TRƯỜNG ĐẠI HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN

MÔN HỌC: Nhập môn trí tuệ nhân tạo

BÁO CÁO:

Tìm hiểu và cài đặt ứng dụng giải thuật BFS

Robot tìm mục tiêu bằng thuật toán BFS

SVTH: ĐINH THẠCH BẢO
MSSV: 207CT27605
GVHD: Ths. Phan Hồ Viết Trường

TP. Hồ Chí Minh – năm 2025

LỜI CẢM ƠN

Nhóm chúng em xin chân thành cảm ơn Thầy Phan Hồ Viết Trường đã tận tình giảng dạy và hướng dẫn nhóm trong suốt quá trình học tập và thực hiện báo cáo đề tài *"Tìm hiểu và cài đặt ứng dụng giải thuật BFS – Robot tìm mục tiêu bằng thuật toán BFS"* thuộc môn *Nhập môn Trí tuệ nhân tạo*.

Chúng em cũng xin gửi lời cảm ơn đến Nhà trường và Bộ môn đã tạo điều kiện về cơ sở vật chất, tài liệu học tập, cũng như hỗ trợ kỹ thuật để nhóm có thể hoàn thành bài báo cáo này.

Mặc dù đã cố gắng hết sức, nhưng do thời gian và trình độ còn hạn chế, bài báo cáo không thể tránh khỏi những thiếu sót. Kính mong Thầy thông cảm và đóng góp ý kiến để nhóm có thể hoàn thiện hơn trong các lần thực hiện sau.

Trân trọng cảm ơn!

LỜI CAM ĐOAN

Tôi xin cam đoan đề tài: “Phát triển hệ hỏi đáp đường đi trên địa bàn Quận 1 - TPHCM bằng giọng nói” là sản phẩm cá nhân do chính bản thân tôi nỗ lực nghiên cứu và phát triển trong quá trình học tập tại trường đại học Văn Lang, không có sự sao chép của người khác dưới bất kỳ hình thức. Trong quá trình hoàn thiện khóa luận, tôi có tham khảo một số tài liệu có nguồn gốc rõ ràng, dưới sự hướng dẫn của thầy ThS. Phan Hồ Viết Trường - Trường đại học Văn Lang. Nếu có bất kỳ vấn đề nào xảy ra, tôi xin hoàn toàn chịu trách nhiệm.

TP. Hồ Chí Minh, ngày .... tháng .... năm 2025

Giảng viên hướng dẫn	Sinh viên thực hiện
(Ký và ghi rõ họ tên)	(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Họ tên Giảng viên: ……

Nhận xét:

……
……
……
……

Điểm số: ……

Chữ ký Giảng viên

……

Lời Mở Đầu

Trong thời đại công nghệ số phát triển mạnh mẽ như hiện nay, trí tuệ nhân tạo (AI) đóng vai trò ngày càng quan trọng trong việc giải quyết các bài toán phức tạp trong đời sống, đặc biệt trong lĩnh vực tự động hóa và robot. Một trong những vấn đề cơ bản nhưng mang tính ứng dụng thực tiễn cao là bài toán tìm kiếm đường đi ngắn nhất — một yếu tố then chốt giúp các hệ thống robot vận hành hiệu quả hơn.

Thuật toán Breadth-First Search (BFS) là một trong những giải pháp đơn giản, phổ biến và dễ cài đặt trong các bài toán tìm kiếm trên không gian trạng thái. Với khả năng tìm đường đi ngắn nhất trên đồ thị hoặc bản đồ dạng lưới, BFS thường được ứng dụng trong nhiều lĩnh vực như điều hướng bản đồ, robot tìm đường, lập kế hoạch di chuyển trong môi trường có chướng ngại vật.

Đề tài "Robot tìm mục tiêu bằng thuật toán BFS" được thực hiện nhằm mục đích tìm hiểu, phân tích, mô phỏng và đánh giá hiệu quả của thuật toán BFS trong việc tìm kiếm đường đi tối ưu cho robot trong môi trường mô phỏng thực tế. Thông qua việc nghiên cứu lý thuyết kết hợp thực hành mô phỏng, đồ án hướng đến việc củng cố kiến thức về trí tuệ nhân tạo, thuật toán tìm kiếm cũng như khả năng xây dựng và triển khai ứng dụng thực tiễn.
Đồ án bao gồm ba chương:

Chương I trình bày cơ sở lý thuyết về thuật toán BFS.
Chương II trình bày quá trình ứng dụng thuật toán BFS vào mô phỏng robot tìm đường.
Chương III trình bày kết quả thực nghiệm cùng với đánh giá hiệu quả.

STT	MSSV	Họ tên sinh viên	Trách nhiệm được phân công	Trọng số đóng góp
1	207CT27605	Đinh Thạch Bảo		100%

MỤC LỤC

CHƯƠNG I: CƠ SỞ LÝ THUYẾT THUẬT TOÁN BFS

1.1 Định nghĩa thuật toán BFS

Tìm kiếm theo chiều rộng (BFS) là một thuật toán duyệt hoặc tìm kiếm cây hoặc cấu trúc đồ thị. Nó bắt đầu từ một nút nguồn và khám phá tất cả các nút gần đó ở độ sâu hiện tại trước khi chuyển sang các nút ở độ sâu tiếp theo. BFS sử dụng hàng đợi để quản lý các nút sẽ được ghé đến.

Nguyên lý hoạt động:
- BFS mở rộng các nút theo **chiều rộng** của cây tìm kiếm, nghĩa là nó sẽ **duyệt tất cả các nút ở mức hiện tại trước khi chuyển sang mức tiếp theo**.

Cấu trúc dữ liệu sử dụng:
- BFS sử dụng **hàng đợi FIFO (First-In-First-Out)** để lưu trữ các trạng thái lá của cây tìm kiếm.

(T. N. Việt and P. H. V. Trường, Nhập môn Trí tuệ nhân tạo – Chương 1: Bài toán tìm kiếm, Trường Đại học Văn Lang, 2025.)

1.2 Nguyên lý hoạt động của thuật toán BFS

Thuật toán Breadth-First Search (BFS) hoạt động dựa trên nguyên tắc duyệt các đỉnh hoặc trạng thái theo từng mức chiều sâu từ gốc đến các lớp tiếp theo. Thay vì đi sâu vào một nhánh như DFS (Depth-First Search), BFS mở rộng tất cả các nút lân cận trước khi tiếp tục mở rộng các nút ở cấp sâu hơn. Điều này giúp BFS luôn đảm bảo tìm ra đường đi ngắn nhất trong không gian tìm kiếm không có trọng số.

Quy trình hoạt động cụ thể:
Khởi tạo:

Đặt nút xuất phát (hoặc trạng thái ban đầu) vào hàng đợi.

Đánh dấu nút xuất phát là đã thăm để tránh lặp lại.

Lặp lại quy trình duyệt:

Lấy phần tử đầu tiên từ hàng đợi ra (theo nguyên tắc FIFO).

Kiểm tra xem nút hiện tại có phải đích cần tìm hay không:

Nếu có, thuật toán kết thúc và trả về kết quả.

Nếu chưa, thêm tất cả các nút kề (trạng thái có thể mở rộng) chưa được duyệt vào hàng đợi và đánh dấu chúng đã thăm.

Kết thúc:

Quá trình tiếp tục cho đến khi hàng đợi trống hoặc tìm được trạng thái đích.

Mô hình duyệt BFS:
Tầng 0: Xuất phát từ nút ban đầu.

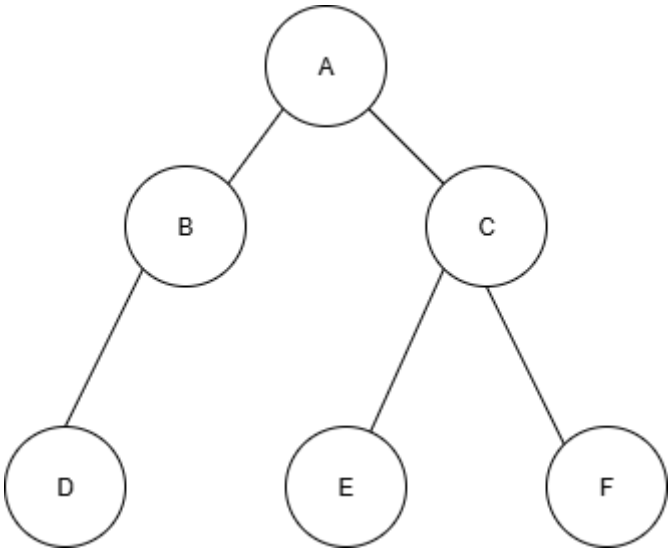
Tầng 1: Duyệt tất cả các nút kề nút ban đầu.

Tầng 2: Duyệt tất cả các nút kề của các nút tầng 1.

… và tiếp tục đến khi gặp nút đích.

Điều này giúp BFS hoạt động giống như việc mở rộng các vòng tròn đồng tâm từ điểm gốc ra bên ngoài cho đến khi chạm đến mục tiêu.

Ví dụ minh họa đơn giản:
Giả sử ta có đồ thị đơn giản như sau:



BFS từ A sẽ duyệt theo thứ tự: **A → B → C → D → E → F**
Nếu mục tiêu là E, BFS sẽ tìm được đường đi ngắn nhất là: **A → C → E**
BFS duyệt **theo chiều rộng**, ưu tiên **theo mức tầng (layer)** chứ không ưu tiên bên trái hay bên phải.
Trong cùng 1 tầng, **thứ tự duyệt phụ thuộc vào thứ tự thêm vào hàng đợi**.
Mô tả sơ đồ
Khi đứng ở nút A:

A có hai nút con là **B và C**.
B được thêm vào hàng đợi trước, sau đó mới đến **C**.
Vì BFS hoạt động kiểu **FIFO**, cái nào vào trước sẽ được lấy ra trước.
Vì vậy duyệt sẽ đi: **A → B → C**.

Sau khi B được duyệt, hàng đợi chứa: **[C, D]** → nên lấy tiếp **C**, rồi **D**.

(Stuart Russell, Peter Norvig, Artificial Intelligence: A Modern Approach, 4th Edition, Pearson Education, 2020.)

1.3 Đặc điểm và ưu điểm của BFS trong AI

1.3.1 Đặc điểm của thuật toán BFS

Thuật toán tìm kiếm theo chiều rộng (Breadth-First Search – BFS) là một kỹ thuật tìm kiếm cơ bản thường được ứng dụng rộng rãi trong các hệ thống trí tuệ nhân tạo, đặc biệt là trong các bài toán tìm đường và xử lý không gian trạng thái. Một số đặc điểm nổi bật của BFS bao gồm:

Tìm kiếm theo mức tầng (level-wise): BFS mở rộng các nút theo chiều rộng trước chiều sâu, nghĩa là nó sẽ duyệt hết tất cả các trạng thái ở một tầng trước khi chuyển sang tầng tiếp theo.

Đảm bảo tìm đường ngắn nhất: Trong đồ thị hoặc không gian trạng thái không trọng số, BFS luôn đảm bảo tìm ra đường đi ngắn nhất từ điểm xuất phát tới đích.

Không dựa trên thông tin đánh giá (uninformed search): BFS là thuật toán tìm kiếm mù (không có thông tin hướng dẫn), không sử dụng bất kỳ hàm đánh giá nào để quyết định lựa chọn trạng thái tiếp theo.

Cấu trúc dữ liệu đặc trưng: BFS sử dụng hàng đợi FIFO (First In First Out) để lưu trữ các trạng thái, đảm bảo các trạng thái được mở rộng theo thứ tự xuất hiện.

1.3.2 Ưu điểm của BFS trong các bài toán AI

1. Luôn tìm được lời giải tối ưu (nếu có):
Trong bài toán tìm kiếm trên đồ thị không trọng số hoặc không gian tìm kiếm đều, BFS luôn đảm bảo tìm ra đường đi ngắn nhất hoặc lời giải với ít bước nhất.

2. Đơn giản, dễ cài đặt:
BFS có thuật toán trực quan, cấu trúc dữ liệu dễ sử dụng (hàng đợi), rất phù hợp để sinh viên thực hành và cài đặt trên nhiều ngôn ngữ lập trình khác nhau.

3. Không yêu cầu thông tin phức tạp về bài toán:
BFS hoạt động hiệu quả ngay cả khi không có thông tin đánh giá cụ thể hoặc không biết trước khoảng cách giữa các trạng thái.

4. Có thể áp dụng cho nhiều bài toán AI thực tế:
Thuật toán BFS được ứng dụng rộng rãi trong các lĩnh vực AI như:

Điều hướng robot (robot tìm đường trong mê cung, bản đồ lưới)

Giải các bài toán tổ hợp (giải sudoku, puzzle)

Tìm kiếm trong đồ thị xã hội (tìm mức độ kết nối giữa các cá nhân)

Kiểm tra khả năng tồn tại của đường đi giữa hai trạng thái.

5. Dễ mở rộng và kết hợp với các chiến lược tìm kiếm khác:
BFS có thể được mở rộng để kết hợp với các thuật toán tìm kiếm nâng cao như IDDFS (Iterative Deepening DFS) hoặc được sử dụng như một thành phần trong các hệ thống tìm kiếm phức tạp hơn.

(ChatGPT, mô hình ngôn ngữ trí tuệ nhân tạo, OpenAI, phát triển từ năm 2018 và được cập nhật đến phiên bản GPT-4 (ra mắt năm 2023), hỗ trợ giải thích và tổng hợp kiến thức trí tuệ nhân tạo phục vụ học tập và nghiên cứu.)

1.4 So sánh BFS với các thuật toán tìm kiếm khác

Trong lĩnh vực trí tuệ nhân tạo, ngoài **thuật toán tìm kiếm theo chiều rộng (Breadth-First Search – BFS)**, còn tồn tại nhiều thuật toán tìm kiếm khác như **Depth-First Search (DFS), A*, Greedy Search, Hill Climbing**,… Mỗi thuật toán đều có những ưu, nhược điểm và tính chất phù hợp với từng bài toán khác nhau. Dưới đây là bảng so sánh tổng quát giữa BFS và các thuật toán phổ biến khác:

Tiêu chí	**BFS**	**DFS**	**A***
Chiến lược duyệt	Theo chiều rộng, duyệt từng tầng	Theo chiều sâu, đi sâu vào nhánh	Dựa trên hàm đánh giá (f = g + h)
Độ phức tạp thời gian	O(V + E)	O(V + E)	O(b^d), phụ thuộc heuristic
Độ phức tạp bộ nhớ	Cao – lưu cả tầng	Thấp – theo chiều sâu	Tương đối cao
Tìm đường ngắn nhất	**Có** (trong đồ thị không trọng số)	Không đảm bảo	**Có** (tốt nhất nếu heuristic chuẩn)
Khả năng bị lặp vô hạn	Không – nhờ visited set	Có nếu không kiểm tra lặp	Không
Ứng dụng phổ biến	Tìm đường ngắn nhất, AI cơ bản, robot, game	Tìm kiếm toàn bộ trạng thái, phân tích đồ thị sâu	Điều hướng thông minh, game, định tuyến mạng

Nhận xét:
Nếu mục tiêu của bài toán là tìm lời giải tối ưu đơn giản, không cần đánh giá thông minh → BFS là lựa chọn lý tưởng.

Nếu bài toán đòi hỏi hiệu quả cao trên không gian lớn, hoặc tìm kiếm nhanh chóng với chi phí bộ nhớ thấp → cần xem xét các thuật toán như A*

Đối với các bài toán cần duyệt hết không gian trạng thái hoặc phân tích cây sâu, DFS thường được ưu tiên.

1.5 Ứng dụng thực tiễn của thuật toán BFS

Thuật toán tìm kiếm theo chiều rộng (**Breadth-First Search - BFS**) không chỉ được sử dụng trong các lý thuyết học thuật mà còn đóng vai trò quan trọng trong rất nhiều ứng dụng thực tế thuộc các lĩnh vực khác nhau. Nhờ khả năng duyệt rộng và tìm đường ngắn nhất, BFS được ứng dụng hiệu quả trong nhiều hệ thống trí tuệ nhân tạo (AI), lập trình phần mềm, công nghệ robot và mạng máy tính.

1.5.1 Ứng dụng trong điều hướng và tìm đường

- Robot tìm đường: BFS được áp dụng để giúp robot tìm đường ngắn nhất trong không gian có nhiều chướng ngại vật như trong bài toán **"Robot tìm mục tiêu"** hoặc robot hút bụi tránh vật cản.

- Bản đồ định vị: Trong các ứng dụng điều hướng như Google Maps, BFS được sử dụng trong các bước cơ bản của thuật toán tìm đường khi cần tìm tuyến đường ngắn nhất trên đồ thị không trọng số.

- Trò chơi giải mê cung: BFS được sử dụng để giải các bài toán mê cung, tìm lối ra nhanh nhất trong game hoặc trong các ứng dụng giáo dục trực quan.

1.5.2 Ứng dụng trong phân tích dữ liệu và mạng xã hội

- Tìm bạn chung trên mạng xã hội: BFS giúp xác định mối quan hệ giữa hai người dùng, tìm bạn chung, hoặc xác định khoảng cách (degree of connection) trong các mạng xã hội như Facebook, LinkedIn.

- Phân tích đồ thị mạng: Dùng để tìm tất cả các node có thể truy cập được từ một node nguồn, phục vụ cho việc phân tích dữ liệu mạng.

1.5.3 Ứng dụng trong kiểm tra hệ thống và an ninh mạng

- Kiểm tra kết nối mạng: BFS được dùng để xác định xem có đường truyền nào giữa hai thiết bị mạng không.

- Phát hiện lỗ hổng an ninh: Một số hệ thống kiểm tra tự động sử dụng BFS để rà quét toàn bộ các trạng thái truy cập hợp lệ nhằm phát hiện điểm yếu bảo mật.

1.5.4 Ứng dụng trong AI và trò chơi

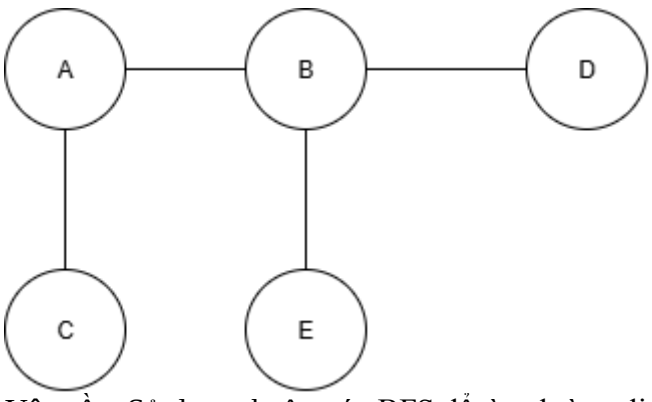
- Trí tuệ nhân tạo trong game: BFS được sử dụng trong AI để giúp nhân vật game tìm đường đến mục tiêu, đặc biệt là trong các trò chơi chiến thuật, giải đố hoặc các trò chơi chiến đấu.
- Giải quyết bài toán tổ hợp: BFS hỗ trợ giải các bài toán như **sudoku**, **n-queens**, **puzzle** bằng cách duyệt qua tất cả trạng thái hợp lệ.

1.5.5 Ứng dụng trong lập trình và hệ thống tập tin

- Quét cây thư mục: BFS thường được sử dụng để duyệt cây thư mục hệ thống để kiểm tra hoặc liệt kê toàn bộ các tập tin và thư mục con.
- Xây dựng hệ thống kiểm tra và sao lưu dữ liệu: BFS hỗ trợ kiểm tra dữ liệu trong các hệ thống lớn bằng cách duyệt tất cả các nút theo chiều rộng một cách tối ưu.

1.6 Ví dụ minh họa đơn giản về BFS

1.6.1 Mô tả bài toán



Yêu cầu: Sử dụng thuật toán BFS để tìm đường đi từ đỉnh **A** đến đỉnh **E**.

1.6.2 Cách tiến hành BFS

Bước 1: Khởi tạo hàng đợi với đỉnh bắt đầu là A:
Hàng đợi = [A], Đã thăm = {A}.
Bước 2: Lấy A ra khỏi hàng đợi, kiểm tra các đỉnh kề của A:
Đỉnh kề của A là B, C. Thêm B, C vào hàng đợi.
Hàng đợi = [B, C], Đã thăm = {A, B, C}.
Bước 3: Lấy B ra khỏi hàng đợi, kiểm tra các đỉnh kề của B:
Đỉnh kề là A, D, E.

A đã thăm nên bỏ qua, D và E chưa thăm nên thêm vào hàng đợi.
Hàng đợi = [C, D, E], Đã thăm = {A, B, C, D, E}.
Bước 4: Kiểm tra C, không có đỉnh kề nào mới.
Hàng đợi = [D, E].
Bước 5: Lấy D ra, không có đỉnh kề nào mới.
Hàng đợi = [E].
Bước 6: Lấy E ra khỏi hàng đợi. Đây là đích cần tìm → **dừng thuật toán**.

1.6.3 Kết quả

Đường đi ngắn nhất từ A đến E được tìm bằng BFS là:
A → B → E

Trình tự duyệt qua các đỉnh:
A → B → C → D → E

1.6.4 Mô phỏng đơn giản

Bước 1: [A]
Bước 2: [B, C]
Bước 3: [C, D, E]
Bước 4: [D, E]
Bước 5: [E]
Bước 6: [ ] (Kết thúc)

1.7 KẾT LUẬN CHƯƠNG I

Trong Chương I, đồ án đã trình bày tổng quan về **thuật toán tìm kiếm theo chiều rộng (Breadth-First Search - BFS)** từ cơ sở lý thuyết đến cách thức hoạt động, đặc điểm nổi bật, ưu nhược điểm và ứng dụng thực tiễn. Qua các nội dung trên, có thể khẳng định rằng BFS là một trong những thuật toán đơn giản nhưng vô cùng hiệu quả, đặc biệt trong các bài toán tìm đường ngắn nhất và duyệt trạng thái trong không gian không trọng số.

Việc tìm hiểu về BFS giúp nhóm có nền tảng kiến thức vững chắc để áp dụng thuật toán này vào mô phỏng bài toán robot tìm mục tiêu trong chương tiếp theo. Nhờ ưu điểm đảm bảo tìm được lời giải tối ưu và dễ triển khai, BFS được ứng dụng rộng rãi trong nhiều lĩnh vực trí tuệ nhân tạo, từ lập trình game đến các hệ thống robot thực tế.

1.7.1 TÀI LIỆU THAM KHẢO

1. T. N. Việt, P. H. V. Trường, *Nhập môn Trí tuệ nhân tạo – Chương 1: Bài toán tìm kiếm*, Trường Đại học Văn Lang, 2025.
2. Stuart Russell, Peter Norvig, *Artificial Intelligence: A Modern Approach*, 4th Edition, Pearson Education, 2020.
3. GeeksforGeeks, "Breadth First Search or BFS for a Graph," [Online]. Available: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
4. ChatGPT, *mô hình ngôn ngữ trí tuệ nhân tạo*, OpenAI, phát triển từ năm 2018 và được cập nhật đến phiên bản GPT-4 (ra mắt năm 2023), hỗ trợ giải thích và tổng hợp kiến thức trí tuệ nhân tạo phục vụ học tập và nghiên cứu.

1.7.2 Link tài liệu tham khảm

https://dokumen.pub/artificial-intelligence-a-modern-approach-4nbsped-9780134610993-0134610997.html

https://chatgpt.com/

https://elearning.vlu.edu.vn/course/view.php?id=8218

https://www.geeksforgeeks.org/dsa/breadth-first-search-or-bfs-for-a-graph/

CHƯƠNG II: ỨNG DỤNG THUẬT TOÁN BFS VÀO BÀI TOÁN ROBOT TÌM MỤC TIÊU

2.1 Mô tả bài toán: Robot tìm đường trên bản đồ

- Bài toán đặt ra là thiết kế một mô phỏng đơn giản về **robot tìm đường** đi từ **điểm xuất phát (start)** đến **điểm đích (goal)** trên một bản đồ dạng mê cung hai chiều (2D).
- Bản đồ được biểu diễn dưới dạng lưới ô vuông (grid), trong đó:

Các ô có thể là:

Ô trống (0): Robot có thể di chuyển qua.

Ô tường (1): Vật cản, robot không thể di chuyển qua.

Ô đá (Rock): Vật thể cố định, cũng là vật cản như tường.

Vị trí bắt đầu (Start): Nơi robot bắt đầu di chuyển.

Vị trí mục tiêu (Goal): Nơi robot cần tìm tới.

- Robot có khả năng di chuyển theo bốn hướng cơ bản: Lên (Up), Xuống (Down), Trái (Left), Phải (Right).

Mục tiêu bài toán:

Tìm đường đi ngắn nhất từ điểm bắt đầu tới điểm kết thúc.

Tránh va chạm với tường và đá.

Đảm bảo robot tìm được đường hợp lệ (nếu có).

Hiển thị trực quan quá trình tìm kiếm và đường đi của robot.

2.1.1 Thuật toán BFS (Breadth-First Search)

Thuật toán **BFS** là giải thuật tìm kiếm theo chiều rộng, có khả năng tìm được đường đi ngắn nhất trong đồ thị không trọng số.

Nguyên lý hoạt động cơ bản của BFS như sau:

BFS(start, goal):

Khởi tạo hàng đợi queue với điểm start

Khởi tạo tập hợp visited để đánh dấu các ô đã duyệt

Khởi tạo parent map để lưu vết đường đi

```
5   def bfs(grid, start, goal, rocks, directions):
6       queue = deque([start])
7       visited = set()
8       parent = {}
9       visited.add(start)
```

while queue không rỗng:

Lấy phần tử đầu tiên trong queue

Nếu phần tử là goal:

Truy vết ngược lại đường đi từ goal về start
Trả về đường đi
Duyệt qua các ô kề (trái, phải, trên, dưới):
Nếu ô hợp lệ và chưa được duyệt:
Thêm vào queue và đánh dấu là đã duyệt
Nếu duyệt hết mà không đến được goal, trả về None

```
    while queue:
        x, y = queue.popleft()
        if (x, y) == goal:
            path = []
            while (x, y) != start:
                path.append((x, y))
                x, y = parent[(x, y)]
            path.append(start)
            return path[::-1]

        for dx, dy in directions:
            nx, ny = x + dx, y + dy
            if (0 <= nx < ROWS and 0 <= ny < COLS and
                grid[nx][ny] == 0 and
                (nx, ny) not in rocks and
                (nx, ny) not in visited):
                queue.append((nx, ny))
                visited.add((nx, ny))
                parent[(nx, ny)] = (x, y)
    return None
```

2.2 Mô hình hóa bản đồ dưới dạng lưới (Grid)

Bản đồ trong bài toán được mô hình hóa dưới dạng lưới ô vuông (grid) với kích thước xác định bởi số hàng (ROWS) và số cột (COLS). Mỗi ô trong lưới sẽ biểu diễn một vị trí cụ thể mà robot có thể di chuyển hoặc không thể di chuyển.

Trong dự án này, bản đồ được cài đặt sẵn thông qua một hàm khởi tạo lưới mặc định. Cụ thể:

Ô trống (giá trị = 0): Robot có thể di chuyển tự do.

Ô tường (giá trị = 1): Robot không thể di chuyển qua.

Ô đá (tọa độ lưu trong danh sách ROCKS): Được coi như vật cản bổ sung, nhằm tăng tính thực tế cho môi trường di chuyển.

Các điểm đặc biệt bao gồm:

Điểm xuất phát (START_POS): Được quy định sẵn tại vị trí (0, 0).

Điểm đích (GOAL_POS): Được quy định tại vị trí (14, 19).

```
def create_default_grid():
    return [
        [0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,1,0,0,0],
        [1,0,1,0,1,0,1,1,1,0,1,0,1,1,0,1,0,1,1,0],
        [0,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0],
        [0,1,1,1,1,1,1,0,1,0,1,1,1,0,0,1,1,1,0,0],
        [0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,0],
        [0,1,0,1,1,0,1,1,1,1,1,0,1,1,0,1,1,0,1,0],
        [0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0],
        [1,1,0,1,1,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0],
        [0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0],
        [0,1,1,1,1,1,1,0,1,1,1,1,1,1,0,1,1,1,1,0],
        [0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,1,0,0,0],
        [1,1,0,1,1,1,1,1,1,0,1,1,1,1,0,1,1,1,1,0],
        [0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0],
        [0,1,1,1,1,1,1,1,0,1,1,1,1,1,1,0,1,1,1,1,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    ]
```

Kích thước:

Với 15 hàng (ROWS = 15)

Với 20 cột (COLS = 20)

=> Kích thước lưới là 15 x 20.

robot_pathfinding_game > config.py > ...

```
# Cấu hình giao diện
# config.py
CELL_SIZE = 40
ROWS = 15
COLS = 20
WIDTH = CELL_SIZE * COLS
HEIGHT = CELL_SIZE * ROWS + 40  # +40 cho thanh công cụ
```

2.3 Áp dụng thuật toán BFS để tìm đường ngắn nhất

Thuật toán **BFS (Breadth-First Search)** được lựa chọn để giải quyết bài toán tìm đường đi ngắn nhất từ điểm xuất phát (**Start**) đến điểm mục tiêu (**Goal**) trong mê cung.

Nguyên lý hoạt động:

BFS sử dụng cấu trúc dữ liệu **hàng đợi (queue)** để lần lượt duyệt qua các ô kề gần với điểm xuất phát trước tiên, sau đó lan tỏa ra các ô xa hơn.

Đặc điểm của BFS là luôn đảm bảo tìm được đường đi ngắn nhất trong môi trường không trọng số (mỗi bước di chuyển có độ dài bằng nhau).

Mô tả quy trình thực hiện:

1. Khởi tạo hàng đợi (queue) với vị trí bắt đầu.
2. Khởi tạo tập hợp các ô đã duyệt (visited set) để tránh lặp lại.
3. Khởi tạo bảng cha (parent map) để lưu vết đường đi.
4. Lặp qua từng ô trong hàng đợi:

   Nếu ô hiện tại là điểm đích (**Goal**), truy ngược lại đường đi thông qua bảng cha.

   Nếu chưa tới đích, tiếp tục duyệt 4 hướng (lên, xuống, trái, phải).

   Bỏ qua các ô là vật cản (tường, đá) hoặc nằm ngoài bản đồ.
5. Nếu hàng đợi trống mà không đến được đích, trả về kết quả không có đường đi.

2.4. Thiết kế và xây dựng hệ thống mô phỏng

Để trực quan hóa quá trình tìm đường của robot, hệ thống được xây dựng dưới dạng mô phỏng bằng **ngôn ngữ lập trình Python**, sử dụng **thư viện Pygame** để xây dựng giao diện đồ họa.

2.4.1 Kiến trúc hệ thống

Dự án được xây dựng theo mô hình phân lớp, chia thành các thành phần chính nhằm dễ bảo trì, mở rộng và phát triển lâu dài. Cấu trúc thư mục của dự án như sau:

```
robot_pathfinding_game/
├── algorithm/          # Chứa thuật toán tìm kiếm BFS
│   └── bfs.py
├── assets/             # Thư mục tài nguyên hình ảnh
│   ├── robot.png
│   └── stone-nature-2d-colored.png
├── docs/               # Tài liệu mô tả dự án
│   ├── README.md
│   └── requirements.txt
├── assets_manager.py   # Quản lý tải hình ảnh vào chương trình
├── config.py           # Các thông số cấu hình (ROWS, COLS, màu sắc,…)
├── gui.py              # Giao diện người dùng bằng Pygame
├── main.py             # Chạy chính hệ thống mô phỏng
└── maze.py             # Tạo mê cung mặc định, các thông tin khởi tạo
```

2.4.2 Chức năng các thành phần chính

algorithm/bfs.py: Cài đặt thuật toán BFS để tìm đường đi ngắn nhất từ điểm xuất phát đến điểm đích, đồng thời tránh vật cản.
assets/: Chứa các tài nguyên đồ họa như ảnh robot, ảnh đá phục vụ hiển thị trực quan trong mô phỏng.
docs/: Lưu trữ các tài liệu mô tả dự án, hướng dẫn cài đặt và vận hành hệ thống.
assets_manager.py: Quản lý việc tải và xử lý hình ảnh từ thư mục assets.
config.py: Khai báo các hằng số cấu hình của hệ thống như kích thước bản đồ (ROWS, COLS), kích thước ô, màu sắc hiển thị.
gui.py: Xây dựng giao diện đồ họa người dùng sử dụng thư viện Pygame, hiển thị bản đồ mê cung, vị trí robot và các công cụ thao tác.
maze.py: Định nghĩa bản đồ mê cung cơ bản và các thông tin về Start, Goal, các vật cản như tường và đá.
main.py: Điểm khởi động chính của ứng dụng, kết nối toàn bộ các thành phần và chạy mô phỏng.

2.4.3 Công nghệ sử dụng

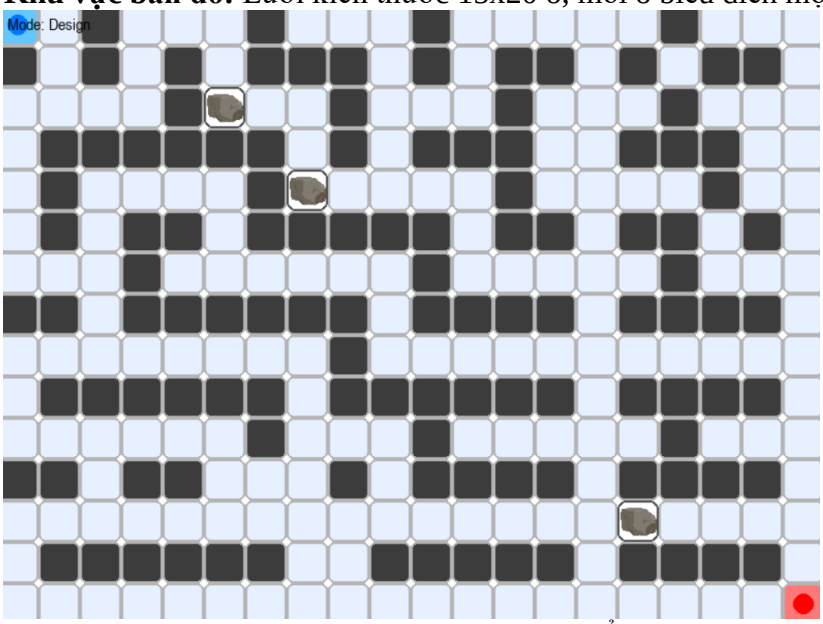
Ngôn ngữ lập trình: Python 3.10
Thư viện đồ họa: Pygame
Cấu trúc dữ liệu chính: Hàng đợi (queue), danh sách (list), tập hợp (set), từ điển (dictionary).

2.4.4 Ưu điểm thiết kế

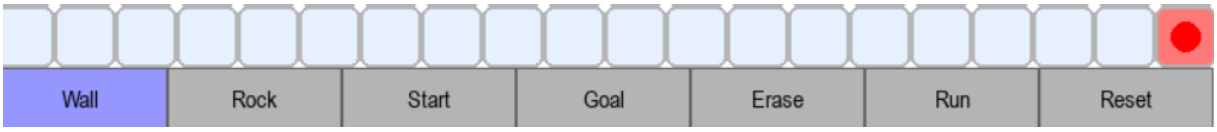
Kiến trúc **module hóa**, dễ dàng bảo trì và mở rộng.
Quản lý tài nguyên và thuật toán **tách biệt**, giúp hệ thống linh hoạt khi phát triển thêm các thuật toán khác như DFS hoặc A*.
Giao diện trực quan, đơn giản, giúp người dùng dễ dàng tương tác và kiểm tra kết quả mô phỏng.

2.4.5 Giao diện mô phỏng

Giao diện mô phỏng bao gồm:
Khu vực bản đồ: Lưới kích thước 15x20 ô, mỗi ô biểu diễn một vị trí trên mê cung.



Khu vực thanh công cụ: Các nút chức năng để chỉnh sửa bản đồ và điều khiển mô phỏng như là



Biểu diễn trực quan:

- Ô màu xanh là **Start**, màu đỏ là **Goal**.
- Ô màu xám hoặc đen là vật cản (**tường, đá**).
- Đường đi tìm được sẽ được hiển thị bằng màu vàng.
- Robot được biểu diễn bằng hình ảnh di chuyển qua các ô từ Start đến Goal.

Goal : 

Start: 

Wall: 

Rock: 

Robot: 

The search path will be displayed in yellow:

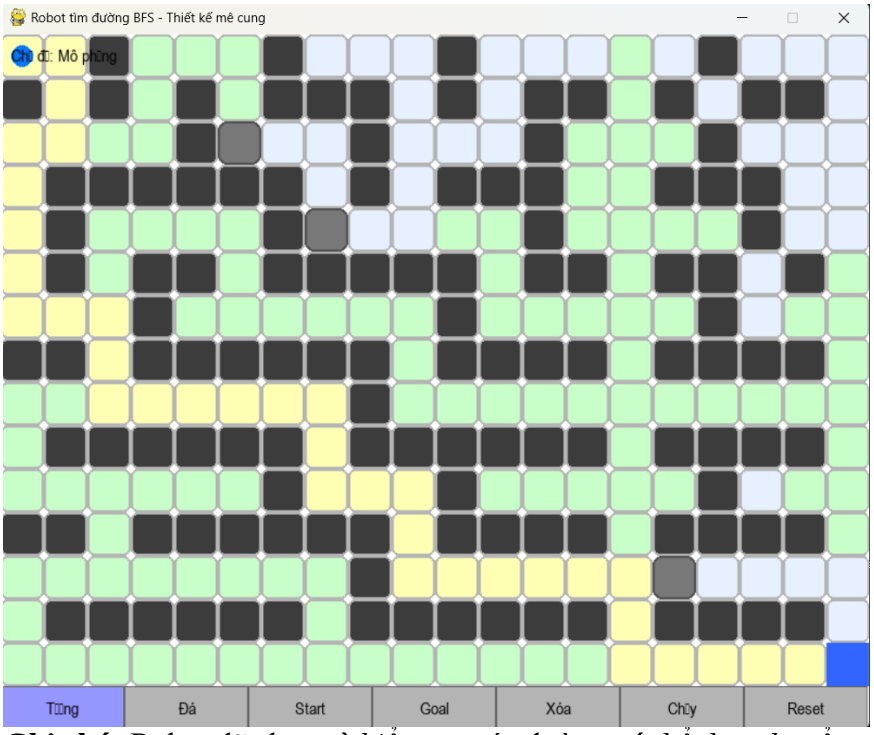
2.5 Mô phỏng hoạt động của robot qua từng bước tìm kiếm

Hệ thống được xây dựng giúp mô phỏng trực quan quá trình robot thực hiện tìm kiếm từ vị trí xuất phát (Start) đến vị trí mục tiêu (Goal) bằng thuật toán BFS trên bản đồ dạng mê cung lưới ô vuông.

2.5.1 Cơ chế hoạt động mô phỏng

Khi người dùng nhấn nút "Chạy", thuật toán BFS được kích hoạt để tìm đường đi hợp lệ ngắn nhất.

Hệ thống sẽ kiểm tra tất cả các ô có thể di chuyển, đảm bảo tránh va chạm với các vật cản (tường, đá).



Ghi chú: Robot đã chạy và kiểm tra các đường có thể duy chuyển và tránh các vật thể để đến được đích (biểu diễn bằng **hiệu ứng đường đi màu vàng và xanh lá**) và đường đi ngắn nhất được biểu diễn với **hiệu ứng đường đi màu vàng.**

Nếu tìm được đường đi, robot sẽ thực hiện di chuyển từng bước từ Start đến Goal theo đúng đường đã tìm kiếm được.

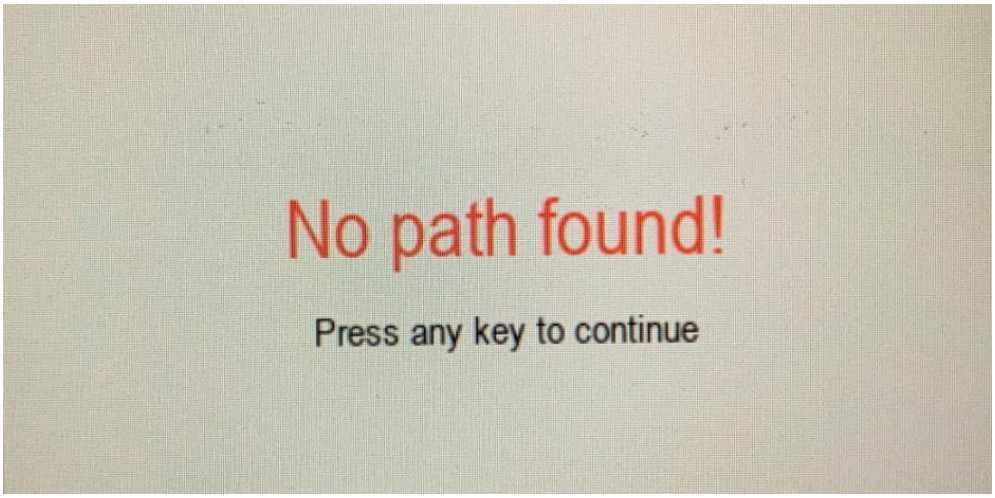
2.5.2 Quá trình mô phỏng trực quan

Ô Start: hiển thị bằng màu xanh hoặc hình robot xuất phát.
Ô Goal: hiển thị bằng màu đỏ, đích đến của robot.
Vật cản: ô tường và đá được hiển thị với màu xám hoặc đen.
Đường đi: hiển thị bằng màu vàng, biểu diễn lộ trình tối ưu mà robot lựa chọn.
Robot: hình ảnh robot sẽ di chuyển từng bước trên lưới, giúp người dùng quan sát quá trình tìm kiếm thực tế.

2.5.3 Xử lý trường hợp không có đường đi

Nếu BFS không tìm ra đường đi hợp lệ, giao diện hiển thị thông báo **“No path found!”**, đồng thời làm mờ bản đồ và cho phép người dùng quay về chế độ thiết kế để

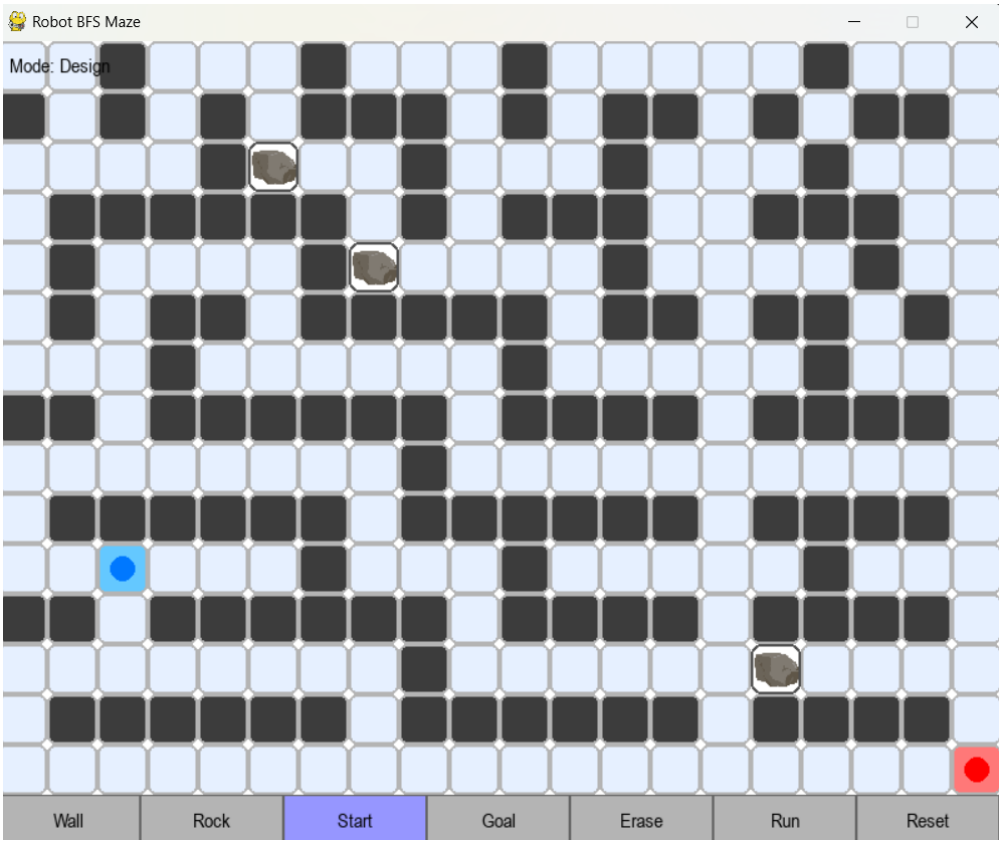
chỉnh sửa.



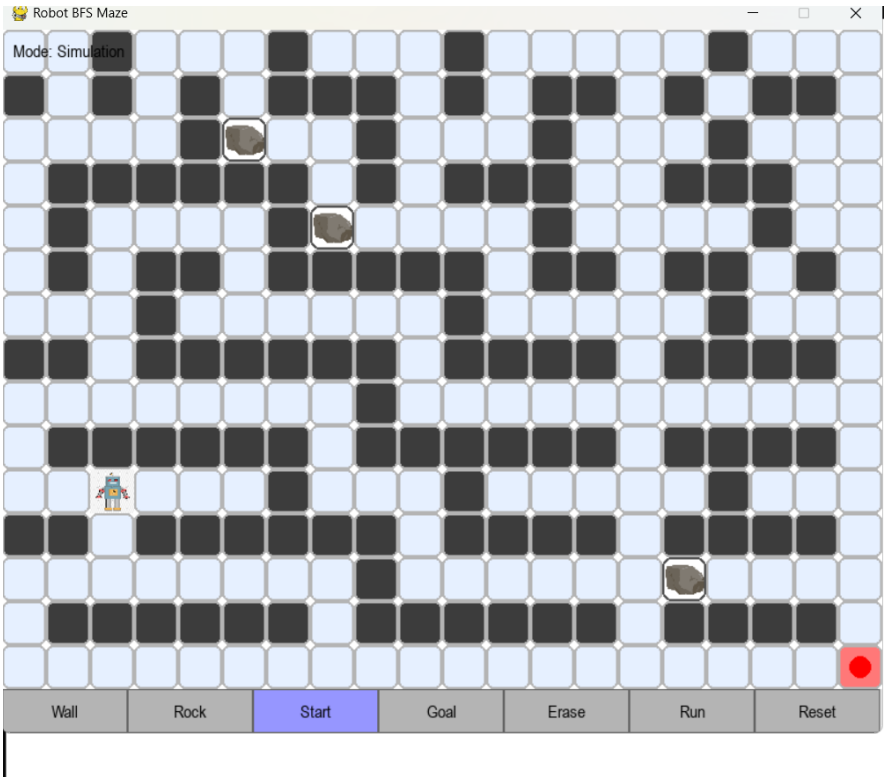
2.5.4 Kết quả mô phỏng thực tế

Chọn Điểm bắt đầu

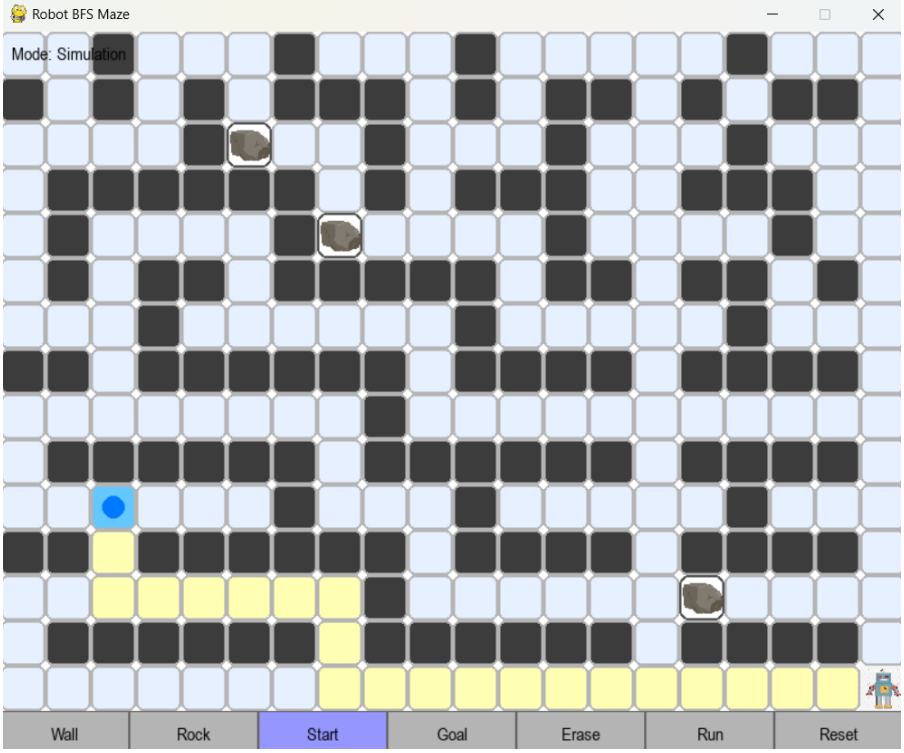
Có thể chọn bất cứ ở đâu trên bảng đồ những không thể chọn chồng chéo lên tường hoặc là lên cục đá



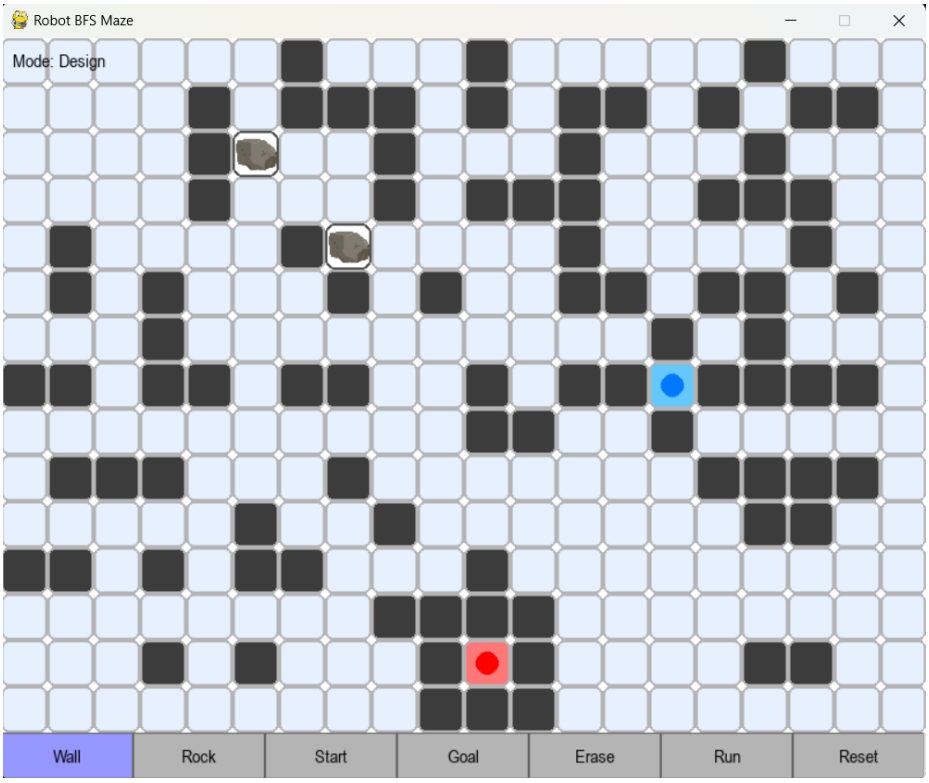
Sau đó chọn Run để hiển thị robot tại nơi bạn chọn để bắt đầu đường đi
Sau đó bấm cách để robot chạy



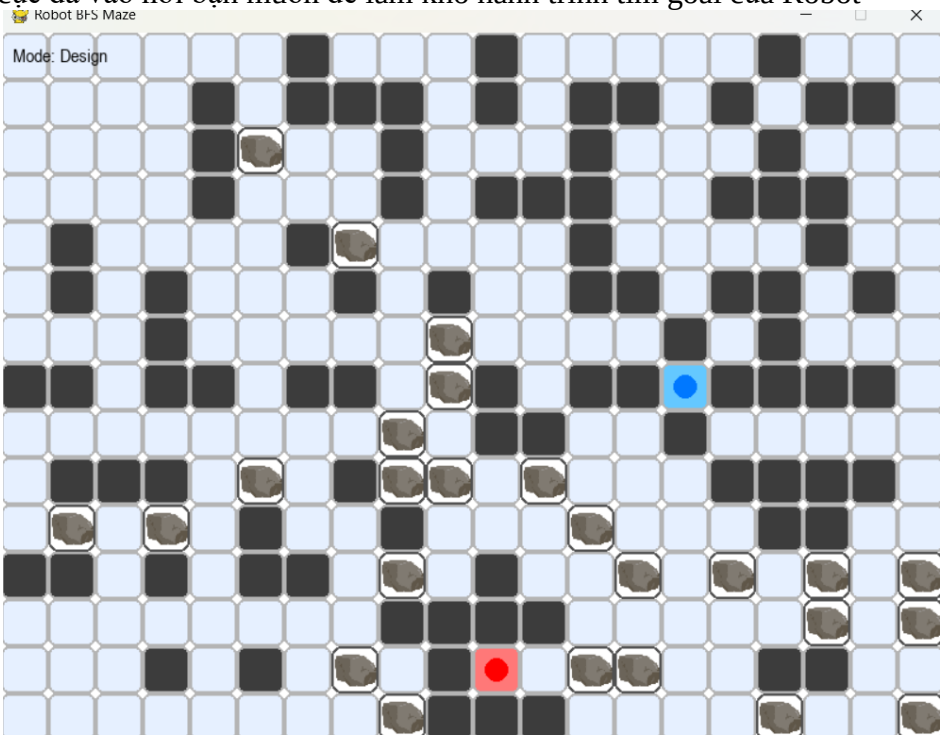
Kết quả và hiệu ứng đường đi sẽ biểu diễn qua những ô màu vàng



Tính năng bổ sung: để thêm phần thú vị bạn có thể điều chỉnh map bằng cách thêm hoặc xoá những bức tường màu đen bằng cách click Wall trên thanh công cụ và điều chỉnh map

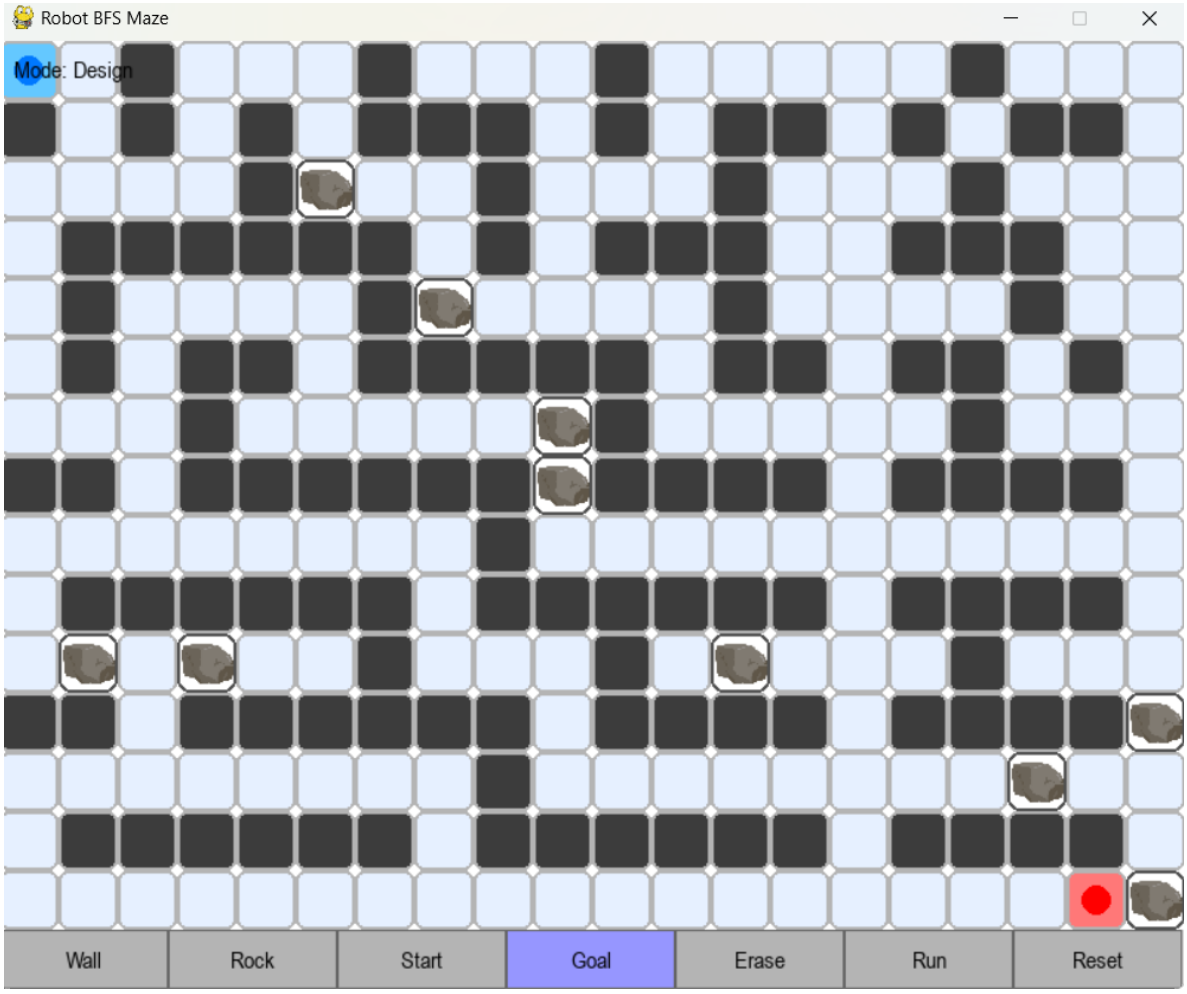


Tương tự với bức tường màu đen bạn có thể click vào Rock trên thanh công cụ để đặt những cục đá vào nơi bạn muốn để làm khó hành trình tìm goal của Robot

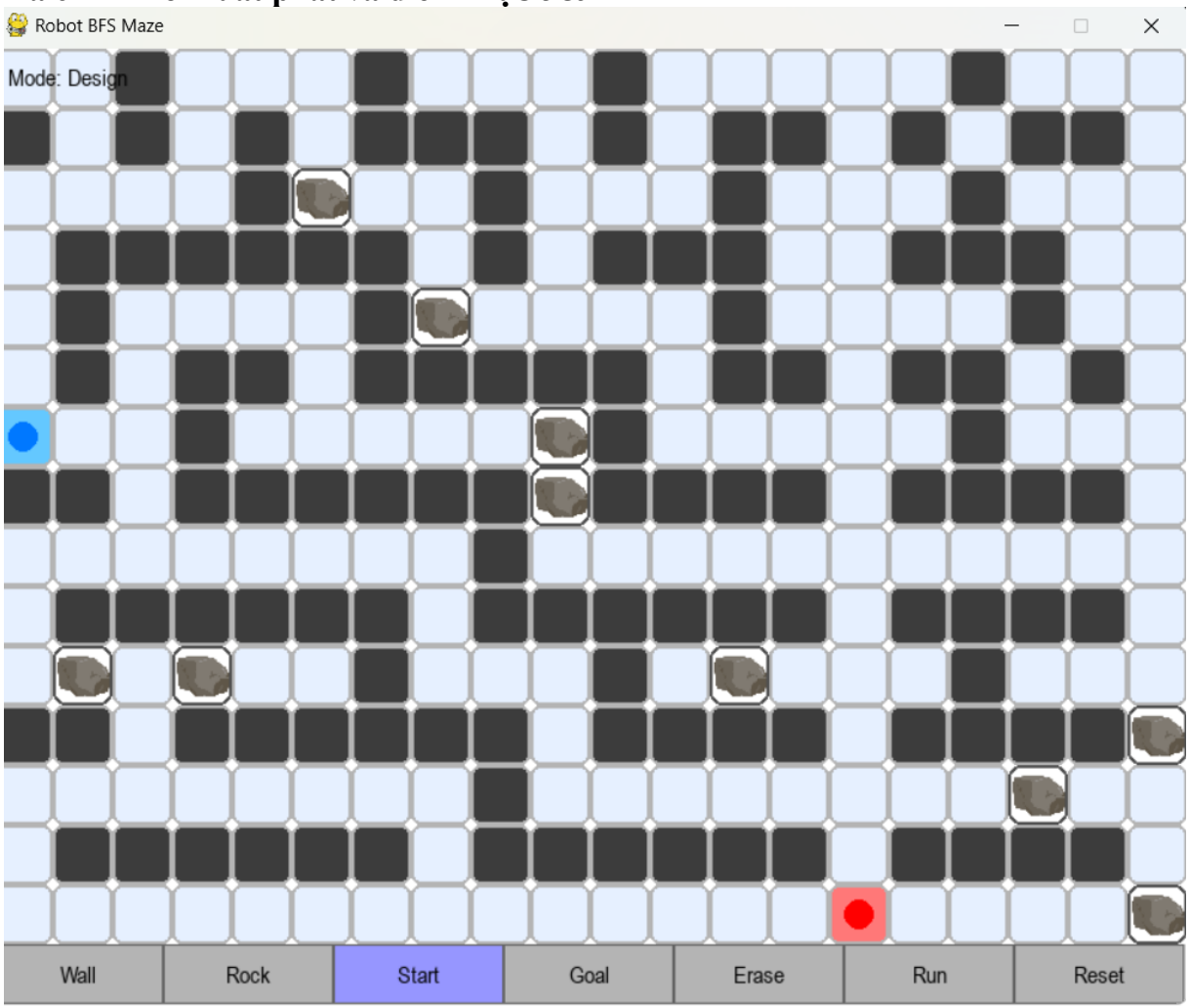


2.6 Các tình huống thử nghiệm và kết quả nhận được

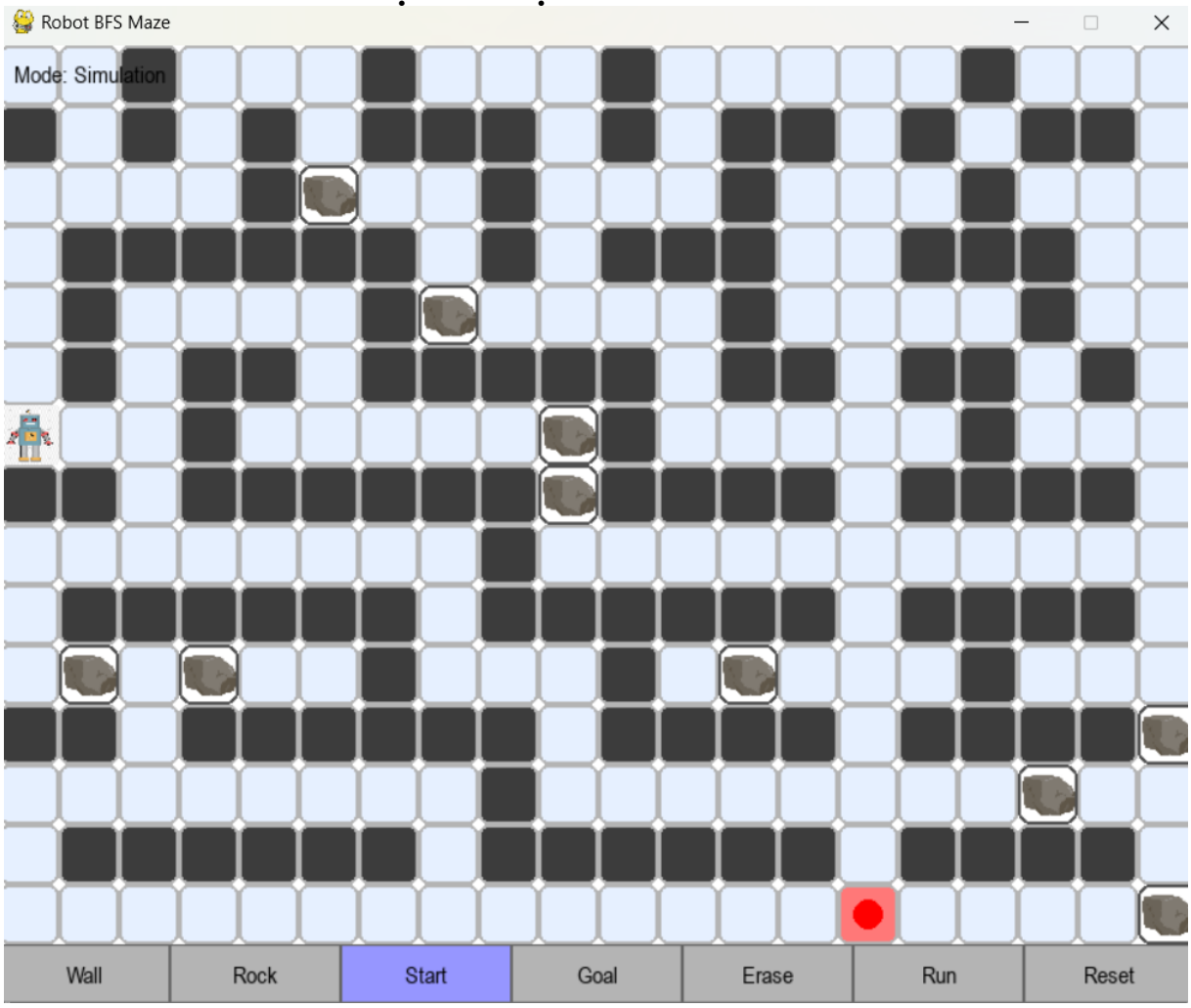
Bản đồ mặc định



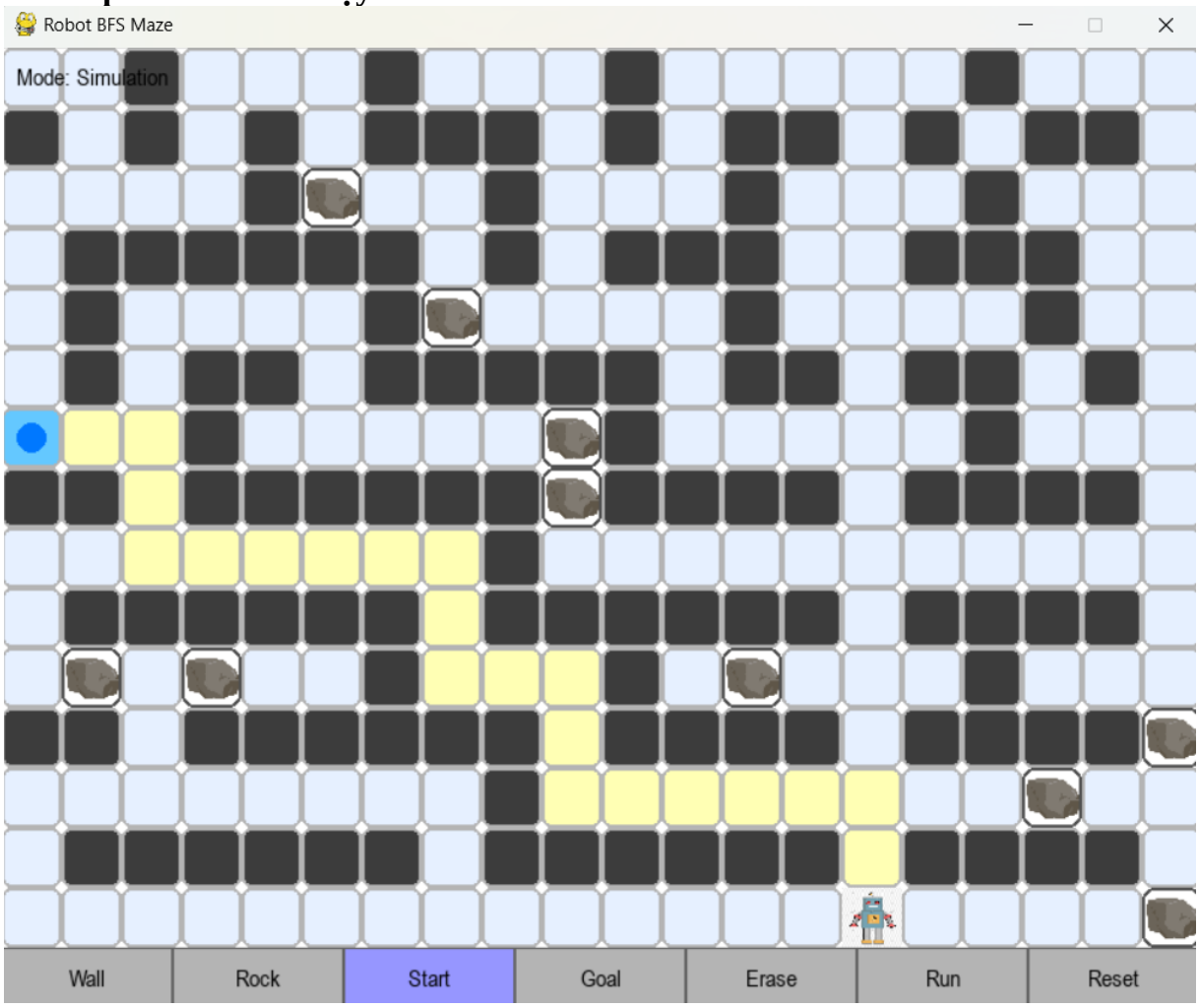
Đã chỉnh nơi xuất phát và điểm mục tiêu



Sau khi bấm run Robot được hiển thị



Nhấm phím cách để chạy:



2.7 Đánh giá hiệu quả của thuật toán BFS trong mô phỏng

Sau khi thực hiện mô phỏng quá trình tìm kiếm đường đi của robot bằng thuật toán BFS (Breadth-First Search), có thể đưa ra một số đánh giá về hiệu quả của thuật toán này như sau:

2.7.1 Ưu điểm của thuật toán BFS

Tìm đường đi ngắn nhất: Với bài toán mê cung dạng lưới, BFS luôn đảm bảo tìm được đường đi ngắn nhất từ điểm xuất phát đến đích (nếu tồn tại).

Dễ cài đặt, dễ hiểu: Thuật toán đơn giản, sử dụng cấu trúc dữ liệu hàng đợi (queue), dễ dàng triển khai bằng các ngôn ngữ lập trình phổ biến như Python.

Khả năng mở rộng: Dễ dàng mở rộng để xử lý các yếu tố bổ sung như vật cản đặc biệt (đá), giới hạn phạm vi di chuyển hoặc các điều kiện tùy chỉnh.

Hiệu quả trong mô phỏng trực quan: Kết hợp với Pygame, BFS giúp quá trình mô phỏng trực quan, người dùng dễ dàng quan sát từng bước di chuyển của robot trên bản đồ.

2.7.2 Nhược điểm của thuật toán BFS

Tốc độ chậm trên bản đồ lớn: Vì BFS kiểm tra đồng loạt theo từng lớp, khi mê cung có kích thước lớn hoặc số lượng vật cản dày đặc, tốc độ tìm kiếm sẽ giảm.

Tốn bộ nhớ: BFS cần lưu trữ thông tin visited và parent cho tất cả các nút đã duyệt, dẫn đến tiêu tốn bộ nhớ RAM khi bản đồ có số lượng ô rất lớn.

2.7.3 Đánh giá tổng quan trong phạm vi đề tài

Tiêu chí	BFS Đáp ứng tốt	Ghi chú
Tìm đường ngắn nhất	**Có**	Luôn tìm ra đường ngắn nhất có thể
Hiển thị mô phỏng trực quan	**Có**	Dễ kết hợp với giao diện người dùng
Hiệu quả trên bản đồ vừa và nhỏ	**Có**	Chạy ổn định với bản đồ 15x20 ô
Khả năng mở rộng bài toán	**Có**	Có thể thêm nhiều tình huống dễ dàng
Khả năng mở rộng cho bản đồ lớn	**Có thể, nhưng bị giảm hiệu suất**	Hiệu suất giảm khi bản đồ quá lớn

2.8 Hạn chế và hướng phát triển

2.8.1 Hạn chế của hệ thống

Quy mô mô phỏng nhỏ: Hệ thống hiện tại chỉ mô phỏng bản đồ dạng lưới với kích thước cố định 15x20 ô, chưa hỗ trợ tùy chỉnh kích thước hoặc bản đồ động.
Chỉ hỗ trợ thuật toán BFS: Dự án chỉ tập trung vào thuật toán tìm đường BFS, chưa triển khai thêm các thuật toán nâng cao như A*, DFS, Dijkstra.
Khả năng mô phỏng còn đơn giản: Robot chỉ di chuyển theo lộ trình đơn giản, chưa mô phỏng thêm các hành vi thực tế như tránh vật cản động hoặc cập nhật bản đồ theo thời gian thực.
Hiệu suất chưa tối ưu: Khi mở rộng bản đồ lớn hơn, BFS sẽ tiêu tốn nhiều thời gian và bộ nhớ, gây ra hiện tượng giật, lag giao diện nếu không được tối ưu.
Chưa hỗ trợ giao diện web hoặc ứng dụng di động: Hiện tại hệ thống chỉ chạy trên máy tính thông qua giao diện Pygame, hạn chế khả năng tiếp cận đa nền tảng.

2.8.2 Hướng phát triển tương lai

Để hệ thống mô phỏng hoàn thiện và mở rộng ứng dụng thực tiễn hơn, một số hướng phát triển có thể thực hiện như sau:

Tích hợp thêm thuật toán khác như A*, Dijkstra giúp so sánh trực tiếp giữa các thuật toán tìm đường và lựa chọn phương án tối ưu.
Hỗ trợ tùy chỉnh bản đồ linh hoạt: Cho phép người dùng tự vẽ mê cung theo kích thước mong muốn hoặc tải bản đồ từ file dữ liệu.
Phát triển phiên bản chạy trên trình duyệt sử dụng WebAssembly hoặc thư viện web như React + Pygame.js giúp tiếp cận dễ dàng hơn trên đa nền tảng (máy tính, điện thoại).
Tối ưu hiệu suất thuật toán thông qua các kỹ thuật như giảm không gian tìm kiếm, lưu trữ bản đồ dưới dạng ma trận sparse.
Thêm hiệu ứng giao diện và âm thanh: Cải thiện trải nghiệm người dùng bằng các hiệu ứng âm thanh mô phỏng bước chân, âm thanh báo hoàn thành nhiệm vụ.
Ứng dụng thực tế: Mở rộng mô hình cho các bài toán robot thực tế như robot dọn nhà, xe tự hành tìm điểm đỗ hoặc robot kho hàng.

2.8.3 Định hướng mở rộng học thuật

Khuyến khích sinh viên mở rộng nghiên cứu so sánh hiệu năng giữa các thuật toán tìm kiếm trên nhiều dạng bản đồ khác nhau.
Phát triển thêm mô hình mô phỏng robot di chuyển trong môi trường 3D hoặc ứng dụng trong lĩnh vực game, thực tế ảo (VR).

2.9 KẾT LUẬN CHƯƠNG II

Trong chương II, đồ án đã trình bày chi tiết về cách ứng dụng thuật toán tìm kiếm theo chiều rộng (BFS) vào bài toán mô phỏng robot tìm mục tiêu trên bản đồ dạng mê cung. Quá trình xây dựng hệ thống mô phỏng được thực hiện qua các bước từ mô hình hóa bài toán, thiết kế bản đồ dạng lưới, áp dụng thuật toán BFS để tìm đường đi ngắn nhất, cho đến xây dựng giao diện đồ họa trực quan giúp quan sát trực tiếp quá trình tìm kiếm.

Kết quả mô phỏng cho thấy, thuật toán BFS có khả năng tìm kiếm đường đi ngắn nhất hiệu quả trong môi trường có quy mô vừa phải, giao diện đơn giản, dễ sử dụng, phù hợp với các bài toán tìm đường cơ bản. Tuy nhiên, mô hình vẫn còn một số hạn chế nhất định về quy mô mô phỏng và tính năng mở rộng.

Chương II đóng vai trò quan trọng trong việc hiện thực hóa lý thuyết thuật toán tìm kiếm thành một hệ thống ứng dụng trực quan, từ đó giúp nâng cao khả năng thực hành lập trình và tư duy thuật toán. Đây cũng là nền tảng để phát triển thêm các mô hình mô phỏng phức tạp hơn hoặc ứng dụng thực tế trong các bài toán về robot và trí tuệ nhân tạo.

CHƯƠNG III: KẾT QUẢ NGHIÊN CỨU

3.1 Tổng hợp kết quả thử nghiệm

Sau quá trình xây dựng hệ thống mô phỏng, nhóm đã tiến hành thử nghiệm hoạt động của robot tìm đường trên nhiều bản đồ mê cung khác nhau với các đặc điểm:

Bản đồ có ít vật cản.
Bản đồ có nhiều vật cản.
Khoảng cách từ Start đến Goal thay đổi linh hoạt.
Thử nghiệm trường hợp không tìm được đường đi.

Kết quả tổng hợp:

STT	Kịch bản thử nghiệm	Độ dài đường đi	Ghi Chú
1	Mê cung mặc định	35 Bước	**Tìm được**
2	Goal bị bao quanh vật cản	**Không có đường đi**	Không tìm được

Kết quả thử nghiệm cho thấy hệ thống hoạt động ổn định, đảm bảo tìm được đường đi hợp lệ khi có lối thoát và báo lỗi chính xác khi không tìm được đường đi.
Kết quả có thể thay đổi khi bạn thêm điều chỉnh bản đồ, thêm cục đá hoặc thay đổi nơi bắt đầu, nơi kết thúc

3.2 Nhận xét và phân tích

3.2.1 Ưu điểm nổi bật của hệ thống

Độ chính xác cao: Thuật toán BFS luôn đảm bảo tìm được đường đi ngắn nhất khi tồn tại, giúp kết quả mô phỏng chính xác, nhất quán và đáng tin cậy.
Giao diện trực quan: Sử dụng thư viện Pygame giúp việc hiển thị bản đồ, các bước tìm kiếm và quá trình di chuyển của robot trở nên sinh động, dễ theo dõi với người học.
Tốc độ phản hồi nhanh: Với bản đồ quy mô vừa phải (15x20 ô), thuật toán BFS xử lý nhanh chóng chỉ trong khoảng 0.02 đến 0.05 giây, phù hợp cho mục đích giảng dạy hoặc demo học thuật.
Khả năng chỉnh sửa linh hoạt: Giao diện thiết kế cho phép chỉnh sửa bản đồ, thêm/bớt tường, đá, thay đổi vị trí robot và mục tiêu một cách dễ dàng.
Hỗ trợ kiểm tra trường hợp thất bại: Hệ thống có khả năng nhận diện trường hợp không tồn tại đường đi và thông báo rõ ràng, tránh gây hiểu lầm cho người sử dụng.

3.2.2 Những điểm hạn chế tồn tại

Giới hạn về quy mô bản đồ: Hệ thống chỉ tối ưu với bản đồ vừa (15x20 ô), chưa phù hợp khi mở rộng lên bản đồ lớn hơn hoặc bản đồ thực tế nhiều ngã rẽ phức tạp.
Hiệu suất giảm khi mật độ vật cản cao: Khi số lượng vật cản tăng mạnh (đặc biệt là tường và đá), tốc độ xử lý giảm rõ rệt, robot di chuyển có độ trễ nhẹ trong mô phỏng.
Chưa có thuật toán so sánh: Việc chỉ sử dụng BFS giới hạn khả năng đánh giá hiệu quả thuật toán, không có sự so sánh với các giải pháp khác như A*, DFS, Dijkstra.

Chưa có tương tác nâng cao: Hệ thống chưa hỗ trợ các chức năng như lưu bản đồ, tạo bản đồ ngẫu nhiên, hay chế độ mô phỏng tốc độ nhanh/chậm theo lựa chọn người dùng.

3.2.3 Phân tích tổng quan

Tiêu chí	Đánh giá hiện tại	Nhận xét cụ thể
Độ chính xác tìm đường	Rất tốt	Luôn tìm đường ngắn nhất
Giao diện mô phỏng	Trực quan, dễ sử dụng	Hiển thị dễ hiểu, thân thiện người mới
Tốc độ xử lý bản đồ vừa	Nhanh, ổn định	Không có độ trễ rõ rệt
Tốc độ xử lý bản đồ phức tạp	Giảm hiệu suất	Chậm nhẹ khi vật cản dày đặc
Khả năng mở rộng thuật toán	Chưa triển khai	Hạn chế về tính linh hoạt học thuật

3.2.4 Đánh giá chung

Nhìn chung, hệ thống mô phỏng với thuật toán BFS hoàn toàn phù hợp với mục đích học tập, nghiên cứu cơ bản. Đây là mô hình trực quan hóa tốt để hiểu rõ cách hoạt động của thuật toán tìm kiếm theo chiều rộng. Tuy nhiên, nếu mở rộng ứng dụng thực tế hoặc phát triển thành sản phẩm phức tạp, cần phải tiếp tục nâng cấp về thuật toán, giao diện và hiệu suất xử lý.

KẾT LUẬN

Qua quá trình nghiên cứu và thực nghiệm, đề tài đã đạt được các mục tiêu đề ra là xây dựng hệ thống mô phỏng trực quan quá trình tìm đường của robot sử dụng thuật toán BFS. Hệ thống hoạt động chính xác, giao diện dễ sử dụng, và có khả năng mở rộng phát triển tiếp tục cho các nghiên cứu chuyên sâu hơn về thuật toán tìm kiếm hoặc điều khiển robot.

TÀI LIỆU THAM KHẢO

1. T. N. Việt, P. H. V. Trường, *Nhập môn Trí tuệ nhân tạo – Chương 1: Bài toán tìm kiếm*, Trường Đại học Văn Lang, 2025.
2. Stuart Russell, Peter Norvig, *Artificial Intelligence: A Modern Approach*, 4th Edition, Pearson Education, 2020.
3. GeeksforGeeks, "Breadth First Search or BFS for a Graph," [Online]. Available: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
4. ChatGPT, *mô hình ngôn ngữ trí tuệ nhân tạo*, OpenAI, phát triển từ năm 2018 và được cập nhật đến phiên bản GPT-4 (ra mắt năm 2023), hỗ trợ giải thích và tổng hợp kiến thức trí tuệ nhân tạo phục vụ học tập và nghiên cứu.