**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN HỌC PHẦN**

TÊN HỌC PHẦN: **TRÍ TUỆ NHÂN TẠO (Artificial Intelligence: AI)**

MÃ SỐ LỚP HP: **ARIN330585 - Nhóm 03 (Sáng thứ 3)**

Tên đề tài:**PHÁT TRIỂN HỆ THỐNG GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP VỚI IDS**

Họ tên sinh viên: **Lê Thị Minh Nguyệt**

**Mã số sinh viên: 19110413 [31]**

**Lớp: 191102A**

**Ngày nộp: 19/06/2021**

**Ký tên:**

**TP.HCM, ngày 10 tháng 06 năm 2021**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỀ TÀI ĐỒ ÁN HỌC PHẦN**

**Giảng viên giảng dạy: VÕ XUÂN THỂ**

TÊN HỌC PHẦN: **TRÍ TUỆ NHÂN TẠO (Artificial Intelligence: AI)**

MÃ SỐ LỚP HP: **ARIN330585 - Nhóm 03 (Sáng thứ 3)**

Tên đề tài: **PHÁT TRIỂN HỆ THỐNG GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP VỚI IDS**

Họ tên sinh viên: **Lê Thị Minh Nguyệt**

**Mã số sinh viên: 19110413 [31]**

**Lớp: 191102A**

**Tên sản phẩm đề tài: N3S3.31.LeThiMinhNguyet.19110413.DoaAnHP.AI.rar**

**Công cụ sử dụng (phiên bản): Spyder (Anaconda3)**

**Ngôn ngữ lập trình sử dụng: Python (python 3.8)**

**Nhận xét của giảng viên:**

**………………………………………………………..**

**………………………………………………………..**

**Điểm đánh giá: …………. ( …………………)**

**Ngày……. / …….. /2021**

**Giảng viên Ký tên**

**TP.HCM, ngày 10 tháng 06 năm 2021**

# LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cám ơn sâu sắc nhất đến thầy Võ Xuân Thể.

Trong quá trình tìm hiểu và học tập môn Trí tuệ nhân tạo em đã nhận được sự giảng dạy và hướng dẫn tận tình, tâm huyết từ thầy, giúp chúng em tích lũy thêm rất nhiều kiến thức. Từ những kiến thức thầy đã truyền đạt chúng em xin trình bày lại những gì mình đã tìm hiểu về đề tài: PHÁT TRIỂN HỆ THỐNG GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP IDS.

Tuy nhiên, kiến thức về môn học của em vẫn còn những hạn chế nhất định và không tránh khỏi những thiếu sót trong quá trình hoàn thành bài báo cáo này. Em mong nhận được lời góp ý, nhận xét đến từ thầy và các bạn để đề tài báo cáo của em được hoàn thiện hơn.

Chúc em kính chúc thầy sức khỏe, hạnh phúc và thành công hơn nữa trong sự nghiệp trồng người để tiếp tục dìu dắt nhiều thế hệ sinh viên đến những bến bờ tri thức.

Chúng em xin chân thành cảm ơn thầy!

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
|  | |
| CSDL hoặc DB | Cơ sở dữ liệu: DataBase |
|  | |
| AI | Trí tuệ (Trí thông minh) nhân tạo: Artificial Intelligence |
| ANN | Mạng nơ ron nhân tạo: Artificial Neural Network |
| ES | Hệ thống chuyên gia: Expert Systems |
| ML | Máy học = Học máy: Machine Learning |
| NLP | Xử lý ngôn ngữ tự nhiện: Natural Language Processing |

# DANH MỤC CÁC THUẬT NGỮ

|  |  |
| --- | --- |
| Bài toán (vấn đề) AI:  AI Problem | Là các tình huống thực tế mà con người cần giải quyết thuộc một lĩnh vực nào đó, thông thường phải dựa vào “trí khôn” con người. |
| Giao diện người dùng  (User-Interface) | Là hệ thống các màn hình giao tiếp cho phép người sử dụng tương tác với các thành phần phần mềm, điều khiển phần mềm hoạt động theo yêu cầu của người dùng - tương ứng các chức năng hiện có của phần mềm. |
| Người dùng (User):  Tài khoản (Account) | Là một quyền làm việc trên hệ thống phần mềm được cấp phát cho một cá nhân thông qua tên tài khoản (username) và mật khẩu (password). |
|  |  |
| BigData | Dữ liệu lớn: là một tập hợp dữ liệu rất lớn và phức tạp, không thể xử lý dữ liệu bằng các phương pháp truyền thống. |
| Heuristics | Là các kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, nhằm đưa ra một giải pháp mà không được đảm bảo là tối ưu |

# MỤC LỤC

[LỜI CẢM ƠN 3](#_Toc74233411)

[DANH MỤC CÁC TỪ VIẾT TẮT 4](#_Toc74233412)

[DANH MỤC CÁC THUẬT NGỮ 5](#_Toc74233413)

[MỤC LỤC 6](#_Toc74233414)

[PHIẾU GIAO NHIỆM VỤ ĐỀ TÀI ĐỒ ÁN HỌC PHẦN 9](#_Toc74233415)

[Chương 1: GIỚI THIỆU VỀ ĐỀ TÀI 10](#_Toc74233416)

[**1.1.** Tổng quan về đề tài 10](#_Toc74233417)

[**1.2.** Nội dung chuyên môn chính của đề tài 10](#_Toc74233418)

[**1.3.** Ngôn ngữ lập trình và công cụ sử dụng 12](#_Toc74233419)

[**1.4.** Sản phẩm của đề tài 12](#_Toc74233420)

[**1.5.** Bố cục của báo cáo 12](#_Toc74233421)

[Chương 2: CƠ SỞ LÝ THUYẾT VỀ TRÍ TUỆ NHÂN TẠO 13](#_Toc74233422)

[**2.1.** Tổng quan về Trí tuệ nhân tạo (AI) 13](#_Toc74233423)

[2.1.1. Các khái niệm 13](#_Toc74233424)

[2.1.2. Vai trò AI 13](#_Toc74233425)

[2.1.3. Nền tảng kỹ thuật của AI 13](#_Toc74233426)

[2.1.4. Các lĩnh vực nghiên cứu và ứng dụng cơ bản của AI 13](#_Toc74233427)

[**2.2.** Giới thiệu về các bài toán (vấn đề) AI 13](#_Toc74233428)

[2.2.1. Một số khái niệm 13](#_Toc74233429)

[2.2.2. Tìm kiếm lời giải (searching) cho AI Problem 13](#_Toc74233430)

[**2.3.** Biểu diễn một số bài toán AI 13](#_Toc74233431)

[2.3.1. Các bài toán Trò chơi 13](#_Toc74233432)

[2.3.2. Cây không gian trạng thái của Bài toán AI 13](#_Toc74233433)

[**2.4.** Các phương pháp biểu diễn tri thức 13](#_Toc74233434)

[2.4.1. Phân loại tri thức theo phương pháp biểu diễn 14](#_Toc74233435)

[2.4.2. Phân loại tri thức theo nền tảng hình thành Cơ sở tri thức 14](#_Toc74233436)

[2.4.3. Logic mệnh đề và Logic vị từ 14](#_Toc74233437)

[2.4.4. Biểu diễn tri thức bằng Frame (Cấu trúc = "Khung") 14](#_Toc74233438)

[2.4.5. Suy diễn tri thức bằng luật dẫn xuất 14](#_Toc74233439)

[2.4.6. Biểu diễn suy luận bằng đồ thị AND/OR 14](#_Toc74233440)

[2.4.7. Các biểu diễn tương đương trong Logic mệnh đề và Logic vị từ 14](#_Toc74233441)

[2.4.8. Xử lý ngôn ngữ tự nhiên (Natural Language Processing = NLP): Voice Assistant 14](#_Toc74233442)

[**2.5.** Các phương pháp tìm kiếm lời giải của bài toán AI 14](#_Toc74233443)

[2.5.1. Giới thiệu 14](#_Toc74233444)

[2.5.2. Các phương pháp tìm kiếm mù (Blind Search) = tìm kiếm không có thêm thông tin (uninformed search) 15](#_Toc74233445)

[2.5.3. Các phương pháp tìm kiếm theo kinh nghiệm (Heuristic Search) = tìm kiếm với thông tin bổ sung (Informed Search) = Cải tiến BFS 15](#_Toc74233446)

[2.5.4. Lập trình Logic (Logic Programming) phát triển cơ chế lập luận logic 15](#_Toc74233447)

[Chương 3: GIỚI THIỆU VỀ MÔ HÌNH BÀI TOÁN AI: GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP VỚI IDS 16](#_Toc74233448)

[3.1. Mô tả bài toán AI 16](#_Toc74233449)

[3.2. Các yếu tố xác định bài toán AI 16](#_Toc74233450)

[Chương 4: BIỂU DIỄN BÀI TOÁN AI VÀ CƠ SỞ TRI THỨC: GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP VỚI IDS 19](#_Toc74233451)

[4.1. Giới thiệu 19](#_Toc74233452)

[4.2. Biểu diễn bài toán AI 19](#_Toc74233453)

[4.3. Xây dựng cơ sở tri thức (KBS) cho bài toán AI 19](#_Toc74233454)

[Chương 5: PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI CHO BÀI TOÁN AI: GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP VỚI IDS 20](#_Toc74233455)

[5.1. Giới thiệu 20](#_Toc74233456)

[5.2. Mô tả sơ bộ về mô hình cây không gian trạng thái của bài toán AI 20](#_Toc74233457)

[5.3. Xác định phương pháp tìm kiếm lới giải cho bài toán AI: game ghép tranh dựa trên game 8-puzzle dựa vào giải thuật DLS kết hợp với IDS 21](#_Toc74233458)

[5.4. Cài đặt (lập trình) phương pháp tìm kiếm lời giải cho bài toán AI: Game ghép tranh dựa trên game 8-puzzle dựa vào giải thuật DLS kết hợp với IDS 22](#_Toc74233459)

[Chương 6: GIỚI THIỆU SẢN PHẨM CỦA ĐỀ TÀI AI: GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP VỚI IDS 24](#_Toc74233460)

[6.1. Giới thiệu sản phẩm bài toán AI: game ghép tranh dựa trên game 8-puzzle dựa vào giải thuật DLS kết hợp với IDS 24](#_Toc74233461)

[6.2. Kết quả sản phẩm bài toán AI: Game ghép tranh dựa trên game 8-puzzle dựa vào giải thuật DLS kết hợp với IDS 35](#_Toc74233462)

[6.3. Nhận xét và đánh giá về sản phẩm bài toán AI: game ghép tranh dựa trên game 8-puzzle dựa vào giải thuật DLS kết hợp với IDS 37](#_Toc74233463)

[Chương 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI 38](#_Toc74233464)

[7.1. Kết luận 38](#_Toc74233465)

[7.1.1. Những kết quả đạt được 38](#_Toc74233466)

[7.1.2. Hạn chế 38](#_Toc74233467)

[7.2. Hướng phát triển 38](#_Toc74233468)

[7.2.1. Hướng khắc phục các hạn chế 38](#_Toc74233469)

[7.2.2. Hướng mở rộng đề tài 38](#_Toc74233470)

[DANH MỤC TÀI LIỆU THAM KHẢO 39](#_Toc74233471)

[ĐÁNH GIÁ KẾT QUẢ HỌC TẬP HỌC PHẦN 40](#_Toc74233472)

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

# PHIẾU GIAO NHIỆM VỤ ĐỀ TÀI ĐỒ ÁN HỌC PHẦN

|  |  |
| --- | --- |
| Giảng viên giảng dạy: **VÕ XUÂN THỂ** | |
| Tên học phần: **Trí tuệ nhân tạo (AI)** Mã số lớp HP: **ARIN330585 – N3S3** | |
| Tên đề tài: **PHÁT TRIỂN HỆ THỐNG GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT TRÍ TUỆ NHÂN TẠO (GIẢI THUẬT DLS KẾT HỢP VỚI IDS)** | |
| Sinh viên thực hiện: **Lê Thị Minh Nguyệt, 19110413** | |
| Thời gian thực hiện: **23/02/2021** đến **10/06/2021** | |
| **Yêu cầu của đề tài**  **Lý thuyết:** Vận dụng AI vào thực tiễn và tiếp cận thành tựu tiên tiến trong lĩnh vực AI:  + Nền tảng và đặc trưng của các dạng bài toán AI và giải thuật tương ứng.  + Một số phương pháp biểu diễn không gian (trạng thái) giải quyết bài toán AI: biểu diễn tri thức và suy diễn logic.  + Một số phương pháp thông dụng trong tìm kiếm lời giải bài toán AI = giải quyết bài toán thông qua tìm kiếm lời giải (Solving Problems by Searching), đặc biệt là DFS và BFS, …  **Thực hành:** PHÁT TRIỂN HỆ THỐNG GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO CÁC GIẢI THUẬT TRÍ TUỆ NHÂN TẠO (GIẢI THUẬT DLS KẾT HỢP VỚI IDS)  + Thư viện chính được dùng: itertools  + Giải thuật chính: tìm kiếm sâu dần IDS  + Các chức năng chính của sản phẩm: trò chơi tự phá, giải xếp hình 9 ô gạch | |
| **GIẢNG VIÊN** | **Ngày 23 tháng 02 năm 2021**  **SV Thực hiện** |

# GIỚI THIỆU VỀ ĐỀ TÀI

## Tổng quan về đề tài

Phần mềm được viết trên nền Python, sử dụng giao diện đồ họa để mô phỏng trò chơi và giải thuật DLS kết hợp với IDS để tìm đường đi. Ở trạng thái ban đầu, các ô được sắp xếp ngẫu nhiên, và nhiệm vụ của người chơi là tìm được cách đưa chúng về trạng thái đích (ô đầu trống, các ô khác theo thứ tự tăng dần từ trái qua phải, từ trên xuống dưới). Người dùng có thể sử dụng chuột/bàn phím chơi với các kích thước khác nhau và với hình ảnh khác nhau hoặc có thể sử dụng chức năng tìm lời giải nhờ giải thuật DLS kết hợp IDS.

Phầm mềm sử dụng một số thư viện như random, sys, collection, time, tkinter, PIL và đặc biệt sử dụng thư viện intertools hỗ trợ nhiều trong sử dụng giải thuật IDS để giải quyết bài toán.

## Nội dung chuyên môn chính của đề tài

* **Các thư viện được sử dụng trong đề tài:**

+ Thư viện random: để tạo các giá trị ngẫu nhiên để tạo ngẫu nhiên trạng thái của các mảnh ghép của bức hình.

+ Thư viện sys: cung cấp các hàm và các biến được sử dụng để thao tác các phần khác nhau của môi trường chạy Python.

+ Module collection trong Python được định nghĩa là một bộ chứa được sử dụng để lưu trữ các bộ sưu tập dữ liệu, ví dụ: list, dict, set và tuple, …

+ Module time dùng để xử lý các tác vụ liên quan đến thời gian bắt đầu và kết thúc để tìm ra giải thuật.

+ Thư viện tkinter: để tạo giao diện cho người dùng.

+ Thư viện PIL: dùng để xử lý hình ảnh tức bức tranh để ghép.

+ Thư viện itertools: tạo các trình vòng lặp phức tạp, giúp đạt được thời gian thực thi nhanh hơn và viết mã tiết kiệm bộ nhớ được ứng dụng trong thuật toán IDS.

* **Giải thuật: sử dụng thuật toán tìm kiếm sâu dần IDS kết hợp với DLS**

+ Phương pháp: tìm theo DFS nhưng không bao giờ mở rộng các nút có độ sâu quá một giới hạn nào đó. Giới hạn độ sâu sẽ được tăng dần cho đến khi tìm được lời giải.

+ Mô tả cơ bản về giải thuật:

𝐼𝐷𝑆(𝑄, 𝑆, 𝐺, 𝑃)

(𝑄: không gian trạng thái, 𝑆: trạng thái bắt đầu, 𝐺: đích, 𝑃: hành động)

Đầu vào: bài toán tìm kiếm

Đầu ra: đường tới nút đích

Khởi tạo: tập các nút biên (nút mở) 𝑂 = 𝑆

𝑐 = 0 là độ sâu hiện thời

while (1) do

1. while (𝑂 ≠ Ø) do

a. lấy nút đầu tiên 𝑛 khỏi 𝑂

b. if 𝑛 ∈ 𝐺, return (đường đi tới n)

c. if 𝑑𝑒𝑝𝑡ℎ(𝑛) ≤ 𝑐 then thêm 𝑃(𝑛) vào đầu 𝑂

2. 𝑐 + +; 𝑂 =S

## Ngôn ngữ lập trình và công cụ sử dụng

Ngôn ngữ lập trình: Python (Python 3.8)

Công cụ sử dụng: Spyder (Anaconda3)

## Sản phẩm của đề tài

Sản phẩm là một trò chơi ghép tranh có thể tự đưa ra lời giải cho số bước di chuyển ngắn nhất để hoàn thành trò chơi trong thời gian nhanh nhất bằng giải thuật trí tuệ nhân tạo (cụ thể là thuật toán tìm kiếm sâu dần IDS kết hợp với DLS).

## Bố cục của báo cáo

Chương 1: Giới thiệu tổng quan về đề tài

Chương 2: Cơ sở lý thuyết của học phần: gồm những lý thuyết cơ bản về AI liên quan đến đề tài.

Chương 3: Giới thiệu về thư viện và giải thuật AI: Phát triển hệ thống game ghép tranh dựa trên game 8-puzzle dựa vào các giải thuật trí tuệ nhân tạo (thuật toán DLS kết hợp với IDS)

Chương 4: Mô hình bài toán AI và cài đặt giải thuật tìm kiếm lời giải

Chương 5: Giới thiệu về sản phẩm: hệ thống AI Phát triển hệ thống game ghép tranh dựa trên game 8-puzzle dựa vào các giải thuật trí tuệ nhân tạo (thuật toán DLS kết hợp với IDS)

Chương 6: Kết luận về kết quả đạt được và những tồn tại, trên cơ cở đó đề xuất các giải pháp khắc phục tồn và hướng mở rộng đề tài.

# CƠ SỞ LÝ THUYẾT VỀ TRÍ TUỆ NHÂN TẠO

## Tổng quan về Trí tuệ nhân tạo (AI)

### Các khái niệm

* + - 1. Trí tuệ nhân tạo = AI (Artificial Intelligence)
      2. **Phân loại hệ thống Trí tuệ nhân tạo**
      3. **Turing Test kiểm tra khả năng khôn của các hệ thống AI**
      4. **Tác tử (Agent: Intelligent Agent) trong hệ thống AI**
      5. **Tri thức là gì (Knowledge)?**
      6. **Cơ sở tri thức (Knowledge Base: KB)**
      7. **Hệ quản trị cơ sở tri thức (KBMS)**

### Vai trò AI

### Nền tảng kỹ thuật của AI

### Các lĩnh vực nghiên cứu và ứng dụng cơ bản của AI

## Giới thiệu về các bài toán (vấn đề) AI

### Một số khái niệm

* + - 1. **AI Problem (Bài toán AI = Vấn đề AI)**
      2. **Các yếu tố cơ bản xác định bài toán AI**
      3. **(Hàm) Mục tiêu của Bài toán AI**

### Tìm kiếm lời giải (searching) cho AI Problem

* + - 1. **Một số dạng bài toán (vấn đề) AI**
      2. **Các cấp độ tìm kiếm lời giải cho bài toán (vấn đề) AI**

## Biểu diễn một số bài toán AI

### Các bài toán Trò chơi

### Cây không gian trạng thái của Bài toán AI

* + - 1. **Định nghĩa**
      2. **Phương pháp giải quyết bài toán AI**
         1. **Phương pháp gia tăng**
         2. **Phương pháp trạng thái**
      3. **Sự bùng nổ không gian trạng thái**

## Các phương pháp biểu diễn tri thức

### Phân loại tri thức theo phương pháp biểu diễn

* + - 1. **Tri thức sự kiện**
      2. **Tri thức mô tả**
      3. **Tri thức thủ tục**
      4. **Tri thức mêta**
      5. **Tri thức có cấu trúc**

### Phân loại tri thức theo nền tảng hình thành Cơ sở tri thức

* + - 1. **Tri thức hệ Heuristic**
      2. **Tri thức hệ chuyên gia (ES = Expert System)**
      3. **So sánh Tri thức hệ Heuristic với tri thức hệ chuyên gia (ES)**

### Logic mệnh đề và Logic vị từ

* + - 1. **Logic Mệnh đề**
      2. **Logic vị từ** 
         1. **Vị từ và lượng từ**
         2. **Biểu diễn Logic vị từ và chân trị của biểu diễn tri thức vị từ**

##### Mở rộng Logic vị từ

##### Logic vị từ bậc||cấp cao

##### Logic tình huống

##### Logic trạng thái (modal)

##### Logic xác suất và Logic khả xuất

### Biểu diễn tri thức bằng Frame (Cấu trúc = "Khung")

### Suy diễn tri thức bằng luật dẫn xuất

#### Khái niệm

#### Luật dẫn xuất trong Cơ sở tri thức: Suy diễn tiến

### Biểu diễn suy luận bằng đồ thị AND/OR

### Các biểu diễn tương đương trong Logic mệnh đề và Logic vị từ

### Xử lý ngôn ngữ tự nhiên (Natural Language Processing = NLP): Voice Assistant

## Các phương pháp tìm kiếm lời giải của bài toán AI

### Giới thiệu

* + - 1. **Các tiêu chí đánh giá**
      2. **Phân loại các phương pháp (chiến lược) tìm kiếm lời giải**

### Các phương pháp tìm kiếm mù (Blind Search) = tìm kiếm không có thêm thông tin (uninformed search)

* + - 1. **Depth-First Search (DFS): Tìm kiếm theo chiều sâu**
      2. **Breadth-First Search (BFS): Tìm kiếm theo chiều rộng || ngang**
      3. **Uniform Cost Search (UCS): Tìm kiếm với chi phí ĐỀU (không thay đổi) = chi phí cực tiểu [Áp dụng thuật toán Dijkstra]**
      4. **Interative Deepening Search (IDS) = Tìm kiếm sâu dần = cải tiến DFS**
      5. **Depth-Limited Search (DLS) = Tìm kiếm giới hạn độ sâu = cải tiến DFS**

### Các phương pháp tìm kiếm theo kinh nghiệm (Heuristic Search) = tìm kiếm với thông tin bổ sung (Informed Search) = Cải tiến BFS

* + - 1. **Best first: BestBFS (Greedy\_BFS)**
      2. **A\* và SMA\* (A\* với bộ nhớ giới hạn): tìm kiếm trên đồi thị (Graph Search)**
      3. **Hill-Climbing Search (Tìm kiếm tối ưu cục bộ || leo đồi|| lep núi) = HCS**
      4. **Chiến lựơc tìm kiếm đối kháng (Board Game) với Giải thuật MiniMax||MinMax**

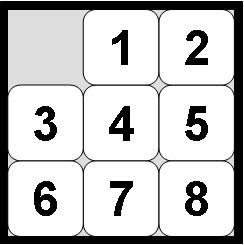
### Lập trình Logic (Logic Programming) phát triển cơ chế lập luận logic

# GIỚI THIỆU VỀ MÔ HÌNH BÀI TOÁN AI: GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP VỚI IDS

## Mô tả bài toán AI

Game ghép hình hay ghép tranh (N-Puzzle) là một trò chơi khá hay và trí tuệ, nó được biết đến với nhiều phiên bản và tên gọi khác nhau như: 8-puzzle, 15-puzzle, Gem puzzle, Boss puzzle. Bài toán N-puzzle là vấn đề cổ điển cho mô hình thuật toán liên quan đến trí tuệ nhân tạo.

Bài toán gồm một bảng 3x3 với các ô số được đánh từ 1-> 8 và một ô trống. Ở trạng thái bắt đầu, các ô được sắp đặt ngẫu nhiên, và nhiệm vụ của người chơi là di chuyển các ô của bảng với số lần di chuyển càng ít càng tốt về trạng thái đích (lớn dần từ trái qua phải và từ trên xuống) như minh họa dưới:



Về thuật toán giải bài toán này, có khá là nhiều cách:

+ Breadth-first Search (BFS)

+ Depth-first Search (DFS)

+ Iterative Deepening Search (IDS)

+ Greedy Search

+ A\* Search

Phần mềm của đề tài này được sử dụng giải thuật DLS kết hợp với IDS

## Các yếu tố xác định bài toán AI

* **4 yếu tố cơ bản xác định bài toán AI**:

+ Tập hợp các trạng thái của bài toán: tổ hợp vị trí các ô

+ Trạng thái bắt đầu: một trạng thái bất kì của các ô mảnh ghép của bức tranh

+ Mục tiêu đặt ra: số lần di chuyển ít nhất để trạng thái giống với trạng thái đích (cho trước)

+ Cơ chế chuyển từ trạng thái này sang trạng thái khác: di chuyển ô trống trái, phải, lên, xuống trên khung hình tương ứng với trạng thái ban đầu

* **Viết và cài đặt hàm mục tiêu**

|  |
| --- |
| class EightPuzzleProblem:  def \_\_init\_\_(self, initial, goal = (0, 1, 2, 3, 4, 5, 6, 7, 8)):  self.initial = initial # trạng thái ban đầu  self.goal = goal # trạng thái đích  def action (self, state): # các action có thể thực hiện  """Trả về chuỗi các Action có thể thực hiện ở trạng thái State hiện tại"""  possible\_actions = ['UP', 'DOWN', 'LEFT', 'RIGHT']  index\_blank\_square = self.find\_blank\_square(state)  # Xóa bớt các nước đi bị hạn chế  if index\_blank\_square % 3 == 0:  possible\_actions.remove('LEFT')  if index\_blank\_square < 3:  possible\_actions.remove('UP')  if index\_blank\_square % 3 == 2:  possible\_actions.remove('RIGHT')  if index\_blank\_square > 5:  possible\_actions.remove('DOWN')  return possible\_actions  def path\_cost(self, c, state1, action, state2):  return c + 1  def goal\_test(self, state):  """Trả về giá trị True nếu State-trạng thái hiện tại trùng với Goal-trạng thái đích"""  return state == self.goal  def find\_blank\_square(self, state):  """Trả về vị trí của ô trống-số 0 trong State"""  return state.index(0)  def Result(self, state, action):  """Ở State hiện tại thực hiện Action để đạt được State mới  Trả về Trạng thái mới-new\_state"""  blank = self.find\_blank\_square(state)  new\_state = list(state)  delta = {'UP': -3, 'DOWN': 3, 'LEFT': -1, 'RIGHT': 1}  neighbor = blank + delta[action]  new\_state[blank], new\_state[neighbor] = new\_state[neighbor], new\_state[blank] # tráo đổi 2 trạng thái (di chuyển ô trắng)  return tuple(new\_state) |

# BIỂU DIỄN BÀI TOÁN AI VÀ CƠ SỞ TRI THỨC: GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP VỚI IDS

## Giới thiệu

Bài toán AI ghép tranh là bài toán có mục tiêu là ghép lại thành bức tranh hoàn chỉnh với số bước di chuyển ngắn nhất. Gồm nhiều trạng thái trong bài toán này và được tập hợp thành cơ sở tri thức. Có sự chuyển đổi trạng thái trong bài toán qua mỗi bước di chuyển trong bài toán.

## Biểu diễn bài toán AI

Bài toán này có nhiều trạng thái khác nhau: mỗi bước di chuyển là một trạng thái của bài toán.

Trạng thái bắt đầu của bài toán các mảnh ghép của bức tranh được sắp đặt một cách ngẫu nhiên.

Sự chuyển đổi trạng thái của bài toán là khi người chơi di chuyển ô trống.

Mục tiêu của bài toán là đưa các mảnh ghép về đúng vị trí của nó, khi đó ta sẽ được một bức tranh hoàn chỉnh.

## Xây dựng cơ sở tri thức (KBS) cho bài toán AI

Khi các mảnh ghép của bức tranh được sắp đặt một cách ngẫu nhiên thì máy tính sẽ bắt đầu tính toán trong thời gian nhanh nhất có thể để tìm ra tất cả các lời giải và đưa ra lời giải có số bước di chuyển ngắn nhất để hoàn thiện bức tranh.

# PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI CHO BÀI TOÁN AI: GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP VỚI IDS

## Giới thiệu

Đề tài này sử dụng phương pháp tìm kiếm mù: IDS và DLS

Nhận xét về phương pháp này là mang ưu điểm của cả BFS và DFS:

+ Có tính đầy đủ tức thuật toán có khả năng tìm ra lời giải

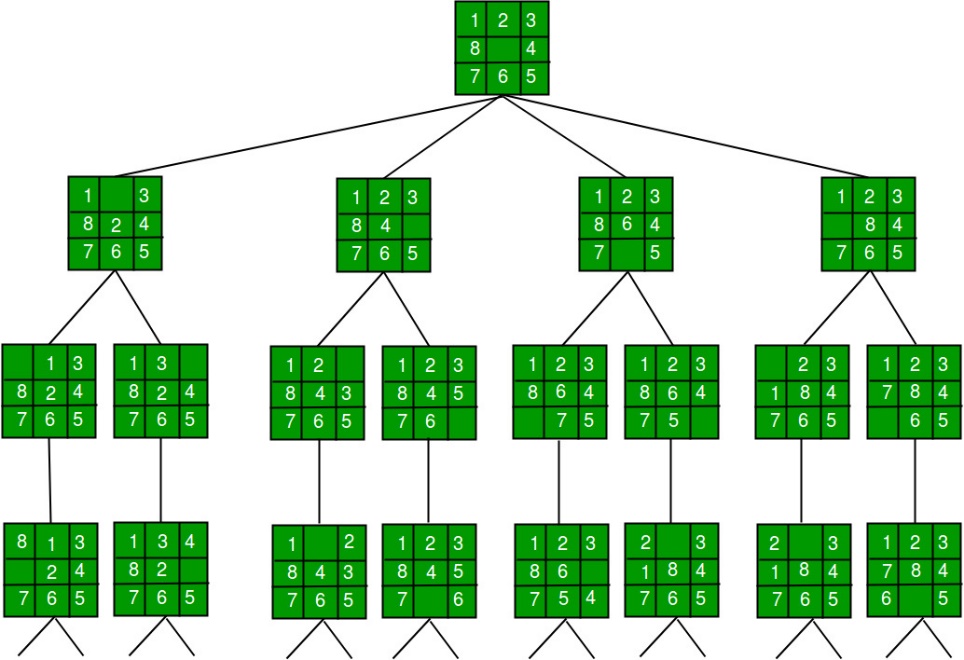
+ Có tính tối ưu: nếu có nhiều lời giải, có thể tìm ra lời giải gần gốc nhất

+ Bộ nhớ: 0 (bd): nhỏ

+ Thời gian: (𝑑 + 1)1 + 𝑑𝑏 + (𝑑 – 1) 𝑏2+ . . . + 2𝑏𝑑−1 + 1𝑏𝑑 = 𝑂(𝑏d)

## Mô tả sơ bộ về mô hình cây không gian trạng thái của bài toán AI

Chúng ta có thể thực hiện tìm kiếm theo chiều sâu trên không gian trạng thái (Tập hợp tất cả các cấu hình của một bài toán đã cho tức là tất cả các trạng thái có thể đạt được từ trạng thái ban đầu) cây.



Trong giải pháp này, các bước di chuyển liên tiếp có thể đưa chúng ta đi khỏi mục tiêu hơn là đưa chúng ta đến gần hơn. Việc tìm kiếm cây không gian trạng thái đi theo con đường ngoài cùng bên trái từ gốc bất kể trạng thái ban đầu.

## Xác định phương pháp tìm kiếm lới giải cho bài toán AI: game ghép tranh dựa trên game 8-puzzle dựa vào giải thuật DLS kết hợp với IDS

Việc tìm kiếm nút trả lời thường có thể được tăng tốc bằng cách sử dụng hàm xếp hạng “thông minh”, còn được gọi là hàm chi phí gần đúng để tránh tìm kiếm trong các cây con không chứa nút trả lời.

Hàm chi phí (số lần di chuyển):

Mỗi nút X trong cây tìm kiếm được liên kết với một chi phí. Hàm chi phí rất hữu ích để xác định nút E tiếp theo. Nút E tiếp theo là nút có chi phí thấp nhất. Hàm chi phí được định nghĩa là:

C (X) = g (X) + h (X), trong đó:

g (X) = chi phí đến được nút hiện tại từ gốc

h (X) = chi phí đến được nút trả lời từ X.

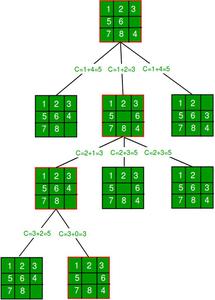
Tôi cho rằng việc di chuyển một ô theo hướng nào sẽ có một chi phí 1 đơn vị. Lưu ý điều đó, chúng tôi xác định hàm chi phí cho thuật toán 8 puzzle như sau:

c (x) = f (x) + h (x), trong đó

f (x) là độ dài của đường đi từ gốc đến x (số lần di chuyển cho đến nay)

h (x) là số ô trống không ở vị trí mục tiêu của họ (số lượng ô xếp không đúng vị trí). Có ít nhất h (x) di chuyển để chuyển trạng thái x thành trạng thái mục tiêu

Sơ đồ dưới đây cho thấy đường dẫn được theo sau bởi thuật toán để đạt được vị trí đích từ vị trí ban đầu đã cho của 8-Puzzle. Lưu ý rằng chỉ sử dụng các nút có giá trị nhỏ nhất của hàm chi phí.



## Cài đặt (lập trình) phương pháp tìm kiếm lời giải cho bài toán AI: Game ghép tranh dựa trên game 8-puzzle dựa vào giải thuật DLS kết hợp với IDS

Lập trình phương pháp tìm kiếm lời giải cho bài toán:

|  |
| --- |
| def DepthLimitSearch(node, problem, limit):  frontier = list([node]) # Stack ngăn xếp chưa các node sẽ duyệt qua  explored = list() # chứa các trạng thái đã được duyệt qua  explored\_depth = list() # chứa độ sâu tương ứng của mỗi trạng thái đã duyệt qua  while frontier:  node = frontier.pop()  if node.state not in explored:  explored.append(node.state)  explored\_depth.append(node.depth)  if problem.goal\_test(node.state):  return node  else:  for child in node.Expand(problem):  # Kiểm tra child thỏa 3 tiêu chí:  # - Không nằm trong danh sách đã duyệt (nếu đã có thì độ sâu depth phải thấp hơn độ sâu của trạng thái đã duyệt)  # - Không nằm trong danh sách sẽ duyệt  # - Độ sâu hiện tại không vượt quá Limit  if child.depth <= limit:  if child not in frontier:  if child.state not in explored:  if problem.goal\_test(child.state):  return child  else: frontier.append(child) # Đẩy node mới thỏa điều kiện vào Hàng đợi  elif explored\_depth[explored.index(child.state)] > child.depth:  explored\_depth[explored.index(child.state)] = child.depth  if problem.goal\_test(child.state):  return child  else: frontier.append(child) # Đẩy node mới thỏa điều kiện vào Hàng đợi  else: break  return None  def IDLS(problem):  """Duyệt qua limit từ 0 đến vô cùng để thực hiện thuật toán tìm kiếm theo chiều sâu  Trả về Kết quả result"""  depthlimit = 0  # Sử dụng itertools để chạy vòng lặp vô tận ( mục đích duyệt limit đến vô cùng )\_Thuật toán IDLS  for depthlimit in itertools.count():  result = DepthLimitSearch(node, problem, limit=depthlimit)  if DepthLimitSearch(node, problem, limit=depthlimit) != None:  break  return result |

# GIỚI THIỆU SẢN PHẨM CỦA ĐỀ TÀI AI: GAME GHÉP TRANH DỰA TRÊN GAME 8-PUZZLE DỰA VÀO GIẢI THUẬT DLS KẾT HỢP VỚI IDS

## Giới thiệu sản phẩm bài toán AI: game ghép tranh dựa trên game 8-puzzle dựa vào giải thuật DLS kết hợp với IDS

Tổng quan về sản phẩm:

+ Tên gọi: game ghép tranh

+ Ngôn ngữ lập trình: python (python 3.8)

+ Công cụ cài đặt: Spyder (Anaconda3)

Full code cài đặt các hàm xử lý thuật toán của bài toán AI:

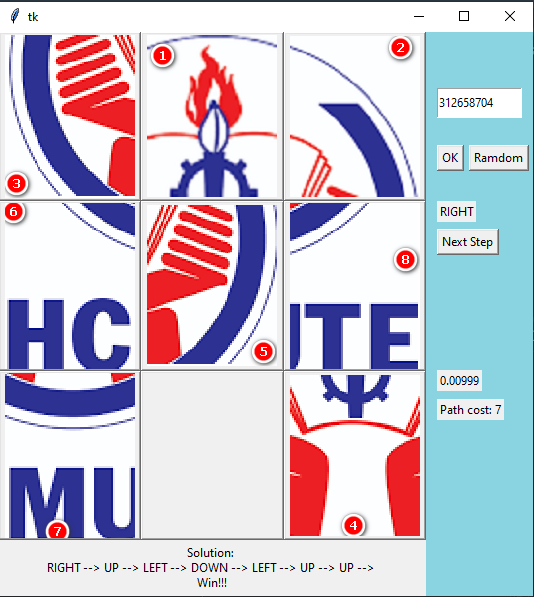
|  |
| --- |
| import sys  import random as rd  from random import choice  from collections import deque  import time  import itertools  class Node:  def \_\_init\_\_(self, state, parent = None, action = None, path\_cost = 0):  self.state = state # trạng thái hiện tại của hình cần xếp  self.parent = parent # trạng thái trước đó  self.action = action # bước đi ~ 'UP', 'DOWN', 'LEFT', 'RIGHT'  self.path\_cost = path\_cost  self.depth = 0 # độ sâu hiện tại  if parent:  self.depth = parent.depth + 1  def Expand(self, problem): # Tập các node con được sinh ra  """Trả về danh sách các Node con được sinh ra với các Action-bước đi tương ứng"""  return [self.Child\_Node(problem, action) for action in problem.action(self.state)]  def Child\_Node(self, problem, action): # Khởi tạo node con kế tiếp  """Trả về một ChildNode-Nút con mới tạo ra khi thực hiện Action"""  next\_state = problem.Result(self.state, action)  next\_node = Node(next\_state, self, action, problem.path\_cost(self.path\_cost, self.state, action, next\_state))  return next\_node  def solution(self): # chuỗi action  """Xuất ra danh sách thứ tự các action dẫn đến trạng thái node hiện tại"""  return [node.action for node in self.path()[1:]]  def path(self): # đường đi  """Xuất ra danh sách các node đã đi qua để dẫn đến trạng thái node hiện tại"""  node, path\_back = self, []  while node:  path\_back.append(node)  node = node.parent  return list(reversed(path\_back))  def DepthLimitSearch(node, problem, limit):  frontier = list([node]) # Stack ngăn xếp chưa các node sẽ duyệt qua  explored = list() # chứa các trạng thái đã được duyệt qua  explored\_depth = list() # chứa độ sâu tương ứng của mỗi trạng thái đã duyệt qua  while frontier:  node = frontier.pop()  if node.state not in explored:  explored.append(node.state)  explored\_depth.append(node.depth)  if problem.goal\_test(node.state):  return node  else:  for child in node.Expand(problem):  # Kiểm tra child thỏa 3 tiêu chí:  # - Không nằm trong danh sách đã duyệt (nếu đã có thì độ sâu depth phải thấp hơn độ sâu của trạng thái đã duyệt)  # - Không nằm trong danh sách sẽ duyệt  # - Độ sâu hiện tại không vượt quá Limit  if child.depth <= limit:  if child not in frontier:  if child.state not in explored:  if problem.goal\_test(child.state):  return child  else: frontier.append(child) # Đẩy node mới thỏa điều kiện vào Hàng đợi  elif explored\_depth[explored.index(child.state)] > child.depth:  explored\_depth[explored.index(child.state)] = child.depth  if problem.goal\_test(child.state):  return child  else: frontier.append(child) # Đẩy node mới thỏa điều kiện vào Hàng đợi  else: break  return None  def IDLS(problem):  """Duyệt qua limit từ 0 đến vô cùng để thực hiện thuật toán tìm kiếm theo chiều sâu  Trả về Kết quả result"""  depthlimit = 0  # Sử dụng itertools để chạy vòng lặp vô tận ( mục đích duyệt limit đến vô cùng )\_Thuật toán IDLS  for depthlimit in itertools.count():  result = DepthLimitSearch(node, problem, limit=depthlimit)  if DepthLimitSearch(node, problem, limit=depthlimit) != None:  break  return result  class EightPuzzleProblem:  def \_\_init\_\_(self, initial, goal = (0, 1, 2, 3, 4, 5, 6, 7, 8)):  self.initial = initial # trạng thái ban đầu  self.goal = goal # trạng thái đích  def action (self, state): # các action có thể thực hiện  """Trả về chuỗi các Action có thể thực hiện ở trạng thái State hiện tại"""  possible\_actions = ['UP', 'DOWN', 'LEFT', 'RIGHT']  index\_blank\_square = self.find\_blank\_square(state)  # Xóa bớt các nước đi bị hạn chế  if index\_blank\_square % 3 == 0:  possible\_actions.remove('LEFT')  if index\_blank\_square < 3:  possible\_actions.remove('UP')  if index\_blank\_square % 3 == 2:  possible\_actions.remove('RIGHT')  if index\_blank\_square > 5:  possible\_actions.remove('DOWN')  return possible\_actions  def path\_cost(self, c, state1, action, state2):  return c + 1  def goal\_test(self, state):  """Trả về giá trị True nếu State-trạng thái hiện tại trùng với Goal-trạng thái đích"""  return state == self.goal  def find\_blank\_square(self, state):  """Trả về vị trí của ô trống-số 0 trong State"""  return state.index(0)  def Result(self, state, action):  """Ở State hiện tại thực hiện Action để đạt được State mới  Trả về Trạng thái mới-new\_state"""  blank = self.find\_blank\_square(state)  new\_state = list(state)  delta = {'UP': -3, 'DOWN': 3, 'LEFT': -1, 'RIGHT': 1}  neighbor = blank + delta[action]  new\_state[blank], new\_state[neighbor] = new\_state[neighbor], new\_state[blank] # tráo đổi 2 trạng thái (di chuyển ô trắng)  return tuple(new\_state)  def random(problem, random\_level):  """Trả về một Node có state bất kì"""  x = rd.randint(20,random\_level)  node = Node(problem.goal)  exlored = set()  exlored.add(node.state)  while x > 0:  temp = choice(node.Expand(problem))  while temp.state in exlored:  temp = choice(node.Expand(problem))  node = temp  exlored.add(node)  x = x - 1  return node  def final(initialState):  global problem  #initial = tuple([int(input()),int(input()),int(input()),int(input()),int(input()),int(input()),int(input()),int(input()),int(input())]  problem = EightPuzzleProblem(initial=None, goal=(0, 1, 2, 3, 4, 5, 6, 7, 8))  problem.initial = initialState #random(problem,random\_level=20).state  # print(problem.initial)  global node  node = Node(problem.initial)  start = time.time()  result = IDLS(problem)  end = time.time()  # print(end - start)  # print(len(result.solution()))  yield result.solution()  yield end-start |

Code tạo giao diện và chạy thử nghiệm trên một bức tranh:

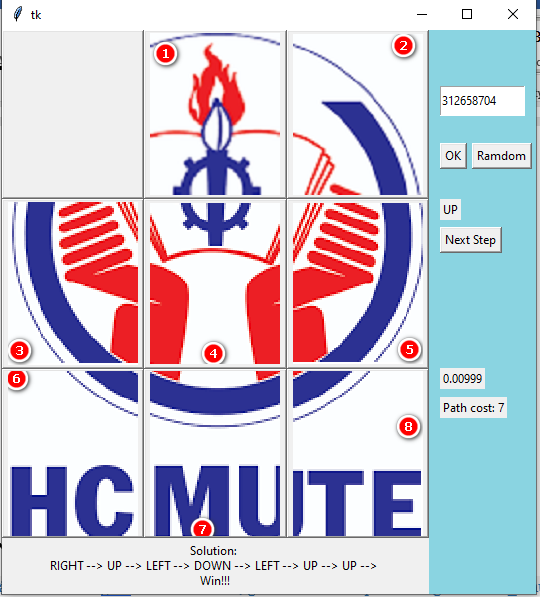
|  |
| --- |
| import tkinter as tk  import tkinter.ttk as ttk  import sys  import time  from Eight\_Puzzle\_Final import \*  from PIL import Image, ImageTk  import time  import itertools  solution =[]  num=[]  #đưa hình ảnh xếp ngẫu nhiên khi click nút random  def set\_text(num):  entryInput.delete(0, tk.END)  text=''  text = ''.join(str(v) for v in num)  entryInput.insert(0,text)  return  #hình ảnh và trạng thái trở lại ban đầu  def clearButton():  for i in range(3):  for j in range(3):  listBt[i][j]['text']=''  listBt[i][j]['image']=''  listBt[i][j]['state']='normal'    def createTable(num):  for i in range(3):  for j in range(3):  listBt[i][j]["text"]=num[i\*3+j]  if(listBt[i][j]["text"]!=0):  listBt[i][j]["image"]=photo[num[i\*3+j]]  else:  listBt[i][j]["text"]=''  #hiện thị phương pháp giải ở cuối màn hình  def setLabel():  solutionString = 'Solution: \n'  global solution  for i in range(len(solution)):  solutionString=solutionString+ solution[i] +' --> '  if(i%7==6):  solutionString=solutionString+'\n'  solutionString = solutionString +'Win!!!'  lbOutput['text']=solutionString  lbPath['text']='Path cost: ' + str(len(solution))  #hoán đổi nút ô trống với nút kế bên  def swapButton(bt0, btAny):  bt0['text']=btAny['text']  bt0['state'] = 'normal'  bt0['image'] = btAny['image']  btAny['text']=''  btAny['image']= ''  btAny['state']='disabled'  #di chuyển theo phương pháp giải  def move():  indx = 0  indy=0  step=''  global solution  if(solution!= []):  step=solution.pop(0)  if(solution!= []):  lbNextstep['text']=solution[0]  for i in range(3):  for j in range(3):  if(listBt[i][j]["text"]==''):  indx=i  indy=j  break  if(step=='LEFT'):  swapButton(listBt[indx][indy], listBt[indx][indy-1])  if(step=='RIGHT'):  swapButton(listBt[indx][indy], listBt[indx][indy+1])  if(step=='UP'):  swapButton(listBt[indx][indy], listBt[indx-1][indy])  if(step=='DOWN'):  swapButton(listBt[indx][indy], listBt[indx+1][indy])    #tìm phương pháp từ một số đưa vào  def findSolution(num):  global solution  solution, time = final(tuple(num))  lbNextstep['text']=solution[0]  lbTime['text'] = "{:.5f}".format(float(time))  setLabel()  #click nút Ok để tìm giải pháp  def ClickOK():  clearButton()  if(len(str(entryInput.get())))==9 and str(entryInput.get())[0]!='0':  num=[int(x) for x in str(entryInput.get())]  createTable(num)  findSolution(num)  #nút Random sẽ làm mất trạng thái hiện tại và tạo ra trạng thái mới của các mảnh ghép và cách giải  def ClickAuto():  clearButton()  problem = EightPuzzleProblem(initial=None, goal=(0, 1, 2, 3, 4, 5, 6, 7, 8))  num = random(problem,random\_level=25).state  set\_text(num)  createTable(num)  findSolution(num)      root = tk.Tk()  #thêm tiện ích như màu sắc, kích thước của các nút  canvas = tk.Canvas(root, height=560, width=530)  canvas.pack()  frame = tk.Frame(root, bg='gray')  frame.place(relwidth=0.8, relheight=0.9)  soluFrame = tk.Frame(root,bg='white')  soluFrame.place(relwidth=0.8, relheight=0.1, relx=0, rely=0.9)  rightFrame = tk.Frame(root, bg='#8ad4e1')  rightFrame.place(relwidth=0.2, relheight=1, relx=0.8, rely=0)  entryInput = tk.Entry(rightFrame)  entryInput.place(relx=0.1, rely=0.1, relwidth=0.8, height=30)  btOK =tk.Button(rightFrame, text='OK', command=ClickOK)  btOK.place(relx=0.1, rely=0.2)  lbNextstep = tk.Label(rightFrame)  lbNextstep.place(relx=0.1, rely=0.3)  btNextstep = tk.Button(rightFrame, text='Next Step', command=move)  btNextstep.place(relx=0.1, rely=0.35)  btAuto = tk.Button(rightFrame, text='Ramdom', command = ClickAuto)  btAuto.place(relx=0.4, rely=0.2)  lbTime = tk.Label(rightFrame)  lbTime.place(relx=0.1, rely=0.6)  lbPath=tk.Label(rightFrame)  lbPath.place(relx=0.1, rely=0.65)  lbOutput = tk.Label(soluFrame)  lbOutput.place(relwidth=1, relheight=1, relx=0, rely=0)  #thêm hình ảnh vào  image0 = Image.open("ute0.png")  image1 = Image.open("ute1.png")  image2 = Image.open("ute2.png")  image3 = Image.open("ute3.png")  image4 = Image.open("ute4.png")  image5 = Image.open("ute5.png")  image6 = Image.open("ute6.png")  image7 = Image.open("ute7.png")  image8 = Image.open("ute8.png")  photo0 = ImageTk.PhotoImage(image0)  photo1 = ImageTk.PhotoImage(image1)  photo2 = ImageTk.PhotoImage(image2)  photo3 = ImageTk.PhotoImage(image3)  photo4 = ImageTk.PhotoImage(image4)  photo5 = ImageTk.PhotoImage(image5)  photo6 = ImageTk.PhotoImage(image6)  photo7 = ImageTk.PhotoImage(image7)  photo8 = ImageTk.PhotoImage(image8)  photo =[photo0, photo1, photo2, photo3, photo4, photo5, photo6, photo7, photo8]  #tạo ra các nút trên màn hình  bt0 = tk.Button(frame)  bt0.place(relx=0, rely=0, relwidth=1/3, relheight=1/3)  bt1 = tk.Button(frame)  bt1.place(relx=1/3, rely=0, relwidth=1/3, relheight=1/3)  bt2 = tk.Button(frame)  bt2.place(relx=2/3, rely=0, relwidth=1/3, relheight=1/3)  bt3 = tk.Button(frame)  bt3.place(relx=0, rely=1/3, relwidth=1/3, relheight=1/3)  bt4 = tk.Button(frame)  bt4.place(relx=1/3, rely=1/3, relwidth=1/3, relheight=1/3)  bt5 = tk.Button(frame)  bt5.place(relx=2/3, rely=1/3, relwidth=1/3, relheight=1/3)  bt6 = tk.Button(frame)  bt6.place(relx=0, rely=2/3, relwidth=1/3, relheight=1/3)  bt7 = tk.Button(frame)  bt7.place(relx=1/3, rely=2/3, relwidth=1/3, relheight=1/3)  bt8 = tk.Button(frame)  bt8.place(relx=2/3, rely=2/3, relwidth=1/3, relheight=1/3)  listBt =[[bt0, bt1, bt2],  [bt3, bt4, bt5],  [bt6, bt7, bt8]]  root.mainloop() |

## Kết quả sản phẩm bài toán AI: Game ghép tranh dựa trên game 8-puzzle dựa vào giải thuật DLS kết hợp với IDS

Trạng thái ban đầu:



Trạng thái đích



## Nhận xét và đánh giá về sản phẩm bài toán AI: game ghép tranh dựa trên game 8-puzzle dựa vào giải thuật DLS kết hợp với IDS

Ưu điểm: đề tài này là sản phẩm dễ code, code ngắn gọn dễ hiểu, có nhiều nguồn tham khảo, giải thuật chạy ổn định và hiệu quả đạt được mục đích đề ra.

Nhược điểm: giải thuật còn chưa tối ưu, giao diện còn chưa đẹp chưa bắt mắt người dùng.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

## Kết luận

### Những kết quả đạt được

Trò chơi có thể tự tìm ra số bước di chuyển ít nhất để chiến thắng trong thời gian nhanh nhất.

Thông qua việc tìm hiểu và nghiên cứu đề tài này giúp có cái nhìn toàn diện hơn trong việc ứng dụng trí tuệ nhân tạo vào giải quyết vấn đề thực tế. Đây là bài toán cổ điển trong trí tuệ nhân tạo cho các thuật toán mô hình hóa liên quan đến tìm kiếm có tri thức bổ sung.

### Hạn chế

Thuật toán IDS tuy nhanh nhưng chưa đảm bảo tính tối ưu trong việc tìm đường đi.

## Hướng phát triển

### Hướng khắc phục các hạn chế

Để khắc phục tính tối ưu trong việc tìm đường đi ta có thể sử dụng thuật toán heuristic nhưng không mở rộng được với mọi không gian trạng thái vì khi A\* xét các nút thì phải tính toán heuristic gây tốn nhiều thời gian.

Hiện vẫn chưa tìm ra cách giải quyết tối ưu cho tất cả không gian trạng thái trò chơi vì kích thước tăng không gian trạng thái sẽ tăng lên rất nhanh, nên phần khắc phục hạn chế sẽ quay lại bổ sung sau.

### Hướng mở rộng đề tài

Sử dụng nhiều thuật toán khác để giải bài toán giúp bài toán được tối ưu hơn như heuristic search, …

Game hiện tại là dựa trên game 8-puzzle thì sẽ tăng số 8 đó lên cao hơn như 15…

Giao diện đẹp bắt mắt, có nhiều tranh cho người chơi lựa chọn.

Người chơi có thể sử dụng giọng nói để chơi game thay vào các nút di chuyển thông thường.

# DANH MỤC TÀI LIỆU THAM KHẢO

1. Võ Xuân Thể (2021), *Bài giảng học phần Trí tuệ nhân tạo*, TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM (lưu hành nội bộ).

[a]. Mẫu giải thuật IDS: <https://drive.google.com/file/d/1ETiq6jE12yHC6CXmrIniGPabz7FQTh_-/view?usp=sharing>

[b]. [www.cs.umsl.edu/~sanjiv/classes/cs5130/lectures/bb.pdf](http://www.cs.umsl.edu/~sanjiv/classes/cs5130/lectures/bb.pdf)   
<https://www.seas.gwu.edu/~bell/csci212/Branch_and_Bound.pdf>

# ĐÁNH GIÁ KẾT QUẢ HỌC TẬP HỌC PHẦN

ĐỒ ÁN HP THAY CHO BÀI THI KẾT THÚC HỌC PHẦN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Hình thức KT** | **Nội dung** | **Chuẩn đầu ra đánh giá** | **Trình độ năng lực** | **Phương pháp đánh giá** | **Công cụ đánh giá** | **Tỉ lệ (%)** |
| **BÀI TẬP LẬP TRÌNH** | | | | | |  |
| BL#1 | Bài tập lập trình theo từng Buổi thực hành = tính bình quân  Kiểm tra viết | G1.1 | 3 | Bài tập  Kiểm tra | Bài tập  Kiểm tra | 50 |
| G2.1 | 3 |
| G3.1  G3.2 | 4  3 |
| G4.1 | 5 |
| **ĐỒ ÁN HỌC PHẦN (Project)** | | | | | |  |
| ĐA#1 | Phát triển một hệ thống AI đơn giản:  + Đề tài cá nhân  + Xây dựng và thực nghiệm giải thuật AI theo bài toán cá nhân  => báo cáo vào ngày thi. | G1.1 | 3 | Báo cáo, Demo và thuyết trình | Rubric | 50 |
| G2.1 | 3 |
| G3.1 | 4 |
| G3.2  G4.1 | 3  5 |
| G2.1 | 4 |
| G3.1 | 4 |
| G3.2  G4.1 | 3  5 |