TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM

KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN CUỐI KỲ

Môn học: Trí Tuệ Nhân Tạo

Đề tài:

Thiết kế và xây dựng game N-puzzle sử dụng thuật toán AI

GVHD: PGS.TS. Hoàng Văn Dũng

MÃ MÔN HỌC: ARIN330585_23_1_03CLC

Danh sách sinh viên thực hiện

Mã số SV	Họ và tên	Mức độ đóng góp (%)
21110221	Võ Chí Khương	100%
21110282	Nguyễn Khánh Quy	100%
21110166	Nguyễn Hồng Thông Điệp	100%

Thành phố Hồ Chí Minh

Tháng 11 Năm 2023

Lời Cảm Ơn.

Chân thành cảm ơn thầy PGS.TS. Hoàng Văn Dũng đã dành thời gian và tâm huyết để phụ trách bộ môn "Trí tuệ nhân tạo". Chúng em xin bày tỏ lòng biết ơn sâu sắc đối với sự hướng dẫn và hỗ trợ mà thầy đã mang lại trong suốt học kỳ vừa qua.

Thầy không chỉ là người giáo viên mà còn là người đồng hành tận tâm, luôn sẵn sàng chia sẻ kiến thức chuyên sâu và kinh nghiệm thực tế. Những góp ý và lời khuyên của thầy đã giúp chúng em hiểu rõ hơn về lĩnh vực trí tuệ nhân tạo, từ đó làm giàu thêm kiến thức và kỹ năng của chúng em.

Chúng em xin bày tỏ lòng biết ơn với sự hỗ trợ mà thầy đã mang lại cho đề tài của nhóm chúng em. Thầy không chỉ giúp định hình rõ ràng hơn về đề tài mà còn đưa ra những hướng đi và giải pháp khả thi. Nhờ vào sự hướng dẫn tận tâm của thầy mà nhóm chúng em đã có cơ hội thực hiện đề tài một cách có hiệu quả và chất lượng.

Tuy nhiên cũng không thể tránh khỏi những sai sót đáng tiếc sẽ xảy ra, hay những hạn chế trong quá trình thực hiện, mong thầy đóng góp ý kiến, bổ sung kiến thức, kinh nghiệm và sửa chữa để bài báo cáo cuối kỳ của nhóm được hoàn thiện hơn.

Nhóm sinh viên thực hiện:

Nhóm trưởng:

Nguyễn Khánh Quy

Thành viên:

Nguyễn Hồng Thông Điệp

Võ Chí Khương

Lời Mở Đầu.

Trong thế giới hiện đại, Trí tuệ nhân tạo (AI) đã trở thành một trụ cột quan trọng không thể thiếu, hiện diện mạnh mẽ trong mọi khía cạnh của cuộc sống. AI không chỉ xuất hiện ở mọi nơi mà còn đảm nhận đa dạng các nhiệm vụ, từ việc dịch ngôn ngữ, công nghệ nhận dạng hình ảnh và tự động lái xe cho đến những trợ lý ảo thông minh, các ứng dụng y tế tiên tiến, ...

Các thuật toán tìm kiếm đóng vai trò quan trọng và không thể phủ nhận trong lĩnh vực Trí tuệ nhân tạo (AI) Các thuật toán tìm kiếm được áp dụng để giải quyết các vấn đề tìm kiếm thông tin và tối ưu hóa, và chúng đóng một vai trò quan trọng trong việc giải quyết các bài toán phức tạp.

Nhóm em xin được lựa chọn đề tài "thiết kế và cài đặt game N-puzzle sử dụng thuật toán AI" một phiên bản mở rộng của Eight-puzzle, một trong những bài toán tiêu biểu về lĩnh vực trí tuệ nhân tạo và các thuật toán tìm kiếm, để tổng hợp lại và áp dụng những kiến thức, thuật toán đã được học, cũng như tạo cơ hội để tìm hiểu về các thuật toán tìm kiếm khác, tạo ra một cơ hội để hiểu rõ hơn về sức mạnh và ứng dụng của AI trong việc giải quyết các vấn đề thực tế.

Mục Lục.

Lời Cảm Ơn. 2
Lời Mở Đầu.....3
Mục Lục.....4
Chương 1. Giới thiệu về đề tài. 6
1.1. Giới thiệu về bài toán. 6
1.2. Mục tiêu và yêu cầu của đề tài....6
1.3. Đối tượng và phạm vi nghiên cứu....7
1.3.1. Đối tượng nghiên cứu....7
1.3.2. Phạm vi nghiên cứu. 7
1.3.3. Cơ sở lý thuyết. 7
Chương 2. Cơ sở lý thuyết. 8
2.1. Tìm hiểu cụ thể về bài toán N-puzzle. 8
2.1.1. Các đặc điểm và yêu cầu của bài toán. 8
2.1.2. Các trạng thái có thể có của bảng. 8
2.2. Tìm hiểu về các thuật toán tìm kiếm. 8
2.2.1. BFS. 8
2.2.2. DFS....9
2.2.3. DLS....9
2.2.4. UCS. 10
2.2.5. ID....11
2.2.6. Greedy....11
2.2.7. A-star. 12
2.2.8. IDA-Star. 13
2.2.9. Hill Climbing. 13
2.2.10. Beam Search. 14
2.2.11. Ý tưởng giải thuật. 14
Chương 3. Phân tích và thiết kế. 15
3.1. Thiết kế giao diện. 15

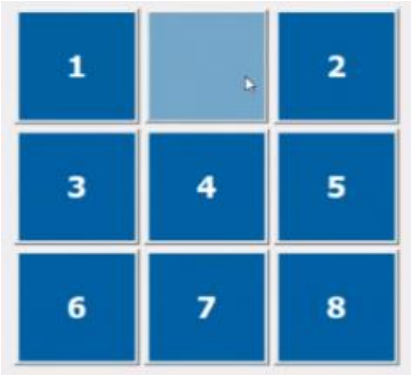
3.1.1. Giao diện chính. 15
3.1.2. Giao diện sau khi tạo puzzle. 15
3.1.3. Giao diện hướng dẫn giải và lịch sử. 15
3.2. Tính năng mở rộng. 15
3.3. Ý tưởng mở rộng. 15
Chương 4. Thực nghiệm và đánh giá. 16
4.1. Giao diện Menu chính. 16
4.1.1. Thông tin. 16
4.1.2. Kích thước hàng và kích thước cột. 16
4.1.3. Tạo Puzzle và thoát. 16
4.2. Giao diện khi tạo Puzzle. 16
4.2.1. Thông tin. 16
4.2.2. Tải ảnh. 17
4.2.3. Hiện ảnh gợi ý. 18
4.2.4. Thay đổi kích thước. 18
4.2.5. Xáo trộn. 18
4.2.6. Đặt lại. 18
4.2.7. Lưu trạng thái. 18
4.2.8. Hướng dẫn giải. 18
4.2.9. Chọn thuật toán. 19
4.2.10. Lịch sử. 20
4.2.11. Giải. 21
4.2.12. Tăng tốc. 21
4.2.13. Thoát. 21
4.3. Đánh giá. 21
Chương 5. Kết Luận. 22
Tài Liệu Tham Khảo. 23
Bảng Phân Chia Công Việc. 24

Chương 1. Giới thiệu về đề tài.

1.1. Giới thiệu về bài toán.

Bài toán Eight-puzzle, hay còn được gọi là bài toán "Taquin" hoặc "Sliding Puzzle," là một trong những bài toán kinh điển trong lĩnh vực trí tuệ nhân tạo và thách thức giải quyết vấn đề trong thế giới thực. Bài toán này thường được sử dụng để minh họa các thuật toán tìm kiếm và giải quyết vấn đề trong lĩnh vực trí tuệ nhân tạo. Bài toán Eight-puzzle gồm một bảng ô vuông kích thước 3x3, có tám ô được đánh số từ 1 tới 8 và một ô trống. Trạng thái ban đầu, các ô được sắp xếp một cách ngẫu nhiên, người chơi có thể di chuyển ô trống để đạt được trạng thái kết quả mong muốn. Trong quá trình giải bài toán, tại mỗi bước, chỉ có ô trống là được di chuyển, như vậy, tối đa một ô trống có thể có 4 khả năng di chuyển (lên trên, xuống dưới, sang trái, sang phải)

Bài toán N-puzzle là một phiên bản mở rộng của Eight-puzzle, chúng ta có thể tùy ý lựa chọn kích thước của bảng, 2x2, 3x3, 3x4, 4x4,...



Hình 1. 8 Puzzle.

1.2. Mục tiêu và yêu cầu của đề tài.

Mục tiêu của đề tài là xây dựng được chương trình giải quyết bài toán N-puzzle thông qua việc áp dụng các thuật toán tìm kiếm, nâng cao hiệu suất và mở rộng chương trình, đưa ra nhiều thuật toán hơn, nhiều thiên hướng phát triển hơn trong tương lai.

Và để làm được điều đó ta cần phải tìm hiểu về các khái niệm cơ bản của trí tuệ nhân tạo, đặc biệt là trong lĩnh vực tìm kiếm và giải quyết vấn đề, tìm hiểu về các thuật toán tìm kiếm như: BFS, USC, Greedy, A-Star,... Ngoài ra cần phải nắm vững các khái niệm lý thuyết liên quan đến bài toán N-puzzle, như heuristic, tìm kiếm thông minh, và các chiến lược giải quyết vấn đề, đánh giá và so sánh hiệu suất của chương trình.

1.3. Đối tượng và phạm vi nghiên cứu.

1.3.1. Đối tượng nghiên cứu.

- Bài toán Eight-puzzle.
- Các thuật toán tìm kiếm.

1.3.2. Phạm vi nghiên cứu.

- Cách tìm kiếm lời giải cho các thuật toán tìm kiếm đối với N-puzzle.
- So sánh hiệu suất của các thuật toán và đưa ra tiêu chí đánh giá.
- Phân tích các thiên hướng có thể mở rộng cho chương trình.

1.3.3. Cơ sở lý thuyết.

- Tổng hợp lại các thuật toán đã được học trên lớp, tìm hiểu thêm một số thuật toán khác thông qua tìm kiếm thông tin trên internet.
- IDE: Visual Studio Code.
- Ngôn Ngữ: Python.

Chương 2. Cơ sở lý thuyết.

2.1. Tìm hiểu cụ thể về bài toán N-puzzle.

2.1.1. Các đặc điểm và yêu cầu của bài toán.

- Kích thước bảng là n*m (với n và m là 2, 3, 4, 5).
- Ô trống mặc định sẽ là ô số 0.
- Có trạng thái bắt đầu và trạng thái đích.
- Di chuyển các ô vào vị trí của ô trống vào mỗi lượt.

2.1.2. Các trạng thái có thể có của bảng.

- Trạng thái xuất phát.
- Trạng thái đích.

2.2. Tìm hiểu về các thuật toán tìm kiếm.

2.2.1. BFS.

Giới Thiệu Chung.

Breadth First Search (BFS) là một trong những thuật toán tìm kiếm cơ bản và thiết yếu của đồ thị. Thuật toán tìm đường đi từ đỉnh xuất phát đến đỉnh kết thúc bằng cách duyệt theo chiều rộng.

Ưu điểm:

- ✓ Dễ cài đặt.
- ✓ Nếu số đỉnh là hữu hạn và số đỉnh nhỏ thì thuật toán sẽ tìm ra kết quả nhanh chóng.

Nhược điểm:

- ✓ Vì duyệt qua hầu hết các đỉnh nên nó mang tính chất vét cạn, không nên áp dụng nếu duyệt qua số đỉnh quá lớn.
- ✓ Chính vì là tìm kiếm mù, không chú ý đến thông tin đường đi do đến dẫn đến duyệt qua các đỉnh một cách mù quáng và gây tốn thời gian, bộ nhớ.

Ý tưởng thuật toán.

Lưu các đỉnh kề thành 1 danh sách và lấy từng phần tử trong danh sách các đỉnh kề ra để xét. Khi thêm một phần tử đỉnh kề vào danh sách thì phần tử đó sẽ được thêm ở cuối danh sách, còn lấy phần tử ra thì sẽ lấy ra ở đầu danh sách. Danh sách hoạt động như thế này còn được gọi là hàng đợi (queue).

2.2.2. DFS

Giới thiệu về thuật toán.

Depth First Search (DFS) là một thuật toán tìm kiếm trong đồ thị, cũng như BFS, nhưng khác biệt ở chỗ thuật toán này duyệt theo chiều sâu. DFS bắt đầu từ một đỉnh xuất phát và tiếp tục duyệt sâu vào các đỉnh kề trước khi quay lại duyệt các đỉnh khác.

Ưu điểm:

- ✓ Dùng ít bộ nhớ hơn so với BFS.
- ✓ Thích hợp cho việc kiểm tra các thành phần liên thông trong đồ thị.

Nhược điểm:

- ✓ Không đảm bảo tìm ra đường đi ngắn nhất.
- ✓ Nếu đồ thị lớn và không có hạn chế về số đỉnh, thuật toán có thể tiêu tốn thời gian lớn.

Ý tưởng thuật toán.

Sử dụng đệ quy hoặc ngăn xếp để duyệt qua các đỉnh và lưu trữ thông tin về các đỉnh đã đi qua. Khi gặp một đỉnh chưa được duyệt, tiếp tục đệ quy hoặc thêm đỉnh đó vào ngăn xếp và duyệt tiếp theo chiều sâu.

2.2.3. DLS.

Giới thiệu chung.

Depth-Limited Search (DLS) là một thuật toán tìm kiếm trong đồ thị, thuộc nhóm thuật toán tìm kiếm mù. Ngược lại với BFS, DLS tập trung vào việc theo đuổi chiều sâu theo một đường đi cụ thể và giới hạn độ sâu của đường đi này.

Ưu điểm:

- ✓ Phù hợp khi chỉ quan tâm đến các đường đi có độ sâu nhỏ.
- ✓ Tiết kiệm bộ nhớ hơn so với BFS vì chỉ duyệt qua một số đỉnh có giới hạn độ sâu.

Nhược điểm:

- ✓ Không đảm bảo tìm kiếm được đường đi ngắn nhất nếu độ sâu giới hạn quá nhỏ.
- ✓ Có thể bỏ qua đường đi tối ưu nếu giới hạn độ sâu không đủ lớn.

Ý tưởng giải thuật.

DLS sử dụng cơ chế giới hạn độ sâu để kiểm soát quá trình tìm kiếm. Thuật toán duyệt qua các đỉnh theo chiều sâu, bắt đầu từ đỉnh xuất phát, và tiếp tục theo đường đi cho đến khi đạt đến độ sâu giới hạn được xác định.

2.2.4. UCS.

Giới thiệu chung.

Thuật toán Uniform Cost Search (UCS) là một thuật toán tìm kiếm trong đồ thị, thuộc nhóm thuật toán tìm kiếm thông minh. UCS tập trung vào việc tìm kiếm đường đi có chi phí thấp nhất từ đỉnh xuất phát đến đỉnh kết thúc. UCS thường được sử dụng khi cần tìm kiếm đường đi ngắn nhất trong đồ thị.

Ưu điểm:

- ✓ Tìm được đường đi ngắn nhất.
- ✓ Không bỏ qua đường đi tối ưu.

Nhược điểm:

- ✓ Tốn nhiều bộ nhớ hơn so với DLS và BFS vì phải lưu trữ các đỉnh đã xét qua.

Ý tưởng giải thuật.

UCS sử dụng cơ chế tìm kiếm đường đi có chi phí thấp nhất để kiểm soát quá trình tìm kiếm. Thuật toán duyệt qua các đỉnh theo chiều rộng, bắt đầu từ đỉnh xuất phát, và tiếp tục theo đường đi có chi phí thấp nhất cho đến khi đạt đến đỉnh kết thúc. UCS có thể được triển khai bằng cách sử dụng hàng đợi ưu tiên để duyệt qua các đỉnh. Khi đạt đến đỉnh kết thúc, thuật toán sẽ trả về đường đi có chi phí thấp nhất.

2.2.5. ID.

Giới thiệu chung.

Iterative Deepening (ID) là một thuật toán tìm kiếm trong đồ thị, được sử dụng để tìm kiếm đường đi từ đỉnh xuất phát đến đỉnh kết thúc. Đây là một thuật toán kết hợp giữa chiến lược tìm kiếm theo chiều sâu (DFS) và tìm kiếm theo chiều rộng (BFS).

Ưu điểm:

- ✓ Đảm bảo tìm kiếm đường đi ngắn nhất: ID đảm bảo rằng nếu có đường đi từ đỉnh xuất phát đến đỉnh kết thúc, thuật toán sẽ tìm ra đường đi ngắn nhất.
- ✓ Dễ cài đặt: Tương tự như BFS, ID cũng là một thuật toán dễ cài đặt.

Nhược điểm:

- ✓ Tốn thời gian: ID có thể tốn nhiều thời gian hơn so với một số thuật toán tìm kiếm thông minh khác, đặc biệt là trên các đồ thị lớn.

Ý tưởng giải thuật.

ID thực hiện tìm kiếm theo chiều sâu (DFS) với giới hạn độ sâu tăng dần ở mỗi vòng lặp. Ở mỗi vòng lặp, nó duyệt qua đỉnh và cạnh trong đồ thị đến khi đạt được giới hạn độ sâu cho vòng lặp đó. Nếu đỉnh đích không được tìm thấy, tăng giới hạn độ sâu và thực hiện vòng lặp tiếp theo.

2.2.6. Greedy.

Giới thiệu chung.

Thuật toán Greedy là một thuật toán tìm kiếm thông minh trong đồ thị. Thuật toán này xây dựng lời giải từng bước một, luôn chọn bước tiếp theo mang lại lợi ích cục bộ lớn

nhất. Vì vậy, các bài toán mà việc chọn lựa tối ưu cục bộ cũng đồng thời dẫn đến lời giải toàn cục là những bài toán phù hợp với Greedy.

Ưu điểm:

- ✓ Thuật toán này có thể được triển khai bằng cách sử dụng hàng đợi ưu tiên để duyệt qua các đỉnh. Khi đạt đến đỉnh kết thúc, thuật toán sẽ trả về lời giải tối ưu cục bộ.

Nhược điểm:

- ✓ Không đảm bảo tìm được lời giải tối ưu toàn cục.
- ✓ Có thể bỏ qua lời giải tối ưu nếu chiến lược tối ưu cục bộ không dẫn đến lời giải tối ưu toàn cục

Ý tưởng giải thuật.

Quá trình biến đổi của thuật toán được thực hiện tương tự như BFS, nhưng lưu trữ dữ liệu theo cơ chế của hàng đợi ưu tiên.

2.2.7. A-star.

Giới thiệu chung.

Thuật toán A* là một thuật toán tìm kiếm thông minh trong đồ thị. Thuật toán này sử dụng hàm heuristic để đánh giá chi phí còn lại để đến được đích. A* tập trung vào việc tìm kiếm đường đi ngắn nhất từ đỉnh xuất phát đến đỉnh kết thúc.

Ưu điểm:

- ✓ Thuật toán sẽ trả về đường đi có chi phí thấp nhất.

Nhược điểm:

- ✓ Không đảm bảo tìm được lời giải tối ưu toàn cục.
- ✓ Có thể bỏ qua lời giải tối ưu nếu nếu hàm heuristic không được thiết kế tốt.

Ý tưởng giải thuật.

Quá trình biến đổi của thuật toán được thực hiện tương tự như BFS, nhưng lưu trữ dữ liệu theo cơ chế của hàng đợi ưu tiên.

2.2.8. IDA-Star.

Giới thiệu chung.

Thuật toán Iterative Deepening A* (IDA*) là một thuật toán tìm kiếm thông minh trong đồ thị. Thuật toán này là sự kết hợp của A* và DFS một phương pháp tìm kiếm đường đi ngắn nhất trong đồ thị có trọng số giữa một đỉnh xuất phát và một tập hợp các đỉnh đích. Nó là một dạng của tìm kiếm độ sâu trước với độ sâu giới hạn được tăng dần theo từng lần lặp.

Ưu điểm:

- ✓ Thuật toán IDA* có thể được triển khai bằng cách sử dụng đệ quy hoặc stack để duyệt qua các đỉnh. Khi đạt đến đỉnh kết thúc, thuật toán sẽ trả về đường đi có chi phí thấp nhất.

Nhược điểm:

- ✓ không đảm bảo tìm được lời giải tối ưu toàn cục và có thể bỏ qua lời giải tối ưu nếu độ sâu giới hạn quá nhỏ.

Ý tưởng giải thuật.

Quá trình biến đổi của thuật toán được thực hiện tương tự như tìm kiếm độ sâu trước, nhưng lưu trữ dữ liệu theo cơ chế của stack hoặc đệ quy.

2.2.9. Hill Climbing.

Giới thiệu chung.

Thuật toán Hill Climbing là một thuật toán tìm kiếm cục bộ liên tục theo hướng tăng dần độ cao/giá trị. Thuật toán này bắt đầu từ một giải pháp ngẫu nhiên và tiến hóa bằng cách di chuyển đến các giải pháp láng giềng tốt hơn. Thuật toán này dừng lại khi không có giải pháp láng giềng nào tốt hơn giải pháp hiện tại.

Ưu điểm:

- ✓ Hill Climbing có thể được sử dụng để giải quyết các bài toán tối ưu hóa đơn giản.

Nhược điểm:

- ✓ Hill Climbing có một số hạn chế, bao gồm khả năng bị sa lầy ở những cực đại cục bộ và không tìm được lời giải tối ưu toàn cục.

Ý tưởng giải thuật.

Hill Climbing đang tìm kiếm giải pháp tối ưu cho bài toán, bắt đầu từ giải pháp ngẫu nhiên và tiếp tục đến giải pháp kết thúc, tiếp tục như vậy cho đến khi tìm ra giải pháp tối ưu cục bộ hoặc không còn giải pháp láng giềng để xét.

2.2.10.Beam Search.

Thuật toán Beam Search là một thuật toán tìm kiếm trong không gian trạng thái để giải quyết các bài toán tìm kiếm và tối ưu hóa. Beam Search mở rộng tập các trạng thái được duyệt theo một số lượng k cố định các trạng thái tốt nhất tại mỗi bước lặp.

Ưu điểm:

- ✓ Beam Search thường cho kết quả tốt hơn so với thuật toán tìm kiếm theo chiều rộng (BFS - Breadth-First Search) và tiết kiệm bộ nhớ hơn thuật toán tìm kiếm theo chiều sâu (DFS - Depth-First Search).

Nhược điểm:

- ✓ Beam Search có thể bỏ qua các lời giải tốt nhất nếu chúng không nằm trong tập hạn chế của các trạng thái tốt nhất.

2.2.11.Ý tưởng giải thuật.

Beam Search bắt đầu từ trạng thái ban đầu và mở rộng ra k trạng thái con tốt nhất dựa trên một hàm đánh giá. Các trạng thái con này tiếp tục được mở rộng theo cùng nguyên tắc, tạo ra một bộ các trạng thái mới, và quá trình này tiếp tục cho đến khi đạt được trạng thái đích hoặc điều kiện dừng được đưa ra.

Chương 3. Phân tích và thiết kế.

3.1. Thiết kế giao diện.

3.1.1. Giao diện chính.

- Thông tin nhóm sinh viên thực hiện.
- Tên đề tài.
- Các nút thực hiện chức năng (tạo puzzle, Thoát,...).

3.1.2. Giao diện sau khi tạo puzzle.

- Bảng trò chơi.
- Các thông số liên quan đến bài toàn (số bước, thời gian, số đỉnh,...).
- Các nút thực hiện chức năng (thay đổi kích thước bảng, tải hình ảnh, khởi tạo puzzle, sinh ra puzzle ngẫu nhiên, các thuật toán khác nhau để lựa chọn, lịch sử những lần giải trước đó, hướng dẫn giải...).

3.1.3. Giao diện hướng dẫn giải và lịch sử.

- Màn hình lịch sử sẽ ghi lại những lần giải bài toán trước đó (gồm thuật toán nào, số bước, thời gian, số đỉnh).
- Hướng dẫn giải sẽ ghi lại những nước đi ở 2 dạng là dạng số hoặc chỉ dẫn dạng hướng.

3.2. Tính năng mở rộng

- Ghi nhớ cách giải của trạng thái đã từng giải qua của mỗi thuật toán.
- Hiện lịch sử giải của từng trạng thái.
- Hiện hướng dẫn giải chi tiết của từng thuật toán.

3.3. Ý tưởng mở rộng.

- Cài đặt thêm thuật toán.
- Thêm tính năng nhập trạng thái đầu và trạng thái đích.
- Thêm tính năng giới hạn thời gian để thử thách người chơi.

Chương 4. Thực nghiệm và đánh giá.

4.1. Giao diện Menu chính.

4.1.1. Thông tin.

- Số thứ tự của nhóm.
- Tên các thành viên nhóm.
- Tên đề tài.

Hình 2. Giao diện menu chính

4.1.2. Kích thước hàng và kích thước cột.

- Cả hàng và cột đề có thể chọn từ 2 đến 5 để quy định kích thước của Puzzle.

4.1.3. Tạo Puzzle và thoát.

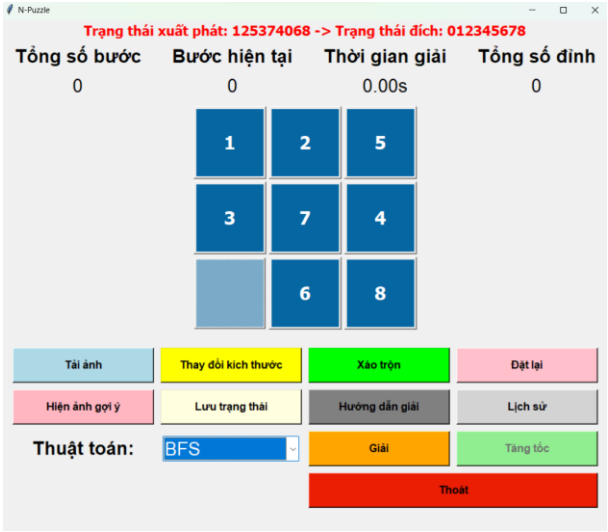
- Nút tạo Puzzle để tạo ra puzzle sau khi chọn kích thước của hàng và cột
- Nút thoát để thoát khỏi chương trình.

4.2. Giao diện khi tạo Puzzle.

4.2.1. Thông tin.

- Dòng trạng thái, bao gồm trạng thái xuất phát và trạng thái đích.
- Các thông tin của trò chơi như tổng số bước, số bước hiện tại, thời gian giải, tổng số đỉnh đã đi qua.

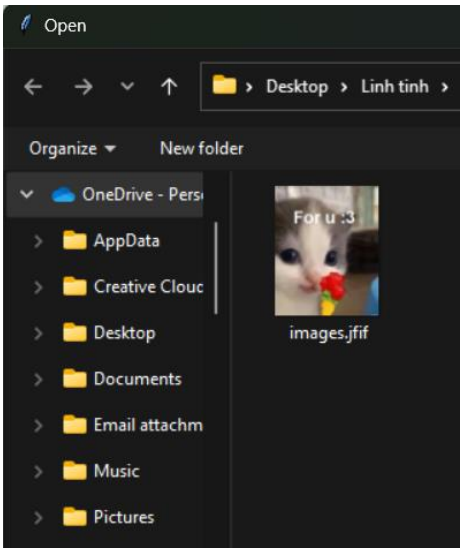
- Bảng Puzzle.

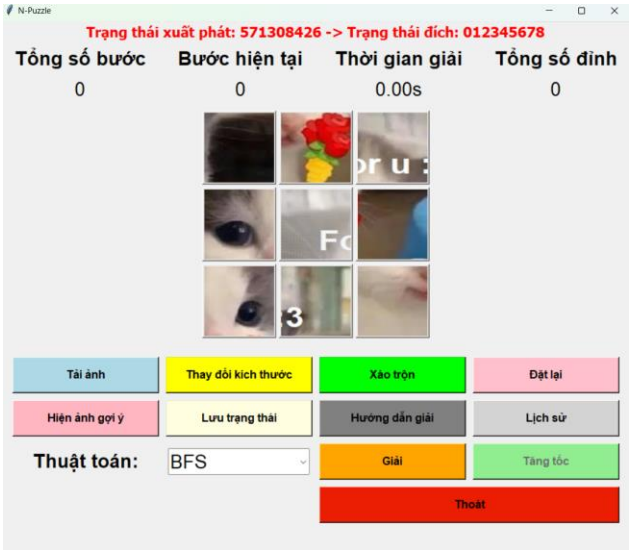


Hình 3. Giao diện khi đã tạo Puzzle.

4.2.2. Tải ảnh.

- Nhấn vào nút tải ảnh, sau đó chọn ảnh chúng ta muốn tải, chương trình sẽ tự động chèn và xáo trộn puzzle có ảnh vừa chọn.

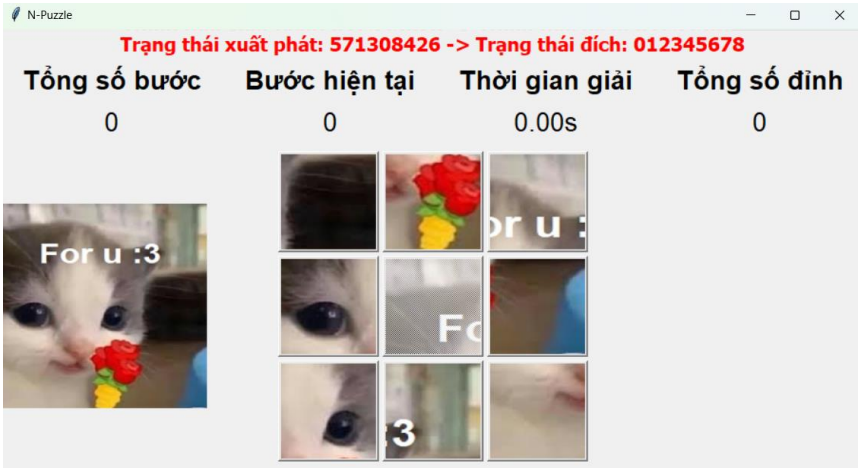




Hình 4. và 5. Chức năng tải ảnh.

4.2.3. Hiện ảnh gợi ý.

- Hiển thị ảnh để gợi ý ảnh gốc.



Hình 6. Hiện ảnh gợi ý.

4.2.4. Thay đổi kích thước.

- Thay đổi kích thước Puzzle.

4.2.5. Xáo trộn.

- Xáo trộn ảnh để tạo ra trạng thái xuất phát ngẫu nhiên.

4.2.6. Đặt lại.

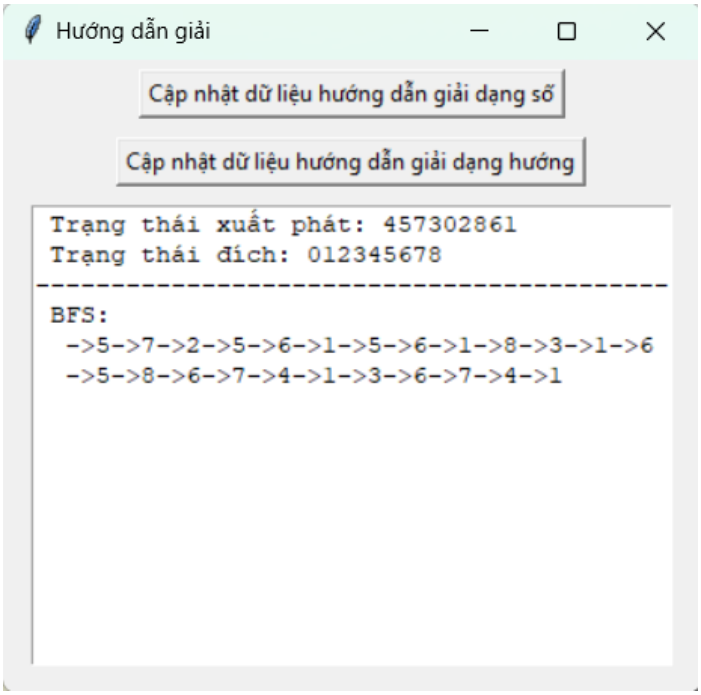
- Giúp quay trở lại trạng thái xuất phát.

4.2.7. Lưu trạng thái.

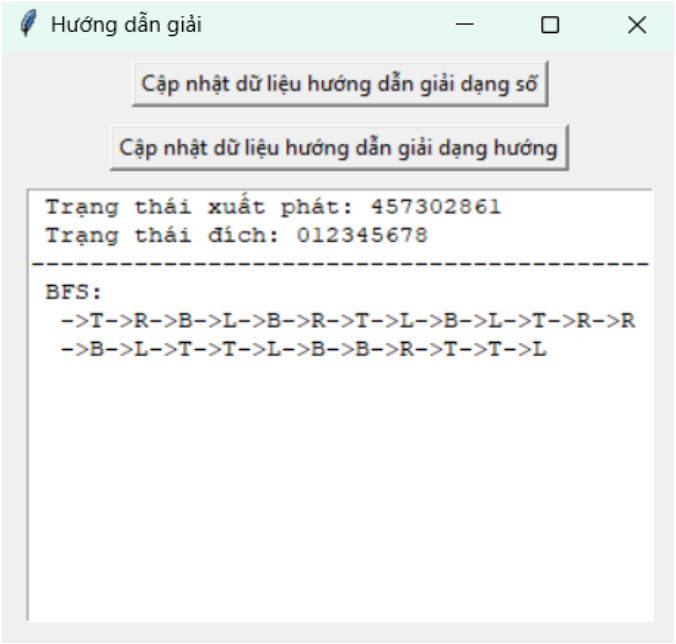
- Lưu trạng thái hiện tại thành trạng thái xuất phát mới.

4.2.8. Hướng dẫn giải.

- Sau khi giải chương trình sẽ lưu lại lời giải vào hướng dẫn giải để người dùng có thể xem cách giải của những chương trình đã được giải.



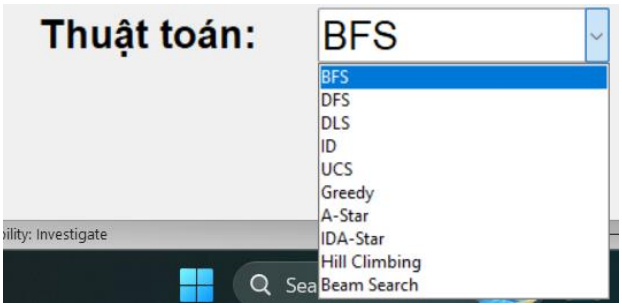
Hình 7-A. Hướng dẫn giải dạng số.



Hình 7-B. Hướng dẫn giải dạng kí tự.

4.2.9. Chọn thuật toán.

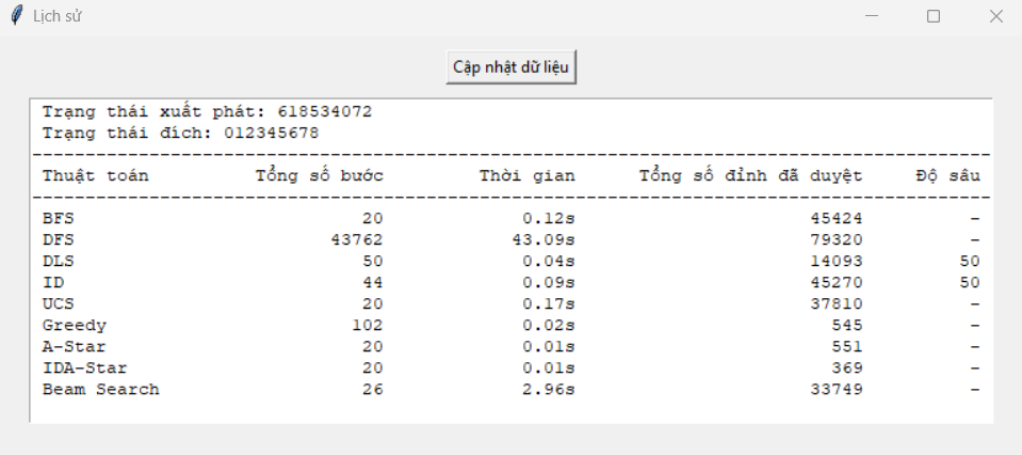
- Gồm 10 thuật toán để chọn lựa.



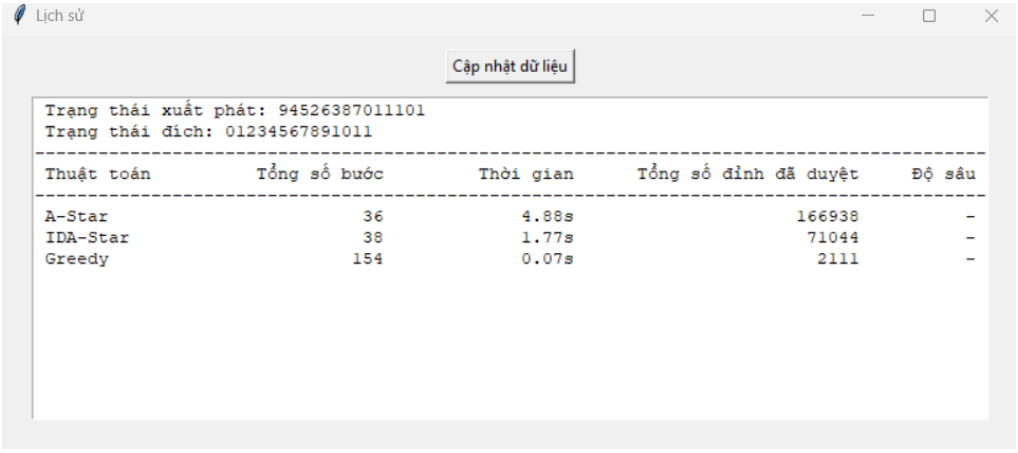
Hình 9. Thuật Toán.

4.2.10.Lịch sử.

- Lưu lại lịch sử của trò chơi.



Hình 8-A. Lịch sử 3x3.



Hình 8-B. Lịch sử 3x4.

4.2.11.Giải.

- Tiến hành giải Puzzle dựa trên thuật toán đã chọn.

4.2.12.Tăng tốc.

- Giúp tua qua nhanh và đưa ta đến kết quả nếu quá trình giải qua lâu và phải đi qua quá nhiều bước.

4.2.13.Thoát.

- Thoát khỏi chương trình.

4.3. Đánh giá.

- BFS, Thường giải nhanh trong trường hợp không gian trạng thái không quá lớn và không có quá nhiều trạng thái, tuy nhiên có thể tiêu tốn nhiều bộ nhớ và thời gian nếu không gian trạng thái lớn.
- DFS, không hiệu quả trong việc giải quyết 8 puzzle nếu không kiểm soát được chiều sâu tối ưu, có thể bị mắc kẹt trong các vòng lặp.
- DLS, có thể hiệu quả nếu được thực hiện một cách cẩn thận với việc kiểm soát độ sâu phù hợp. ID thường được ưu tiên hơn vì nó kết hợp cả DFS và DLS để tìm kiếm.
- UCS, là một phiên bản cải tiến của BFS, có thể hiệu quả hơn khi có trọng số trên các bước di chuyển.
- Greedy, A-Star, IDA-Star, thường có khả năng tìm kiếm tốt hơn so với BFS và DFS trong việc tối ưu hóa thời gian tìm kiếm, với A* thường được ưa chuộng nhất do sử dụng hàm heuristic để ước lượng chi phí còn lại đến mục tiêu.
- Hill Climbing và Beam Search, có thể tìm kiếm tốt trong không gian trạng thái nhỏ, nhưng có thể bị mắc kẹt.

Chương 5. Kết Luận.

Về cơ bản chương trình của nhóm em đã giải quyết được hết những yêu cầu đề ra. Đã tạo ra chương trình với bảng Puzzle có kích thước từ 4 đến 25 ô cùng nhiều chức năng tiện ích, áp dụng các thuật toán tìm kiếm trong chương trình học và tìm hiểu một số thuật toán khác để giải quyết bài toán n-puzzle. Đối với mỗi thuật toán nhóm đã tìm ra được thời gian chạy, tổng số bước, số bước ở hiện tại, số đỉnh đã đi qua, ghi lại lịch sử giải và hiển thị ra để người dùng theo dõi, giao diện còn có thể trực tiếp di chuyển để tạo ra trạng thái mong muốn, có hướng dẫn giải ở cả 2 dạng, số đối với Puzzle thông thường và chữ đối với dạng hình ảnh.

Định hướng phát triển chương trình sẽ là, tối ưu chương trình để chạy nhanh hơn, thiết kế giao diện bắt mắt và thân thiện với người dùng hơn, thêm một số tính năng tiện ích và hướng dẫn người dùng thực hiện, mở rộng quy mô của Puzzle, thêm nhiều thuật toán để giải hơn nữa, thêm phần so sánh giữa các thuật toán để đưa ra lựa chọn tối ưu.

Tài Liệu Tham Khảo.

BFS (Breadth-first search), VNOI, Nguyễn Châu Khanh, đường dẫn: https://vnoi.info/wiki/algo/graph-theory/breadth-first-search.md

Data Structure & Algorithm - Graph Algorithms - Depth First Search (DFS), VIBLO, Thái Thanh Hải, ngày 26 tháng 2 năm 2023, đường dẫn: https://viblo.asia/p/data-structure-algorithm-graph-algorithms-depth-first-search-dfs-qPoL7zyXJvk

[Algorithm] Các thuật toán tìm kiếm trong AI, Flinters, Bùi Quang Hà, ngày 19 tháng 11 năm 2020, đường dẫn: https://labs.flinters.vn/algorithm/algorithm-cac-thuat-toan-tim-kiem-trong-ai/

Các thuật toán cơ bản trong AI - Phân biệt Best First Search và Uniform Cost Search (UCS), VIBLO, Thơ Trần, ngày 13 tháng 1 năm 2019, đường dẫn: https://viblo.asia/p/cac-thuat-toan-co-ban-trong-ai-phan-biet-best-first-search-va-uniform-cost-search-ucs-Eb85omLWZ2G

Single Agent Search Video 6: IDA*, Nathan Sturtevant, ngày 20 tháng 1 năm 2021, đường dẫn: https://www.youtube.com/watch?v=kodg9NEKl0U

Hill Climbing Algorithm trong trí tuệ nhân tạo, websitehcm, Dục Đoàn Trình, ngày 26 tháng 3 năm 2022, đường dẫn: https://websitehcm.com/hill-climbing-algorithm-trong-tri-tue-nhan-tao/

[Cẩm nang AI] Các thuật toán tìm kiếm phổ biến trong AI, viettelidc, ngày 20 tháng 5 năm 2022, đường dẫn: https://viettelidc.com.vn/tin-tuc/cam-nang-ai-cac-thuat-toan-tim-kiem-pho-bien-trong-ai

Bảng Phân Chia Công Việc.

Họ và tên	Công việc	Mức độ hoàn thiện
Nguyễn Khánh Quy	Xây dựng giao diện và các tính năng mở rộng, cài đặt thuật toán BFS, DFS, DLS, ID.	100%
Nguyễn Hồng Thông Điệp	Cài đặt thuật toán UCS, Greedy, A-Star, kiểm tra lỗi và viết báo cáo.	100%
Võ Chí Khương	Cài đặt thuật toán IDA-Star, Hill Climbing, Beam Search, kiểm tra lỗi và làm bài trình chiếu.	100%