo để tìm ra đường đi ngắn nhất trên bản đồ một cách hiệu quả, tiết kiệm thời gian và chi phí cho công ty, giúp cải thiện hiệu suất di chuyển của Robot vận chuyển.

2.2. Mục tiêu cụ thể

- Nghiên cứu và áp dụng các thuật toán tìm đường đi ngắn nhất: Mục tiêu này là nghiên cứu và áp dụng các thuật toán để tìm ra đường đi ngắn nhất trên bản đồ
- Xác định các yếu tố ảnh hưởng đến đường đi: Mục tiêu này là xác định các yếu tố cản đường như: các bức tường, đồ vật, máy móc và các ràng buộc khác có thể ảnh hưởng đến quyết định về đường đi ngắn nhất của robot.
- Xây dựng giao diện người dùng: Mục tiêu này là xây dựng một giao diện người dùng trực quan và dễ sử dụng để người dùng có thể nhập liệu địa điểm xuất phát và đích, từ đó hệ thống hiển thị kết quả đường đi ngắn nhất trên bản đồ.
- Đánh giá và tối ưu hiệu suất: Đánh giá hiệu suất của hệ thống tìm đường đi, bao gồm thời gian tính toán, chi phí di chuyển. Tiến hành tối ưu hóa các thuật toán và phương pháp để đảm bảo hiệu suất tốt nhất.
- Kiểm tra và đánh giá tích hợp: Mục tiêu này là kiểm tra tích hợp của hệ thống tìm đường đi vào các hệ thống và dịch vụ thực tế, đảm bảo tính tương thích và khả năng hoạt động ổn định.
- Đảm bảo bảo mật và quyền riêng tư: Mục tiêu này là đảm bảo rằng hệ thống tìm đường đi bảo vệ thông tin người dùng và tuân thủ các quy định về bảo mật và quyền riêng tư.

3 Đối tượng và phạm vi nghiên cứu

- **Đối tượng nghiên cứu**: Các ứng dụng tìm đường đi ngắn nhất trên bản đồ sử dụng trí tuệ nhân tạo và công nghệ hiện đại. Chúng tôi sẽ tập trung vào các thành phần chính của ứng dụng bao gồm dữ liệu địa lý, yếu tố ảnh hưởng, mô hình dự đoán AI, và giao diện người dùng.

- **Phạm vi nghiên cứu**:

  Tập trung vào các khía cạnh quan trọng của hệ thống, với các điểm chính:
  - Phân tích và so sánh thuật toán: Nghiên cứu có thể tập trung vào việc phân tích và so sánh các thuật toán tìm đường đi ngắn nhất hiện có. Điều này có thể bao gồm đánh giá tính chính xác, hiệu suất, độ phức tạp thuật toán và khả năng áp dụng trong các tình huống cụ thể.
  - Tối ưu hóa đường đi: Nghiên cứu có thể tập trung vào việc tối ưu hóa đường đi ngắn nhất trong các bối cảnh cụ thể.
  - Ứng dụng và mô phỏng: Nghiên cứu có thể tập trung vào việc áp dụng thuật toán tìm đường đi ngắn nhất trong các lĩnh vực cụ thể như giao thông, quản lý đô thị, đi lại và vận tải. Nghiên cứu này có thể xây dựng mô phỏng hoặc hệ thống thực tế để kiểm tra và đánh giá hiệu quả của thuật toán trong các tình huống thực tế.
  - Tích hợp dữ liệu thời gian thực: Nghiên cứu có thể tập trung vào việc tích hợp dữ liệu thời gian thực vào quá trình tìm đường đi ngắn nhất. Điều này có thể bao gồm tích hợp thông tin về tình trạng giao thông, điều chỉnh đường đi dựa trên thông tin thời tiết hoặc sự kiện đặc biệt, hoặc đưa ra dự đoán về tình trạng giao thông trong tương lai.

4 Nhiệm vụ nghiên cứu

- **Nghiên cứu lý thuyết**: Tìm hiểu và phân tích các nguyên lý cơ bản của thuật toán A* và các công cụ, kiến thức liên quan.
- **Xây dựng mô hình**: Thiết kế và triển khai mô hình ứng dụng thuật toán A* để giải quyết bài toán tìm đường đi.

5 Công cụ sử dụng

- **Ngôn ngữ lập trình**: Python. Là ngôn ngữ lập trình bậc cao, có tính linh hoạt. Python thường được dùng để xây dựng trang web và phần mềm, tự động hóa các tác vụ và tiến hành phân tích dữ liệu. Python đã trở thành một trong những ngôn ngữ được sử dụng nhiều nhất hiện nay bởi sự mạnh mẽ, dễ hiểu của nó.
  - Ưu điểm:
    - Thư viện mở rộng
    - Cải thiện hiệu suất

        - Đơn giản và dễ dàng
        - Có thể đọc được
    - Nhược điểm
        - Giới hạn tốc độ
        - Phát hiện lỗi trong mã
        - Tiêu thụ bộ nhớ lớn
- **Môi trường:** Pycharm IDE (Integrated Development Environment) này hỗ trợ mạnh mẽ cho việc phát triển Python, cung cấp các tính năng như debugging, code completion, và tích hợp với các hệ thống quản lý version như Git.
- **Thư viện và công cụ hỗ trợ**:
  NumPy: Sử dụng để quản lý và tính toán các mảng số, rất hữu ích khi làm việc với ma trận trọng số hoặc bản đồ chi phí.
  Matplotlib: Dùng để trực quan hóa lộ trình tìm được bằng thuật toán A* trên bản đồ nhà máy.
  PIL: Một thư viện trong Python, dùng để xử lý và thao tác với hình ảnh
  Heapq: Một module trong Python, cung cấp các chức năng để làm việc với hàng đợi ưu tiên (Priority queue) dưới dạng Heap (Cấu trúc dữ liệu dạng cây nhị phân)

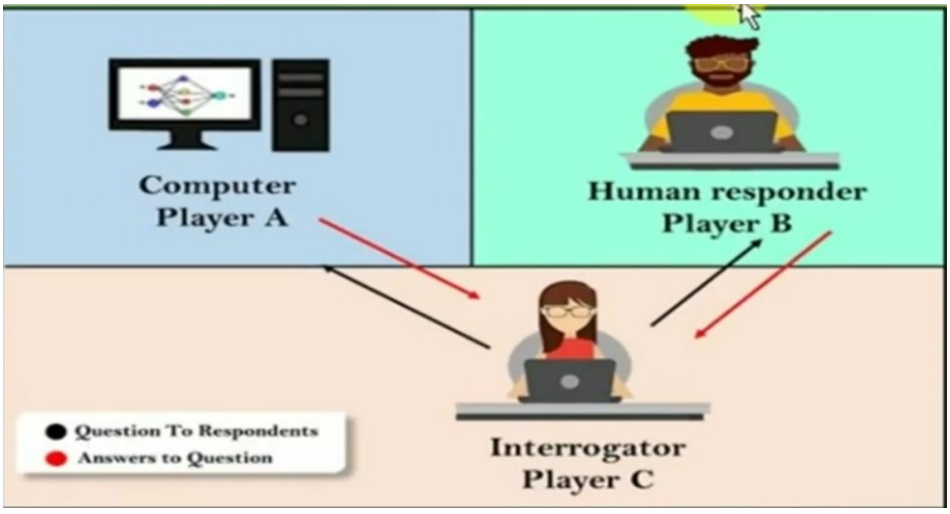
CHƯƠNG 1: Tổng quan về trí tuệ nhân tạo

1.1. Khái niệm trí tuệ nhân tạo

- Trong lĩnh vực Công nghệ thông tin, Trí tuệ nhân tạo (TTNT) cũng có thể hiểu là "thông minh nhân tạo", tức là sự thông minh của máy móc do con người tạo ra, đặc biệt tạo ra cho máy tính, robot, hay các máy móc có các thành phần tính toán điện tử. TTNT là một ngành mới, nhưng phát triển rất mạnh mẽ và đem lại nhiều kết quả to lớn. Mùa hè 1956, tại hội thảo ở Darmouth John McCarthy đã đưa ra thuật ngữ trí tuệ nhân tạo (Artificial Intelligence - AI). Mốc thời gian này được xem là thời điểm ra đời thực sự của lĩnh vực nghiên cứu TTNT.
- Trí tuệ nhân tạo là hướng đi của việc tạo ra máy tính, người máy điều khiển bằng máy tính hay là những phần mềm suy nghĩ thông minh hơn, tương tự như suy nghĩ thông minh của con người.
- Trí tuệ nhân tạo được học như bộ não con người, như cách mà con người học, quyết định và làm việc khi giải quyết một vấn đề, và sau đó sử dụng kết quả của quá trình học đó như là nền tảng của việc phát triển phần mềm và hệ thống thông minh.
- Ở thời điểm hiện tại, Thuật ngữ này thường dùng để nói đến các MÁY TÍNH có mục đích không nhất định và ngành khoa học nghiên cứu về các lý thuyết và ứng dụng của trí tuệ nhân tạo. Tức là mỗi loại trí tuệ nhân tạo hiện nay đang dừng lại ở mức độ những máy tính hoặc siêu máy tính dùng để xử lý một loại công việc nào đó như điều khiển một ngôi nhà, nhận diện hình ảnh, xử lý dữ liệu của bệnh nhân để đưa ra phác đồ điều trị, xử lý dữ liệu để tự học hỏi, khả năng trả lời các câu hỏi về chẩn đoán bệnh, trả lời khách hàng về các sản phẩm của một công ty,...

Hình 1.1: AI là một bộ phận của khoa học máy tính

- Mong muốn làm cho máy có những khả năng của trí thông minh con người đã có từ nhiều thế kỷ trước, tuy nhiên TTNT chỉ xuất hiện khi con người sáng tạo ra máy tính điện tử. Alan Turing – nhà toán học lỗi lạc người Anh, người được xem là cha đẻ của Tin học do đưa ra cách hình thức hóa các khái niệm thuật toán và tính toán trên máy Turing – một mô hình máy trừu tượng mô tả bản chất việc xử lý các ký hiệu hình thức - có đóng góp quan trọng cho TTNT vào năm 1950, gọi là phép thử Turing. Theo Turing: "Trí tuệ là những gì có thể đánh giá được thông qua các trắc nghiệm thông minh".



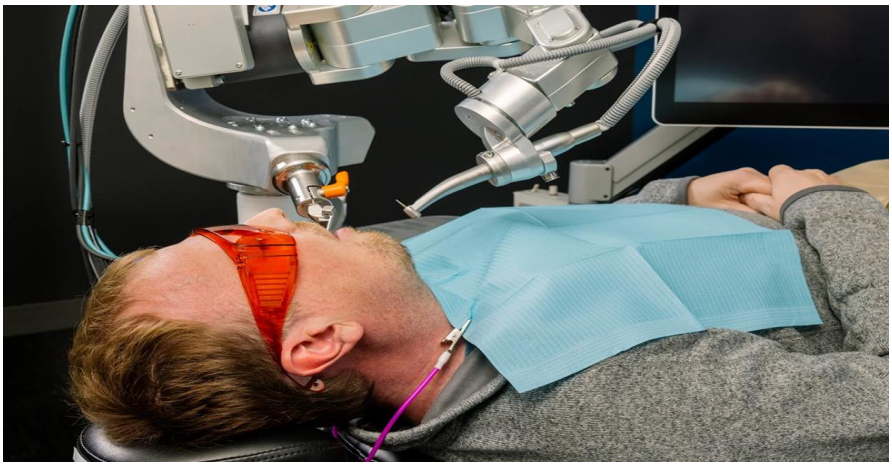
Hình 1.2: (Ảnh:interfect) Phép thử Turing

- Phép thử Turing là một cách để trả lời câu hỏi "Máy tính có biết nghĩ không?". Alan Turing đề xuất bộ kiểm thử (Turing test): Trong trắc nghiệm này, một máy tính và một người tham gia trắc nghiệm được đặt vào trong các căn phòng cách biệt với một người thứ hai (người thẩm vấn). Người thẩm vấn không biết được chính xác đối tượng nào là người hay máy tính, và cũng chỉ có thể giao tiếp với hai đối tượng đó thông qua các phương tiện kỹ thuật như một thiết bị soạn thảo văn bản, hay thiết bị đầu cuối. Người thẩm vấn có nhiệm vụ phân biệt người với máy tính bằng cách chỉ dựa trên những câu trả lời của họ đối với những câu hỏi được truyền qua thiết bị liên lạc này. Trong trường hợp nếu người thẩm vấn không thể phân biệt được máy tính với người thì khi đó theo Turing máy tính này có thể được xem là thông minh.
- Khái niệm trí tuệ đưa ra trong từ điển bách khoa toàn thư:

  Trí tuệ là khả năng phản ứng một cách thích hợp những tình huống mới thông qua hiệu chỉnh hành vi một cách thích đáng. Hiểu rõ những mối liên hệ qua lại của các sự kiện của thế giới bên ngoài nhằm đưa ra những hành động phù hợp đạt tới một mục đích nào đó. Hiện nay nhiều nhà nghiên cứu cho rằng, TTNT là lĩnh vực nghiên cứu sự thiết kế các tác nhân thông minh (bất cứ cái gì tồn tại trong môi trường và hành động một cách thông minh).
- TTNT là một ngành của khoa học máy tính - nghiên cứu xử lý thông tin bằng máy tính, do đó TTNT đặt ra mục tiêu nghiên cứu: làm thế nào thể hiện được các hành vi thông minh bằng thuật toán, rồi nghiên cứu các phương pháp cài đặt các chương trình có thể thực hiện được các hành vi thông minh bằng thuật toán, tiếp theo chúng ta cần chỉ ra tính hiệu quả, tính khả thi của thuật toán thực hiện một nhiệm vụ, và đưa ra các phương pháp cài đặt.
- Mục tiêu chính của AI là tạo ra các máy móc có khả năng suy nghĩ, học hỏi, giải quyết vấn đề và tự động hoá một cách thông minh giống như loài người. Các hệ thống AI được thiết kế để bắt chước cách thức

hoạt động của bộ não con người về khả năng suy luận logic, hiểu ngôn ngữ, nhận biết hình ảnh, ghi nhớ thông tin, v.v...

- Các khái niệm cơ bản về AI:
    - Học máy (Machine learning): Cho phép máy móc tự học cách thực hiện một nhiệm vụ mà không cần lập trình cụ thể.
    - Học sâu (Deep learning): Sử dụng các mạng nơ-ron nhân tạo để học cách nhận dạng mẫu từ dữ liệu lớn.
    - Xử lý ngôn ngữ tự nhiên (NLP): Cho phép máy hiểu và xử lý ngôn ngữ tự nhiên của con người.
    - Thị giác máy tính (Computer vision): Cho phép máy phân tích và hiểu hình ảnh, video.
    - Lập luận (Reasoning): Khả năng suy luận logic và đưa ra kết luận dựa trên kiến thức đã có.



Hình 1.3: Robot nha khoa của Perceptive thực hiện các thao tác nhanh, chính xác hơn so với các bác sĩ (Ảnh: Perceptive).

1.2. Vai trò của trí tuệ nhân tạo (Artifical Intelligence - AI)

Vai trò của AI là vô tận đối với cuộc sống của chúng ta. AI có thể tiếp cận với con người thông qua nhiều lĩnh vực, ngành nghề khác nhau. Ưu điểm của trí tuệ nhân tạo AI là khả năng xử lý dữ liệu khoa học hơn, nhanh hơn, hệ thống hơn so với con người. Việc phát triển và đưa các sản phẩm AI tới tay người dùng đúng cách sẽ thúc đẩy mạnh mẽ sự phát triển của toàn nhân loại. Mở ra một thế giới hoàn toàn mới cùng các giải pháp bù đắp cho những vấn đề mà con người không thể giải quyết.

- **Vai trò của trí tuệ nhân tạo trong y học**

  Công nghệ AI đã mở ra một trang mới cho nền y học thế giới, đặc biệt là nền y học nước nhà. Nó mang đến cho con người những giá trị đáng kinh ngạc trong việc bảo vệ sức khỏe và điều trị bệnh tật. Tại lĩnh vực này, AI có vai trò quan trọng trong việc hỗ trợ điều trị y tế như định lượng thuốc, các phương pháp điều trị khác nhau cho bệnh nhân và quy trình phẫu thuật trong phòng mổ. Chúng sử dụng những thuật toán phân tích để hỗ trợ bệnh nhân theo dõi kết quả điều trị 24/7.

Hình 1.4: Áp dụng AI trong y khoa

- **Vai trò của AI trong tài chính**

  AI là công cụ giúp con người xử lý các hoạt động trong ngân hàng như xử lý giao dịch, theo dõi số dư, quản lý tài sản và các tài khoản tiền gửi lớn một cách nhanh chóng và chính xác nhất. Trí tuệ nhân tạo không những giúp các ngân hàng hợp lý hóa giao dịch mà còn có thể ước tính cung, cầu và định giá chứng khoán một cách dễ dàng hơn.

Hinh 1.5: Áp dụng AI trong lĩnh vực tài chính

- **Vai trò của AI trong trò chơi và công nghệ**
  - Hiện nay, những tập đoàn lớn đang ngày càng thúc đẩy việc sử dụng máy móc thông minh vào dây chuyền sản xuất. AI được sử dụng như các robot có thể thay thế một phần công việc của con người. Khối lượng công việc và thời gian hoàn thành sẽ nhanh chóng và nhẹ nhàng hơn dưới sự hoạt động của máy móc tích hợp trí tuệ nhân tạo.
  - Tiêu biểu là với các sản phẩm như ô tô tự lái và trò chơi điện tử. Trong trò chơi điện tử, trí tuệ nhân tạo AI sẽ tự phân tích các hành vi và đưa ra những đáp án không kém cạnh với trí tuệ con người. Với ô tô tự lái, hệ thống AI tính toán tất cả các dữ liệu bên trong động cơ, tìm hiểu cách đi và ngăn chặn va chạm bởi chướng ngại vật
- **Sự kết hợp hoàn hảo của AI và robot hút bụi**
  - Nhắc đến trí tuệ nhân tạo, có lẽ điều đầu tiên họ thường nghĩ đến là robot. Đối với lĩnh vực dọn dẹp tự động hóa gia đình, AI là điều không thể thiếu. Kết hợp các công nghệ tiên tiến cùng công nghệ AI siêu thông minh, các dòng máy robot hút bụi tự động liên tục được ra mắt trên thị trường. Các sản phẩm tích hợp AI thường là những công cụ cao cấp nhất, đem lại hiệu quả cực lớn trong việc làm sạch sàn nhà của các hộ gia đình.
  - Với thời đại công nghệ 4.0 hiện nay, việc ứng dụng AI không còn xa lạ gì với cuộc sống của chúng ta. Trí tuệ nhân tạo có mặt trong mọi lĩnh vực đời sống từ giải trí cho đến y tế, xã hội. Đây chính là chìa khóa để mở ra một thế hệ mới đầy văn minh, thúc đẩy sự phát triển to lớn của loài người.

Hình 1.6: Áp dụng AI để sản xuất robot hút bụi

- **Vai trò của AI trong bảo tồn văn hóa của dân tộc**
  - Tác giả nhấn mạnh rằng: "Trí tuệ nhân tạo (AI) đóng vai trò quan trọng trong việc phục dựng và bảo tồn các tác phẩm mỹ thuật cổ, đặc biệt là tại Huế, nơi có nhiều di sản văn hóa cần được bảo tồn một cách tinh tế và chính xác" **(Nguyễn, 2021)** [2]

Hình 1.7: AI phục dựng tác phẩm mỹ thuật (Nguồn: Nguyễn, 2021)

- **Sự vượt trội của Kỹ thuật xử lý tri thức trong TTNT so với kỹ luật lập trình truyền thống**

Bảng 1.1: So sánh kỹ thuật lập trình truyền thống và kỹ thuật xử lý tri thức trong AI

Chương trình truyền thống	Kỹ thuật TTNT
Xử lý dữ liệu	Xử lý tri thức
Bản chất chương trình là tính toán, xử lý theo các thuật toán	Bản chất chương trình là lập luận, xử lý theo các thuật giải heuristics
Xử lý tuần tự theo lô	Xử lý theo chế độ tương tác
Xử lý thông tin chính xác đầy đu	Xử lý được các thông thin không chắc chắn, không chính xác
Chương trình = Cấu trúc dữ liệu + Giải thuật	AI = Tri thức + Suy diễn
Không giải thích trong quá trình thực hiện	Có thể giải thích hành vi hệ thống trong quá trình thực hiện

- **Tác động của AI đến sản xuất trong nền công nghiệp 4.0 như sau:**
  - Chất lượng – Năng suất dự đoán: Vai trò của trí tuệ nhân tạo đầu tiên là giảm thiểu các hao tổn trong sản xuất và ngăn ngừa các quy trình sản xuất kém hiệu quả. Khi nhu cầu ngày càng tăng để đáp ứng sự cạnh tranh thì trí tuệ nhân tạo là điều vô cùng cần thiết.
  - Bảo trì sự dự đoán: Một trong những lợi ích của trí tuệ nhân tạo nữa là bảo trì dự đoán. Thay vì việc bảo trì theo lịch trình định trước thì bảo trì dự đoán sẽ sử dụng thuật toán để dự đoán lỗi tiếp theo của một bộ phận/máy móc/hệ thống. Nhờ đó có thể cảnh báo nhân viên thực hiện các quy trình bảo trì tập trung để ngăn chặn sự cố. Bảo trì dự đoán có ưu điểm là giảm đáng kể chi phí trong khi loại bỏ nhu cầu về thời gian ngừng hoạt động theo kế hoạch trong nhiều trường hợp. Ngoài ra, nhờ nó mà Tuổi thọ hữu dụng còn lại của máy móc và thiết bị lâu hơn.
  - Kết hợp giữa robot và con người

Hinh 1.8: AI chiến thắng kiện tướng cờ vua thế giới Gary Kasparov (Nguồn VnExpress)

Tính đến năm 2020, ước tính có khoảng 1,64 triệu robot công nghiệp đang hoạt động trên toàn thế giới. Robot sản xuất được chấp thuận làm việc cùng với con người để tăng năng suất công việc. [3]

Khi áp dụng robot ngày càng nhiều thì AI sẽ đóng một vai trò quan trọng trong việc đảm bảo an toàn cho con người. Đồng thời trao cho robot nhiều trách

nhiệm hơn trong việc đưa ra các quyết định có thể tối ưu hóa các quy trình dựa trên dữ liệu thời gian thực được thu thập từ sàn sản xuất.

- Thiết kế sáng tạo: Nhà sản xuất có thể tận dụng trí tuệ nhân tạo vào giai đoạn thiết kế. Khi có bản tóm tắt thiết kế được xác định rõ rành làm đầu vào thì các nhà kỹ sư, thiết kế có thể sử dụng thuật toán AI. Mục đích để khám phá tất cả các cấu hình có thể có của một giải pháp.
- Nhu cầu cung ứng thị trường: Hiện nay trí tuệ nhân tạo đang hiện hữu ở mọi nơi trong hệ sinh thái công nghiệp 4.0. Nhà sản xuất có thể sử dụng các thuật toán AI để tối ưu hóa chuỗi cung ứng của các hoạt động sản xuất. Đồng thời giúp họ phản ứng và dự đoán tốt hơn những thay đổi trên thị trường.

1.3. Một số kỹ thuật trong Trí tuệ nhân tạo

Trong thế giới thực, Tri thức có một vài thuộc tính như sau:

- Dung lượng đồ sộ, phi thường.
- Tổ chức tốt, định dạng tốt.
- Luôn luôn cập nhật sự thay đổi.

Kỹ thuật Trí tuệ nhân tạo là một cách để tổ chức và sử dụng tri thức có hiệu quả trong những cách sau đây:

- Có thể nhận thức được người đã cung cấp cho nó.
- Có thể sửa đổi dễ dàng để sửa lỗi.
- Nó có thể hữu ích trong một số tình huống dù nó chưa hoàn thiện hoặc chưa chính xác lắm.

Kỹ thuật Trí tuệ nhân tạo nâng cao tốc độ thực thi của những chương trình phức tạp.

Một số kỹ thuật Trí tuệ nhân tạo cơ bản :

- Lý thuyết giải bài toán và suy diễn thông minh
- Lý thuyết tìm kiếm may rủi
- Các ngôn ngữ về TTNT
- Lý thuyết thể hiện tri thức và hệ chuyên gia
- Lý thuyết nhận dạng và xử lý tiếng nói
- Người máy
- Tâm lý học xử lý thông tin
- Xử lý danh sách, kỹ thuật đệ quy, quay lui và xử lý cú pháp hình thức

1.4. Lịch sử phát triển

- **Giai đoạn đầu: Những ý tưởng ban đầu (1940 - 1950)**
    - Khái niệm sơ khai: Ý tưởng về máy móc có khả năng tư duy xuất hiện từ thời Hy Lạp cổ đại. Tuy nhiên, phải đến giữa thế kỷ 20, với sự ra đời của máy tính điện tử, ý tưởng này mới trở nên khả thi.
    - Mạng thần kinh nhân tạo: Warren McCulloch và Walter Pitts đã đề xuất mô hình toán học đầu tiên để xây dựng một mạng lưới thần kinh nhân tạo, đặt nền móng cho một trong những kỹ thuật cốt lõi của AI hiện đại.
- **Giai đoạn vàng: Sự hưng phấn và những kỳ vọng lớn (1950 - 1960)**
    - Hội nghị Dartmouth: Năm 1956, hội nghị Dartmouth đánh dấu sự ra đời chính thức của thuật ngữ "trí tuệ nhân tạo". Các nhà khoa học hàng đầu đã tập trung để thảo luận về khả năng tạo ra máy móc có khả năng học hỏi, suy luận và tự cải thiện.
    - Các nhà nghiên cứu đã phát triển các chương trình có thể giải quyết các bài toán phức tạp bằng cách mô phỏng quá trình suy luận của con người.
- **Mùa đông AI lần thứ nhất: Sự thất vọng và cắt giảm ngân sách (Những năm 1970)**
    - Những hạn chế của máy tính: Các máy tính thời đó còn quá chậm và hạn chế về bộ nhớ để giải quyết các bài toán phức tạp như ngôn ngữ tự nhiên hay nhận dạng hình ảnh.
    - Cắt giảm ngân sách: Khi những kỳ vọng ban đầu không được đáp ứng, chính phủ và các tổ chức tài trợ đã cắt giảm đáng kể ngân sách cho nghiên cứu AI.
- **Sự hồi sinh và phát triển mạnh mẽ (1980 - nay)**
    - Hệ thống chuyên gia: Sự phát triển của hệ thống chuyên gia, những chương trình có thể đưa ra quyết định dựa trên một lượng lớn kiến thức chuyên môn, đã giúp AI tìm lại vị thế của mình.
    - Mạng thần kinh nhân tạo (Deep learning): Sự phát triển của sức mạnh tính toán và thuật toán học sâu đã tạo ra một cuộc cách mạng trong lĩnh vực AI. Các mô hình học sâu đã đạt được những thành tựu đáng kể trong các lĩnh vực như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên,…

- Ứng dụng rộng rãi**:** AI hiện nay được ứng dụng trong rất nhiều lĩnh vực, từ y tế, tài chính đến tự động hóa và xe tự lái.

1.5. Các thành phần trong hệ thống của Trí tuệ nhân tạo

- Hệ thống trí tuệ nhân tạo bao gồm hai thành phần cơ bản đó là biểu diễn tri thức và tìm kiếm tri thức trong miền biểu diễn:

TTNT = Tri thức + Suy diễn

- Tri thức của bài toán có thể được phân ra làm ba loại cơ bản đó là tri thức mô tả, tri thức thủ tục và tri thức điều khiển.
- Để biểu diễn tri thức người ta sử dụng các phương pháp sau đây:
  - Phương pháp biểu diễn nhờ luật
  - Phương pháp biểu diễn nhờ mạng ngữ nghĩa
  - Phương pháp biểu diễn nhờ bộ ba liên hợp OAV
  - Phương pháp biểu diễn nhờ Frame
  - Phương pháp biểu diễn nhờ logic vị từ
- Sau khi tri thức của bài toán đã được biểu diễn, kỹ thuật trong lĩnh vực trí tuệ nhân tạo là các phương pháp tìm kiếm trong miền đặc trưng tri thức về bài toán đó. Với mỗi cách biểu diễn sẽ có các giải pháp tương ứng.

1.6. Các lĩnh vực nghiên cứu và ứng dụng cơ bản

Trí tuệ nhân tạo có những ảnh hưởng vượt trội trong nhiều lĩnh vực như:

● Game: Trí tuệ nhân tạo đóng vai trò cốt yếu trong những game chiến lược như cờ, đánh bài, tic-tac-toe (như cờ caro), ... nơi mà máy móc có thể suy nghĩ số lớn những trường hợp có khả năng xảy ra dựa trên tri thức.

● Xử lý ngôn ngữ tự nhiên: Nó có khả năng tương tác với máy tính, hiểu ngôn ngữ tự nhiên mà con người nói.

● Hệ thống chuyên môn hóa: Có một vài ứng dụng mà các máy móc thông minh, phần mềm và những thông tin đặc biệt để suy luận. Nó giải thích và đưa ra lời khuyên cho người dùng hệ thống đó.

● Hệ thống thị giác: Hệ thống có thể hiểu, phân tích và tiếp thu dữ liệu vào thuộc về thị giác ngay trên máy tính. Ví dụ như:

○ Những máy bay do thám chụp lại hình ảnh, sau đó sử dụng kỹ thuật này để mô hình hóa những thông tin không gian, bản đồ của khu vực.

○ Bác sĩ sử dụng hệ thống buồng bệnh chuyên môn để chẩn đoán cho bệnh nhân.

○ Cảnh sát có thể sử dụng phần mềm máy tính để nhận diện khuôn mặt của tội phạm từ những hình chân dung được vẽ lại bởi những họa sĩ pháp y.

● Nhận diện lời nói: Một vài hệ thống thông minh có khả năng nghe và tiếp thu ngôn ngữ trong cấu trúc và nghĩa của câu trong khi con người nói. Nó có thể nắm bắt được độ nhấn mạnh khác nhau, từ lóng, tiếng ồn phía sau, sự thay đổi trong âm thanh của con người do trời lạnh, ...

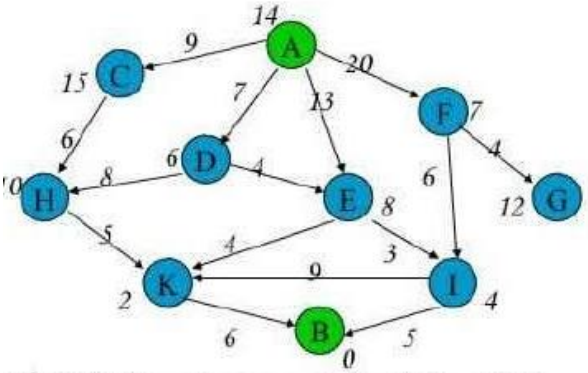
● Nhận diện chữ viết tay được viết trên giấy bằng bút hoặc viết trên màn hình bằng bút cảm ứng. Nó nhận dạng được hình dạng của chữ và chuyển nó thành văn bản có thể chỉnh sửa được.

● Người máy thông minh: Người máy có khả năng thực hiện nhiệm vụ mà con người giao cho. Nó có các cảm biến để nhận dạng các dữ liệu vật lý trong thế giới thực như ánh sáng, hơi nóng, nhiệt độ, sự di chuyển, âm thanh, sự va chạm và áp lực. Nó được trang bị bộ xử lý hiệu quả, đa cảm biến và bộ nhớ lớn để thể hiện sự thông minh. Hơn thế nữa, nó có khả năng học từ lỗi sai của nó và thích nghi với môi trường mới.

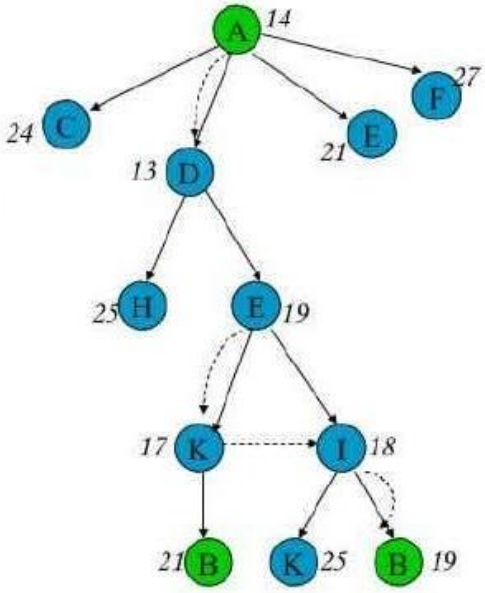
CHƯƠNG 2: Tìm hiểu về thuật toán A*

2.1. Giới thiệu về thuật toán

- Trong khoa học máy tính, A* (A* Search) là 1 thuật toán tìm kiếm trong đồ thị. Thuật toán này tìm một đường đi từ 1 nút khởi đầu tới 1 nút cho trước (hoặc tới 1 nút thỏa mãn 1 điều kiện đích). Thuật toán này sử dụng 1 đánh giá heuristic để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó. Thuật toán này duyệt các nút theo thứ tự của đánh giá heurristic này. Do đó thuật toán A* là 1 ví dụ của tìm kiếm theo lựa chọn tốt nhất (Best - First- Search).
- Thuật toán A* được mô tả lần đầu vào năm 1968 bởi Peter Hart và Bertram Rafael. Trong bài báo của họ, thuật toán được gọi là thuật toán A; khi sử dụng thuật toán này với 1 đánh giá heuristic thích hợp sẽ thu được hoạt động tối ưu, do đó mà có tên là A*.



Đồ thị không gian trạng thái với hàm đánh giá



Cây tìm kiếm theo thuật toán A*

*Hình 2.1: Đồ thị không gian và cây tìm kiếm A**

2.1.1. Heuristic chấp nhận được

Trong kỹ thuật tìm kiếm, để việc tìm kiếm có hiệu quả sẽ sử dụng hàm đánh giá để hướng dẫn tìm kiếm. Các kỹ thuật này thuộc nhóm tìm kiếm Heuristic.

- Giả sử u là một trạng thái đạt tới (có đường đi từ trạng thái đầu u0 tới u); hàm đánh giá được xác định như sau:
  - ✓ *g(u)*: đánh giá độ dài đường đi ngắn nhất từ u_0 tới u.
  - ✓ *h(u)*: đánh giá độ dài đường đi ngắn nhất từ u tới trạng thái đích.Hàm h(u) được gọi là chấp nhận được nếu với mọi trạng thái u, h(u) ≤ độ dài đường đi ngắn nhất thực tế từ u tới trạng thái đích.
  - ✓ Để tăng hiệu quả của quá trình tìm kiếm: *f(u) = g (u) + h(u)*
- Heuristic chấp nhận được trong A*
  - ✓ Heuristic *h(n)* là chấp nhận được nếu với mọi node *n*, *h(n) ≤ h(n),* trong đó *h (n)* là chi phí thực để đi đến đích từ *n*.
  - ✓ Heuristic chấp nhận được không bao giờ đánh giá chi phí cao quá thực tế.
  - ✓ *Định lý:* Nếu *h(n)* là chấp nhận được, A là thuật toán cho lời giải tối ưu.

2.1.2. Ý tưởng

Xét bài toán tìm đường - bài toán mà A* thường được dùng để giải. A* xây dựng tăng dần tất cả các tuyến đường từ điểm xuất phát cho tới khi nó tìm thấy một đường đi chạm tới đích. Tuy nhiên, cũng như tất cả các phương pháp tìm kiếm có thông tin nó chỉ xây dựng các tuyến đường "có vẻ" dẫn về phía đích.

Để biết những tuyến đường nào có khả năng sẽ dẫn tới đích, A* sử dụng một "đánh giá heuristic" về khoảng cách từ điểm bất kỳ cho trước tới đích. Trong trường hợp tìm đường đi, đánh giá này có thể là khoảng cách đường chim bay - một đánh giá xấp xỉ thường dùng cho khoảng cách của đường giao thông.

Điểm khác biệt của A* đối với BFS (Tìm kiếm theo chiều rộng) là nó còn tính đến khoảng cách đã đi qua. Điều đó làm cho A* "đầy đủ" và "tối ưu", nghĩa là, A* sẽ luôn luôn tìm thấy đường đi ngắn nhất nếu tồn tại một đường đi như thế. A* không đảm bảo sẽ chạy nhanh hơn các thuật toán tìm kiếm đơn giản hơn. Trong

một môi trường dạng mê cung, cách duy nhất để đến đích có thể là trước hết phải đi về phía xa đích và cuối cùng mới quay lại. Trong trường hợp đó, việc thử các nút theo thứ tự "gần đích hơn thì được thử trước" có thể gây tốn thời gian.

2.1.3. Mô tả thuật toán

Giả sử n là một trạng thái đạt tới (Có đường đi từ trạng thái n0 tới n).

A* lưu giữ một tập các lời giải chưa hoàn chỉnh, nghĩa là các đường đi qua đồ thị, bắt đầu từ nút xuất phát. Tập lời giải này được lưu trong một hàng đợi ưu tiên. Thứ tự ưu tiên gán cho một đường đi n được quyết định bởi hàm:

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

Trong đó:

- *g(n):* là chi phí của đường đi cho đến thời điểm hiện tại, nghĩa là tổng trọng số của các cạnh đã đi qua.
- *h(n):* là hàm đánh giá heuristic về chi phí nhỏ nhất để đến đích từ n
- *f(n):* ước lượng tổng giá đến đích qua n

2.1.4. Bài báo liên quan

- Bài báo với tiêu đề “Path Planning with Modified A* Algorithm for a Mobile Robot” của nhà xuất bản Elsevier vào 12/2014 đã đề cập đến việc lập kế hoạch đường đi cho một robot di động dựa trên bản đồ lưới. Tác giả František Duchoň và Andrej Babinec cũng nhấn mạnh vào sự ưu việt về khả năng tính toán, tính tối ưu của đường đi. [4]
- Bài báo "Path Planning of Automated Guided Vehicles Based on Improved A-Star Algorithm" của Chunbao Wang, Lin Wang, và Zhengzhi Wu thảo luận về vai trò quan trọng của hệ thống AGV trong việc tăng cường hiệu suất sản xuất và tự động hóa logistics. Các tác giả đã đề xuất một thuật toán A* cải tiến để giải quyết vấn đề lập kế hoạch đường đi ngắn nhất và tránh va chạm trong hệ thống AGV. Thuật toán này đã được kiểm chứng qua các mô phỏng và thí nghiệm, chứng minh tính hiệu quả và khả thi trong thực tế. [5]

2.2. Thuật toán KD-Tree

KD-Tree (K-Dimensional Tree) là một cấu trúc dữ liệu cây được thiết kế để hiệu quả trong việc tìm kiếm không gian đa chiều, đặc biệt là trong các bài toán

khoảng cách và tìm kiếm gần nhất. Thuật toán này thường được sử dụng trong các ứng dụng như xử lý hình ảnh, trí tuệ nhân tạo, xử lý văn bản, và các lĩnh vực khác có liên quan đến dữ liệu đa chiều.

Chúng thường được sử dụng trong các ứng dụng yêu cầu tìm kiếm gần nhất, như trong đồ họa máy tính, xử lý hình ảnh, xử lý văn bản, và nhiều lĩnh vực khác liên quan đến dữ liệu đa chiều.

2.2.1. Cấu trúc của KD-Tree

(1) Khái niệm chung

- Một KD-Tree được sắp xếp dưới dạng cây nhị phân, trong đó mỗi nút của cây đại diện cho một điểm dữ liệu trong không gian nhiều chiều.
- Mỗi nút có một giá trị chia (split value) và một chiều cụ thể (split dimension) mà nó sử dụng để chia không gian dữ liệu.

(2) Xây dựng KD-Tree

- Việc xây dựng cây thường được thực hiện bằng cách lựa chọn điểm dữ liệu làm nút gốc, sau đó chọn chiều và giá trị chia tạo thành các nhánh con.
- Quá trình đệ quy được thực hiện trên các tập dữ liệu con cho đến khi không còn dữ liệu hoặc cây đã đạt đến một độ sâu cố định.

(3) Chọn chiều và giá trị chia

- Đối với mỗi nút, cần chọn chiều và giá trị chia sao cho tạo ra hai tập con gồm các điểm dữ liệu có giá trị trong chiều được chọn nhỏ hơn và lớn hơn giá trị chia.
- Thông thường, chiều được chọn lặp lại theo chu kỳ và giá trị chia được chọn là trung bình hoặc trung vị của các giá trị trong chiều đó.

(4) Tìm kiếm trong KD-Tree

- Khi thực hiện tìm kiếm trong KD-Tree, bắt đầu từ nút gốc và di chuyển xuống cây theo các nhánh dựa trên giá trị của điểm tìm kiếm.
- Khi di chuyển xuống, so sánh giá trị tìm kiếm với giá trị chia của nút để quyết định hướng đi tiếp.

(5) Đánh giá

Bảng 2.1: Đánh giá về thuật toán KD-Tree

Ưu điểm	Hạn chế
Hiệu suất tìm kiếm tốt trong không gian đa chiều so với các phương pháp tìm kiếm trực tiếp. Phù hợp cho dữ liệu thưa thớt: Đặc biệt với dữ liệu không đồng đều phân bố trong không gian.	Không linh hoạt với dữ liệu thay đổi: Khi dữ liệu thay đổi, cây cần được xây dựng lại từ đầu. Điều này làm giảm hiệu suất của KD-Tree Chi phí xây dựng cây có thể tốn kém đối với các tập dữ liệu lớn hoặc có chiều cao lớn. Không hiệu quả đối với không gian thấp: không cao bằng các phương pháp khác như tìm kiếm tuyến tính.

2.2.2. Hàng đợi ưu tiên

"Hàng Đợi Ưu Tiên" (Priority Queue) là một khái niệm quan trọng trong lĩnh vực trí tuệ nhân tạo, đặc biệt là trong các thuật toán tìm kiếm, đồ thị, và xử lý dữ liệu. Dưới đây là một phân tích chi tiết về Hàng Đợi Ưu Tiên trong trí tuệ nhân tạo:

(1) Đặc điểm cơ bản của Hàng Đợi Ưu Tiên

- Ưu Tiên: Mỗi phần tử trong hàng đợi được gán một giá trị ưu tiên. Các phần tử với giá trị ưu tiên cao hơn được ưu tiên xử lý trước.
- Cấu trúc dữ liệu: Hàng đợi ưu tiên thường được triển khai thông qua các cấu trúc dữ liệu như heap (đống) để đảm bảo thời gian truy cập và cập nhật là O(log n), n là số lượng phần tử.

(2) Ứng dụng của Hàng Đợi Ưu Tiên

- Ứng dụng trong trí tuệ nhân tạo:
  - Thuật toán tìm kiếm: Trong các thuật toán như thuật toán A* (A-star) sử dụng hàng đợi ưu tiên để tối ưu hóa việc chọn các đỉnh trong đồ thị.
  - Xử lý đồ thị: Trong các thuật toán như Dijkstra và Prim, hàng đợi ưu tiên giúp duyệt qua các đỉnh theo thứ tự ưu tiên.

- Ứng dụng trong thực tế:
  - Sử dụng hàng đợi ưu tiên để xác định đường đi ngắn nhất từ một điểm đến tất cả các điểm khác. Ví dụ: Đường đi ngắn nhất trong bản đồ…
  - Quản lý tác vụ đồng thời theo thứ tự ưu tiên.
  - Trong một số thuật toán máy học, như học tăng cường hàng đợi ưu tiên có thể được sử dụng để quản lý các trạng thái và hành động theo thứ tự ưu tiên.
  - Lập lịch và quản lý sự kiện: Hàng đợi ưu tiên giúp quản lý và xử lý các sự kiện theo ưu tiên độ ưu tiên cao trong các hệ thống AI thời gian thực

(3) Thực hiện hàng đợi ưu tiên trong trí tuệ nhân tạo

- Heap: Đống là cấu trúc dữ liệu thường được sử dụng để triển khai hàng đợi ưu tiên. Heap có thể là max-heap (phần tử lớn nhất ở đỉnh) hoặc min-heap (phần tử nhỏ nhất ở đỉnh).
- Duyệt đồ thị: Trong thuật toán Dijkstra, mỗi đỉnh được thăm và cập nhật độ dài ngắn nhất từ nguồn đến nó thông qua heap để đảm bảo đỉnh có độ ưu tiên thấp nhất được xử lý trước.

(4) Đánh giá

Bảng 2.2: Đánh giá hàng đợi ưu tiên (Priority Queue)

Ưu điểm	Nhược điểm
○ Hiệu suất cao: Xử lý các phần tử ưu tiên cao mức độ hiệu quả. ○ Linh hoạt: Thích ứng với nhiều ứng dụng khác nhau. ○ Dễ dàng triển khai (Sử dụng các cấu trúc dữ liệu như heap)	○ Cần cập nhật ổn định: Nếu giá trị ưu tiên thay đổi thường xuyên, cần duy trì tính ổn định của hàng đợi ưu tiên. ○ Không phù hợp cho mọi tình huống (Các trường hợp yêu cầu quá trình cập nhật tần suất cao).

2.3. Cài đặt thuật toán

(1) Khởi tạo

- Khởi tạo một danh sách mở (OPEN): Chứa đỉnh khởi đầu T_0
- Khởi tạo một danh sách đóng (CLOSE): Rỗng
- Đặt $g(T_0)=0$: Chi phí từ đỉnh khởi đầu đến đỉnh hiện tại là 0
- Tính toán hàm ước lượng $h(T_0)$: Hàm ước lượng từ T_0 đến đích
- Tính toán: $f(T_0) = g(T_0)+h(T_0)$

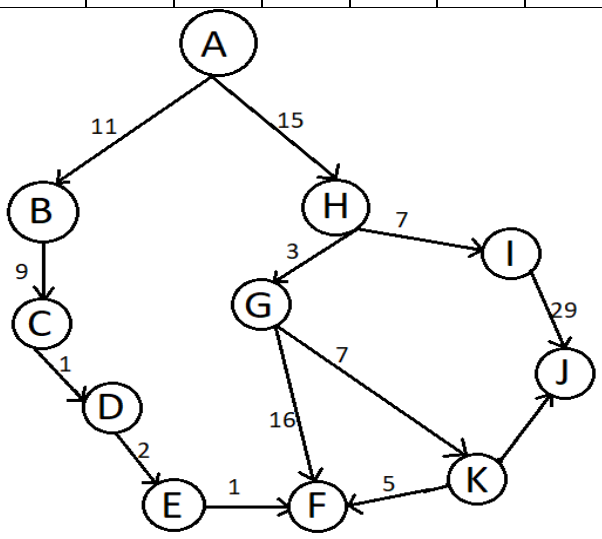
(2) Lặp lại các bước sau cho đến khi OPEN rỗng:

- Chọn đỉnh n từ OPEN sao cho f(n) nhỏ nhất
- Nếu n chính là đích TG, thì trả về đường đi từ T_0 đến TG
- Nếu n không phải đích, kiểm tra các đỉnh kề m:
  - Nếu m chưa có trong OPEN, tính toán các giá trị:
    - $g(m) = g(n) + cost(n, m)$; $h(m)$; $f(m) = g(m) + h(m)$
    - Đặt CHA(m) = n
    - Thêm m vào OPEN
  - Nếu m đã có trong OPEN, và g(m) > g(n) + Cost(n, m)
    - g(n) = g(n) + Cost(n, m)
    - f(m) = g(m) + h(m)
    - CHA(m) = n
  - Nếu m có trong CLOSE, và g(m) > g(n) + Cost(n, m):
    - Bỏ m khỏi CLOSE
    - Thêm m vào OPEN
- Di chuyển đỉnh n từ OPEN sang CLOSE.

2.4. Ví dụ minh họa

Đề bài: Trạng thái ban đầu T0 = A, trạng thái đích Goal = {K}. Các số ghi cạnh các cung là độ dài đường đi. Và ta có bảng chi phí ước lượng như sau:

T_i	A	B	C	D	E	F	G	H	I	J	K
$h(T_i)$	60	53	36	35	35	19	16	38	23	0	7



*Hình 2.2: Sơ đồ A**

Hãy ước lượng khoảng cách từ đỉnh hiện tại đến cho đến đỉnh kết thúc?

Giải

Bước	n	Các đỉnh kề	OPEN	CLOSE
0			A_{60}	
1	A	B_{64}, H_{53}	B_{64}, H_{53}	A
2	H	G_{34}, I_{45}	B_{64}, G_{34}, I_{45}	A, H
3	G	K_{32}, F_{53}	B_{64}, I_{45}, K_{32}, F_{53}	A, H, G
4	K			

Ban đầu:

OPEN = {A, g(A) = 0, f(A) = 60}

Phát triển đỉnh A, sinh ra các đỉnh con: B, H. Tính giá trị của hàm f tại các đỉnh này ta có:

OPEN = {g(B) = 11, f(B) = 11 + 53 = 64, cha(B) = A,
g(H) = 15, f(H) = 15 + 38 = 53, cha(H) = A}

CLOSE = {A, g(A) = 0, f(A) = 60}

Do f(H) = 53 nhỏ nhất → Chọn H để phát triển.

Phát triển đỉnh H, sinh ra các đỉnh con: G, I

g(G) = g(H) + Cost(H, G) = 15 + 3 = 18, f(G) = 18 + 16 = 34

g(I) = g(H) + Cost(H, I) = 15 + 7 = 22, f(I) = 22 + 23 = 45

OPEN = { g(B) = 11, f(B) = 11 + 53 = 64, cha(B) = A,
g(G) = 18, f(G) = 34, cha(G) = H,
g(I) = 22, f(I) = 45, cha(I) = H}

CLOSE = {A, g(A) = 0, f(A) = 60,
g(H) = 15, f(H) = 15 + 38 = 53, cha(H) = A}

Do f(G) = 34 nhỏ nhất → Chọn G để phát triển.

Phát triển đỉnh G, sinh ra các đỉnh con: K, F

g(K) = g(G) + Cost(G, K) = 18 + 7 = 25, f(K) = 25 + 7 = 32

g(F) = g(G) + Cost(G, F) = 18 + 16 = 34, f(F) = 34 + 19 = 53

OPEN = { g(B) = 11, f(B) = 11 + 53 = 64, cha(B) = A,
g(I) = 22, f(I) = 45, cha(I) = H,
g(K) = 25, f(K) = 32, cha(K) = G,

g(F) = 34, f(F) = 53, cha(F) = G}

CLOSE = {A, g(A) = 0, f(A) = 60,

g(H) = 15, f(H) = 15 + 38 = 53, cha(H) = A,

g(G) = 18, f(G) = 34, cha(G) = H }

Do K∈ Goal nên quá trình tìm kiếm kết thúc. Để đưa ra đường đi, ta truy ngược lại trong tập CLOSE. Khi đó đường đi tìm được có chi chí C(p) = 32 và trình tự các đỉnh là p: A → H → G → K

CHƯƠNG 3: Xây dựng chương trình

3.1. Bài toán

3.1.1. Giới thiệu

3.1.2. Phát biểu bài toán

Khi một robot cần vận chuyển hàng hóa từ khu vực này sang khu vực khác trong nhà máy, để tối ưu hóa thời gian và chi phí di chuyển, hệ thống sẽ sử dụng bản đồ của khu vực đó. Hệ thống sẽ tiếp nhận dữ liệu bản đồ dưới dạng hình ảnh và xử lý chúng theo mô hình đã được huấn luyện.

Quá trình bắt đầu bằng việc chuyển đổi bản đồ thành dạng ma trận. Sau khi dữ liệu đã được xử lý, thuật toán tìm kiếm đường đi A* sẽ được áp dụng để xác định lộ trình tối ưu cho robot. Cuối cùng, hệ thống sẽ cung cấp lộ trình tối ưu cùng với tổng chi phí di chuyển toàn bộ quãng đường.

3.1.3. Cơ sở dữ liệu

Cơ sở dữ liệu trong bài toán tìm đường đi cho robot vận chuyển là các hình ảnh bản đồ trong nhà máy có đuôi file .png

3.1.4. Yêu cầu bài toán

- Mục tiêu: Xây dựng được một hệ thống tìm đường đi ngắn nhất trên bản đồ và đưa ra lộ trình tối ưu cho Robot vận chuyển.
- Độ chính xác dữ liệu: Dữ liệu đầu vào được lấy từ bản đồ chi tiết của nhà máy bao gồm vị trí của các chướng ngại vật, lối đi, và các khu vực hoạt động. Nhờ vào việc sử dụng mô hình học máy đã được huấn luyện trên dữ liệu lịch sử, kết hợp với dữ liệu thời gian thực, hệ thống có thể xác định đường đi tối ưu với độ chính xác cao, đảm bảo robot di chuyển hiệu quả và an toàn.
- Hiệu suất và độ chính xác của mô hình: Phải đạt hiệu suất tốt nhất có thể và chính xác để đưa ra lộ trình tối ưu cả về mặt thời gian và chi phí.
- Tính khả thi triển khai: Mô hình cần phải khả thi để triển khai và áp dụng thực tế, bao gồm việc tính toán và xử lý thời gian thực.
- Khả năng mở rộng và quản lý: Hệ thống dễ dàng mở rộng và cũng có thể quản lý một cách dễ dàng.

3.2. Cài đặt chương trình

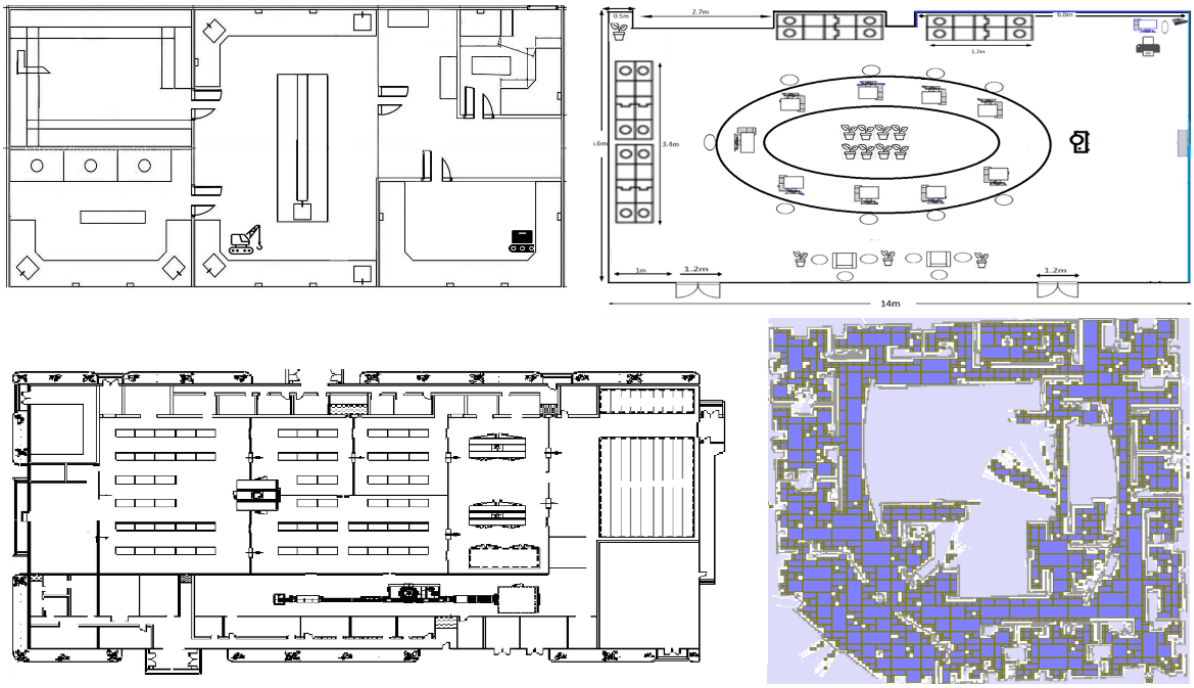
Chương trình tìm đường đi ngắn nhất cho robot vận chuyển được triển khai bằng ngôn ngữ Python dựa trên môi trường cục bộ (Laptop) và sử dụng Pycharm Community Edition

3.2.1. Thuật toán

- Input: Bản đồ (2D) của nhà máy, điểm bắt đầu di chuyển và điểm đích
- Output: Lộ trình đường đi tối ưu, thời gian, chi phí ước lượng để vận chuyển

3.2.2. Cài đặt

- Chương trình sẽ chia làm 3 module:
  - **Module1 (Astar.py):** Module này chứa hàm tính khoảng cách Heuristic, và hàm thuật toán A*
  - **Module2 (Image_Processing.py):** Module này có nhiệm vụ xử lý hình ảnh.
    - load_image( ): Giúp hệ thống mở hình ảnh và chuyển sang grayscale (Thang độ xám, tức là hình ảnh sẽ chỉ có màu đen hoặc trắng)
    - convert_to_matrix( ): Chuyển hình ảnh grayscale ở trên thành ma trận nhị phân.
  - **Module3 (Main.py):** Module chính, nơi các module khác được thực thi
- Chương trình sẽ đọc dữ liệu từ file hình ảnh của bản đồ (.png). Dưới đây là các bản đồ sẽ được tìm đường đi ngắn nhất trên đó:



Hình 3.1: Hình ảnh các bản đồ của nhà máy cần được xử lý

- **Module1 Image_Processing.py**: Xử lý hình ảnh
  - Thêm thư viện numpy và module "Image" từ thư viện PIL
    - Numpy: Một thư viện phổ biến trong Python, dùng để xử lý mảng và tính toán toán học trên các dữ liệu số.
    - PIL: Một thư viện Python có chức năng xử lý các hình ảnh.

```
from PIL import Image
import numpy as np
```

Hình 3.2: Hình ảnh thư viện được sử dụng trong module xử lý hình ảnh

  - Hàm load_image(image_path): Chức năng của hàm này dùng để mở hình ảnh và chuyển sang hình ảnh dạng grayscale

```
def load_image(image_path):
    #Mở hình ảnh và chuyển sang grayscale
    image = Image.open(image_path)
    image = image.convert('L')
    return image
```

Hình 3.3: Hình ảnh hàm mở và chuyển hình ảnh sang dạng grayscale

Trong đó: Image.open(image_path) dùng để mở hình ảnh;
image.convert('L') dùng để chuyển hình ảnh sang dạng Grayscale.
'L' là chế độ trong PIL để chuyển hình ảnh thành 9 bit-pixels (Mỗi pixel có một giá trị từ 0 <Đen> đến 255 <Trắng>)

  - Hàm convert_to_matrix(image, threshold = 128): Chuyển đổi hình ảnh grayscale thành ma trận nhị phân dựa trên giá trị ngưỡng threshold

```
def convert_to_matrix(image, threshold=128):
    #Chuyển hình ảnh grayscale thành ma trận nhị phân
    image_matrix = np.array(image)
    binary_matrix = (image_matrix >= threshold).astype(int)
    return binary_matrix
```

Hình 3.4: Hình ảnh hàm xử lý hình ảnh

*Chú ý: "Threshold": Là ngưỡng cường độ ánh sáng được sử dụng để phân loại các pixel trong hình ảnh. Mặc định là 128.

- **Module2 Astar.py:** Đây là module triển khai thuật toán A* để tìm đường đi ngắn nhất từ điểm bắt đầu đến điểm đích trên ma trận nhị phân được lấy ra từ module Image_Processing. Trong ma trận này, giá trị '1' đại diện cho một ô có thể đi qua, và '0' đại diện cho vật cản.
    - 'heapq': Là thư viện được sử dụng để quản lý hàng đợi ưu tiên, trong đó các phần tử được sắp xếp dựa trên chi phí nhỏ nhất. Đây là một cấu trúc dữ liệu quan trọng trong thuật toán A* để luôn mở rộng các đỉnh có chi phí ước lượng thấp nhất trước.
    - Hàm 'heuristic': Tính toán khoảng cách Manhattan giữa hai điểm a và b. Khoảng cách Manhattan là tổng giá trị tuyệt đối của sự khác biệt giữa các tọa độ tương ứng của hai điểm. Đây là một cách ước lượng phổ biến trong các bài toán tìm đường trên lưới.

```python
def heuristic(a, b):
    # Tính khoảng cách Heuristic bằng cách sử dụng khoảng cach Manhattan
    return abs(a[0] - b[0]) + abs(a[1] - b[1])
```

Hình 3.5: Hình ảnh hàm heuristic tính khoảng cách Manhattan giữa 2 điểm

- Hàm 'a_star_search':

```python
def a_star_search(binary_matrix, start, goal):
    # Thuật toán A*
    rows, cols = binary_matrix.shape
    MO = []
    heapq.heappush(MO, (0, start))
    DONG = {}
    g_n = {start: 0}
    f_n = {start: heuristic(start, goal)}
```

Hình 3.6: Hình ảnh hàm a_star_search

- 'rows', 'cols': Số hàng và số cột của ma trận
- 'MO': Tập mở được quản lý bởi hàng đợi ưu tiên, ban đầu được khởi tạo bởi điểm bắt đầu (start) với tổng chi phí 0
- 'DONG': Theo dõi các điểm đã được xét và lưu điểm cha của từng điểm (để xây dựng đường đi).
- 'g_n': Từ điển lưu chi phí từ điểm bắt đầu đến mỗi điểm.
- 'f_n': Từ điển lưu chi phí tổng 'f(x) = g(x) + h(x)' cho mỗi điểm.

- Vòng lặp chính của thuật toán:

```
while MO:
    #Điểm có chi phí min lấy đc lấy từ MO
    n = heapq.heappop(MO)[1]
    if n == goal:
        path = []
        while n in DONG:
            path.append(n)
            n = DONG[n]  # Cập nhật n để di chuyển về điểm bắt đầu
        path.append(start)
        path.reverse()
        return path
```

*Hình 3.7: Hình ảnh vòng lặp chính của thuật toán A**

- Lấy điểm n có chi phí thấp nhất trong MO
- Nếu 'n' là điểm đích, thuật toán sẽ xây đựng đường đi bằng cách truy vết ngược lại từ 'goal' về 'start' qua các điểm lưu trong 'DONG'.

- Xét các điểm lân cận:

```
for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:   #Tìm các điểm lân cận: Trên, dưới, trái, phải
    m = (n[0] + dx, n[1] + dy)
    if 0 <= m[0] < rows and 0 <= m[1] < cols:
        if binary_matrix[m[0], m[1]] == 0:  # Kiểm tra vật cản
            continue

        tentative_g_n = g_n[n] + 1

        if m not in g_n or tentative_g_n < g_n[m]:
            DONG[m] = n
            g_n[m] = tentative_g_n
            f_n[m] = tentative_g_n + heuristic(m, goal)
            heapq.heappush(MO, (f_n[m], m))
```

Hình 3.8: Hình ảnh vòng lặp xét các điểm lân cận và tính toán chi phí

- dx, dy: Các cặp tọa độ đại diện cho 4 hướng di chuyển (trên dưới, trái, phải).
- m: Tọa độ của điểm lân cận được tính dựa trên tọa độ của 'n'.
- Kiểm tra tính hợp lệ của điểm lân cận m:
  - Điểm lân cận phải nằm trong phạm vi ma trận.
  - Điểm lân cận không phải là vật cản.
- Tính toán chi phí g(x):

- Nếu ‘m’ chưa có trong ‘g_n’ hoặc đã có trong ‘g_n’ nhưng chi phí mới thấp hơn chi phí cũ thì cập nhật điểm cha của ‘m’, chi phí ‘g_n[m]’ và ‘f_n[m]’.
- Thêm điểm ‘m’ vào tập mở MO với chi phí ‘f_n[m]’.
- Nếu không tìm thấy đường đi trả về ‘None’.

- **Module3 Main.py:** Là module chính, nơi tất cả các module khác được thực thi chương trình. Ở đây, người dùng có thể chọn điểm bắt đầu, đích trên hình ảnh, sau đó thực hiện thuật toán A* rồi hiển thị kết quả lên màn hình.
  - Hàm lấy tọa độ: Thực hiện xử lý lấy tọa độ khi ấn lên hình ảnh được hiển thị bằng matplotlib.

```
def onclick(event, location):
    #Click chuột & Lấy tọa độ
    if event.xdata and event.ydata:
        location.append((int(event.ydata), int(event.xdata)))
        plt.plot( *args: event.xdata, event.ydata, 'go' if len(location) == 1 else 'ro')
        plt.draw()
```

Hình 3.9: Hình ảnh hàm onclick lấy tọa độ

Giải thích:

- if event.xdata and event.ydata: Điều kiện để kiểm tra tọa độ (x, y) có hợp lệ hay không. Điều kiện sẽ lỗi khi thiếu 1 trong 2 tọa độ x hoặc y; hoặc nhấp chuột vào khu vực nằm ngoài hình ảnh.
- location.append((int(event.ydata), int(event.xdata))): Tọa độ dưới dạng tuple sẽ được thêm vào một List location.
  *Chú ý: Tọa độ được chuyển về kiểu số nguyên để dễ xử lý
- plt.plot(event.xdata, event.ydata, 'go' if len(location) == 1 else 'ro'): Vẽ điểm trên hình ảnh có tọa độ nhấp chuột.

Bảng 3.1: Giải thích màu của điểm bắt đầu, điểm đích

‘go’(Green Circle)	‘ro’ (Red Circle)
Nếu List Location chỉ có MỘT điểm (Điểm bắt đầu)	Nếu List Location có nhiều hơn một điểm (Điểm đích)

- plt.draw( ): Vẽ và hiển thị lên màn hình các thay đổi.

  - Hàm chọn điểm bắt đầu và điểm kết thúc:

```python
def select_points(binary_matrix):
    #Chon điểm bắt đầu, đích
    coords = []
    plt.imshow(binary_matrix, cmap='gray')
    plt.gcf().canvas.mpl_connect( s: 'button_press_event', lambda event: onclick(event, coords))
    plt.show()
    return coords
```

Hình 3.10: Hình ảnh hàm chọn điểm bắt đầu và điểm kết thúc

Giải thích:

- Coords = [ ]: Là một list rỗng lưu trữ tọa độ các điểm được chọn
- plt.imshow(binary_matrix, cmap='gray'): Hàm imshow( ) của thư viện matplotlib có tác dụng hiển thị hình ảnh từ mã nhị phân binary_matrix với bảng màu xám.
- plt.gcf(): Trả về đối tượng hình ảnh matplotlib đang làm việc
- lambda event: onclick(event, coords): Hàm ẩn danh (Lambda) dùng để gọi hàm onclick( )
- .canvas.mpl_connect(…, …): Kết nối một sự kiện nhấp chuột có tên "button_press_event" với một hàm xử lý sự kiện.
- Plt.show( ): Hiển thị cửa sổ hình ảnh, lúc này người dùng có thể nhập điểm bắt đầu, đích bằng cách nhấp chuột trái.
- Return coords: Trả về list coords chứa các tọa độ điểm mà người dùng đã chọn.

- Hàm vẽ đường đi trong ma trận

```python
def display_result(binary_matrix, path):
    #Vẽ đường đi trong ma trận
    plt.imshow(binary_matrix, cmap='gray')
    if path:
        plt.plot( *args: *zip(*[(y, x) for x, y in path]), color='blue', linewidth=2)
    plt.show()
```

Hình 3.11: Hình ảnh hàm vẽ đường đi trong ma trận

Giải thích:

- plt.imshow(binary_matrix, cmap='gray'): Hàm imshow( ) của thư viện matplotlib có tác dụng hiển thị hình ảnh từ mã nhị phân binary_matrix với bảng màu xám.

- If(path): Kiểm tra xem ‘path’ có chứa dữ liệu hay không. ‘Path’ là danh sách các tọa độ (Các điểm trên đường đi). Nếu có, câu lệnh bên dưới sẽ được thực thi.
- [(y, x) for x, y in path]: Biểu thức List Comprehension này chuyển đổi tuple từ định dạng (x, y) về dạng (y, x). Bởi vì:

Bảng 3.2: So sánh tọa độ điểm ở trong ma trận và ở Matplotlib

Trong ma trận (binary_matrix)	Trong hệ tọa độ Matplotlib
Tọa độ được biểu diễn dưới dạng (row, column), tương ứng với tọa độ (y, x)	Tọa độ được xử lý dưới dạng là (x, y). Với x: Tọa độ ngang và y: Tọa độ dọc.

- zip(*[(y, x) for x, y in path]): Sau khi đã chuyển về tọa độ dạng (y, x), hàm zip sẽ chia danh sách tuple(y, x) thành hai danh sách riêng biệt: Một danh sách chứa toàn bộ các giá trị y (Hàng), và một danh sách chưa toàn bộ các giá trị x (Cột).
- plt.show( ): Hiển thị hình ảnh.

- Đoạn code thực hiện chương trình chính:

```
if len(points) == 2:
    start, goal = points
    if matrix[start[0], start[1]] == 1 and matrix[goal[0], goal[1]] == 1:
        path, f_n = a_star_search(matrix, start, goal)
        display_result(matrix, path) if path else print("Không tìm thấy đường đi.")
        if f_n is not None:
            print("Tổng chi phí: ", f_n[goal])
    else:
        print("Lỗi khi chọn điểm")
```

Hình 3.12: Hình ảnh đoạn code thực hiện chương trình chính

Giải thích:

- If len(points) == 2: Đảm bảo đã có điểm bắt đầu và điểm đích
- Start, goal = points: Gán lần lượt Start là tọa độ điểm bắt đầu, goal là tọa độ điểm đích.
- if matrix[start[0], start[1]] == 1 and matrix[goal[0], goal[1]] == 1: Kiểm tra xem cả điểm bắt đầu và điểm đích đều không nằm trên các vật cản.

- path = a_star_search(matrix, start, goal): Danh sách chứa tọa độ các điểm trên đường đi từ điểm bắt đầu (Start) tới điểm đích (Goal)
- display_result(matrix, path) if path else print("Không tìm thấy đường đi."): Nếu thuật toán A* không tìm được đường đi thì dòng thông báo sẽ được in ra màn hình.
- else: print("Lỗi khi chọn điểm"): Ngược lại, dòng thông báo trên sẽ được in ra, cảnh báo lỗi cho người dùng biết rằng có lỗi trong quá trình chọn điểm bắt đầu, điểm đích.

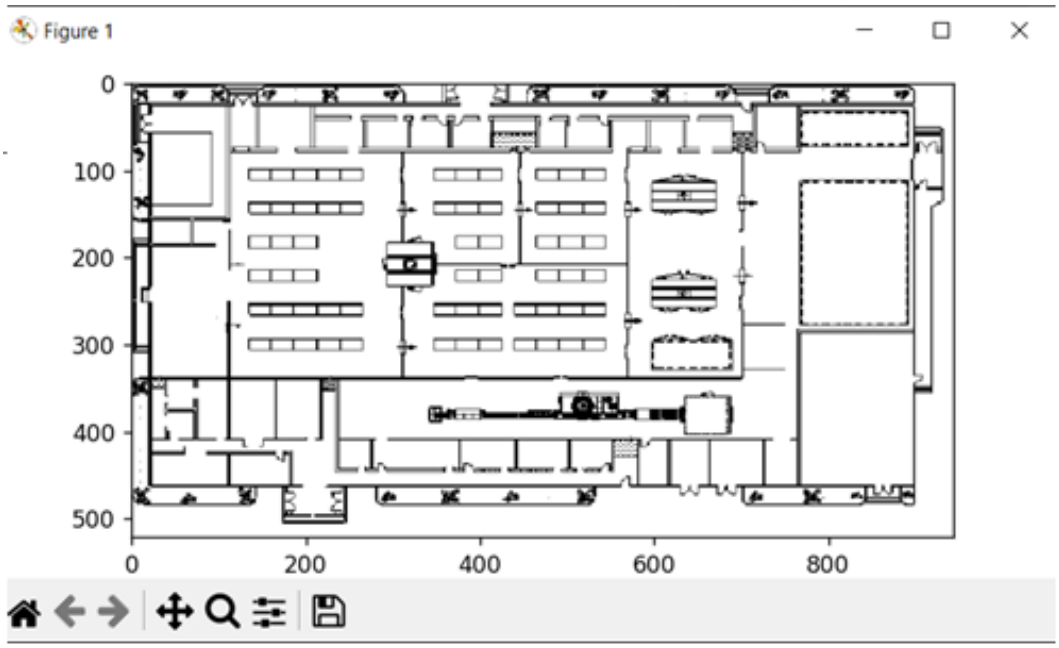
*Chú ý:

- Trước khi import các thư viện, bạn phải đảm bảo rằng thư viện đã tồn tại trong máy. Nếu thư viện chưa tồn tại, bạn có thể tải bằng cách sử dụng câu lệnh "pip" ở terminal của Pycharm.
- **heapq**: Là module chuẩn (Có sẵn khi tải Python), không cần cài đặt.

Pip install matplotlib numpy pillow

3.3. Quy trình và kết quả

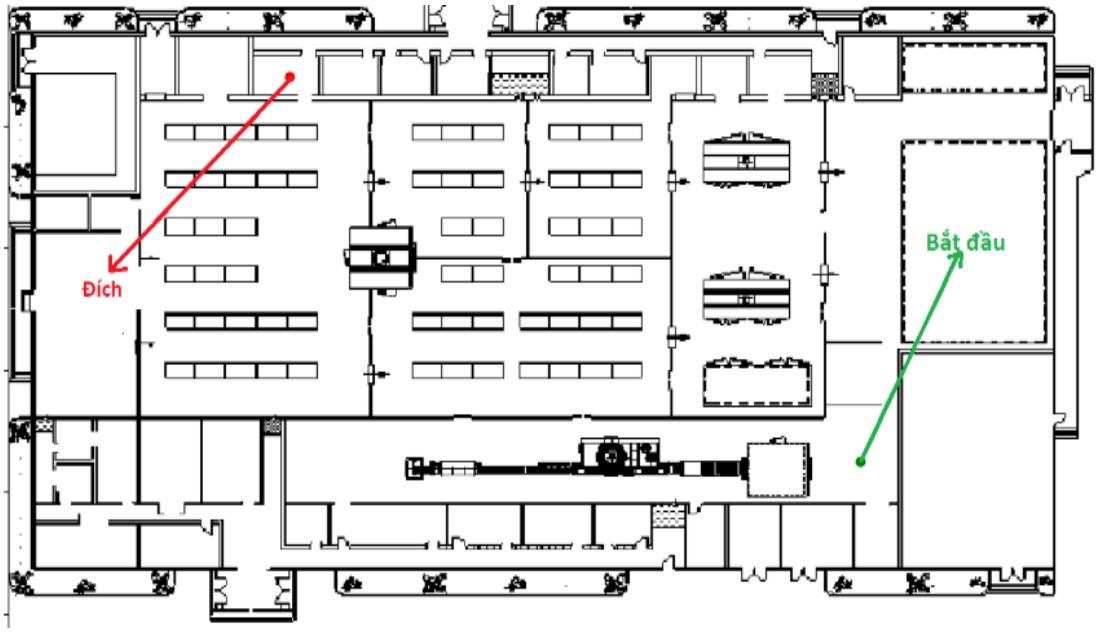
(1) Khi người dùng chạy chương trình trong module Main.py, hệ thống sẽ hiển thị giao diện của thư viện matplotlib



Hình 3.13: Hình ảnh giao diện bản đồ trong matplotlib

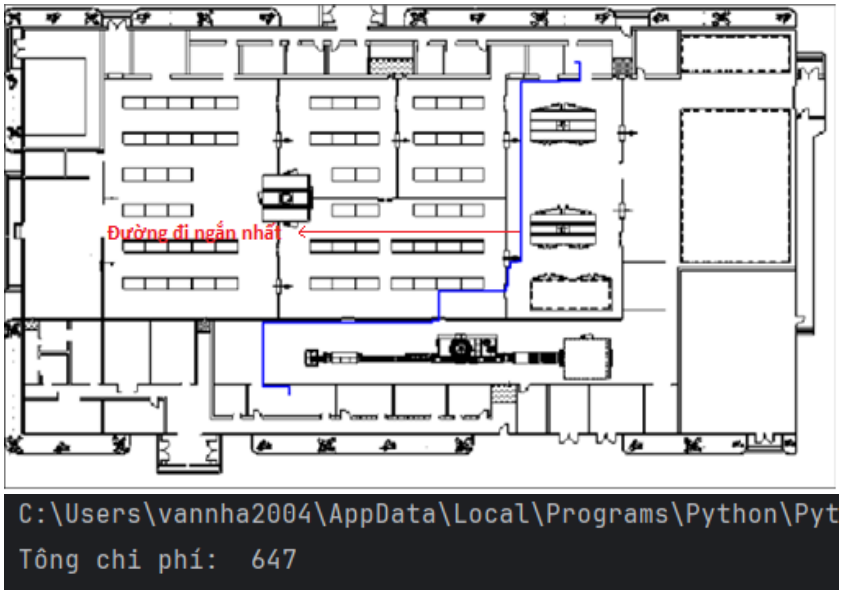
(2) Tiếp theo, người dùng chọn điểm bắt đầu và điểm đích bằng cách nhấn chuột trái trực tiếp lên màn hình. (Điểm bạn lựa chọn phải nằm ở trong khung, không được chọn vùng trắng ở ngoài bản đồ)

- Chấm xanh: Biểu tượng cho điểm bắt đầu
- Chấm đỏ: Biểu tượng cho điểm đích.



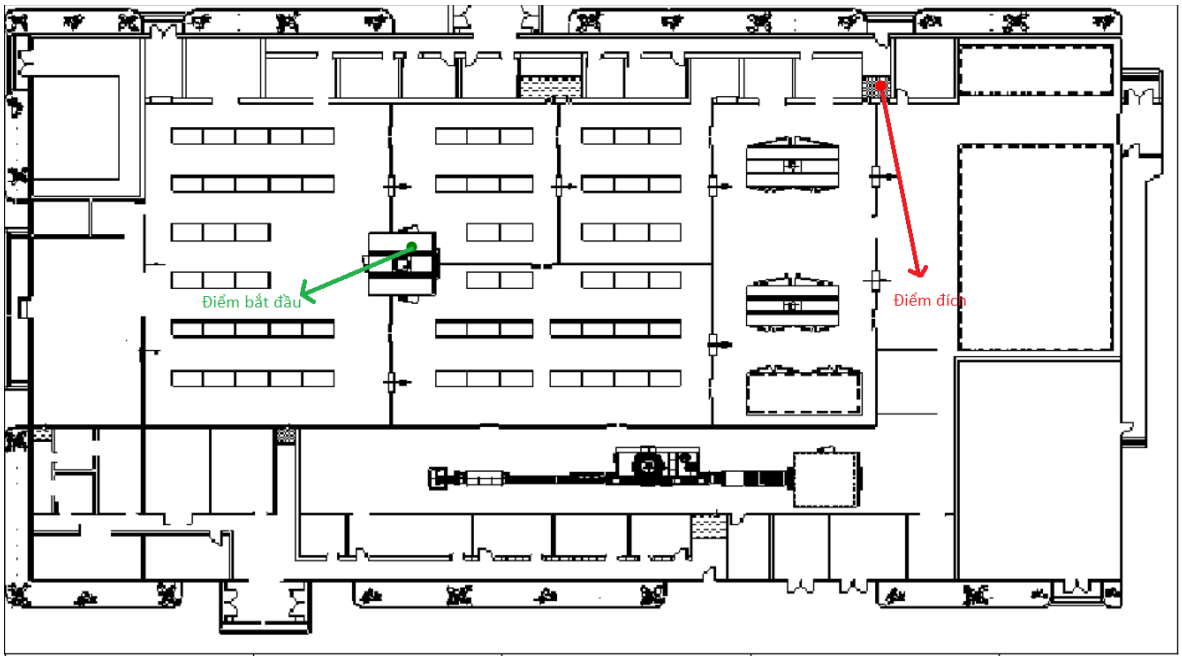
Hình 3.14: Hình ảnh bản đồ khi chọn điểm bắt đầu, điểm đích

(3) Nếu điểm bắt đầu, điểm đích được chọn đúng cách, và tồn tại đường đi thì hệ thống sẽ hiển thị lộ trình trực tiếp ở trên bản đồ và chi phí của lộ trình:

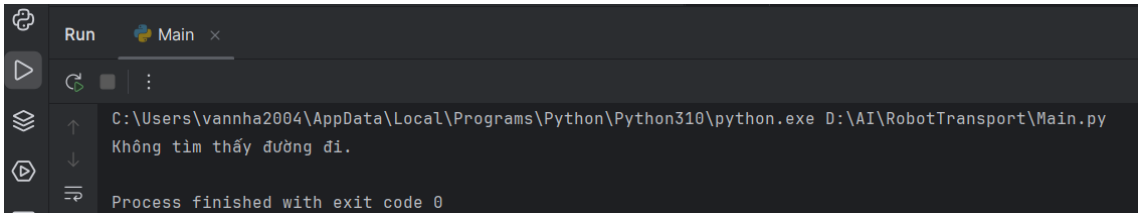


Hình 3.15: Hình ảnh lộ trình đường đi ngắn của robot trên bản đồ

Nếu không tồn tại đường đi, hoặc điểm bắt đầu, đích trùng với vật cản thì hệ thống sẽ thông báo không tìm được đường đi. Sau đó chương trình kết thúc:



Hình 3.16: Hình ảnh điểm bắt đầu, điểm đích không thỏa mãn



Hình 3.17: Hệ thống thông báo không tìm thấy đường đi

KẾT LUẬN

Hệ thống tìm đường đi ngắn nhất cho robot vận chuyển đã thành công trong việc xây dựng một giải pháp hiệu quả để xác định lộ trình tối ưu. Việc sử dụng thuật toán A* kết hợp với xử lý hình ảnh để tạo ma trận nhị phân giúp hệ thống hoạt động với độ chính xác cao và hiệu suất tốt. Toàn bộ quy trình được triển khai trong môi trường Python với cấu trúc module rõ ràng, bao gồm xử lý hình ảnh, thực thi thuật toán tìm đường, và giao diện người dùng để nhập điểm bắt đầu và điểm đích. Thuật toán A* không chỉ cải thiện đáng kể khả năng di chuyển của robot mà còn đảm bảo rằng robot có thể tránh được các vật cản, tối ưu hóa lộ trình và tiết kiệm năng lượng.

Một yếu tố quan trọng dẫn đến thành công của ứng dụng này là việc chuyển đổi môi trường thực tế thành lưới ô vuông, trong đó mỗi ô vuông đại diện cho một vị trí mà robot có thể hoặc không thể đi qua. Kỹ thuật này cho phép thuật toán A* hoạt động hiệu quả trong việc tìm ra đường đi ngắn nhất bằng cách phân tích và so sánh các ô lân cận cho đến khi đạt được mục tiêu.

Tóm lại, bài tập lớn này không chỉ minh chứng tiềm năng của thuật toán A* trong việc giải quyết các bài toán thực tế mà còn mở ra nhiều cơ hội phát triển trong lĩnh vực robot và tự động hóa. Việc áp dụng thuật toán A* đã mang lại những kết quả tích cực, góp phần vào sự phát triển của các hệ thống robot thông minh và tự hành trong tương lai.

TÀI LIỆU THAM KHẢO

[1] SaoNguyen, *Robot vận chuyển hàng trong nhà máy*, RTC Technology, 2024 https://rtc.edu.vn/robot-van-chuyen-hang-trong-nha-may/#:~=Robot%20v%E1%BA%ADn%20chuy%E1%BB%83n%20h%C3%A0ng%20h%C3%B3a,m%C3%A1y%20s%E1%BA%A3n%20xu%E1%BA%A5t%20tin%20d%C3%B9ng. [Truy cập 12/08/2024]

[2] Trần Thị Hoài Diễm, *Suy nghĩ về vai trò của trí tuệ nhân tạo (AI) trong phục dựng, bảo tồn tác phẩm mỹ thuật cổ ở Huế*, ResearchGate, 2018, https://www.researchgate.net/publication/349477446_Suy_nghi_ve_vai_tro_cua_tri_tue_nhan_tao_AI_trong_phuc_dung_bao_ton_tac_pham_my_thuat_co_o_Hue [Truy cập ngày 15/8/2024]

[3] Bảo Lâm, *Ván cờ lịch sử giữa con người và 'AI đời đầu'*, VnExpressm, 2023, https://vnexpress.net/van-co-lich-su-giua-con-nguoi-va-ai-doi-dau-4608531.html, [Truy cập ngày 16/5/2024]

[4] František Duchoň, Andrej Babinec, *Path Planning with Modified a Star Algorithm for a Mobile Robot*, Elsevier, 2014, https://www.sciencedirect.com/science/article/pii/S187770581403149X?ref=pdf_download&fr=RR-2&rr=8b6130794c8204c8 [Truy cập ngày 20/08/2024]

[5] C.Wang, L.Wang, Z.Wu, *Path planning of automated guided vehicles based on improved A-Star algorithm*, IEEE Xplore, 2020, https://ieeexplore.ieee.org/abstract/document/9296641