BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

KHOA CÔNG NGHỆ THÔNG TIN

HCMUTE

ĐỒ ÁN CUỐI KỲ

HỌC KỲ 2 – NĂM HỌC 2024-2025

MÔN HỌC: TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI: XÂY DỰNG GAME FLOW FREE VÀ ỨNG DỤNG CÁC THUẬT TOÁN TÌM KIẾM

Mã lớp học phần: ARIN230585_05

Giảng viên hướng dẫn: TS. Phan Thị Huyền Trang

Danh sách sinh viên thực hiện:

MSSV	Họ tên
23133061	Phan Trọng Quí
23133056	Phan Trọng Phú
23133030	Đỗ Kiến Hưng

Thành phố Hồ Chí Minh, 12 tháng 05 năm 2025

Nhận xét của giảng viên

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TP. Hồ Chí Minh, ngày 12 tháng 05 năm 2025

Giảng viên ký tên

KẾ HOẠCH PHÂN CÔNG NHIỆM VỤ THỰC HIỆN

ĐỀ TÀI CUỐI KỲ MÔN TRÍ TUỆ NHÂN TẠO

HỌC KỲ 2 - NĂM HỌC 2024-2025

1. Mã lớp môn học: ARIN230585_05

2. Giảng viên hướng dẫn: TS. Phan Thị Huyền Trang

3. Tên đề tài: XÂY DỰNG GAME FLOW FREE VÀ ỨNG DỤNG CÁC THUẬT TOÁN TÌM KIẾM

4. GitHub: https://github.com/darktheDE/

5. Nhóm thực hiện: Nhóm 6

6. Bảng phân công nhiệm vụ:

Sinh viên thực hiện	Nội dung thực hiện
Phan Trọng Quí	Nghiên cứu, kiểm thử, thuyết trình, Thiết kế thuật toán
Phan Trọng Phú	Xây dựng mô hình, giao diện, quản lý github
Đỗ Kiến Hưng	Thuật toán chính, giao diện

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

Thuật ngữ	Ý nghĩa
AI	Trí tuệ nhân tạo
GUI	Giao diện người dùng đồ họa
BFS	Tìm kiếm theo chiều rộng
DFS	Tìm kiếm theo chiều sâu
UCS	Tìm kiếm chi phí đồng nhất
A*	Thuật toán A sao
CSP	Bài toán thỏa mãn ràng buộc

LỜI NÓI ĐẦU

Trí tuệ nhân tạo đang ngày càng khẳng định vai trò là một trong những trụ cột công nghệ của thế kỷ 21, với những ứng dụng sâu rộng trong mọi mặt của đời sống và khoa học. Trong đó, các thuật toán tìm kiếm và giải quyết vấn đề luôn là một lĩnh vực nghiên cứu cốt lõi, đặt nền móng cho nhiều hệ thống thông minh. Môn học Trí Tuệ Nhân Tạo đã trang bị cho chúng em những kiến thức nền tảng và chuyên sâu về các phương pháp tìm kiếm. Các phương pháp này bao gồm từ những thuật toán cổ điển đến các kỹ thuật giải quyết vấn đề trong môi trường không chắc chắn, điển hình là tìm kiếm không có quan sát.

Với mong muốn củng cố kiến thức lý thuyết đã học và phát triển kỹ năng ứng dụng vào thực tiễn, nhóm chúng em đã lựa chọn thực hiện đề tài. Đề tài có tên "Xây dựng Game Flow Free ứng dụng các thuật toán tìm kiếm AI".

Game Flow Free, một trò chơi giải đố trực quan và hấp dẫn, đặt ra một bài toán tìm kiếm thú vị. Trong đó, các đường đi phải được hình thành để kết nối các cặp điểm màu mà không giao nhau và phủ kín toàn bộ lưới. Đây là một nền tảng lý tưởng để triển khai, thử nghiệm và so sánh hiệu quả của nhiều thuật toán tìm kiếm khác nhau.

Báo cáo này sẽ trình bày chi tiết về cơ sở lý thuyết trọng tâm là tìm kiếm không có quan sát. Cùng với đó là quá trình phân tích, thiết kế và xây dựng Game Flow Free. Đặc biệt, báo cáo sẽ đi sâu vào việc áp dụng một số thuật toán tìm kiếm tiêu biểu đã được học trong chương trình để giải quyết bài toán Flow Free. Nhóm sẽ tiến hành thực nghiệm và đánh giá kết quả thu được. Qua đó, chúng em hy vọng không chỉ làm rõ hơn các khái niệm AI đã học mà còn rút ra được những kinh nghiệm thực tế quý báu trong việc phát triển một ứng dụng AI cụ thể.

Chúng em xin chân thành cảm ơn TS. Phan Thị Huyền Trang đã tận tình hướng dẫn và tạo điều kiện để chúng em hoàn thành đề tài này. Mặc dù đã rất cố gắng, nhưng do kiến thức và kinh nghiệm còn hạn chế, báo cáo chắc chắn không tránh khỏi những thiếu sót. Chúng em rất mong nhận được những ý kiến đóng góp quý báu từ quý thầy cô để đề tài được hoàn thiện hơn.

CHƯƠNG 1. GIỚI THIỆU

1.1. Lý do chọn đề tài

Việc lựa chọn đề tài "Xây dựng Game Flow Free ứng dụng các thuật toán tìm kiếm AI" xuất phát từ nhiều cơ sở quan trọng. Trước hết, tầm quan trọng của lý thuyết tìm kiếm trong AI là không thể phủ nhận. Các thuật toán tìm kiếm là nền tảng của nhiều ứng dụng AI, từ giải quyết vấn đề, lập kế hoạch đến học máy. Việc tìm hiểu sâu về một khía cạnh đặc thù như tìm kiếm không có quan sát giúp mở rộng hiểu biết về cách AI đối phó với sự không chắc chắn, một thách thức phổ biến trong thế giới thực. Bên cạnh đó, đề tài tạo cơ hội để nhóm áp dụng trực tiếp các kiến thức đã học từ môn Trí Tuệ Nhân Tạo. Các kiến thức này bao gồm lý thuyết về không gian trạng thái, các thuật toán tìm kiếm mù, tìm kiếm có thông tin và các khái niệm liên quan đến môi trường không chắc chắn.

Ngoài ra, tính hấp dẫn và phù hợp của Game Flow Free cũng là một yếu tố thúc đẩy. Flow Free là một game giải đố với luật chơi đơn giản nhưng việc tìm ra lời giải có thể rất phức tạp, đòi hỏi tư duy logic và chiến lược. Điều này làm cho nó trở thành một đối tượng thú vị để mô hình hóa dưới dạng bài toán tìm kiếm AI. Cấu trúc của game cho phép dễ dàng áp dụng và so sánh hiệu quả của nhiều thuật toán tìm kiếm khác nhau, từ đó làm rõ hơn ưu nhược điểm của từng phương pháp. Hơn nữa, game có thể được mở rộng với nhiều kích thước lưới và số lượng màu khác nhau, tạo ra các bài toán với độ khó đa dạng để kiểm thử.

Cuối cùng, quá trình thực hiện đề tài giúp rèn luyện các kỹ năng quan trọng như phân tích vấn đề, thiết kế thuật toán, lập trình, gỡ lỗi, làm việc nhóm, và trình bày báo cáo khoa học.

1.2. Mục tiêu đề tài

Đề tài hướng đến việc đạt được nhiều mục tiêu chính, bao gồm cả lý thuyết, ứng dụng và đánh giá. Về mặt lý thuyết, mục tiêu là nghiên cứu và trình bày một cách có hệ thống các khái niệm cốt lõi của tìm kiếm không có quan sát. Điều này bao gồm định nghĩa, trạng thái niềm tin và các phương pháp tiếp cận để giải quyết bài toán. Đồng thời, nhóm cũng ôn tập và hệ thống hóa kiến thức về các thuật toán tìm kiếm tiêu biểu được học trong chương trình đào tạo.

Về mặt ứng dụng và sản phẩm, đề tài nhắm đến việc phân tích và mô hình hóa bài toán Game Flow Free dưới dạng một bài toán tìm kiếm trong không gian trạng thái. Một mục tiêu quan trọng khác là thiết kế và xây dựng một phiên bản Game Flow Free hoàn

chỉnh bằng ngôn ngữ lập trình Python, có giao diện đồ họa cho phép người dùng tương tác. Song song đó, nhóm sẽ lựa chọn và triển khai một số thuật toán tìm kiếm AI để tự động tìm lời giải cho các màn chơi Flow Free.

Cuối cùng, về mặt đánh giá, đề tài đặt mục tiêu thiết kế các kịch bản thử nghiệm với các màn chơi Flow Free có độ khó khác nhau. Từ đó, nhóm sẽ thực hiện so sánh và đánh giá hiệu năng của các thuật toán đã triển khai trên các màn chơi đó. Kết quả đánh giá sẽ giúp rút ra những nhận xét, kết luận về sự phù hợp và hiệu quả của từng thuật toán đối với bài toán cụ thể này.

1.3. Phạm vi đề tài

Để đảm bảo tính khả thi và tập trung, đề tài được giới hạn trong một số phạm vi nhất định. Trong phạm vi lý thuyết, nghiên cứu sẽ tập trung sâu về tìm kiếm không có quan sát theo các tài liệu và giáo trình chính. Nhóm cũng sẽ tổng quan các thuật toán tìm kiếm cơ bản và thông dụng được giảng dạy trong môn học, không đi sâu vào các biến thể quá phức tạp hoặc các lĩnh vực AI khác không trực tiếp liên quan.

Đối với phạm vi sản phẩm, Game Flow Free được phát triển cho nền tảng máy tính cá nhân. Giao diện đồ họa sẽ được thiết kế đơn giản, tập trung vào chức năng cốt lõi là chơi game và hiển thị quá trình giải bằng AI. Game sẽ hỗ trợ các màn chơi trên lưới hình chữ nhật với các cặp điểm màu cố định. Về phạm vi thuật toán áp dụng, nhóm sẽ lựa chọn một số thuật toán tiêu biểu từ các nhóm tìm kiếm mù và tìm kiếm có thông tin để triển khai giải game. Cuối cùng, phạm vi đánh giá sẽ dựa trên các tiêu chí cơ bản như thời gian thực thi và số lượng trạng thái được duyệt trên một tập hợp các màn chơi được thiết kế trước.

1.4. Đối tượng nghiên cứu

Các đối tượng chính được tập trung nghiên cứu trong đề tài này bao gồm:

Lý thuyết "Tìm kiếm không có quan sát" (Searching with no observation). Bao gồm các khái niệm, đặc điểm, và phương pháp giải quyết.

Các thuật toán tìm kiếm trong Trí tuệ nhân tạo bao gồm các thuật toán tìm kiếm mù (BFS, DFS, ...), thuật toán tìm kiếm có thông tin (Greedy Best-First Search, A*,...), và các khái niệm liên quan như hàm heuristic.

Bài toán giải đố Game Flow Free. Luật chơi, cấu trúc bài toán, và cách biểu diễn bài toán dưới dạng tìm kiếm AI.

Ngôn ngữ lập trình Python và các thư viện hỗ trợ. Đặc biệt là thư viện dùng để xây dựng giao diện đồ họa (ví dụ: Pygame, Tkinter) và các cấu trúc dữ liệu phù hợp.

1.5. Phương pháp nghiên cứu

Để đạt được các mục tiêu đề ra, nhóm đã áp dụng kết hợp các phương pháp nghiên cứu sau:

Nghiên cứu tài liệu. Thu thập và nghiên cứu các tài liệu khoa học, giáo trình, sách chuyên khảo, bài báo liên quan đến lý thuyết tìm kiếm AI, đặc biệt là "Tìm kiếm không có quan sát", và các thuật toán tìm kiếm cụ thể. Tìm hiểu về Game Flow Free, các biến thể và các phương pháp giải đã có.

Phân tích và Thiết kế hệ thống. Phân tích chi tiết yêu cầu của bài toán Game Flow Free. Thiết kế cấu trúc dữ liệu để biểu diễn trạng thái game, các hành động và không gian tìm kiếm. Thiết kế kiến trúc tổng thể cho ứng dụng game và các module thuật toán.

Phương pháp thực nghiệm. Hiện thực hóa Game Flow Free và các thuật toán tìm kiếm đã chọn bằng ngôn ngữ Python. Thiết kế các bộ dữ liệu thử nghiệm (các màn chơi Flow Free với các đặc điểm khác nhau). Tiến hành chạy thử nghiệm, thu thập dữ liệu về hiệu năng của các thuật toán.

Phương pháp so sánh và đánh giá. So sánh kết quả thực nghiệm của các thuật toán dựa trên các tiêu chí đã định trước (thời gian, số nút duyệt). Phân tích, đánh giá ưu nhược điểm và sự phù hợp của từng thuật toán đối với bài toán Flow Free.

Phương pháp phát triển lặp. Quá trình xây dựng sản phẩm có thể diễn ra theo các vòng lặp nhỏ, từ việc xây dựng các chức năng cơ bản đến hoàn thiện và tích hợp các thuật toán, đồng thời kiểm thử và cải tiến liên tục.

1.6. Bố cục đề tài

Ngoài phần mở đầu, kết luận và tài liệu tham khảo, báo cáo được cấu trúc thành 05 chương chính như sau:

CHƯƠNG 1: GIỚI THIỆU

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VỀ TÌM KIẾM KHÔNG CÓ QUAN SÁT

CHƯƠNG 3: PHÂN TÍCH HỆ THỐNG VÀ XÂY DỰNG SẢN PHẨM

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VỀ TÌM KIẾM KHÔNG CÓ QUAN SÁT

2.1. Giới thiệu về Tìm kiếm không có quan sát

Tìm kiếm không có quan sát (Searching with No Observation) là một dạng bài toán trong trí tuệ nhân tạo, trong đó tác nhân (agent) hoạt động trong môi trường không có cảm biến để xác định trạng thái hiện tại. Điều này có nghĩa là tác nhân không biết chính xác mình đang ở đâu hoặc môi trường xung quanh đang như thế nào.

Trong trường hợp này, tác nhân phải sử dụng suy luận logic để đưa ra quyết định thông qua các hành động mà nó thực hiện, thay vì dựa trên thông tin cảm nhận. Trạng thái mà tác nhân duy trì không phải là trạng thái vật lý cụ thể, mà là một tập hợp các khả năng được biểu diễn bằng trạng thái niềm tin (Belief State).

Mục tiêu cuối cùng là xây dựng được chuỗi hành động mà dù xuất phát từ bất kỳ trạng thái vật lý ban đầu nào, tác nhân vẫn có thể đạt được trạng thái mục tiêu.

2.2. Trạng thái niềm tin trong tìm kiếm không có quan sát

Trong môi trường không cảm biến (sensorless), tác nhân không thể xác định chính xác trạng thái hiện tại của mình. Thay vào đó, nó duy trì một tập hợp các trạng thái có thể xảy ra – tập hợp này gọi là trạng thái niềm tin (Belief State). Đây là cách biểu diễn tri thức của tác nhân về tình huống hiện tại, bao gồm tất cả các khả năng mà tác nhân có thể đang ở trong.

Trạng thái ban đầu là tập hợp các trạng thái vật lý mà tác nhân có thể bắt đầu từ đó. Trong quá trình thực hiện các hành động, trạng thái niềm tin sẽ thay đổi dựa vào mô hình chuyển trạng thái (Transition Model).

Tuy nhiên, trong không gian trạng thái niềm tin, bài toán không còn là hoàn toàn có thể quan sát được (fully observable) nữa, vì tác nhân không thể biết chắc chắn trạng thái thật sự của hệ thống tại thời điểm đó.

2.3. Phương pháp giải quyết bài toán tìm kiếm không có quan sát

Quá trình giải bài toán tìm kiếm trong không gian trạng thái niềm tin gồm 3 bước chính:

2.3.1. Xác định bài toán cơ bản

Cần mô hình hóa môi trường dưới dạng tìm kiếm thông thường, với các thành phần như sau. Tập các trạng thái vật lý đại diện cho mọi khả năng tồn tại. Tập các hành động hợp lệ được định nghĩa cho mỗi trạng thái. Mô hình chuyển trạng thái xác định kết quả khi

thực hiện một hành động trong một trạng thái cụ thể. Trạng thái mục tiêu các trạng thái đạt được điều kiện thành công. Chi phí hành động: nếu có, để tìm giải pháp tối ưu.

2.3.2. Mở rộng sang không gian trạng thái niềm tin

Ta chuyển từ trạng thái đơn lẻ sang tập hợp trạng thái, bằng cách như sau. Cách 1, ta có thể xác định hành động trong trạng thái niềm tin. Nếu một hành động không ảnh hưởng môi trường, nó có thể áp dụng cho mọi trạng thái trong trạng thái niềm tin. Nếu hành động dẫn đến kết thúc trò chơi trong một vài trạng thái, cần lấy giao của các hành động hợp lệ. Cách 2, cập nhật trạng thái niềm tin. Sau mỗi hành động, trạng thái niềm tin mới B' là tập hợp tất cả các trạng thái kết quả có thể xảy ra từ các trạng thái hiện tại.

$$B' = \text{RESULT}(B, a) = \bigcup_{s \in B} \text{RESULTS}(s, a)$$

2.3.3. Xây dựng thuật toán tìm kiếm

Dựa vào các thuật toán tìm kiếm quen thuộc (BFS, DFS, A*), ta tiến hành. Khởi tạo trạng thái niềm tin ban đầu B_0. Duyệt qua các trạng thái niềm tin, mở rộng dần từng nút bằng cách xác định các hành động hợp lệ tại B. Tính trạng thái niềm tin mới sau hành động a. Nếu đạt mục tiêu, dừng. Ghi nhớ các trạng thái niềm tin đã duyệt để tránh lặp. Áp dụng thuật toán tìm kiếm BFS, DFS hoặc A* tùy yêu cầu bài toán.

2.4. Triển khai giải quyết bài tập cá nhân 8-Puzzle

Để giải bài toán 8-puzzle bằng tìm kiếm không có quan sát với thuật toán BFS, ta cần làm việc trong không gian trạng thái niềm thay vì trạng thái đơn lẻ như trong tìm kiếm bình thường.

Trạng thái vật lý. Mỗi trạng thái là một cách sắp xếp các số từ 1 đến 8 và một ô trống (0) trong bảng 3x3. Có tổng cộng 9! = 362,880 trạng thái vật lý (chỉ một nửa trong số đó là solvable).

Trạng thái niềm tin. Thay vì biết chính xác trạng thái hiện tại, tác nhân chỉ biết mình có thể đang ở một trong nhiều trạng thái. Trạng thái niềm tin: một tập hợp con các trạng thái vật lý có thể xảy ra.

Mục tiêu là đưa hệ thống đến một trạng thái niềm tin sao cho mọi trạng thái trong đó đều là trạng thái đích (tức là mọi trạng thái vật lý trong đó đều là đích), hoặc tối thiểu có thể đảm bảo chắc chắn đã đến trạng thái đích bất kể ban đầu là gì.

Ý tưởng chính là khởi đầu với trạng thái niềm tin là tất cả các trạng thái solvable của bài toán 8-puzzle. Ở mỗi bước, thử tất cả các hành động (UP, DOWN, LEFT, RIGHT) trên mọi trạng thái trong trạng thái niềm tin hiện tại. Cập nhật trạng thái niềm tin mới bằng

cách hợp kết quả hành động trên từng trạng thái vật lý. Dùng BFS để tìm chuỗi hành động đầu tiên sao cho trạng thái niềm tin đạt tập con chỉ chứa trạng thái mục tiêu.

Initial belief state (chỉ có 1 trạng thái vật lý ban đầu):

2 6 5

0 8 7

4 3 1

Goal state:

1 2 3

4 5 6

7 8 0

Biểu diễn trạng thái. Ta sẽ biểu diễn mỗi trạng thái dưới dạng tuple 1 chiều gồm 9 phần tử (từ trái sang phải, từ trên xuống dưới):

initial_state = (2, 6, 5, 0, 8, 7, 4, 3, 1)**,** goal_state = (1, 2, 3, 4, 5, 6, 7, 8, 0)

Xây dựng hàm chuyển trạng thái. Khi di chuyển ô trống (0), có 4 hướng:

Hành động	**Điều kiện**	**Di chuyển index**
UP	hàng > 0	-3
DOWN	hàng < 2	+3
LEFT	cột > 0	-1
RIGHT	cột < 2	+1

Mô phỏng BFS từ trạng thái initial_state. BFS Queue, mỗi phần tử là (belief_state, path).

queue = [({(2,6,5,0,8,7,4,3,1)}, [])]

Bước 1:

belief = {(2,6,5,0,8,7,4,3,1)} path = [].

Thực hiện 4 hành động:

UP: Vị trí 0 là ở index 3 → chuyển lên index 0**;** Swap 0 ↔ 2 → (0,6,5,2,8,7,4,3,1)

DOWN: index 3 → chuyển xuống index 6**;** Swap 0 ↔ 4 → (2,6,5,4,8,7,0,3,1)

RIGHT: index 3 (cột 0) → index 4**;** Swap 0 ↔ 8 → (2,6,5,8,0,7,4,3,1)

LEFT: không hợp lệ vì cột = 0

Thêm vào queue:

queue = [

({(0,6,5,2,8,7,4,3,1)}, ['UP']),

({(2,6,5,4,8,7,0,3,1)}, ['DOWN']),

({(2,6,5,8,0,7,4,3,1)}, ['RIGHT']),

]

Bước 2:

belief: {(0,6,5,2,8,7,4,3,1)}

path = ['UP']

Vị trí 0 là index 0

DOWN: (swap 0 ↔ 2) → (2,6,5,0,8,7,4,3,1) → trở lại initial

RIGHT: (0 ↔ 6) → (6,0,5,2,8,7,4,3,1)

Cập nhật queue:

queue = [

({(2,6,5,4,8,7,0,3,1)}, ['DOWN']),

({(2,6,5,8,0,7,4,3,1)}, ['RIGHT']),

({(6,0,5,2,8,7,4,3,1)}, ['UP', 'RIGHT']),

]

Bước 3:

Tiếp tục như trên, mỗi bước:

1. Lấy belief state đầu tiên trong queue.
2. Với mỗi trạng thái trong đó, áp dụng 4 hướng hợp lệ.
3. Nếu ra trạng thái đã có rồi → bỏ qua.
4. Nếu ra goal_state → kết thúc.

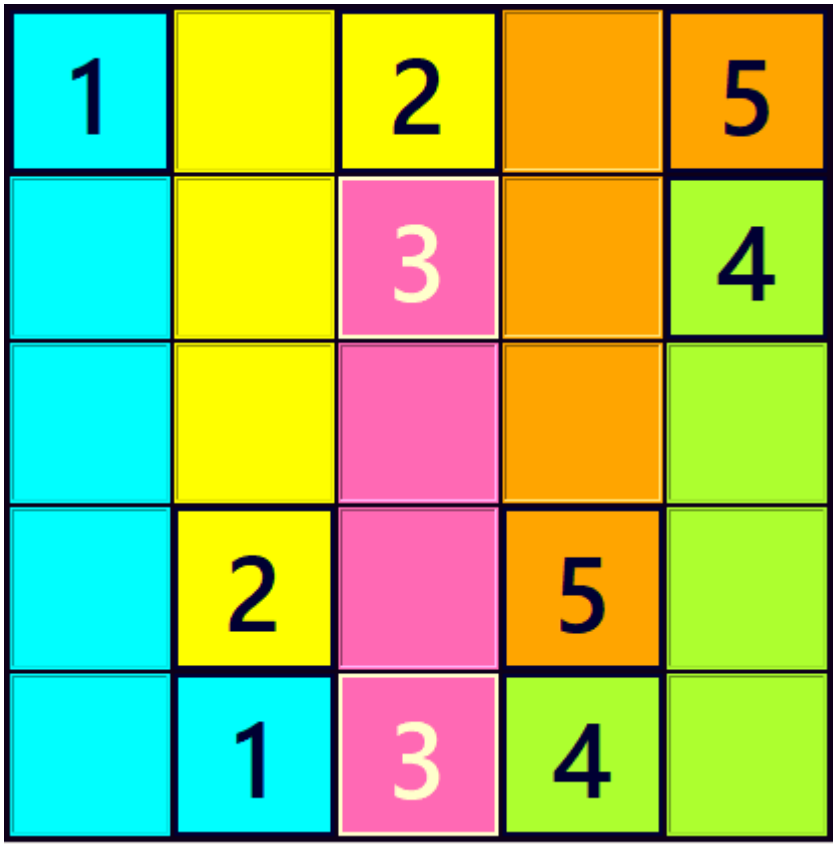
CHƯƠNG 3. PHÂN TÍCH HỆ THỐNG VÀ XÂY DỰNG SẢN PHẨM

3.1. Phân tích bài toán Game Flow Free

Game Flow Free là một trò chơi giải đố phổ biến, thu hút người chơi bởi sự đơn giản trong luật chơi nhưng lại ẩn chứa độ phức tạp cao trong việc tìm kiếm lời giải. Việc phân tích kỹ lưỡng bài toán này là bước đầu tiên và quan trọng để có thể áp dụng hiệu quả các thuật toán Trí tuệ nhân tạo.

3.1.1. Mô tả Game Flow Free

Trong Game Flow Free, người chơi được cung cấp một lưới ô vuông. Trên lưới này, có một số cặp điểm được đánh dấu bằng các màu sắc khác nhau. Nhiệm vụ của người chơi là nối các cặp điểm cùng màu lại với nhau bằng cách vẽ các đường đi trên các ô của lưới. Các đường đi này phải tuân theo một số quy tắc nhất định. Thứ nhất, mỗi đường đi chỉ được nối hai điểm cùng màu. Thứ hai, các đường đi của những cặp màu khác nhau không được phép cắt nhau hoặc đi chồng lên nhau trên cùng một ô. Thứ ba, một yêu cầu quan trọng để hoàn thành một màn chơi là tất cả các ô trên lưới phải được lấp đầy bởi các đường đi, không được để trống bất kỳ ô nào. Một màn chơi được coi là giải thành công khi tất cả các cặp điểm màu được nối đúng cách và toàn bộ lưới được phủ kín. Độ khó của trò chơi thường tăng lên theo kích thước của lưới và số lượng cặp màu cần nối.



3.1.2. Định nghĩa bài toán tìm kiếm cho Game Flow Free

Để áp dụng các thuật toán Trí tuệ nhân tạo, chúng ta cần mô hình hóa Game Flow Free thành một bài toán tìm kiếm không gian trạng thái. Điều này bao gồm việc xác định các thành phần cơ bản.

Một là, không gian trạng thái. Mỗi trạng thái trong bài toán này biểu diễn cấu hình hiện tại của lưới game. Nó bao gồm vị trí cố định của các cặp điểm màu ban đầu, các đoạn đường đã được vẽ cho từng màu, và các ô còn lại trên lưới đang ở trạng thái trống. Một cách phổ biến để biểu diễn trạng thái là sử dụng một ma trận hai chiều, trong đó mỗi phần tử của ma trận lưu trữ thông tin về màu sắc của đường đi đang chiếm giữ ô đó, hoặc đánh dấu ô đó là điểm đầu hoặc cuối của một màu, hoặc là ô trống.

Hai là, trạng thái ban đầu. Đây là trạng thái xuất phát của quá trình tìm kiếm. Trong Game Flow Free, trạng thái ban đầu là một lưới chỉ chứa các cặp điểm màu tại vị trí đã cho, còn tất cả các ô khác trên lưới đều ở trạng thái trống.

Ba là, các hành động. Một hành động trong bài toán này tương ứng với việc mở rộng một đường đi của một màu cụ thể. Cụ thể hơn, từ điểm cuối hiện tại của một đường đi chưa hoàn chỉnh của một màu, tác nhân có thể chọn một ô trống kề cận, ví dụ như Lên, Xuống, Trái, hoặc Phải, để vẽ tiếp đường đi đó. Một hành động được coi là hợp lệ nếu ô được chọn nằm trong phạm vi lưới, ô đó đang trống, và việc vẽ thêm đoạn đường đó không tạo ra tình huống bị cô lập cho các ô trống khác, trừ khi đó là bước đi cuối cùng để hoàn thành một đường đi và lấp đầy lưới.

Bốn là, hàm chuyển tiếp, hay còn gọi là mô hình chuyển trạng thái. Hàm này mô tả cách trạng thái của lưới thay đổi sau khi một hành động hợp lệ được thực hiện. Khi một ô trống được chọn để mở rộng đường đi của một màu, trạng thái mới được tạo ra bằng cách cập nhật thông tin của ô đó trong ma trận biểu diễn lưới, gán cho nó màu tương ứng.

Năm là, trạng thái đích, hay còn gọi là kiểm tra mục tiêu. Một trạng thái được coi là trạng thái đích nếu nó thỏa mãn tất cả các điều kiện để hoàn thành màn chơi. Cụ thể, tất cả các cặp điểm màu phải được nối hoàn chỉnh với nhau bằng các đường đi riêng biệt. Các đường đi của các màu khác nhau không được giao nhau. Và quan trọng nhất, không còn ô nào trên lưới ở trạng thái trống, tức là toàn bộ lưới đã được lấp đầy bởi các đường đi.

Sáu là, chi phí đường đi. Trong bài toán Flow Free cơ bản, mục tiêu chính là tìm ra một lời giải bất kỳ. Tuy nhiên, nếu xét đến khía cạnh tối ưu, chúng ta có thể coi mỗi hành động mở rộng đường đi, tức là vẽ vào một ô mới, có chi phí là 1. Các thuật toán như Tìm

kiếm chi phí đồng nhất hoặc A sao sẽ cố gắng tìm lời giải với tổng chi phí thấp nhất, mặc dù trong Flow Free, tất cả các lời giải hợp lệ đều sẽ lấp đầy cùng một số lượng ô.

3.2. Lựa chọn và Áp dụng Thuật toán cho Game Flow Free

Sau khi đã định nghĩa rõ ràng bài toán, bước tiếp theo là lựa chọn các thuật toán tìm kiếm phù hợp từ những kiến thức đã học để triển khai giải quyết Game Flow Free. Việc lựa chọn này dựa trên đặc điểm của bài toán và mục tiêu so sánh hiệu năng giữa các phương pháp khác nhau.

3.2.1. Lý giải lựa chọn các thuật toán được triển khai

Nhóm đã quyết định lựa chọn một tập hợp các thuật toán đại diện cho các nhóm tìm kiếm khác nhau để có cái nhìn toàn diện về cách chúng hoạt động trên bài toán Flow Free. Các thuật toán được chọn bao gồm Thuật toán quay lui, Tìm kiếm theo chiều rộng, Thuật toán A sao, và kỹ thuật Truyền bá ràng buộc.

Việc lựa chọn Thuật toán quay lui là vì đây là một kỹ thuật cơ bản và trực quan để giải quyết các bài toán tìm kiếm tổ hợp. Tìm kiếm theo chiều rộng được chọn vì tính đầy đủ và khả năng tìm ra lời giải có số bước đi ngắn nhất trong trường hợp chi phí mỗi bước như nhau, rất tốt để làm cơ sở so sánh. Thuật toán A sao là một đại diện mạnh mẽ của nhóm tìm kiếm có thông tin, có khả năng tìm giải pháp tối ưu một cách hiệu quả nếu có hàm heuristic tốt. Cuối cùng, việc đưa vào kỹ thuật Truyền bá ràng buộc, thường gắn liền với Bài toán thỏa mãn ràng buộc, nhằm mục đích khám phá một hướng tiếp cận khác, có tiềm năng cắt tỉa không gian tìm kiếm hiệu quả. Việc triển khai nhiều thuật toán cho phép chúng ta không chỉ kiểm chứng tính đúng đắn của chúng mà còn so sánh được hiệu suất và các đặc tính khác khi áp dụng vào cùng một bài toán thực tế.

3.2.2. Tổng quan về các thuật toán được chọn và cách áp dụng

Dưới đây là ý tưởng cơ bản và cách mỗi thuật toán được chọn được áp dụng để giải quyết Game Flow Free.

3.2.2.1. Thuật toán quay lui

Ý tưởng cơ bản của Thuật toán quay lui là một kỹ thuật giải quyết vấn đề bằng cách thử xây dựng một giải pháp từng bước một. Tại mỗi bước, nếu lựa chọn hiện tại không dẫn đến giải pháp hoặc vi phạm một ràng buộc nào đó, thuật toán sẽ "quay lui" lại bước trước đó và thử một lựa chọn khác. Nó duyệt qua không gian các giải pháp tiềm năng một cách có hệ thống.

Quy trình áp dụng giải Game Flow Free bằng Thuật toán quay lui bắt đầu từ một trạng thái lưới trống, chỉ có các điểm màu. Sau đó, thuật toán chọn một cặp màu chưa được nối hoặc một đường đi của màu đang dang dở. Tiếp theo, nó thử mở rộng đường đi của màu đó sang một ô trống kề cận hợp lệ, đảm bảo không cắt đường khác và không ra ngoài lưới. Nếu việc mở rộng này dẫn đến một trạng thái hợp lệ, ví dụ không tạo ra ngõ cụt cho các màu khác và không vi phạm luật chơi, thuật toán sẽ tiếp tục đệ quy để giải quyết phần còn lại của lưới, bao gồm các màu khác hoặc phần còn lại của màu đang vẽ. Nếu việc giải quyết phần còn lại thành công, tức là tìm được trạng thái đích, thì giải pháp hiện tại là đúng và quá trình kết thúc. Ngược lại, nếu việc mở rộng không dẫn đến giải pháp, ví dụ không còn nước đi hợp lệ, hoặc các bước tiếp theo đều thất bại, hoặc nếu việc giải quyết phần còn lại thất bại, thì thuật toán sẽ "quay lui". Điều này có nghĩa là xóa bỏ bước mở rộng vừa thực hiện, trả ô đó về trạng thái trống, và thử một hướng mở rộng khác cho màu đó tại bước trước. Nếu tất cả các khả năng đã được thử mà không tìm thấy giải pháp, thuật toán kết thúc và báo không thành công. Thuật toán quay lui có thể tìm ra giải pháp, nhưng hiệu quả của nó phụ thuộc nhiều vào thứ tự thử các lựa chọn và các kỹ thuật cắt tỉa để loại bỏ sớm các nhánh tìm kiếm không hứa hẹn.

3.2.2.2. Tìm kiếm theo chiều rộng

Ý tưởng cơ bản của Tìm kiếm theo chiều rộng, hay BFS, là khám phá tất cả các trạng thái, tức là cấu hình lưới, có thể đạt được từ trạng thái ban đầu theo từng lớp hoặc mức. Nó bắt đầu từ trạng thái ban đầu, sau đó xét tất cả các trạng thái có thể đạt được sau một hành động, rồi tất cả các trạng thái có thể đạt được sau hai hành động, và cứ thế tiếp tục. BFS sử dụng một cấu trúc dữ liệu hàng đợi để lưu trữ các trạng thái cần xét.

Quy trình áp dụng giải Game Flow Free bằng BFS bắt đầu với trạng thái ban đầu của lưới Flow Free, chỉ có các điểm màu, và đưa trạng thái này vào hàng đợi. Quá trình lặp lại cho đến khi hàng đợi rỗng hoặc tìm thấy lời giải. Trong mỗi lần lặp, một trạng thái lưới được lấy ra khỏi đầu hàng đợi. Nếu trạng thái này là trạng thái đích, nghĩa là tất cả các màu đã nối và lưới đã đầy, thì thuật toán dừng lại và thông báo thành công. Nếu không, thuật toán tạo ra tất cả các trạng thái kế tiếp hợp lệ từ trạng thái hiện tại bằng cách thử mở rộng một đường đi của một màu chưa hoàn thành sang một ô trống kề cận. Với mỗi trạng thái kế tiếp hợp lệ chưa được thăm, thuật toán sẽ đánh dấu đã thăm và đưa vào cuối hàng đợi. BFS đảm bảo sẽ tìm ra lời giải nếu có, và đó sẽ là lời giải có số lần mở rộng đường đi ít nhất để phủ kín lưới.

3.2.2.3. Thuật toán A sao

Ý tưởng cơ bản của Thuật toán A sao là một thuật toán tìm kiếm có hướng dẫn. Nó sử dụng một hàm đánh giá f(n) bằng tổng của g(n) và h(n) để ưu tiên mở rộng các trạng thái có vẻ hứa hẹn nhất. Trong đó, g(n) là chi phí thực tế để đi từ trạng thái ban đầu đến trạng thái n, và h(n) là chi phí ước lượng, hay heuristic, từ trạng thái n đến trạng thái đích. Thiết kế hàm Heuristic h(n) cho Flow Free, ví dụ như hàm tổng khoảng cách Manhattan, là một bước quan trọng. Khoảng cách Manhattan là tổng chênh lệch tuyệt đối theo tọa độ hàng và tọa độ cột giữa hai điểm. Hàm tổng khoảng cách Manhattan tính toán, với mỗi cặp màu chưa nối hoàn chỉnh, tổng khoảng cách Manhattan giữa các điểm cuối hiện tại của đường đi và các điểm đích tương ứng của chúng. Hoặc nó có thể là khoảng cách giữa điểm đầu và điểm cuối nếu chưa bắt đầu vẽ. Giá trị h(n) là tổng của tất cả các khoảng cách Manhattan này cho tất cả các cặp màu chưa hoàn thành. Một giá trị h(n) nhỏ hơn cho thấy trạng thái hiện tại có vẻ "gần" với trạng thái đích hơn.

Quy trình áp dụng giải Game Flow Free bằng A sao bắt đầu với trạng thái ban đầu. Thuật toán tính giá trị g bằng 0 và h dựa vào hàm heuristic đã chọn, từ đó tính f bằng g cộng h. Trạng thái này được đưa vào một hàng đợi ưu tiên, sắp xếp theo giá trị f tăng dần. Quá trình lặp lại cho đến khi hàng đợi ưu tiên rỗng hoặc tìm thấy lời giải. Trong mỗi lần lặp, trạng thái lưới n có giá trị f nhỏ nhất được lấy ra khỏi hàng đợi ưu tiên. Nếu n là trạng thái đích, thuật toán dừng lại và thông báo thành công. Với mỗi trạng thái kế tiếp hợp lệ m của n, thuật toán tính chi phí mới g(m), tính h(m) bằng hàm heuristic, và tính f(m) bằng g(m) cộng h(m). Nếu m chưa có trong hàng đợi ưu tiên hoặc giá trị f(m) mới tốt hơn giá trị cũ, thì thuật toán thêm hoặc cập nhật m vào hàng đợi ưu tiên. Nếu hàm heuristic là "chấp nhận được", A sao đảm bảo tìm ra lời giải tối ưu về mặt chi phí g(n).

3.2.2.4. Truyền bá ràng buộc trong giải quyết bài toán thỏa mãn ràng buộc

Ý tưởng cơ bản của việc áp dụng Truyền bá ràng buộc là mô hình hóa Bài toán Flow Free như một Bài toán thỏa mãn ràng buộc, hay CSP. Trong đó, các biến có thể là các ô trên lưới, và miền giá trị của mỗi biến là các màu có thể có hoặc trạng thái trống. Các ràng buộc bao gồm các quy tắc của trò chơi như các điểm đầu và cuối của mỗi màu phải có màu tương ứng, các ô liền kề trên cùng một đường đi phải cùng màu, các đường đi khác màu không được giao nhau, và tất cả các ô phải được tô màu. Truyền bá ràng buộc là một kỹ thuật được sử dụng trong CSP để giảm không gian tìm kiếm. Khi một giá trị được gán cho một biến, ví dụ một ô được tô một màu cụ thể, kỹ thuật này sẽ "truyền bá" ảnh hưởng của

việc gán đó đến các biến lân cận hoặc liên quan, loại bỏ các giá trị không còn hợp lệ khỏi miền giá trị của chúng.

Quy trình áp dụng giải Game Flow Free, thường kết hợp với một thuật toán tìm kiếm như quay lui, bắt đầu bằng việc mô hình hóa bài toán như đã nêu. Sau đó, quá trình tìm kiếm bắt đầu với một lưới trống. Thuật toán chọn một biến chưa được gán, ví dụ một ô chưa được tô màu. Nó thử gán một giá trị, tức là một màu, cho biến đó. Sau khi gán, thuật toán áp dụng truyền bá ràng buộc để kiểm tra và cập nhật miền giá trị của các biến khác. Nếu bất kỳ biến nào không còn giá trị hợp lệ nào trong miền của nó, thì lựa chọn hiện tại là không thể và cần quay lui. Nếu truyền bá ràng buộc không dẫn đến mâu thuẫn, thuật toán tiếp tục đệ quy để gán giá trị cho các biến còn lại. Nếu tìm được một phép gán hoàn chỉnh thỏa mãn tất cả ràng buộc, tức là lưới được giải, quá trình kết thúc. Nếu một lựa chọn dẫn đến ngõ cụt, thuật toán sẽ quay lui và thử một giá trị khác cho biến hiện tại. Truyền bá ràng buộc giúp phát hiện sớm các mâu thuẫn, từ đó cắt tỉa đáng kể không gian tìm kiếm.

3.3. Xây dựng sản phẩm (Game Flow Free)

Quá trình xây dựng sản phẩm Game Flow Free bao gồm nhiều công đoạn, từ việc lựa chọn công nghệ phù hợp đến việc hiện thực hóa các chức năng và thuật toán.

3.3.1. Lựa chọn công cụ phát triển

Để phát triển Game Flow Free, nhóm đã quyết định sử dụng ngôn ngữ lập trình Python. Lý do chính cho sự lựa chọn này là tính dễ đọc, dễ học của Python, cùng với hệ sinh thái thư viện phong phú hỗ trợ phát triển nhanh chóng. Đặc biệt, Python có nhiều thư viện mạnh mẽ cho việc xây dựng giao diện người dùng đồ họa.

Đối với việc xây dựng giao diện người dùng đồ họa, nhóm đã lựa chọn thư viện Tkinter. Tkinter được chọn vì nó đơn giản, dễ tích hợp và có sẵn trong bộ cài đặt chuẩn của Python, phù hợp cho các ứng dụng GUI không quá phức tạp về đồ họa.

3.3.2. Thiết kế cấu trúc dữ liệu

Việc thiết kế cấu trúc dữ liệu hiệu quả là rất quan trọng để biểu diễn trạng thái game và hỗ trợ các thuật toán tìm kiếm. Các cấu trúc dữ liệu chính được sử dụng bao gồm việc biểu diễn lưới game. Một ma trận hai chiều, ví dụ danh sách các danh sách trong Python hoặc mảng NumPy, được sử dụng để lưu trữ trạng thái của từng ô trên lưới. Mỗi phần tử trong ma trận có thể lưu một giá trị số nguyên hoặc một đối tượng để biểu thị ô trống, một điểm màu cố định và màu của nó, hoặc một phần của đường đi và màu của đường đi đó.

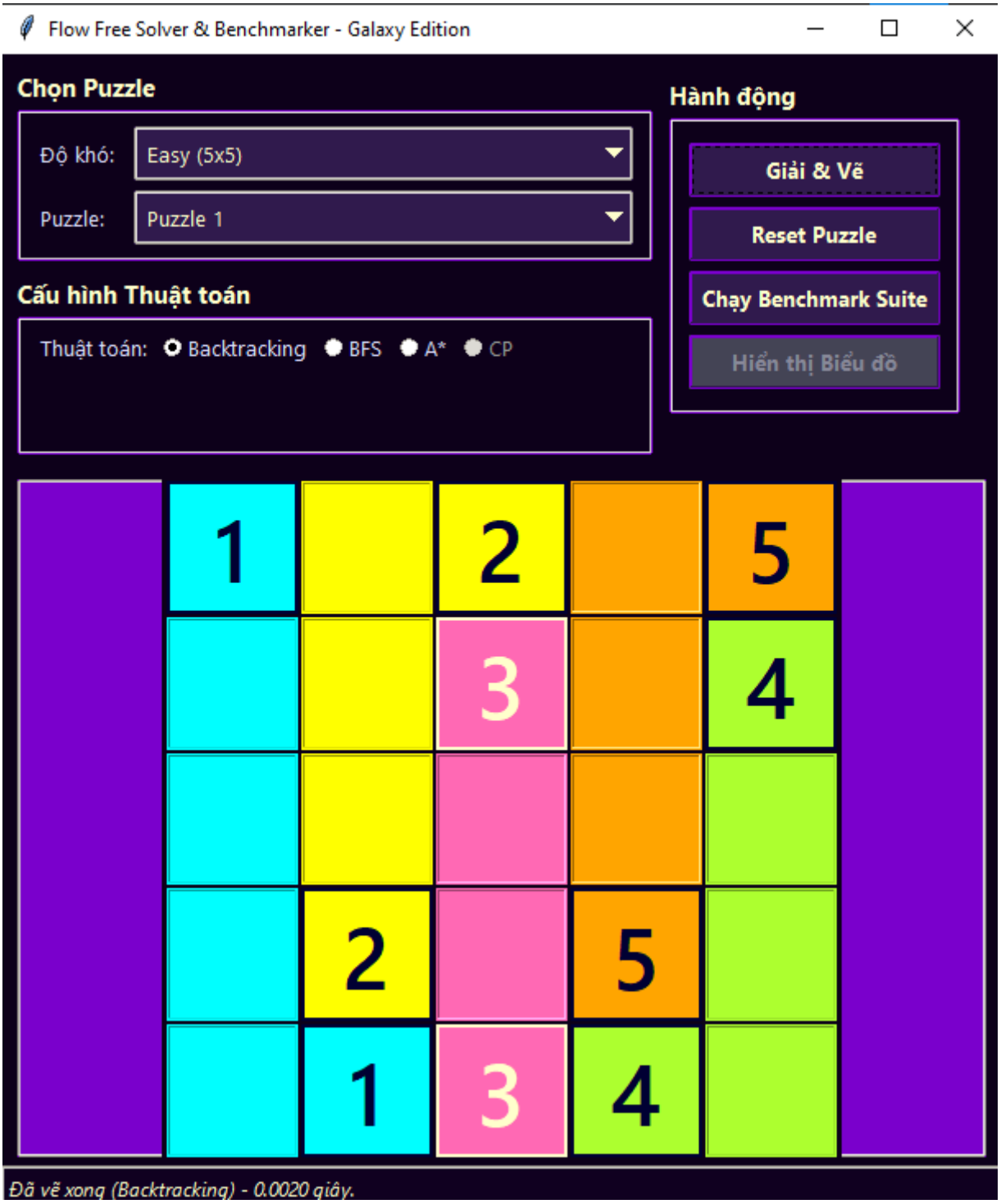
Bên cạnh đó, thông tin các cặp điểm màu cũng cần được lưu trữ. Một danh sách các tuple hoặc đối tượng được sử dụng để lưu trữ thông tin về các cặp điểm màu cần nối. Mỗi phần tử chứa thông tin về màu sắc, tọa độ điểm bắt đầu và tọa độ điểm kết thúc của cặp màu đó.

Cuối cùng, việc lưu trữ đường đi hiện tại trong quá trình tìm kiếm cũng rất cần thiết. Đối với mỗi màu đang được vẽ, một danh sách các tọa độ có thể được sử dụng để lưu trữ các ô mà đường đi của màu đó đã đi qua. Điều này giúp cho việc kiểm tra tính hợp lệ của nước đi, ví dụ không tự cắt chính mình, và quay lui khi cần thiết.

3.3.3. Thiết kế giao diện người dùng

Giao diện người dùng được thiết kế với mục tiêu đơn giản, trực quan và dễ sử dụng. Các thành phần chính của giao diện bao gồm khu vực hiển thị lưới game. Đây là thành phần trung tâm, hiển thị trực quan các ô lưới, các điểm màu cố định, và các đường đi được vẽ bởi người chơi hoặc bởi thuật toán AI.

Một chức năng quan trọng khác là chọn màn chơi, cho phép người dùng chọn một màn chơi từ danh sách các màn chơi có sẵn. Giao diện cũng cần có chức năng chọn thuật toán giải. Chức năng này cung cấp các nút hoặc menu để người dùng lựa chọn thuật toán AI như BFS, DFS, A sao, mà họ muốn sử dụng để giải màn chơi hiện tại. Các nút điều khiển như "Bắt đầu giải", "Reset màn chơi", "Thoát game" cũng cần được tích hợp. Cuối cùng, một khu vực để hiển thị thông tin hoặc kết quả là cần thiết. Khu vực này có thể hiển thị thông báo như "Đang giải...", "Đã tìm thấy lời giải!", thời gian giải, hoặc số bước thực hiện.



3.3.4. Hiện thực hóa các thuật toán đã chọn để giải Flow Free

Sau khi đã có cấu trúc dữ liệu và giao diện cơ bản, bước tiếp theo là hiện thực hóa các thuật toán tìm kiếm đã được trình bày ở mục 3.2.2. Mỗi thuật toán được cài đặt dưới dạng một hàm hoặc một lớp riêng biệt.

Quá trình hiện thực hóa đòi hỏi việc chuyển đổi các khái niệm lý thuyết của thuật toán, như hàng đợi cho BFS, ngăn xếp cho DFS, hàng đợi ưu tiên và hàm heuristic cho A sao, thành mã lệnh Python. Các hàm phụ trợ cần được xây dựng, ví dụ như hàm kiểm tra

một trạng thái có phải là trạng thái đích không, hàm tạo ra các trạng thái kế tiếp hợp lệ từ một trạng thái hiện tại, hàm tính toán giá trị heuristic cho A sao. Một điểm quan trọng là cách các thuật toán này tương tác với trạng thái game Flow Free. Khi một thuật toán mở rộng một trạng thái, nó cần tạo ra một bản sao của lưới hiện tại để thử nghiệm các nước đi mới, tránh làm thay đổi trạng thái gốc một cách không mong muốn. Việc quản lý các trạng thái đã thăm cũng rất quan trọng để tránh lặp lại công việc và các vòng lặp vô hạn, đặc biệt với BFS và DFS.

3.3.5. Quy trình giải một màn chơi bằng AI

Khi người dùng chọn một màn chơi và một thuật toán AI để giải, quy trình tổng thể diễn ra qua nhiều bước. Đầu tiên là khởi tạo, chương trình nạp thông tin của màn chơi đã chọn và thiết lập trạng thái ban đầu của lưới. Tiếp theo, thuật toán tìm kiếm được chọn, ví dụ A sao, được gọi với trạng thái ban đầu làm đầu vào.

Sau đó, quá trình tìm kiếm bắt đầu. Thuật toán duyệt qua không gian trạng thái theo logic đã được cài đặt, ví dụ mở rộng các nút theo thứ tự f(n) đối với A sao. Giao diện có thể được cập nhật để hiển thị một số thông tin về tiến trình, ví dụ số nút đã duyệt, hoặc thậm chí là hình ảnh hóa các bước đi thử nghiệm nếu có.

Cuối cùng là xử lý kết quả. Nếu tìm thấy lời giải, thuật toán sẽ trả về chuỗi các hành động hoặc trạng thái cuối cùng là lời giải. Chương trình sẽ dựa vào đó để vẽ các đường đi hoàn chỉnh lên lưới game trên giao diện, đồng thời hiển thị thông báo thành công và các thông số như thời gian giải. Nếu không tìm thấy lời giải sau một thời gian nhất định hoặc đã duyệt hết không gian trạng thái có thể, chương trình sẽ thông báo cho người dùng rằng không tìm thấy lời giải. Sau đó, người dùng có thể chọn chơi lại, chọn màn chơi khác, hoặc thoát game.

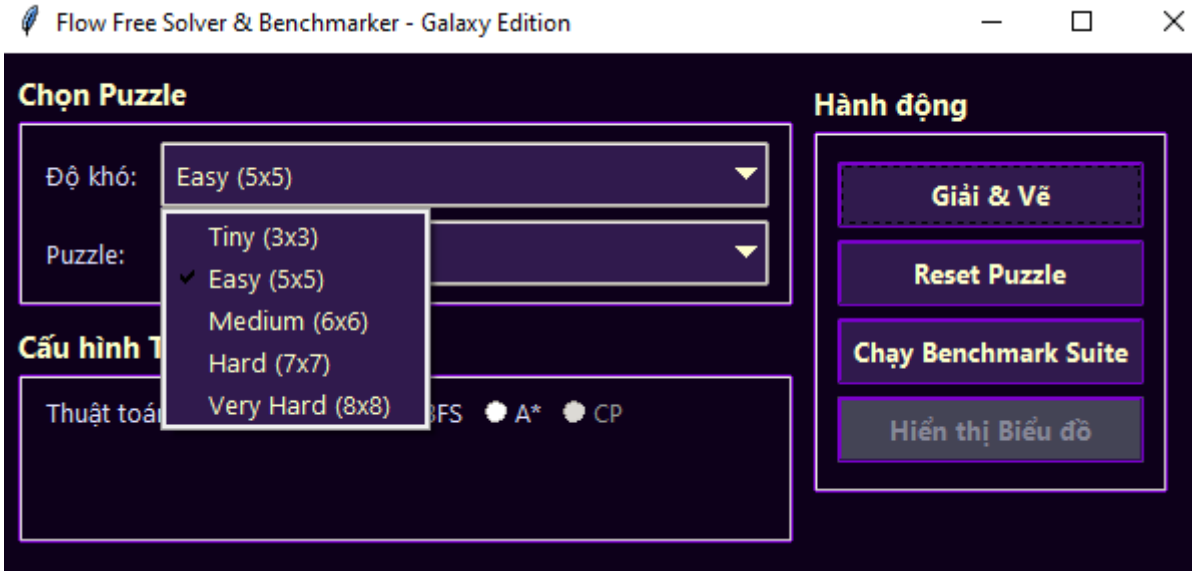
CHƯƠNG 4. KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1. Môi trường thử nghiệm

Tất cả các thử nghiệm được thực hiện trên một hệ thống máy tính cá nhân với cấu hình cụ thể. Hệ điều hành được sử dụng là [Windows 10 Pro 64-bit]. Phần cứng bao gồm bộ vi xử lý [Intel Core i5-8250U @ 1.60GHz], bộ nhớ RAM [Dung lượng RAM, 8GB]. Ngôn ngữ lập trình chính được sử dụng để phát triển game và các thuật toán là Python, phiên bản [Phiên bản Python, 3.9.7]. Thư viện đồ họa được lựa chọn để xây dựng giao diện người dùng là Tkinker. Các thư viện Python tiêu chuẩn khác cũng được sử dụng khi cần thiết.

4.2. Các màn chơi thử nghiệm

Để đánh giá hiệu năng của các thuật toán một cách khách quan, nhóm đã thiết kế và lựa chọn một tập hợp các màn chơi thử nghiệm với các đặc điểm khác nhau. Các màn chơi này đa dạng về kích thước lưới và số lượng cặp màu cần nối, từ đó tạo ra các mức độ khó khác nhau. Cụ thể, các kích thước lưới được sử dụng bao gồm 5x5, 6x6, 7x7, 8x8, 9x9. Đối với mỗi kích thước lưới, số lượng cặp màu cũng được thay đổi. Việc lựa chọn đa dạng các màn chơi giúp kiểm tra khả năng mở rộng và tính ổn định của các thuật toán khi đối mặt với các bài toán có độ phức tạp tăng dần.

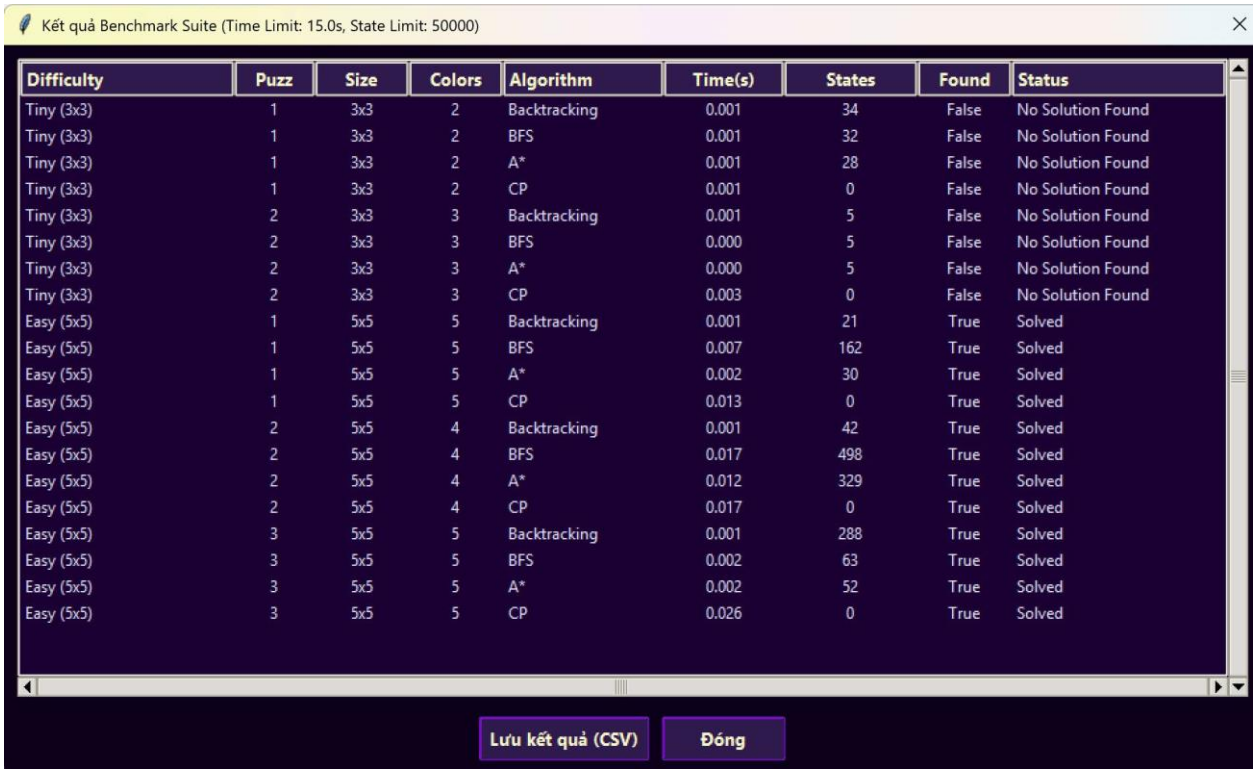


4.3. Kết quả thực nghiệm của các thuật toán

Sau khi chuẩn bị môi trường và các màn chơi thử nghiệm, nhóm đã tiến hành chạy từng thuật toán đã triển khai (Bao gồm Thuật toán quay lui, Tìm kiếm theo chiều rộng, Thuật toán A sao với các hàm heuristic khác nhau, và phương pháp Truyền bá ràng buộc) trên mỗi màn chơi. Các thông số chính được thu thập để đánh giá hiệu năng bao gồm thời

gian thực thi để tìm ra lời giải (tính bằng giây hoặc mili giây) và số lượng nút hoặc trạng thái đã được duyệt trong quá trình tìm kiếm. Kết quả được ghi nhận và tổng hợp dưới dạng bảng biểu để dễ dàng so sánh.

Bảng 4.1. Kết quả thử nghiệm trên các màn chơi



Difficulty	Puzz	Size	Colors	Algorithm	Time(s)	States	Found	Status
Tiny (3x3)	1	3x3	2	Backtracking	0.001	34	False	No Solution Found
Tiny (3x3)	1	3x3	2	BFS	0.001	32	False	No Solution Found
Tiny (3x3)	1	3x3	2	A*	0.001	28	False	No Solution Found
Tiny (3x3)	1	3x3	2	CP	0.001	0	False	No Solution Found
Tiny (3x3)	2	3x3	3	Backtracking	0.001	5	False	No Solution Found
Tiny (3x3)	2	3x3	3	BFS	0.000	5	False	No Solution Found
Tiny (3x3)	2	3x3	3	A*	0.000	5	False	No Solution Found
Tiny (3x3)	2	3x3	3	CP	0.003	0	False	No Solution Found
Easy (5x5)	1	5x5	5	Backtracking	0.001	21	True	Solved
Easy (5x5)	1	5x5	5	BFS	0.007	162	True	Solved
Easy (5x5)	1	5x5	5	A*	0.002	30	True	Solved
Easy (5x5)	1	5x5	5	CP	0.013	0	True	Solved
Easy (5x5)	2	5x5	4	Backtracking	0.001	42	True	Solved
Easy (5x5)	2	5x5	4	BFS	0.017	498	True	Solved
Easy (5x5)	2	5x5	4	A*	0.012	329	True	Solved
Easy (5x5)	2	5x5	4	CP	0.017	0	True	Solved
Easy (5x5)	3	5x5	5	Backtracking	0.001	288	True	Solved
Easy (5x5)	3	5x5	5	BFS	0.002	63	True	Solved
Easy (5x5)	3	5x5	5	A*	0.002	52	True	Solved
Easy (5x5)	3	5x5	5	CP	0.026	0	True	Solved

Ngoài các bảng biểu, việc sử dụng các biểu đồ cột để so sánh trực quan thời gian giải hoặc số nút duyệt giữa các thuật toán trên cùng một màn chơi hoặc trên các nhóm màn chơi khác nhau cũng rất hữu ích. Các biểu đồ này giúp làm nổi bật sự khác biệt về hiệu năng và xu hướng của từng thuật toán.

4.4. Phân tích và Đánh giá hiệu năng

Dựa trên các kết quả thực nghiệm đã thu thập, chúng ta có thể tiến hành phân tích và đánh giá hiệu năng của các thuật toán.

Về thời gian giải, có thể thấy rằng, thuật toán A sao với hàm heuristic Manhattan Sum thường cho thời gian giải nhanh hơn so với Tìm kiếm theo chiều rộng và Thuật toán quay lui trên các màn chơi có kích thước trung bình và lớn. Tìm kiếm theo chiều rộng tỏ ra hiệu quả trên các màn chơi nhỏ nhưng thời gian tăng nhanh khi kích thước lưới tăng. Thuật toán quay lui có thể rất nhanh nếu tìm được nhánh đúng sớm, nhưng cũng có thể rất chậm trong trường hợp xấu nhất. Phương pháp Truyền bá ràng buộc cho thấy tiềm năng trong việc giảm thời gian giải bằng cách loại bỏ các lựa chọn không hợp lệ sớm.

Về số lượng nút duyệt, kết quả cũng cho thấy những khác biệt rõ rệt. Các thuật toán có thông tin như A sao thường duyệt ít nút hơn đáng kể so với các thuật toán tìm kiếm mù như BFS, điều này cho thấy hiệu quả của việc sử dụng hàm heuristic để hướng dẫn quá trình tìm kiếm. Truyền bá ràng buộc cũng giúp giảm số lượng trạng thái cần xem xét so với quay lui thuần túy.

Ảnh hưởng của kích thước lưới và số lượng màu đến hiệu năng của từng thuật toán cũng là một yếu tố quan trọng. Khi kích thước lưới và số lượng màu tăng lên, tất cả các thuật toán đều gặp khó khăn hơn, thể hiện qua thời gian giải và số nút duyệt tăng lên. Tuy nhiên, mức độ ảnh hưởng có thể khác nhau giữa các thuật toán. Các thuật toán có khả năng cắt tỉa không gian tìm kiếm tốt hơn như A sao hoặc CP có thể chịu đựng được sự gia tăng độ phức tạp tốt hơn.

Đánh giá về hàm heuristic sử dụng cho thuật toán A sao cũng cần được xem xét. Hàm heuristic Manhattan Sum đã cho thấy hiệu quả nhất định trong việc hướng dẫn tìm kiếm. Tuy nhiên, có thể có những hàm heuristic khác hoặc sự kết hợp các heuristic mang lại kết quả tốt hơn nữa. Việc so sánh giữa các biến thể heuristic sẽ cung cấp thêm thông tin giá trị.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Qua quá trình thực hiện đề tài "Nghiên cứu lý thuyết về Tìm kiếm không có quan sát và Xây dựng Game Flow Free ứng dụng các thuật toán tìm kiếm AI", nhóm đã đạt được một số kết quả quan trọng. Về mặt lý thuyết, đề tài đã trình bày được các khái niệm cốt lõi và phương pháp tiếp cận của tìm kiếm không có quan sát, một lĩnh vực quan trọng trong Trí tuệ nhân tạo giúp xử lý sự không chắc chắn. Đồng thời, nhóm cũng đã hệ thống hóa và làm rõ ý tưởng cơ bản của các thuật toán tìm kiếm tiêu biểu bao gồm Thuật toán quay lui, Tìm kiếm theo chiều rộng, Thuật toán A sao, và kỹ thuật Truyền bá ràng buộc.

Về mặt ứng dụng, nhóm đã phân tích thành công bài toán Game Flow Free và mô hình hóa nó dưới dạng một bài toán tìm kiếm không gian trạng thái. Một sản phẩm Game Flow Free hoàn chỉnh với giao diện đồ họa đã được xây dựng bằng ngôn ngữ Python và thư viện Tkinker. Các thuật toán tìm kiếm đã lựa chọn đã được hiện thực hóa và tích hợp vào game, cho phép tự động tìm lời giải cho các màn chơi.

Thông qua quá trình thực nghiệm và đánh giá, nhóm đã thu thập được những dữ liệu cụ thể về hiệu năng của từng thuật toán trên các màn chơi Flow Free có độ khó khác nhau. Kết quả cho thấy sự vượt trội của A sao trong nhiều trường hợp, hoặc những ưu điểm cụ thể của CP. Những phân tích này không chỉ giúp hiểu rõ hơn về đặc tính của từng thuật toán mà còn cung cấp cơ sở để lựa chọn phương pháp giải quyết vấn đề phù hợp trong các bài toán tương tự. Quan trọng hơn, quá trình thực hiện đề tài đã giúp các thành viên trong nhóm củng cố kiến thức chuyên môn, rèn luyện kỹ năng lập trình, phân tích vấn đề và làm việc nhóm.

5.2. Hạn chế của đề tài

Mặc dù đã nỗ lực để hoàn thành các mục tiêu đề ra, đề tài vẫn còn một số hạn chế nhất định.

Thứ nhất, giao diện người dùng của game được thiết kế ở mức độ cơ bản, chủ yếu tập trung vào chức năng chính mà chưa có nhiều yếu tố đồ họa hấp dẫn hay các tính năng nâng cao như lưu màn chơi, gợi ý cho người dùng.

Thứ hai, số lượng và sự đa dạng của các màn chơi thử nghiệm có thể chưa đủ lớn để đưa ra những kết luận mang tính tổng quát cao về hiệu năng của các thuật toán trên mọi tình huống.

Thứ ba, việc đánh giá hiệu năng chủ yếu dựa trên thời gian thực thi và số nút duyệt, trong khi các yếu tố khác như mức độ tiêu thụ bộ nhớ chưa được đo lường và phân tích một cách chi tiết. Thứ tư, đối với thuật toán A sao, việc thiết kế và lựa chọn hàm heuristic có thể chưa phải là tối ưu nhất; việc thử nghiệm thêm nhiều biến thể heuristic khác nhau có thể mang lại những cải thiện đáng kể. Cuối cùng, mã nguồn hiện thực hóa các thuật toán, dù đã được kiểm tra, vẫn có thể còn những điểm chưa được tối ưu hoàn toàn về mặt tốc độ hoặc cấu trúc.

5.3. Hướng phát triển

Từ những kết quả đạt được và những hạn chế đã nhận diện, có một số hướng phát triển tiềm năng cho đề tài trong tương lai.

Một hướng quan trọng là cải thiện và nâng cấp sản phẩm Game Flow Free. Điều này bao gồm việc phát triển giao diện đồ họa phong phú và thân thiện hơn, bổ sung các tính năng tiện ích cho người dùng như khả năng tự tạo màn chơi, lưu và tải lại tiến trình, cung cấp gợi ý khi người chơi gặp khó khăn, hoặc hỗ trợ các biến thể phức tạp hơn của trò chơi như Flow Free Bridges hay Flow Free Hexes.

Về mặt thuật toán, có thể nghiên cứu và áp dụng thêm các thuật toán tìm kiếm nâng cao hoặc các kỹ thuật tối ưu hóa khác. Ví dụ, có thể thử nghiệm các biến thể của A sao như IDA sao (Iterative Deepening A star), hoặc các thuật toán tìm kiếm cục bộ tiên tiến hơn nếu bài toán được điều chỉnh phù hợp. Việc nghiên cứu sâu hơn về thiết kế các hàm heuristic hiệu quả hơn cho Flow Free cũng là một hướng đi hứa hẹn. Đối với phương pháp Truyền bá ràng buộc, có thể khám phá các thuật toán nhất quán cung mạnh mẽ hơn.

Ngoài ra, có thể mở rộng phạm vi nghiên cứu bằng cách phân tích sâu hơn về độ phức tạp lý thuyết của bài toán Flow Free. Một hướng đi thú vị khác là áp dụng các kỹ thuật học máy, ví dụ như học tăng cường, để huấn luyện một tác nhân tự động học cách giải Game Flow Free hoặc thậm chí là học cách tạo ra các hàm heuristic tốt. Cuối cùng, việc tìm kiếm và áp dụng lý thuyết về tìm kiếm không có quan sát vào các bài toán thực tế khác, ngoài các ví dụ minh họa, cũng là một hướng phát triển có giá trị.

TÀI LIỆU THAM KHẢO

1. Russell, S. J., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Nhà xuất bản (Pearson).
2. Đề cương chi tiết môn học Trí Tuệ Nhân Tạo, Khoa Công nghệ Thông tin, Trường Đại học Sư phạm Kỹ thuật TP. Hồ Chí Minh.