

BỘ GIAO THÔNG VẬN TẢI
HỌC VIỆN HÀNG KHÔNG VIỆT NAM
KHOA CÔNG NGHỆ THÔNG TIN



TIỂU LUẬN
(ĐÁNH GIÁ LÀM VIỆC NHÓM)
LẬP TRÌNH PYTHON
XÂY DỰNG TRÒ CHƠI N-PUZZLE

Giảng viên hướng dẫn: ThS.Hồ Văn Quý

Sinh viên/ Nhóm sinh viên thực hiện: Nhóm 05

Mã số sinh viên:

Lớp: 010100087202

Thành phố Hồ Chí Minh, tháng 12/2024

BỘ GIAO THÔNG VẬN TẢI
HỌC VIỆN HÀNG KHÔNG VIỆT NAM
KHOA CÔNG NGHỆ THÔNG TIN



TIỂU LUẬN
(ĐÁNH GIÁ LÀM VIỆC NHÓM)
LẬP TRÌNH PYTHON
XÂY DỰNG TRÒ CHƠI N-PUZZLE

Giảng viên hướng dẫn: ThS.Hồ Văn Quý

Sinh viên/ Nhóm sinh viên thực hiện: Nhóm 05

Mã số sinh viên:

Lớp: 010100087202

Thành phố Hồ Chí Minh, tháng 12/2024

Danh sách Nhóm:

STT	Họ và tên	MSSV	Lớp	Ghi chú
1	Đặng Võ Tùng Dương	2254810311	22ĐHTT02	Nhóm Trưởng
2	Đặng Thị Khánh Linh	2254810312	22ĐHTT02	
3	Lê Trần Bảo Quốc	2254810058	22ĐHTT02	
4	Trần Minh Quân	2254810055	22ĐHTT02	
5	Trịnh Thanh Minh	2254810061	22ĐHTT02	

Cán bộ chấm thi 1 <i>(ký và ghi rõ họ tên)</i>	Cán bộ chấm thi 2 <i>(ký và ghi rõ họ tên)</i>
Cán bộ chấm thi phúc khảo 1 <i>(ký và ghi rõ họ tên)</i>	Cán bộ chấm thi phúc khảo 2 <i>(ký và ghi rõ họ tên)</i>

DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT

Ký hiệu, chữ viết tắt	Chữ viết đầy đủ
BFS	Breadth First Search
DFS	Depth First Search
DLS	Depth – Limited Search
UCS	Uniform Cost Search
ID	Iterative Deepening
GUI	Graphical User Interface
OOP	Object-oriented programming

DANH MỤC HÌNH ẢNH

Hình 1.1. Eight-Puzzle.	1
Hình 2.1: Các bước duyệt thuật toán BFS.....	6
Hình 2.2: Các bước duyệt thuật toán DFS.....	6
Hình 2.4: Các bước duyệt thuật toán UCS	8
Hình 2.5: Các bước duyệt thuật toán ID.....	8
Hình 2.6: Các bước duyệt thuật toán Greedy	9
Hình 2.7: Các bước duyệt thuật toán A*	10
Hình 2.8: Các bước duyệt thuật toán IDA*	10
Hình 2.9: Các bước duyệt thuật toán Beam Search.....	11
Hình 3.1 – Biểu đồ use case	13
Hình 3.2. Giao diện chính Puzzle.....	14
Hình 3.3. Chức năng tải ảnh.....	14
Hình 3.4. Hiện ảnh gợi ý.....	15
Hình 3.5. Hướng dẫn giải dạng số.	16
Hình 3.6. Hướng dẫn giải dạng kí tự.....	17
Hình 3.7. Thuật Toán.....	17
Hình 3.8. Lịch sử 3x3.....	18
Hình 3.9. Lịch sử 3x4.....	18
Hình 3.10 Giao diện chọn BFS.....	20
Hình 3.11 Giao diện chọn DFS.....	20
Hình 3.12: Giao diện chọn DLS	21
Hình 3.13 Giao diện chọn UCS	22
Hình 3.14: Giao diện chọn ID	22
Hình 3.15: Giao diện chọn Greedy	23
Hình 3.16: Giao diện chọn A*	23
Hình 3.17 Giao diện chọn IDA*	24
Hình 3.18: Giao diện chọn Beam Search	24

MỤC LỤC

	trang
DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT.....	<i>i</i>
DANH MỤC HÌNH ẢNH.....	<i>ii</i>
MỞ ĐẦU.....	<i>v</i>
CHƯƠNG 1. GIỚI THIỆU.....	<i>1</i>
1.1. Lý do chọn đề tài.....	1
1.2. Mục tiêu đề tài.....	1
1.3. Phạm vi đề tài.....	2
1.4. Đối tượng nghiên cứu.....	2
1.5. Phương pháp nghiên cứu.....	3
1.6. Bố cục đề tài.....	3
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	<i>4</i>
2.1. Tìm hiểu cụ thể về bài toán N-puzzle.....	4
2.1.1. Các đặc điểm và yêu cầu của bài toán.....	4
2.1.2. Các trạng thái có thể có của bảng.....	4
2.2. Ngôn ngữ lập trình Python.....	5
2.3. Tìm hiểu về các thuật toán tìm kiếm.....	5
2.3.1. BFS.....	5
2.3.2. DFS.....	6
2.3.3. DLS.....	6
2.3.4. UCS.....	7
2.3.5. ID.....	8
2.3.6. Greedy.....	9
2.3.7. A-star.....	9
2.3.8. IDA-Star.....	10
2.3.9. Beam Search.....	10
2.4. Các thư viện hỗ trợ.....	11
2.4.1. Thư viện Tkinter.....	11
2.4.2. Thư viện Pillow.....	11
CHƯƠNG 3. PHÂN TÍCH HỆ THỐNG VÀ XÂY DỰNG SẢN PHẨM.....	<i>13</i>
3.1. Phân tích hệ thống.....	13
3.1.1. Giao diện tạo Puzzle.....	13
3.1.1.1. Thông tin.....	13
3.1.1.2. Tải ảnh.....	14

3.1.1.3. Hiện ảnh gợi ý.....	15
3.1.1.4. Thay đổi kích thước	15
3.1.1.5. Xáo trộn	15
3.1.1.6. Đặt lại	16
3.1.1.7. Lưu trạng thái.	16
3.1.1.8. Hướng dẫn giải.	16
3.1.1.9. Chọn thuật toán.....	17
3.1.1.10. Lịch sử.....	17
3.1.1.11. Giải	19
3.1.1.12. Tăng tốc.....	19
3.1.1.13. Thoát.	19
3.2. Giao diện các thuật toán tìm kiếm.....	19
3.2.1. Giao diện BFS.....	19
3.2.2. Giao diện DFS.....	20
3.2.3. Giao diện DLS	21
3.2.4. Giao diện UCS	21
3.2.5. Giao diện ID.....	22
3.2.6. Giao diện Greedy	23
3.2.7. Giao diện A-Star.....	23
3.2.8. Giao diện IDA-Star	24
3.2.9. Giao diện Beam Search.....	24
3.3. Đánh giá.	25
KẾT LUẬN.....	26
TÀI LIỆU THAM KHẢO	27

MỞ ĐẦU

Chân thành cảm ơn thầy ThS. Hồ Văn Quý đã dành thời gian và tâm huyết để phụ trách bộ môn "Lập trình Python ". 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.

Nhóm em xin được lựa chọn đề tài “thiết kế và cài đặt game N-puzzle sử dụng thuật toán” một phiên bản mở rộng của Eight-puzzle, chúng em không chỉ tập trung vào việc thiết kế và cài đặt trò chơi mà còn tìm hiểu sâu 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.

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.

CHƯƠNG 1. GIỚI THIỆU

1.1. Lý do chọn đề tài

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 3×3 , 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, 2×2 , 3×3 , 3×4 , 4×4 ,...



Hình 1.1. Eight-Puzzle.

1.2. Mục tiêu 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. Phạm vi đề tài

Đề tài sẽ nghiên cứu và áp dụng các thuật toán tìm kiếm như BFS, DFS, A-Star, IDA-Star, Greedy Search, và Uniform Cost Search để giải quyết bài toán N-puzzle. Mỗi thuật toán sẽ được phân tích về khả năng tìm kiếm đường đi ngắn nhất và sử dụng hiệu quả các chiến lược tìm kiếm như heuristic và mở rộng không gian tìm kiếm

Hiệu suất của các thuật toán sẽ được so sánh dựa trên thời gian thực hiện, số lượng bước di chuyển, và bộ nhớ sử dụng. Tiêu chí đánh giá sẽ được xây dựng để xác định thuật toán nào hiệu quả nhất cho từng kích thước và mức độ phức tạp của bài toán N-puzzle, giúp đưa ra những kết luận về sự phù hợp của từng thuật toán

Đề tài sẽ phân tích khả năng mở rộng chương trình như tối ưu hóa thuật toán, hỗ trợ các phiên bản N-puzzle lớn hơn, và áp dụng các phương pháp học máy để cải thiện hiệu suất. Ngoài ra, việc áp dụng các giải pháp từ N-puzzle vào các bài toán và trò chơi tương tự trong thực tế cũng sẽ được nghiên cứu.

1.4 Đối tượng nghiên cứu

Đây là một bài toán nổi tiếng trong lĩnh vực trí tuệ nhân tạo, có cấu trúc là một bảng 3x3 với các ô vuông được đánh số từ 1 đến 8 và một ô trống. Nhiệm vụ của người giải là di chuyển các ô vuông bằng cách sử dụng ô trống để đưa các ô về đúng vị trí mục tiêu đã xác định. Nghiên cứu tập trung vào cấu trúc của bài toán, cách giải quyết, và mở rộng bài toán Eight-puzzle lên phiên bản N-puzzle với các kích thước lớn hơn (4x4, 5x5,...).

Các thuật toán tìm kiếm là công cụ quan trọng để giải quyết bài toán N-puzzle. Đề tài sẽ tập trung vào việc nghiên cứu và áp dụng các thuật toán như BFS (tìm kiếm theo chiều rộng), DFS (tìm kiếm theo chiều sâu), A-Star, IDA-Star, Greedy Search, UCS (tìm kiếm theo chi phí đồng nhất), và các thuật toán khác. Mỗi thuật toán sẽ được nghiên cứu về

lý thuyết, cấu trúc, và cách thức hoạt động trong quá trình tìm kiếm lời giải cho bài toán N-puzzle.

1.5. Phương pháp nghiên cứu

Phương pháp nghiên cứu bao gồm việc thu thập và tìm hiểu tài liệu về bài toán N-puzzle và các thuật toán tìm kiếm như BFS, DFS, A-Star, Greedy Search và IDA-Star. Tiếp theo là phân tích và thiết kế các thuật toán dựa trên lý thuyết đã nghiên cứu, sau đó lập trình và cài đặt trong Python. Quá trình thực nghiệm được tiến hành để kiểm thử chương trình với các kích thước N-puzzle khác nhau, từ đó đo lường và so sánh hiệu suất của từng thuật toán. Cuối cùng, các kết quả được đánh giá dựa trên tiêu chí thời gian, bộ nhớ và số bước di chuyển để rút ra kết luận về tính hiệu quả của từng thuật toán.

1.6. Bố cục đề tài

Phần còn lại của báo cáo tiểu luận môn học này được tổ chức theo cấu trúc với các thành phần như sau:

Chương 2 của báo cáo trình bày lý thuyết cơ bản về bài toán N-Puzzle và các thuật toán tìm kiếm. Chương này trình bày các lý thuyết cơ bản liên quan đến bài toán N-Puzzle và các thuật toán tìm kiếm trong trí tuệ nhân tạo. Các thuật toán như BFS (Breadth-First Search), DFS (Depth-First Search), UCS (Uniform Cost Search), A-Star (A^*), Greedy, IDA-Star và Beam Search sẽ được giới thiệu, với các ưu nhược điểm của từng thuật toán. Chương cũng phân tích các ứng dụng của các thuật toán này trong việc giải quyết bài toán N-Puzzle, và cách thức chúng có thể áp dụng để tối ưu hóa quá trình tìm kiếm giải pháp.

Chương 3 của báo cáo trình bày hệ thống và xây dựng sản phẩm giải bài toán N-Puzzle. Chương này giới thiệu về thiết kế và triển khai hệ thống giải bài toán N-Puzzle sử dụng các thuật toán tìm kiếm trong trí tuệ nhân tạo. Các thành phần chính của hệ thống bao gồm giao diện người dùng (giao diện tạo Puzzle, tải ảnh, thay đổi kích thước, xáo trộn, lưu trạng thái, hướng dẫn giải), và các thuật toán tìm kiếm để giải quyết bài toán. Chương này mô tả cách thức thực hiện từng chức năng của hệ thống, bao gồm việc xây dựng giao diện người dùng, cách thức các thuật toán tìm kiếm được tích hợp vào ứng dụng, và cách chương trình xử lý các sự kiện như giải Puzzle, tăng tốc, và lưu lịch sử. Bên cạnh đó, phần đánh giá hiệu quả của từng thuật toán sẽ được trình bày, giúp người dùng hiểu rõ ưu nhược điểm của mỗi thuật toán khi giải quyết bài toán N-Puzzle.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Để có thể hoàn thành tiểu luận một cách tốt nhất, thì ngôn ngữ lập trình, thuật toán là phần không thể thiếu, vì vậy em đã nghiên cứu và tóm tắt những tài liệu và trọng điểm chính sau đây.

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

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

Bài toán N-puzzle có một số đặc điểm và yêu cầu cơ bản cần được hiểu rõ trước khi triển khai giải quyết. Trước tiên, kích thước bảng của N-puzzle có thể thay đổi linh hoạt theo các giá trị $n*m$, với n và m thường là 2, 3, 4, hoặc 5, tùy thuộc vào độ phức tạp mà người chơi mong muốn. Trong bài toán này, ô trống được mặc định là ô có số hiệu 0. Mục tiêu của trò chơi là di chuyển các ô có số thứ tự vào vị trí của ô trống ở mỗi lượt, sao cho trạng thái bắt đầu được chuyển thành trạng thái đích.

Trạng thái bắt đầu là một cấu hình ngẫu nhiên của các ô trên bảng, trong khi trạng thái đích là cấu hình chuẩn mà các ô cần đạt được. Các thao tác hợp lệ bao gồm di chuyển ô trống theo bốn hướng: lên, xuống, trái, phải. Để giải quyết bài toán, người chơi phải sắp xếp lại các ô vuông sao cho tất cả các ô đều được đưa về đúng vị trí theo trạng thái đích đã đặt ra.

2.1.2. Các trạng thái có thể có của bảng.

Trạng thái xuất phát: Đây là cấu hình ban đầu của bảng khi trò chơi bắt đầu. Các ô vuông được sắp xếp ngẫu nhiên trên bảng, bao gồm một ô trống (được ký hiệu bằng số 0). Trạng thái xuất phát có thể thay đổi tùy theo cách xáo trộn các ô và là điểm bắt đầu cho quá trình giải bài toán.

Trạng thái đích: Đây là cấu hình mong muốn của bảng, khi tất cả các ô vuông đã được sắp xếp theo thứ tự từ 1 đến $n*m - 1$, với ô trống nằm ở vị trí cuối cùng của bảng.

Trạng thái đích là mục tiêu mà người chơi cần đạt được bằng cách thực hiện các bước di chuyển hợp lệ.

2.2. Ngôn ngữ lập trình Python.

Python là một ngôn ngữ lập trình bậc cao, đa năng, dễ học và được sử dụng rộng rãi trong nhiều lĩnh vực công nghệ. Python cung cấp cú pháp đơn giản, dễ đọc, cho phép các nhà phát triển tập trung vào giải quyết vấn đề hơn là cấu trúc ngôn ngữ. Một trong những tính năng nổi bật của Python là thư viện tiêu chuẩn phong phú, hỗ trợ nhiều tác vụ từ xử lý chuỗi, làm việc với tệp, đến tính toán khoa học và phát triển web Python hỗ trợ cả lập trình hướng đối tượng (OOP) lẫn lập trình hàm (Functional Programming), giúp người dùng linh hoạt trong cách tiếp cận và triển khai các giải pháp. Với tính năng tích hợp tốt và cộng đồng lớn mạnh, Python sở hữu hàng nghìn thư viện mã nguồn mở phục vụ cho các nhu cầu cụ thể như NumPy, Pandas, TensorFlow, OpenCV, và Flask.

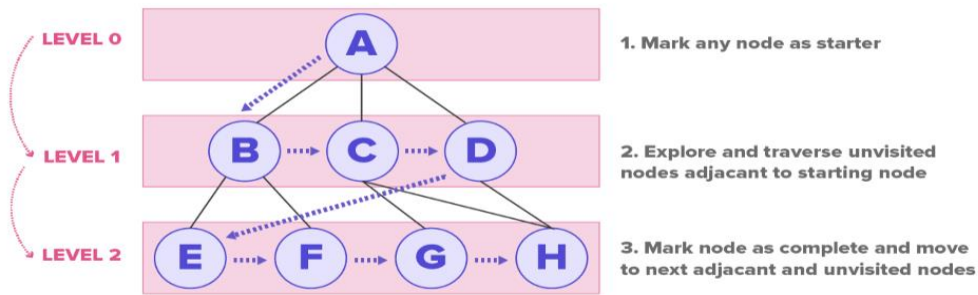
2.3. Tìm hiểu về các thuật toán tìm kiếm.

2.3.1. BFS

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.

Ý tưởng giải thuật là 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).

ARCHITECTURE OF BFS

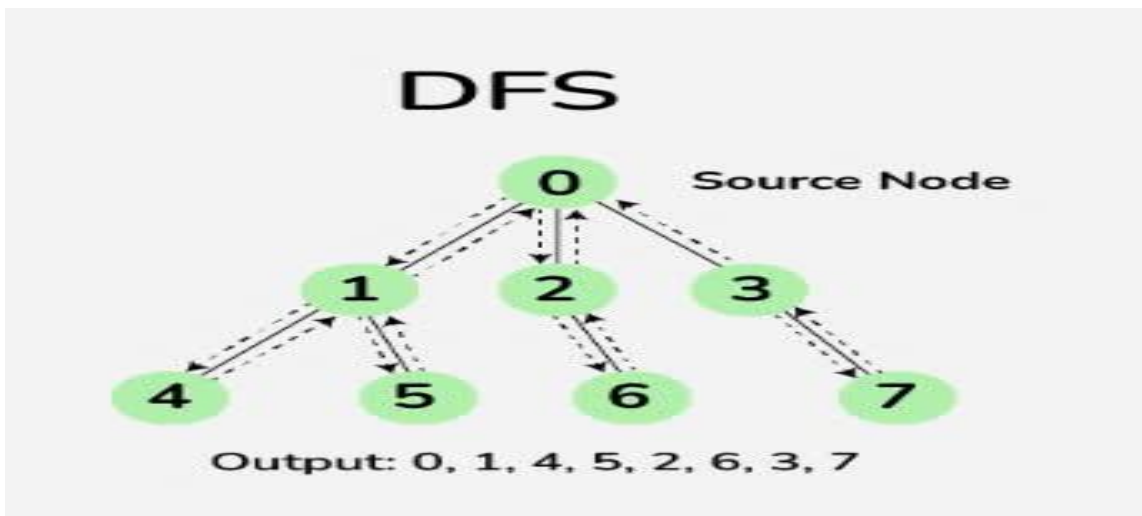


Hình 2.1: Các bước duyệt thuật toán BFS

2.3.2. DFS.

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.

Ý tưởng giải thuật là 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.



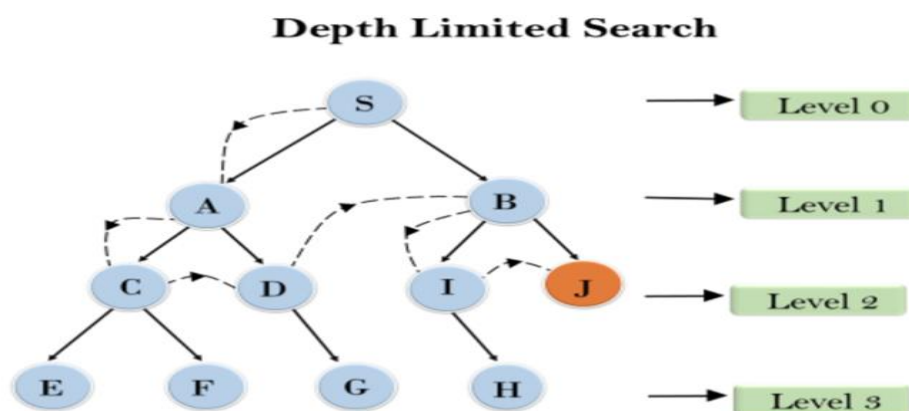
Hình 2.2: Các bước duyệt thuật toán DFS

2.3.3. DLS.

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.

Ý tưởng giải thuật là 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.

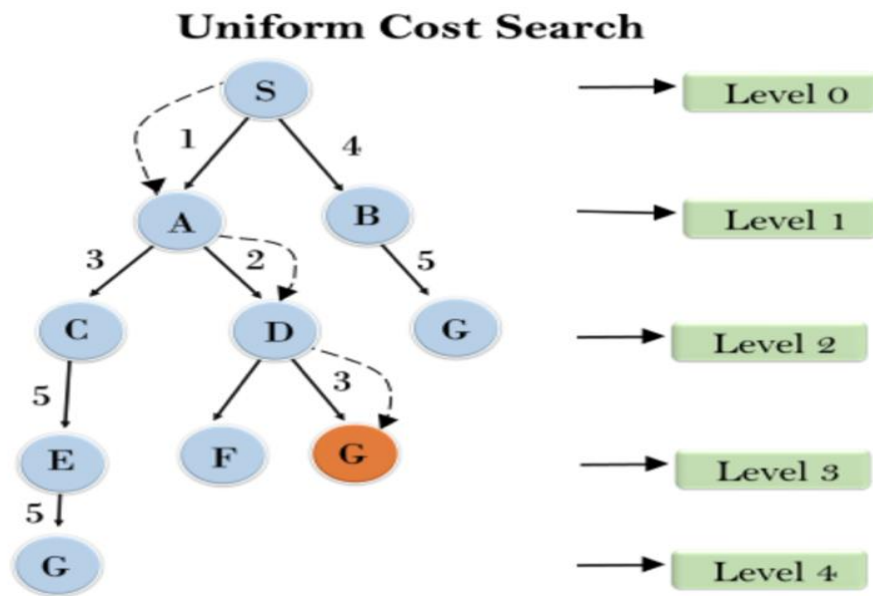


Hình 2.3: Các bước duyệt thuật toán DLS

2.3.4. UCS

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.

Ý tưởng giải thuật là 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.

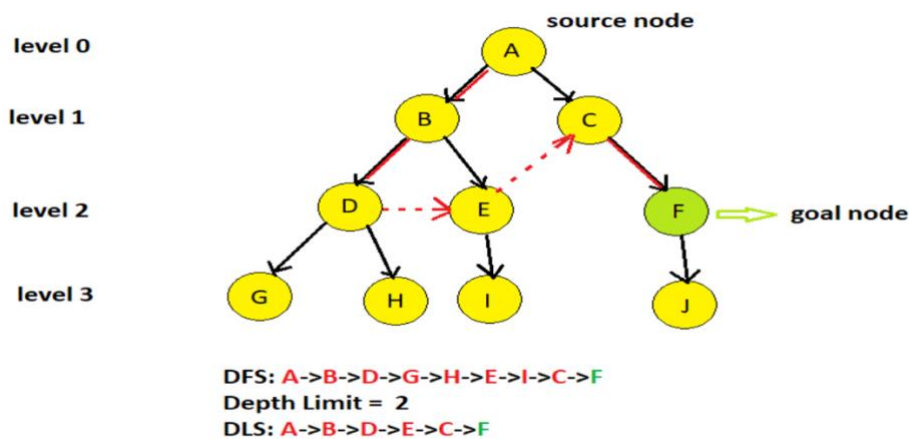


Hình 2.4: Các bước duyệt thuật toán UCS

2.3.5. ID

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).

Ý tưởng giải thuật là 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.

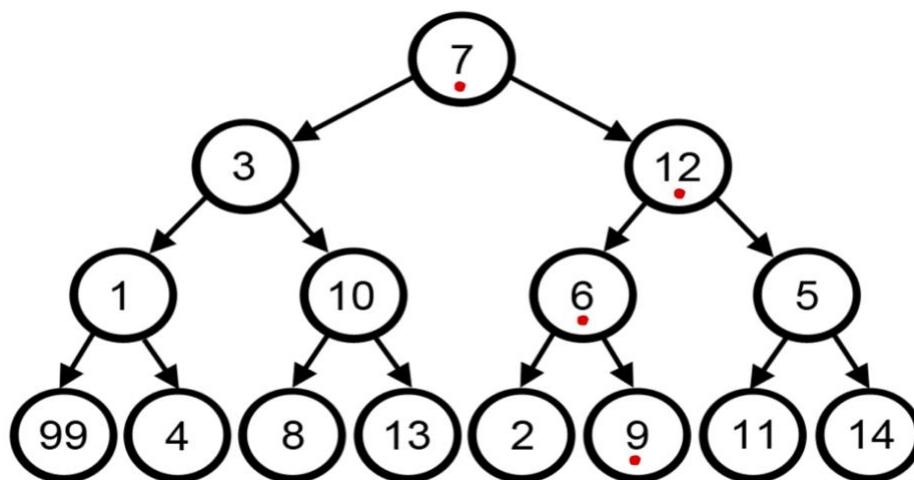


Hình 2.5: Các bước duyệt thuật toán ID

2.3.6. Greedy

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.

Ý tưởng giải thuật là 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.

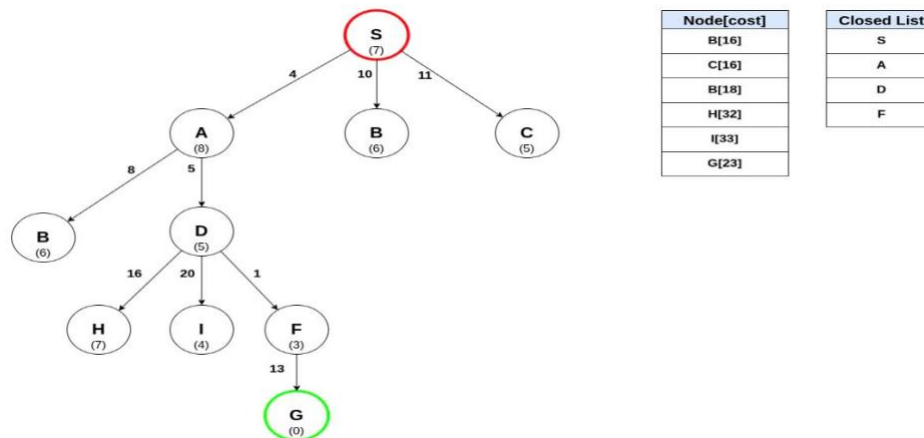


Hình 2.6: Các bước duyệt thuật toán Greedy

2.3.7. A-star

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.

Ý tưởng giải thuật là 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.

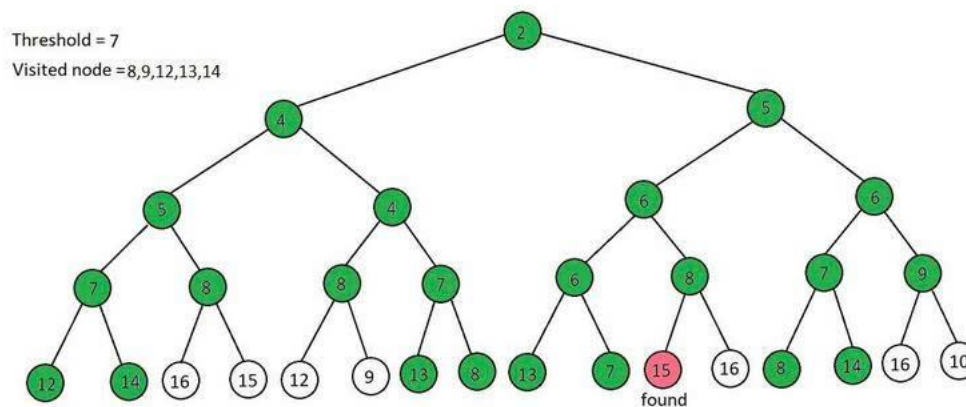


Hình 2.7: Các bước duyệt thuật toán A*

2.3.8. IDA-Star

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.

Ý tưởng giải thuật là 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.



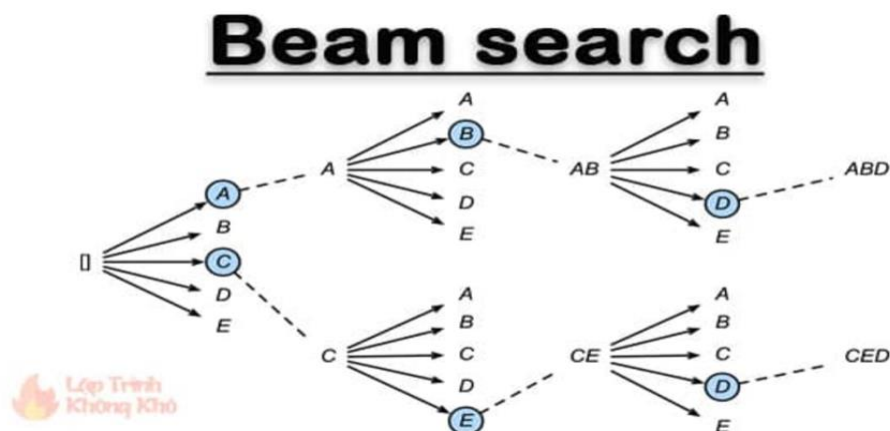
Hình 2.8: Các bước duyệt thuật toán IDA *

2.3.9. 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.

Ý tưởng giải thuật là 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.



Hình 2.9: Các bước duyệt thuật toán Beam Search

2.4. Các thư viện hỗ trợ

2.4.1. Thư viện Tkinter

Tkinter là một thư viện tích hợp sẵn trong Python, cung cấp các tính năng mạnh mẽ để xây dựng giao diện người dùng đồ họa (GUI). Thư viện hỗ trợ tạo các widget cơ bản như nút bấm, nhãn, hộp văn bản, hộp chọn, thanh cuộn và khung hình, giúp người dùng dễ dàng xây dựng các ứng dụng trực quan. Tkinter cung cấp các phương pháp bố trí giao diện linh hoạt như `pack()`, `grid()` và `place()`, đồng thời hỗ trợ xử lý sự kiện như nhấn nút, nhập liệu từ bàn phím, hoặc di chuyển chuột. Các widget có thể tùy chỉnh linh hoạt về màu sắc, font chữ, và kích thước để phù hợp với nhu cầu thiết kế.

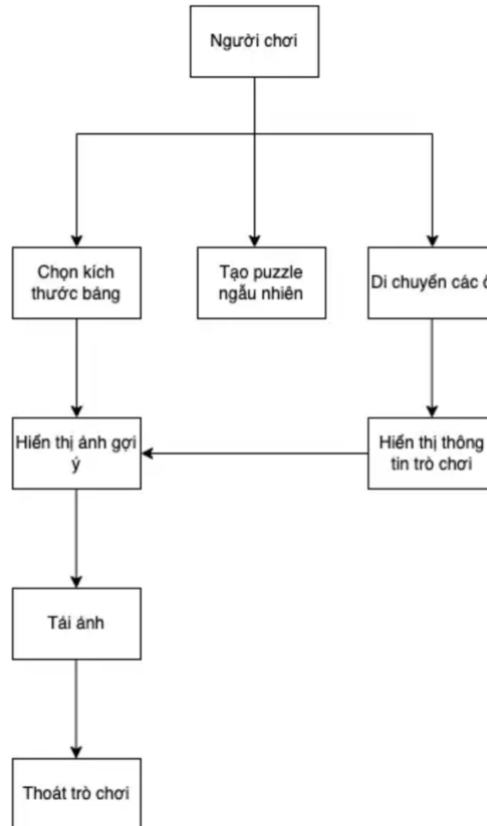
2.4.2. Thư viện Pillow

Pillow là một thư viện Python mạnh mẽ được phát triển từ nền tảng của PIL (Python Imaging Library), cung cấp các công cụ xử lý và thao tác trên hình ảnh một cách hiệu quả. Đây là phiên bản nâng cấp của PIL, được duy trì và phát triển để hỗ

trợ các phiên bản Python hiện đại, đặc biệt là Python 3.x, giúp khắc phục những hạn chế của PIL. Pillow hỗ trợ nhiều tính năng mạnh mẽ, từ mở, chỉnh sửa, lưu trữ hình ảnh đến áp dụng các hiệu ứng và bộ lọc tiên tiến.

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

3.1. Phân tích hệ thống.



Hình 3.1 – Biểu đồ use case

Người dùng có thể thực hiện các thao tác như tạo bảng Puzzle, tải ảnh, thay đổi kích thước bảng, xáo trộn, lưu trạng thái, và xem lịch sử. Ngoài ra, người dùng có thể chọn thuật toán giải quyết vấn đề (như BFS, DFS, A-Star, v.v.), xem hướng dẫn giải, và theo dõi quá trình giải thuật qua các bước di chuyển. Các chức năng này giúp người dùng dễ dàng tham gia vào trải nghiệm giải đố hoặc kiểm tra hiệu suất của các thuật toán tìm kiếm trong trò chơi.

3.1.1. Giao diện tạo Puzzle.

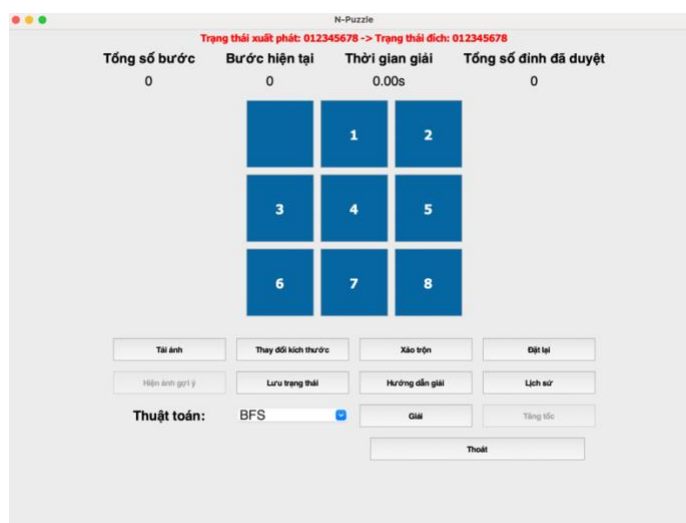
3.1.1.1. Thông tin.

Dòng trạng thái hiển thị hai trạng thái quan trọng của trò chơi là trạng thái xuất phát và trạng thái đích. Trạng thái xuất phát cho biết cấu hình ban đầu của bảng Puzzle,

trong khi trạng thái đích là mục tiêu cần đạt được để hoàn thành trò chơi.

Các thông tin của trò chơi như tổng số bước, số bước hiện tại, thời gian giải, tổng số đỉnh đã duyệt.

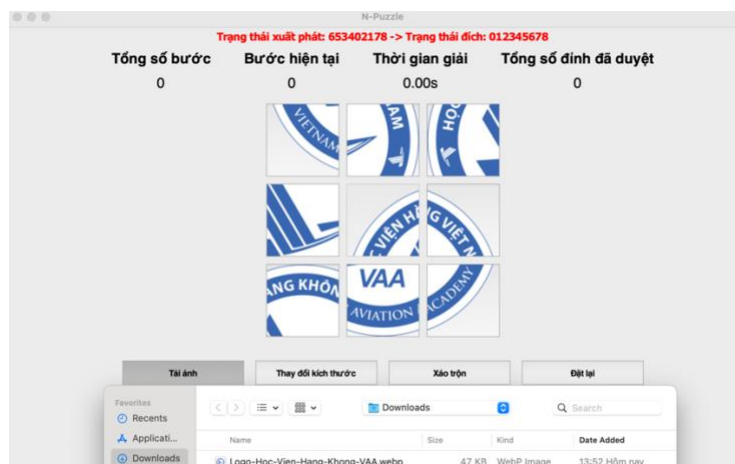
Bảng Puzzle hiển thị trạng thái hiện tại của bảng Puzzle, cho phép người dùng thấy rõ vị trí của các ô số và ô trống, đồng thời theo dõi các thay đổi mỗi khi thực hiện một bước di chuyển.



Hình 3.2. Giao diện chính Puzzle.

3.1.1.2. Tải ảnh

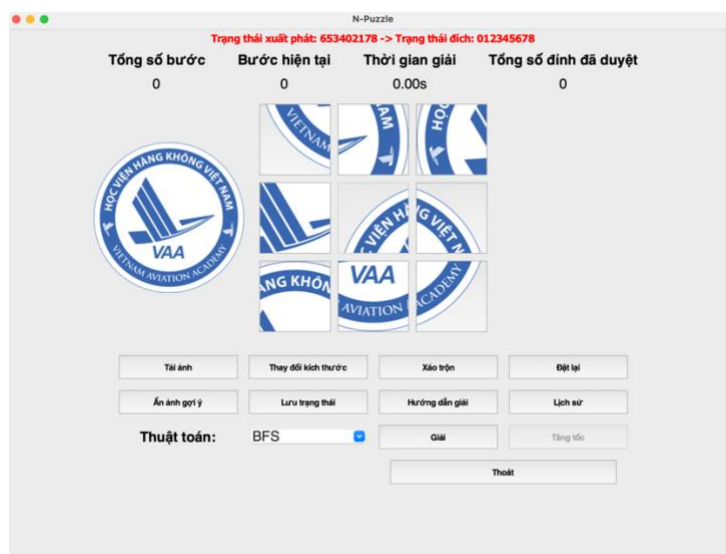
Nhấn vào nút tải ảnh, sau đó chọn ảnh chúng ta muốn tải, chương trình sẽ tự động chèn và xáo trộn puzzle có ảnh vừa chọn.



Hình 3.3. Chức năng tải ảnh.

3.1.1.3. Hiện ảnh gợi ý.

Trong giao diện của chương trình N-Puzzle, tính năng “Hiện ảnh gợi ý” cho phép người dùng hiển thị hình ảnh gợi ý về trạng thái đích của trò chơi. Chức năng này sẽ hiển thị một bức ảnh gốc tương ứng với trạng thái hoàn chỉnh của Puzzle, giúp người chơi có cái nhìn trực quan về cách sắp xếp các ô số để đạt được trạng thái đích. Tính năng này đặc biệt hữu ích khi người chơi gặp khó khăn trong quá trình giải quyết bài toán và cần một tham chiếu để định hướng các bước di chuyển tiếp theo.



Hình 3.4. Hiện ảnh gợi ý

3.1.1.4. Thay đổi kích thước

Trong giao diện của chương trình N-Puzzle, tính năng “Thay đổi kích thước” cho phép người dùng điều chỉnh lại kích thước của bảng Puzzle. Người dùng có thể thay đổi số hàng và số cột của bảng tùy theo mức độ thử thách mong muốn. Khi kích thước được thay đổi, một bảng Puzzle mới với trạng thái xuất phát tương ứng sẽ được tạo ra. Tính năng này giúp người dùng có thêm sự linh hoạt trong việc chọn độ khó của trò chơi, từ các bảng đơn giản (như 2x2) đến các bảng phức tạp hơn (như 5x5).

3.1.1.5. Xáo trộn

Chức năng “Xáo trộn” trong giao diện của chương trình N-Puzzle cho phép người dùng tạo ra trạng thái xuất phát ngẫu nhiên cho bảng Puzzle. Khi nhấn nút xáo trộn, các ô số sẽ được hoán đổi vị trí một cách ngẫu nhiên để tạo ra một trạng thái xuất

phát mới, từ đó người chơi sẽ bắt đầu giải bài toán. Tính năng này giúp trò chơi trở nên đa dạng và thú vị hơn, mang lại trải nghiệm mới mẻ mỗi khi người dùng chơi lại.

3.1.1.6. Đặt lại

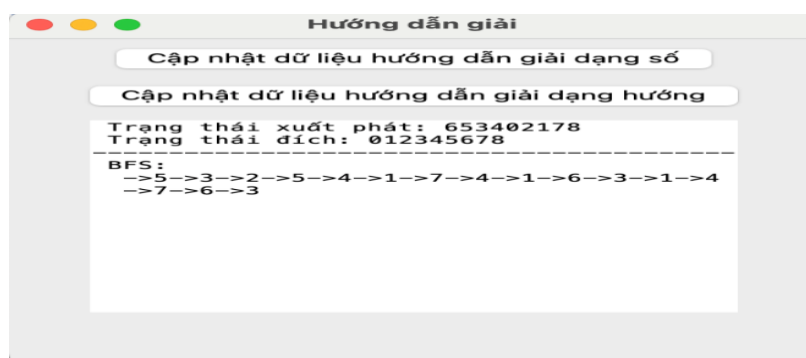
Chức năng “Đặt lại” trong giao diện của chương trình N-Puzzle cho phép người dùng quay trở lại trạng thái xuất phát ban đầu của bảng Puzzle. Khi nhấn nút đặt lại, tất cả các thay đổi đã thực hiện sẽ được hoàn tác, và bảng Puzzle sẽ trở về cấu hình ngẫu nhiên ban đầu sau khi được xáo trộn. Tính năng này hữu ích khi người chơi muốn thử lại từ đầu mà không cần tạo Puzzle mới.

3.1.1.7. Lưu trạng thái.

Chức năng “Lưu trạng thái” trong giao diện của chương trình N-Puzzle cho phép người dùng lưu lại trạng thái hiện tại của bảng Puzzle và thiết lập nó như trạng thái xuất phát mới. Khi người dùng nhấn nút lưu trạng thái, trạng thái hiện tại của các ô trên bảng sẽ được ghi nhận và lưu lại, từ đó trở thành trạng thái xuất phát cho các lần chơi tiếp theo. Tính năng này hữu ích trong việc giữ lại tiến độ hoặc lưu trữ trạng thái khi người dùng muốn tiếp tục giải bài toán từ điểm đang dừng.

3.1.1.8. 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.



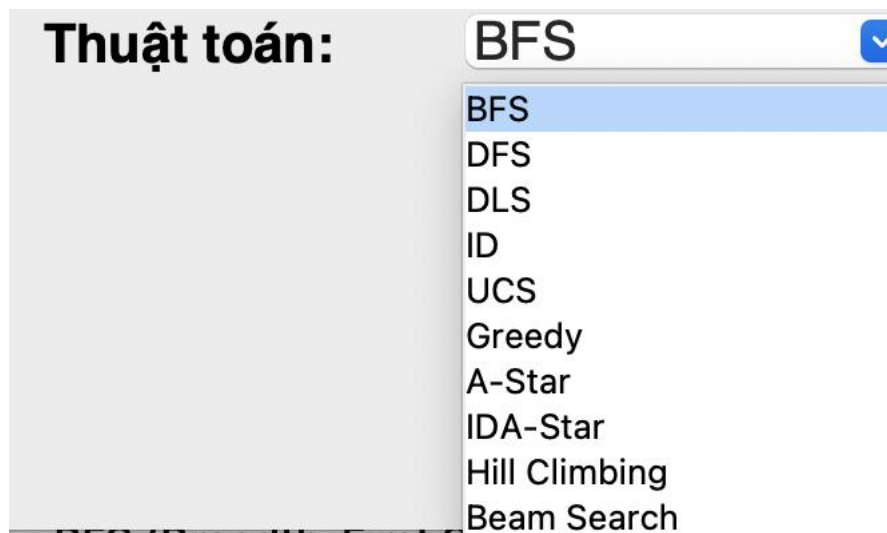
Hình 3.5. Hướng dẫn giải dạng số.



Hình 3.6. Hướng dẫn giải dạng kí tự.

3.1.1.9. Chọn thuật toán.

Gồm 9 thuật toán: BFS, DFS, DLS, ID, UCS, Greedy, A-Star, IDA-Star, Beam Search,...

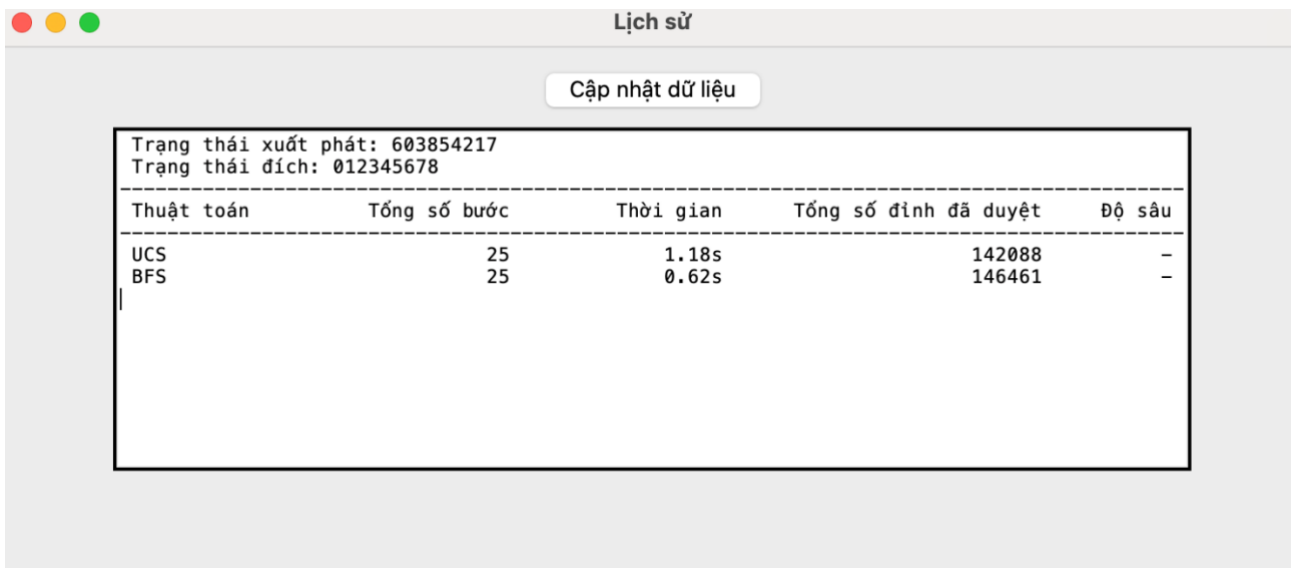


Hình 3.7. Thuật Toán.

3.1.1.10. Lịch sử.

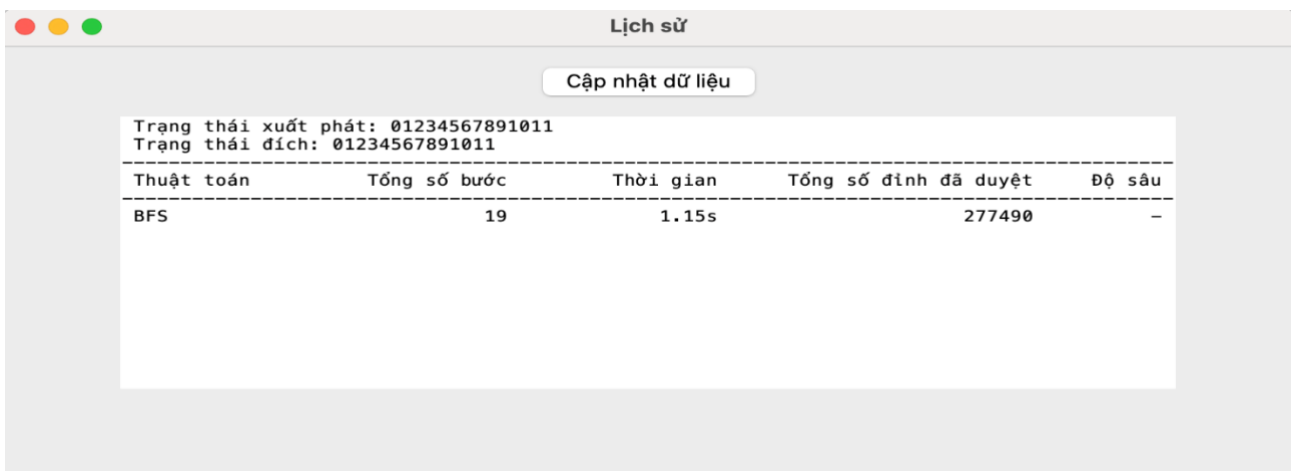
Chức năng “Lịch sử” trong giao diện của chương trình N-Puzzle cho phép lưu lại lịch sử của trò chơi, bao gồm thông tin về các lần giải trước đó. Lịch sử sẽ ghi nhận các bước di chuyển, thời gian hoàn thành, và trạng thái của bảng

Puzzle trong từng lần chơi. Tính năng này giúp người chơi có thể xem lại các lần chơi trước đó, theo dõi tiến độ và so sánh hiệu suất giữa các lần giải Puzzle. Việc lưu lại lịch sử cũng tạo điều kiện cho người dùng phân tích chiến lược giải và cải thiện kết quả trong các lần chơi tiếp theo.



Lịch sử				
Cập nhật dữ liệu				
Trạng thái xuất phát: 603854217 Trạng thái đích: 012345678				
Thuật toán	Tổng số bước	Thời gian	Tổng số đỉnh đã duyệt	Độ sâu
UCS	25	1.18s	142088	–
BFS	25	0.62s	146461	–

Hình 3.8. Lịch sử 3x3.



Lịch sử				
Cập nhật dữ liệu				
Trạng thái xuất phát: 01234567891011 Trạng thái đích: 01234567891011				
Thuật toán	Tổng số bước	Thời gian	Tổng số đỉnh đã duyệt	Độ sâu
BFS	19	1.15s	277490	–

Hình 3.9. Lịch sử 3x4.

3.1.1.11. Giải

Chức năng “Giải” trong giao diện của chương trình N-Puzzle cho phép tiến hành giải bảng Puzzle dựa trên thuật toán mà người dùng đã chọn. Sau khi lựa chọn thuật toán tìm kiếm phù hợp (như BFS, A-Star, hoặc các thuật toán khác), chương trình sẽ tự động thực hiện các bước di chuyển để đưa bảng Puzzle từ trạng thái xuất phát đến trạng thái đích. Quá trình này sẽ hiển thị các bước giải trên giao diện và cập nhật thông tin về thời gian giải, số bước đi, và các trạng thái trung gian, giúp người dùng theo dõi quá trình giải quyết của thuật toán đã chọn

3.1.1.12. Tăng tốc

Chức năng “Tăng tốc” cho phép người dùng tua nhanh quá trình giải Puzzle, đặc biệt hữu ích khi thuật toán phải thực hiện quá nhiều bước để tìm ra giải pháp. Khi tính năng này được kích hoạt, chương trình sẽ tăng tốc độ di chuyển giữa các bước, hoặc thậm chí bỏ qua các bước trung gian và đưa người chơi trực tiếp đến kết quả cuối cùng. Điều này giúp tiết kiệm thời gian cho người dùng, nhất là khi giải các bảng Puzzle phức tạp với nhiều bước di chuyển.

3.1.1.13. Thoát.

Chức năng “Thoát” cho phép người dùng đóng chương trình N-Puzzle và kết thúc phiên làm việc. Khi nhấn nút Thoát, toàn bộ cửa sổ của chương trình sẽ được đóng lại, giúp người dùng nhanh chóng thoát khỏi ứng dụng khi không còn nhu cầu tiếp tục giải Puzzle hoặc thực hiện các thao tác khác.

3.2. Giao diện các thuật toán tìm kiếm.

3.2.1. Giao diện BFS

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.

Trạng thái xuất phát: 018237465 -> Trạng thái đích: 012345678

Tổng số bước	Bước hiện tại	Thời gian giải	Tổng số đỉnh đã duyệt
18	18	0.08s	24494

1
2
3
4
5
6
7
8

Tải ảnh

Hiện ảnh gợi ý

Thay đổi kích thước

Lưu trạng thái

Xáo trộn

Hướng dẫn giải

Giải

Đặt lại

Lịch sử

Tăng tốc

Thuật toán: BFS ▼

Thoát

Hình 3.10 Giao diện chọn BFS

3.2.2. Giao diện DFS

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.

Trạng thái xuất phát: 2130 -> Trạng thái đích: 0123

Tổng số bước	Bước hiện tại	Thời gian giải	Tổng số đỉnh đã duyệt
2	2	0.00s	0

1
2
3

Tải ảnh

Hiện ảnh gợi ý

Thay đổi kích thước

Lưu trạng thái

Xáo trộn

Hướng dẫn giải

Giải

Đặt lại

Lịch sử

Tăng tốc

Thuật toán: DFS ▼

Thoát

Hình 3.11 Giao diện chọn DFS

3.2.3. Giao diện DLS

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.

Trạng thái xuất phát: 162754380 -> Trạng thái đích: 012345678

Tổng số bước	Bước hiện tại	Thời gian giải	Tổng số đỉnh đã duyệt
40	40	0.33s	129474

1

2

3

4

5

6

7

8

Tải ảnh

Thay đổi kích thước

Xáo trộn

Đặt lại

Hiện ảnh gợi ý

Lưu trạng thái

Hướng dẫn giải

Lịch sử

Thuật toán: DLS

Giải

Tăng tốc

Giới hạn độ sâu: 50

Thoát

Hình 3.12: Giao diện chọn DLS

3.2.4. Giao diện UCS

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.

Trạng thái xuất phát: 758601432 -> Trạng thái đích: 012345678

Tổng số bước	Bước hiện tại	Thời gian giải	Tổng số đỉnh đã duyệt
26	26	1.31s	162241

1
2

3
4

5
6

7
8

Tải ảnh

Hiện ảnh gợi ý

Thay đổi kích thước

Lưu trạng thái

Xáo trộn

Hướng dẫn giải

Giải

Đặt lại

Lịch sử

Tăng tốc

Thuật toán: UCS

Thoát

Hình 3.13 *Giao diện chọn UCS*

3.2.5. Giao diện ID

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.

Trạng thái xuất phát: 162754380 -> Trạng thái đích: 012345678

Tổng số bước	Bước hiện tại	Thời gian giải	Tổng số đỉnh đã duyệt
50	50	0.06s	23649

1
2

3
4

5
6

7
8

Tải ảnh

Hiện ảnh gợi ý

Thay đổi kích thước

Lưu trạng thái

Xáo trộn

Hướng dẫn giải

Giải

Đặt lại

Lịch sử

Tăng tốc

Thuật toán: ID

Độ sâu dẫn: 50

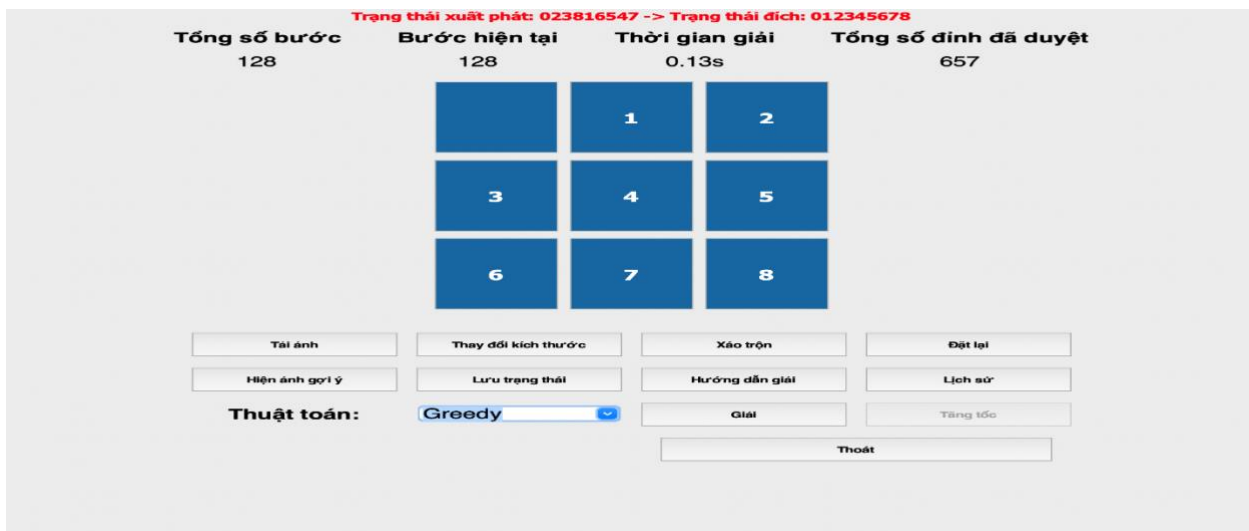
Thoát

Hình 3.14: *Giao diện chọn ID*

3.2.6. Giao diện 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

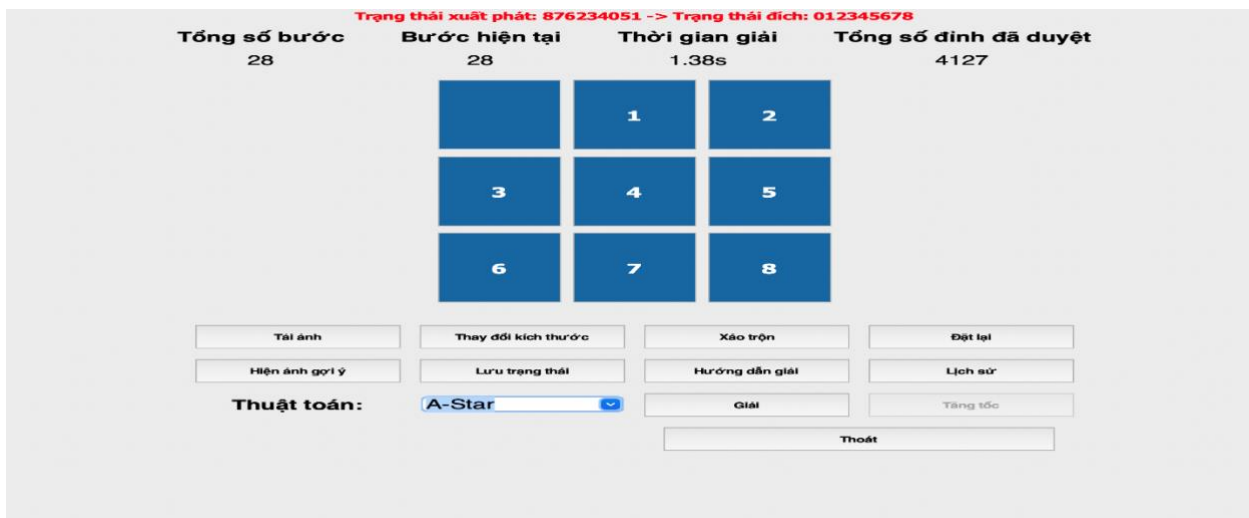


Hình 3.15: Giao diện chọn Greedy

3.2.7. Giao diện A-Star

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 hàm heuristic không được thiết kế tốt.

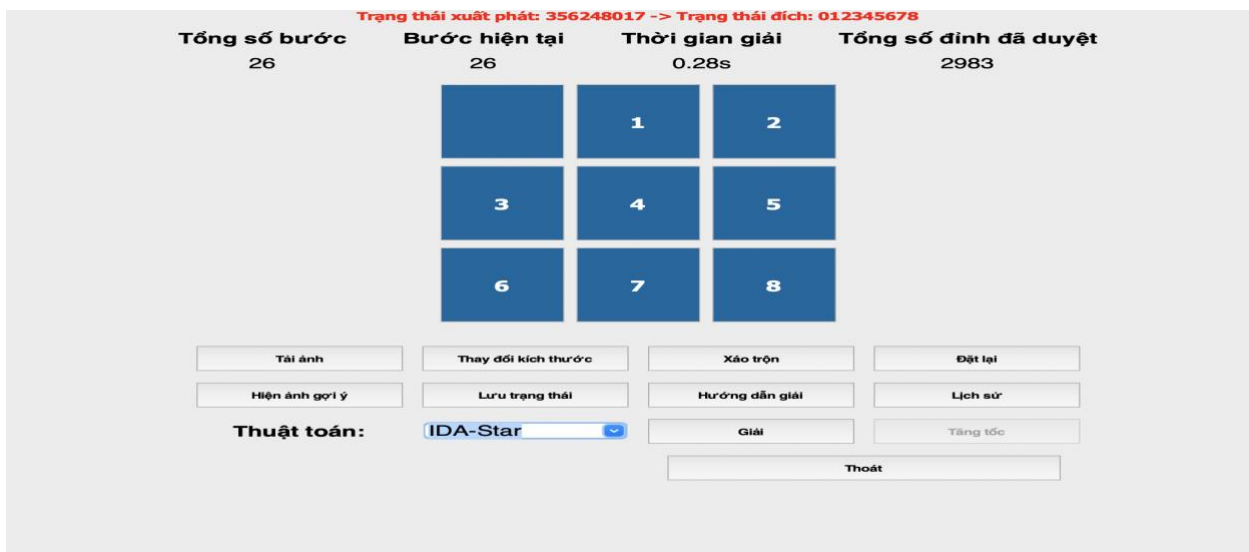


Hình 3.16: Giao diện chọn A*

3.2.8. Giao diện IDA-Star

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.

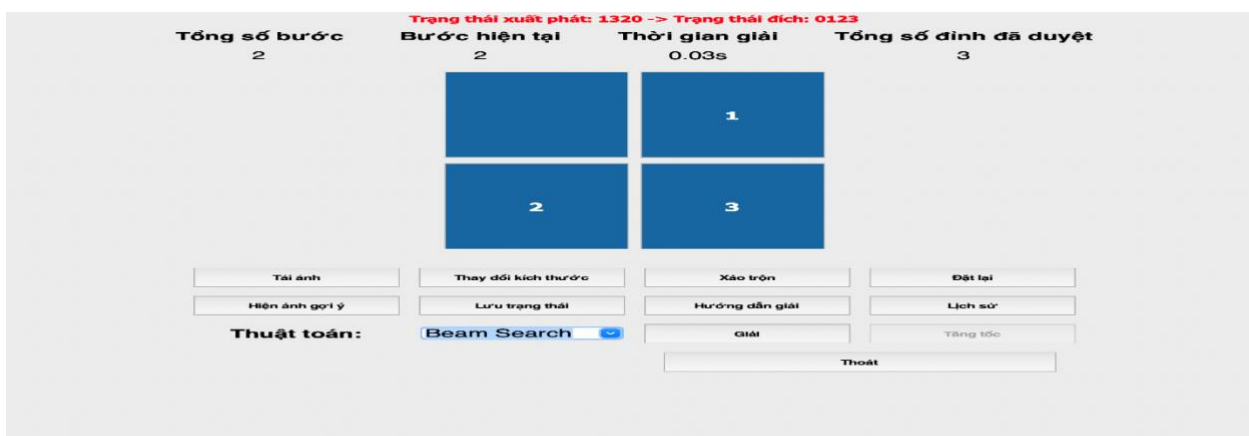


*Hình 3.17 Giao diện chọn IDA**

3.2.9. Giao diện Beam Search.

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.



Hình 3.18: Giao diện chọn Beam Search

3.3. 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.

Beam Search, có thể tìm kiếm tốt trong không gian trạng thái nhỏ, nhưng có thể bị mắc kẹt

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.

Mặc dù các chức năng cơ bản của chương trình N-Puzzle đã được triển khai, nhưng vẫn còn những điểm cần tối ưu và cải tiến để chương trình hoạt động hiệu quả hơn. Trên cơ sở những gì chưa hoàn thiện hoặc chưa đạt mức tối ưu, phần này sẽ trình bày các định hướng phát triển trong tương lai nhằm nâng cấp và hoàn thiện sản phẩm. Trước tiên, cần tiến hành “tối ưu hóa” chương trình để nó có thể chạy nhanh hơn. Việc tối ưu hóa này có thể bao gồm cải thiện hiệu quả thuật toán, giảm thiểu sử dụng bộ nhớ và tăng tốc độ xử lý. Tiếp theo, “thiết kế giao diện” cũng là một yếu tố quan trọng cần cải tiến. Giao diện hiện tại có thể được thay đổi để trở nên bắt mắt, trực quan và thân thiện với người dùng hơn. Bổ sung thêm các “tính năng tiện ích” như gợi ý cho người chơi, cung cấp hướng dẫn rõ ràng và chi tiết để người dùng có thể dễ dàng sử dụng chương trình hơn. Cuối cùng, việc “so sánh giữa các thuật toán” sẽ giúp người dùng đánh giá và lựa chọn thuật toán phù hợp nhất với bài toán cụ thể. Một tính năng cho phép người chơi hoặc người nghiên cứu xem xét hiệu suất của từng thuật toán về thời gian, số bước di chuyển, và độ phức tạp là một bước nâng cấp quan trọng

TÀI LIỆU THAM KHẢO

- [1] Nguyễn Châu Khanh, BFS (Breadth-first search), VNOI, đường dẫn: <https://vnoi.info/wiki/algo/graph-theory/breadth-first-search.md>
- [2] Thái Thanh Hải, Data Structure & Algorithm - Graph Algorithms - Depth First Search (DFS), ngày 26 tháng 2 năm 2023, đường dẫn: <https://viblo.asia/p/data-structure-algorithm-graph-algorithms-depth-first-search-dfs-qPoL7zyXJvk>
- [3] Bùi Quang Hà, Các thuật toán tìm kiếm trong AI, Flinters, ngày 19 tháng 11 năm 2020, đường dẫn: <https://labs.flinters.vn/algorithm/algorithm-cac-thuat-toan-tim-kiem-trong-ai/>
- [4] Thơ Trần, Các thuật toán cơ bản trong AI - Phân biệt Best First Search và Uniform Cost Search (UCS), VIBLO, 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>
- [5] Dục Đoàn Trình, Hill Climbing Algorithm trong trí tuệ nhân tạo, websitehcm, ngày 26 tháng 3 năm 2022, đường dẫn: <https://websitehcm.com/hill-climbing-algorithm-trong-tri-tue-nhan-tao/>
- [6] [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>