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

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN CUỐI KỲ  
MÔN HỌC: TRÍ TUỆ NHÂN TẠO**

**GIẢI VÀ MÔ PHỎNG TRÒ CHƠI SOKOBAN BẰNG CÁC THUẬT TOÁN TÌM KIẾM**

MÃ SỐ HP:241ARIN330585\_03CLC

GVHD: TS. Phan Thị Huyền Trang

SVTH: Nguyễn Hoài Bảo 22110108

Phan Phúc Hảo 22110134

Nguyễn Võ Cát Tường 22110263

**TP.HCM, ngày 02 Tháng 12 năm 2024**

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

**KHOA [CÔNG NGHỆ THÔNG TIN|CHẤT LƯỢNG CAO]**

**ĐỀ TÀI ĐỒ ÁN HỌC PHẦN**

TÊN HỌC PHẦN: **Trí tuệ nhân tạo**

MÃ SỐ HP: **241ARIN330585\_03CLC**

Tên đề tài: **GIẢI VÀ MÔ PHỎNG TRÒ CHƠI SOKOBAN BẰNG CÁC THUẬT TOÁN TÌM KIẾM**

**NHẬN XÉT CỦA GIẢNG VIÊN**

**………………………………………………………..**

**………………………………………………………..**

**………………………………………………………..**

**………………………………………………………..**

**………………………………………………………..**

**………………………………………………………..**

**………………………………………………………..**

**………………………………………………………..**

**Ngày 02/12/2024**

**Giảng viên: Ký tên**

**TP.HCM, ngày 02 tháng 12 năm 2024**

# LỜI CẢM ƠN

Nhóm em xin gửi lời cảm ơn chân thành và sâu sắc đến giảng viên Phan Thị Huyền Trang, người đã không quản ngại thời gian và công sức để tận tâm giảng dạy, hướng dẫn và động viên chúng em trong suốt quá trình học tập và thực hiện đồ án môn Trí tuệ Nhân tạo.

Sự tận tình của cô không chỉ giúp chúng em có được nền tảng kiến thức lý thuyết vững chắc mà còn truyền cảm hứng để chúng em phát huy tinh thần sáng tạo và tư duy logic trong việc áp dụng các thuật toán trí tuệ nhân tạo vào thực tiễn. Trong suốt quá trình làm đồ án, cô không chỉ đóng vai trò là một người hướng dẫn mà còn là một người đồng hành, luôn sẵn sàng giải đáp mọi thắc mắc, hỗ trợ và khuyến khích chúng em vượt qua những khó khăn, thử thách.

Những lời khuyên quý báu của cô không chỉ giúp chúng em hoàn thiện đồ án một cách tốt nhất mà còn mang lại cho chúng em nhiều bài học ý nghĩa, cả trong học tập lẫn cuộc sống. Chính sự nhiệt huyết, chu đáo và sự quan tâm của cô đã tạo động lực mạnh mẽ để chúng em không ngừng cố gắng, hoàn thành nhiệm vụ với tinh thần trách nhiệm cao.

Ngoài ra, chúng em cũng xin gửi lời cảm ơn đến Nhà trường và Khoa Công nghệ Thông tin đã tạo điều kiện để chúng em có cơ hội học tập và thực hiện đồ án trong môi trường giáo dục chất lượng cao, được trang bị các nguồn tài nguyên học thuật phong phú và cơ sở vật chất hiện đại.

Cuối cùng, nhóm em cũng xin cảm ơn nhau – các thành viên trong nhóm, vì tinh thần đoàn kết, sự nỗ lực và phối hợp chặt chẽ để vượt qua những khó khăn trong quá trình thực hiện. Chúng em tin rằng những gì đã học được và trải nghiệm qua đồ án này sẽ là nền tảng vững chắc cho con đường học tập và sự nghiệp trong tương lai.

Một lần nữa, chúng em xin chân thành cảm ơn cô Phan Thị Huyền Trang vì những đóng góp to lớn trong hành trình học tập của chúng em.

**LỜI CAM KẾT**

Chúng em xin cam đoan rằng báo cáo này là kết quả của quá trình làm việc nghiêm túc, sáng tạo và độc lập của các thành viên trong nhóm. Toàn bộ nội dung trong báo cáo, bao gồm các phân tích, thiết kế, và triển khai, đều được thực hiện bởi chính các thành viên trong nhóm dưới sự hướng dẫn tận tình của giảng viên phụ trách.

Trong suốt quá trình nghiên cứu và thực hiện, chúng em đã tuân thủ đầy đủ các quy định về học thuật, đặc biệt là về đạo đức trong nghiên cứu. Chúng em không sao chép, trích dẫn tài liệu, mã nguồn hoặc ý tưởng của người khác mà không có sự ghi nguồn rõ ràng và phù hợp.

Nếu phát hiện bất kỳ hành vi nào liên quan đến sao chép hoặc vi phạm đạo văn trong quá trình thực hiện, chúng em xin hoàn toàn chịu trách nhiệm trước Nhà trường và giảng viên hướng dẫn.

Chúng em hy vọng rằng báo cáo này sẽ góp phần thể hiện năng lực học tập và tinh thần trách nhiệm của nhóm đối với môn học, đồng thời mang lại giá trị học thuật và thực tiễn hữu ích.

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
|  |  |
|  |  |

**DANH MỤC CÁC THUẬT NGỮ**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**MỤC LỤC**

[LỜI CẢM ƠN 3](#_Toc183856528)

[**LỜI CAM KẾT** 4](#_Toc183856529)

[DANH MỤC CÁC TỪ VIẾT TẮT 5](#_Toc183856530)

[**DANH MỤC CÁC THUẬT NGỮ** 5](#_Toc183856531)

[Chương 1: MỞ ĐẦU 8](#_Toc183856532)

[1.1. GIỚI THIỆU VỀ ĐỀ TÀI 8](#_Toc183856533)

[1.2. LÝ DO CHỌN ĐỀ TÀI 8](#_Toc183856534)

[1.3. PHÁT BIỂU BÀI TOÁN 9](#_Toc183856535)

[1.4. CÁC NỘI DUNG DỰ KIẾN THỰC HIỆN 10](#_Toc183856536)

[1.5. BỐ CỤC CỦA BÁO CÁO 10](#_Toc183856537)

[Chương 2: CƠ SỞ LÝ THUYẾT 11](#_Toc183856538)

[2.1. GIỚI THIỆU VỀ TRÍ TUỆ NHÂN TẠO 11](#_Toc183856539)

[2.2. TỔNG QUAN VỀ CÁC THUẬT TOÁN TÌM KIẾM 11](#_Toc183856540)

[2.2.1. Thuật toán tìm kiếm là gì? 11](#_Toc183856541)

[2.2.2. Phát biểu bài toán tìm kiếm 11](#_Toc183856542)

[2.2.3. Quy trình tìm kiếm trong không gian trạng thái 13](#_Toc183856543)

[2.2.4. Không gian trạng thái tường minh và không tường minh 13](#_Toc183856544)

[2.2.5. Thuật toán tìm kiếm tổng quát 14](#_Toc183856545)

[2.2.6. Các tiêu chuẩn đánh giá thuật toán tìm kiếm 15](#_Toc183856546)

[2.2.7. Thuật toán tìm kiếm mù (Uninformed Search) 16](#_Toc183856547)

[2.2.8. Thuật toán tìm kiếm có thông tin (Informed search) 18](#_Toc183856548)

[2.2.9. Thuật toán tìm kiếm cục bộ (Local search) 20](#_Toc183856549)

[2.2.10. Thuật toán tìm kiếm quay lui (Backtracking) 21](#_Toc183856550)

[2.3. GIỚI THIỆU CÁC THƯ VIỆN VÀ CÔNG CỤ THỰC HIỆN 22](#_Toc183856551)

[Chương 3: PHÂN TÍCH VÀ THIẾT KẾ GIẢI PHÁP 25](#_Toc183856552)

[Chương 4: THỰC NGHIỆM, ĐÁNH GIÁ, PHÂN TÍCH KẾT QUẢ 25](#_Toc183856553)

[Chương 5: KẾT LUẬN 25](#_Toc183856554)

[**TÀI LIỆU THAM KHẢO** 25](#_Toc183856555)

1. **MỞ ĐẦU**

## GIỚI THIỆU VỀ ĐỀ TÀI

Sokoban là một trò chơi giải đố kinh điển được phát triển vào những năm 1980 tại Nhật Bản. Từ "Sokoban" trong tiếng Nhật có nghĩa là “người quản lý kho hàng”, thể hiện đúng bản chất của trò chơi. Người chơi sẽ hóa thân thành một nhân vật có nhiệm vụ đẩy các hộp vào các vị trí được chỉ định trong một bản đồ giới hạn.

Điểm hấp dẫn của Sokoban nằm ở sự kết hợp giữa tính giải trí và tư duy logic. Các bản đồ trong trò chơi có độ khó tăng dần, yêu cầu người chơi phải lập kế hoạch cẩn thận để tránh rơi vào trạng thái bế tắc. Chính điều này đã biến Sokoban trở thành một bài toán lý thú trong lĩnh vực trí tuệ nhân tạo (AI), nơi các thuật toán tìm kiếm được áp dụng để giải quyết bài toán một cách hiệu quả.

Trong dự án này, nhóm nghiên cứu và triển khai các thuật toán trí tuệ nhân tạo như Breadth-First Search (BFS), Depth-First Search (DFS), A\* Search, Simulated Annealing, và Backtracking. Mục tiêu là không chỉ giải quyết bài toán mà còn đánh giá hiệu suất của từng thuật toán khi áp dụng trên các bản đồ với độ phức tạp khác nhau. Đồng thời, nhóm cũng xây dựng một hệ thống Sokoban hoàn chỉnh với giao diện trực quan, tích hợp khả năng hỗ trợ giải tự động bằng các thuật toán tìm kiếm.

## LÝ DO CHỌN ĐỀ TÀI

Sokoban không chỉ là một trò chơi giải trí phổ biến mà còn là một bài toán nghiên cứu kinh điển trong lĩnh vực trí tuệ nhân tạo. Trò chơi này mang lại cơ hội thực tiễn để áp dụng và kiểm chứng hiệu quả của các thuật toán tìm kiếm, giúp nhóm làm quen với việc triển khai các thuật toán trên các bài toán có tính ứng dụng cao. Đặc biệt, Sokoban có thể mô phỏng các bài toán tối ưu hóa trong thực tế, như quản lý kho hàng hoặc lập kế hoạch di chuyển.

Việc thực hiện đề tài Sokoban còn giúp nhóm phát triển kỹ năng tư duy logic và lập trình. Trò chơi đặt ra nhiều thách thức, bao gồm xử lý trạng thái bế tắc (deadlock), tối ưu hóa thuật toán để giảm thời gian thực hiện, và thiết kế giao diện đồ họa thân thiện với người dùng. Đây là cơ hội để nhóm rèn luyện các kỹ năng cần thiết cho việc phát triển các ứng dụng trí tuệ nhân tạo thực tiễn.

Ngoài ra, Sokoban mang tính linh hoạt cao, cho phép nhóm thử nghiệm nhiều ý tưởng sáng tạo. Nhóm có thể cải tiến các thuật toán để nâng cao hiệu suất hoặc thiết kế các bản đồ phức tạp hơn để kiểm tra khả năng giải quyết bài toán trong các tình huống đa dạng. Điều này không chỉ giúp nhóm hiểu sâu hơn về các thuật toán mà còn khuyến khích sự sáng tạo trong việc phát triển các giải pháp mới.

Cuối cùng, Sokoban là một đề tài phù hợp với trình độ của nhóm, vừa sức nhưng vẫn đủ thách thức để học hỏi và phát triển. Nó kết hợp hài hòa giữa yếu tố giải trí và học thuật, mang lại một môi trường nghiên cứu và thực nghiệm lý tưởng cho các thuật toán trí tuệ nhân tạo. Với tất cả những lý do trên, nhóm đánh giá rằng Sokoban là lựa chọn lý tưởng để thực hiện đồ án này.

## PHÁT BIỂU BÀI TOÁN

Trò chơi Sokoban có thể được biểu diễn như một bài toán tìm kiếm trong không gian trạng thái. Nhiệm vụ chính của bài toán là di chuyển các hộp (boxes) đến các vị trí đích được chỉ định trên bản đồ, tuân thủ các quy tắc cụ thể. Trạng thái ban đầu được xác định bởi vị trí của công nhân và các hộp trên bản đồ, trong khi trạng thái đích là khi tất cả các hộp được đặt đúng vào vị trí đích.

Trong quá trình thực hiện bài toán, công nhân chỉ có thể đẩy hộp, không được kéo và không được đẩy nhiều hơn một hộp cùng lúc. Các hành động di chuyển phải nằm trong phạm vi giới hạn của bản đồ, đảm bảo không tạo ra trạng thái bế tắc (deadlock). Mỗi bản đồ đại diện cho một thử thách khác nhau với độ phức tạp tăng dần, yêu cầu người chơi hoặc thuật toán phải tính toán đường đi hiệu quả nhất để đạt được mục tiêu.

Bài toán đặt ra yêu cầu tìm một chuỗi hành động (di chuyển và đẩy) để chuyển đổi từ trạng thái ban đầu sang trạng thái đích. Các hành động này cần được tối ưu hóa dựa trên các tiêu chí như số bước di chuyển, thời gian thực hiện, hoặc chi phí tổng thể. Trong dự án này, nhóm đã áp dụng các thuật toán trí tuệ nhân tạo như Breadth-First Search (BFS), Depth-First Search (DFS), A\* Search, Simulated Annealing, và Backtracking để giải quyết bài toán. Mỗi thuật toán được triển khai trên các bản đồ khác nhau nhằm đánh giá hiệu suất và khả năng tìm ra giải pháp tối ưu.

Với cách tiếp cận như trên, bài toán Sokoban không chỉ mang lại tính thách thức mà còn tạo cơ hội để nhóm thực nghiệm các thuật toán tìm kiếm trong trí tuệ nhân tạo, từ đó rút ra bài học về hiệu quả và ứng dụng của từng phương pháp.

## CÁC NỘI DUNG DỰ KIẾN THỰC HIỆN

## BỐ CỤC CỦA BÁO CÁO

1. **CƠ SỞ LÝ THUYẾT**

## GIỚI THIỆU VỀ TRÍ TUỆ NHÂN TẠO

Trí tuệ nhân tạo (AI) là một lĩnh vực nghiên cứu thuộc khoa học máy tính, nhằm phát triển các hệ thống có khả năng thực hiện các nhiệm vụ phức tạp thường đòi hỏi trí tuệ của con người. Các nhiệm vụ này bao gồm: nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, lập luận, và học tập dựa trên dữ liệu.

AI được phân loại thành:

* **AI yếu (Narrow AI)**: Hệ thống chuyên giải quyết một bài toán hoặc nhiệm vụ cụ thể, ví dụ: trợ lý ảo hoặc xe tự lái.
* **AI mạnh (General AI)**: Hệ thống có khả năng thực hiện mọi nhiệm vụ trí tuệ tương tự con người, tuy nhiên loại này hiện vẫn chỉ là lý thuyết.

Trong lịch sử phát triển, AI đã đi từ những ý tưởng ban đầu, như phép thử Turing, đến các ứng dụng thực tế, bao gồm hệ thống dự đoán, hỗ trợ y tế, và tối ưu hóa quy trình.

## TỔNG QUAN VỀ CÁC THUẬT TOÁN TÌM KIẾM

### Thuật toán tìm kiếm là gì?

Thuật toán tìm kiếm là các quy trình được thiết kế để khám phá và xác định con đường từ trạng thái ban đầu đến trạng thái mục tiêu trong không gian trạng thái. Nó được áp dụng rộng rãi trong các bài toán như giải đố, định tuyến đường đi, hoặc tối ưu hóa.

Các yếu tố chính của một bài toán tìm kiếm:

* **Trạng thái**: Tập hợp các điểm hoặc tình huống có thể xảy ra.
* **Hành động**: Các bước di chuyển giữa các trạng thái.
* **Mục tiêu**: Trạng thái cần đạt được.

### Phát biểu bài toán tìm kiếm

Bài toán tìm kiếm là một phương pháp giải quyết vấn đề bằng cách khám phá không gian trạng thái để tìm ra lời giải đáp ứng yêu cầu. Không gian trạng thái bao gồm tất cả các trạng thái liên quan đến bài toán và quá trình tìm kiếm sử dụng các phép chuyển đổi trạng thái để di chuyển từ trạng thái xuất phát đến trạng thái đích.

Bài toán tìm kiếm được mô tả thông qua năm thành phần cơ bản sau:

1. **Tập trạng thái (Q):** Đây là tập hợp toàn bộ các trạng thái có thể xảy ra trong bài toán, được gọi chung là không gian trạng thái. Mỗi trạng thái đại diện cho một tình huống cụ thể hoặc một điểm trong quá trình giải quyết vấn đề.
2. **Trạng thái xuất phát (S):** Là một hoặc nhiều trạng thái ban đầu từ đó thuật toán bắt đầu quá trình tìm kiếm. Các trạng thái xuất phát thường được xác định rõ ràng và nằm trong không gian trạng thái **Q**.
3. **Trạng thái đích (G):** Là một hoặc nhiều trạng thái mà bài toán yêu cầu đạt được. Trạng thái đích có thể được định nghĩa theo hai cách:
   1. **Tường minh:** Cụ thể chỉ ra trạng thái đích là gì (ví dụ: điểm đến cụ thể trên bản đồ).
   2. **Không tường minh:** Được mô tả qua các điều kiện mà trạng thái phải thỏa mãn (ví dụ: trạng thái chiến thắng trong trò chơi cờ vua là khi đối thủ bị chiếu hết).
4. **Các phép chuyển đổi (P):** Là tập hợp các hành động, toán tử hoặc phép biến đổi trạng thái. Mỗi phép chuyển đổi cho phép chuyển từ trạng thái hiện tại sang một hoặc nhiều trạng thái lân cận.
   1. Ký hiệu: **P(n)** là tập hợp các trạng thái có thể đạt được từ trạng thái **n**.
   2. Các trạng thái sinh ra từ một phép chuyển đổi được gọi là **trạng thái lân cận** hoặc **trạng thái hàng xóm**.
5. **Chi phí (c):** Là giá trị đại diện cho chi phí di chuyển giữa hai trạng thái, ký hiệu **c(x,y)**, trong đó **x** và **y** là hai trạng thái liền kề. Chi phí này có thể được xác định dựa trên yêu cầu của bài toán, chẳng hạn:
   1. **Quãng đường:** Chiều dài từ **x** đến **y**.
   2. **Thời gian:** Thời gian cần để di chuyển từ **x** đến **y**.
   3. **Năng lượng tiêu hao:** Chi phí cần để thực hiện hành động.

Tổng chi phí cho một chuỗi chuyển đổi từ trạng thái xuất phát đến trạng thái đích là tổng của tất cả các chi phí di chuyển giữa các trạng thái trung gian. Chi phí này được sử dụng để đánh giá chất lượng của lời giải, với lời giải có chi phí thấp nhất được coi là tối ưu.

### Quy trình tìm kiếm trong không gian trạng thái

1. **Khởi đầu:** Thuật toán bắt đầu từ trạng thái xuất phát **S**.
2. **Khám phá:** Các trạng thái hàng xóm được tạo ra bằng cách áp dụng các phép chuyển đổi **P** lên trạng thái hiện tại.
3. **Kiểm tra mục tiêu:** Sau mỗi lần chuyển đổi, thuật toán kiểm tra xem trạng thái mới có thỏa mãn điều kiện đích **G** hay không.
4. **Kết thúc:** Quá trình tìm kiếm dừng lại khi:
   1. Tìm thấy trạng thái đích (thỏa mãn điều kiện bài toán).
   2. Không còn trạng thái nào để khám phá (với bài toán không có giải).
5. **Trả lời:** Lời giải có thể là:
   1. **Chuỗi chuyển động:** Mô tả các hành động cần thực hiện để đạt đến trạng thái đích.
   2. **Trạng thái đích:** Trong một số bài toán (ví dụ: lập lịch), chỉ cần trả về trạng thái cuối cùng mà không cần chuỗi hành động.

### Không gian trạng thái tường minh và không tường minh

1. **Không gian trạng thái tường minh:** Các trạng thái được liệt kê đầy đủ và cụ thể.

Ví dụ: Trong bài toán tìm đường, mỗi địa điểm được biểu diễn dưới dạng một nút trong đồ thị.

1. **Không gian trạng thái không tường minh:** Không gian trạng thái được suy diễn từ trạng thái xuất phát và các phép chuyển đổi.

Trường hợp này, các trạng thái chỉ xuất hiện khi được tạo ra qua phép chuyển đổi và thường khó xác định toàn bộ không gian trạng thái trước khi tìm kiếm.

1. **Không gian trạng thái có thể là:**
   1. **Hữu hạn:** Chỉ bao gồm số lượng trạng thái giới hạn.
   2. **Vô hạn:** Bao gồm vô số trạng thái, như trong các bài toán suy diễn với số lượng phép chuyển đổi không giới hạn.

### Thuật toán tìm kiếm tổng quát

|  |
| --- |
| Graph\_Search(Q, S, G, P)  Input:  Q: Không gian trạng thái  S: Trạng thái xuất phát  G: Trạng thái đích  P: Tập các hành động  Output:  Chuỗi hành động hoặc trạng thái đích, hoặc thông báo không có giải pháp  1. Khởi tạo:  O ← {S} // Tập các nút biên  D ← ∅ // Tập các nút đã mở rộng (tập đóng)  2. While O ≠ ∅ do:  a. Lấy một nút n từ O  b. If n ∈ G then:  return đường đi tới n  c. Thêm n vào D  d. Sinh các trạng thái lân cận của n (P(n)) và thêm chúng vào O nếu chúng không thuộc D và O  3. Return "Không có giải pháp" |

Dưới đây là sơ đồ minh họa cây tìm kiếm:

* **Start**: Trạng thái xuất phát, tương ứng với gốc của cây.
* **Goal**: Trạng thái đích, là kết quả mà thuật toán tìm kiếm.
* **Các** **nhánh**: Đại diện cho hành động áp dụng trên trạng thái cha để tạo ra trạng thái con.

A diagram of a tree

Description automatically generated

Ưu điểm và hạn chế của thuật toán:

* Ưu điểm:
  + Tổng quát, áp dụng được cho nhiều bài toán khác nhau.
  + Dễ dàng tùy chỉnh thông qua cách mở rộng và kiểm tra trạng thái.
* Hạn chế:
  + Không quy định rõ thứ tự mở rộng nút, do đó hiệu quả phụ thuộc vào chiến lược cụ thể.
  + Có thể tiêu tốn nhiều tài nguyên nếu không gian trạng thái lớn.

### Các tiêu chuẩn đánh giá thuật toán tìm kiếm

Để đánh giá hiệu quả và khả năng áp dụng của các thuật toán tìm kiếm, bốn tiêu chuẩn chính thường được sử dụng như sau:

1. *Tính đầy đủ*: Thuật toán có đảm bảo tìm được lời giải nếu bài toán có lời giải không? Một thuật toán được gọi là đầy đủ nếu luôn tìm thấy lời giải trong trường hợp tồn tại, ngược lại, nó không đầy đủ.

2. *Tính tối ưu*: Thuật toán có thể tìm ra lời giải tốt nhất (với chi phí thấp nhất) khi bài toán có nhiều lời giải không? Tiêu chuẩn tối ưu thường dựa trên giá thành đường đi, thuật toán được coi là tối ưu nếu luôn tìm được lời giải có giá thành nhỏ nhất.

3. *Độ phức tạp tính toán*: Là khối lượng công việc mà thuật toán cần thực hiện để tìm ra lời giải. Điều này được đo lường thông qua số lượng trạng thái (nút) cần sinh ra và kiểm tra trước khi đạt được trạng thái đích.

4. *Yêu cầu bộ nhớ*: Là lượng bộ nhớ tối đa mà thuật toán cần để lưu trữ các trạng thái trong quá trình tìm kiếm. Điều này ảnh hưởng lớn đến khả năng áp dụng thuật toán trong các bài toán có không gian trạng thái lớn.

Các tiêu chuẩn đánh giá thuật toán tìm kiếm phụ thuộc vào độ khó (kích thước) của bài toán, đặc biệt là không gian trạng thái. Khi không gian trạng thái lớn, thuật toán cần nhiều thời gian và tài nguyên hơn để xử lý. Độ khó của bài toán có thể được xác định dựa trên ba tham số sau:

1.*Mức độ rẽ nhánh (b)*: Số lượng trạng thái con tối đa sinh ra từ một trạng thái cha.

2. *Độ sâu của lời giải (d)*: Số bước ngắn nhất từ trạng thái xuất phát đến trạng thái đích.

3. *Độ sâu tối đa của cây tìm kiếm (m)*: Chiều sâu lớn nhất mà cây tìm kiếm có thể đạt được.

### Thuật toán tìm kiếm mù (Uninformed Search)

Thuật toán tìm kiếm mù không sử dụng thông tin bên ngoài để quyết định lựa chọn tiếp theo trong quá trình tìm kiếm. Các thuật toán tìm kiếm mù sẽ mở rộng các trạng thái mà không có sự đánh giá ưu tiên nào, từ đó tìm kiếm tất cả các lựa chọn có thể cho đến khi tìm thấy giải pháp.

BFS là thuật toán tìm kiếm theo chiều rộng, nơi các trạng thái được mở rộng theo từng mức độ, tức là tất cả các trạng thái ở một mức độ hiện tại được xử lý trước khi chuyển sang mức độ tiếp theo. Thuật toán này đảm bảo tìm ra giải pháp tối ưu nếu không gian trạng thái không có trọng số. Tuy nhiên, BFS có nhược điểm là yêu cầu bộ nhớ lớn, đặc biệt khi không gian trạng thái rất lớn.

|  |
| --- |
| BFS(Q, S, G, P)  Input:  Q: Không gian trạng thái  S: Trạng thái xuất phát  G: Trạng thái đích  P: Tập các hành động  Output:  Chuỗi hành động hoặc trạng thái đích, hoặc thông báo "Không có lời giải"  1. Khởi tạo:  O ← {S} // Hàng đợi FIFO với trạng thái xuất phát  D ← ∅ // Tập các trạng thái đã được duyệt (tránh vòng lặp)  2. While O ≠ ∅ do:  a. Lấy nút đầu tiên n từ O và xóa n khỏi O  b. If n ∈ G then:  Return (đường đi tới n)  c. Add n vào D  d. For mỗi trạng thái con c thuộc P(n) do:  If c ∉ D và c ∉ O then:  Thêm c vào cuối O  3. Return "Không có lời giải" |

DFS là thuật toán tìm kiếm theo chiều sâu, trong đó một nhánh được mở rộng đến cùng trước khi quay lại thử các nhánh khác. Thuật toán này tiết kiệm bộ nhớ hơn so với BFS nhưng lại không đảm bảo tìm ra giải pháp tối ưu, đồng thời có thể rơi vào vòng lặp vô hạn nếu không có cơ chế dừng thích hợp.

|  |
| --- |
| DFS(Q, S, G, P)  Input:  Q: Không gian trạng thái  S: Trạng thái xuất phát  G: Tập trạng thái đích  P: Tập các hành động  Output:  Chuỗi hành động hoặc trạng thái đích, hoặc thông báo "Không có lời giải"  1. Khởi tạo:  O ← {S} // Ngăn xếp LIFO với trạng thái xuất phát  D ← ∅ // Tập các trạng thái đã được duyệt (tránh vòng lặp)  2. While O ≠ ∅ do:  a. Lấy nút đầu tiên n từ O và xóa n khỏi O  b. If n ∈ G then:  Return (đường đi tới n)  c. Add n vào D  d. For mỗi trạng thái con c thuộc P(n) do:  If c ∉ D và c ∉ O then:  Thêm c vào đầu O  3. Return "Không có lời giải" |

### Thuật toán tìm kiếm có thông tin (Informed search)

Thuật toán tìm kiếm có thông tin sử dụng hàm heuristic để đánh giá các trạng thái trong không gian tìm kiếm và ưu tiên mở rộng các trạng thái có khả năng dẫn đến mục tiêu tốt hơn. So với tìm kiếm mù, các thuật toán tìm kiếm có thông tin có thể giảm đáng kể thời gian tính toán và bộ nhớ bằng cách lựa chọn các bước đi tối ưu hơn.

A\* là thuật toán tìm kiếm phổ biến nhất trong tìm kiếm có thông tin. Thuật toán này kết hợp chi phí đi từ trạng thái hiện tại đến mục tiêu (hàm heuristic) với chi phí đã tiêu tốn (g), từ đó đưa ra một tổng chi phí (f = g + h) để lựa chọn bước tiếp theo. A\* đảm bảo tìm ra giải pháp tối ưu khi có một hàm heuristic phù hợp. Tuy nhiên, thuật toán này có thể tốn nhiều bộ nhớ và thời gian tính toán nếu không gian trạng thái rất lớn.

|  |
| --- |
| A\_Star(Q, S, G, P, h)  Input:  Q: Không gian trạng thái  S: Trạng thái xuất phát  G: Tập trạng thái đích  P: Tập các hành động  h: Hàm heuristic (ước lượng chi phí từ trạng thái hiện tại đến trạng thái đích)  Output:  Chuỗi hành động hoặc trạng thái đích, hoặc thông báo "Không có lời giải"  1. Khởi tạo:  O ← {S} // Tập mở  g(S) ← 0 // Chi phí từ trạng thái xuất phát đến S  f(S) ← g(S) + h(S) // Ước tính tổng chi phí  D ← ∅ // Tập đóng  2. While O ≠ ∅ do:  a. Lấy nút n từ O với giá trị f(n) nhỏ nhất và xóa n khỏi O  b. If n ∈ G then:  Return (đường đi tới n)  c. Thêm n vào D  d. For mỗi trạng thái con c thuộc P(n) do:  If c ∉ D và c ∉ O then:  g(c) ← g(n) + cost(n, c)  f(c) ← g(c) + h(c)  Thêm c vào O  Else If c ∈ O then:  If g(c) > g(n) + cost(n, c) then:  g(c) ← g(n) + cost(n, c)  f(c) ← g(c) + h(c)  End For  3. Return "Không có lời giải" |

### Thuật toán tìm kiếm cục bộ (Local search)

Thuật toán tìm kiếm cục bộ không tạo ra một cây tìm kiếm đầy đủ mà chỉ tìm kiếm trong không gian trạng thái từ một điểm xuất phát, tập trung vào việc tìm giải pháp gần nhất hoặc tối ưu nhất trong phạm vi hiện tại. Thuật toán này có thể rất hiệu quả trong các bài toán có không gian trạng thái lớn, nhưng đôi khi có thể bị mắc kẹt ở điểm tối ưu cục bộ và không tìm ra giải pháp tối ưu toàn cục.

Simulated Annealing là một thuật toán tìm kiếm cục bộ dựa trên nguyên lý mô phỏng quá trình làm lạnh của kim loại. Thuật toán này sẽ thử nghiệm các lựa chọn ngẫu nhiên trong không gian trạng thái và dần dần giảm mức độ thay đổi (giảm nhiệt độ) để tìm ra giải pháp tối ưu. Simulated Annealing rất hiệu quả trong việc tránh bị mắc kẹt ở các cực trị cục bộ, nhưng việc chọn hàm nhiệt độ phù hợp là rất quan trọng để đạt được hiệu quả tốt.

|  |
| --- |
| Simulated\_Annealing(Start, Goal, T, alpha, h)  Input:  Start: Trạng thái xuất phát  Goal: Tập trạng thái đích  T: Nhiệt độ ban đầu  alpha: Tỷ lệ giảm nhiệt độ (0 < alpha < 1)  h: Hàm heuristic đánh giá trạng thái  Output:  Trạng thái đích, hoặc thông báo "Không có lời giải"  1. Khởi tạo:  n ← Start  2. While T > 0 do:  a. If n ∈ Goal then:  Return (đường đi tới n)  b. Chọn ngẫu nhiên một trạng thái con c thuộc P(n)  c. ΔE ← h(n) - h(c)  d. If ΔE > 0 then:  n ← c // Chuyển đến trạng thái tốt hơn  Else If e^(ΔE / T) > random(0, 1) then:  n ← c // Chuyển đến trạng thái tệ hơn với xác suất giảm dần  e. Giảm nhiệt độ T ← T \* alpha  3. Return "Không có lời giải" |

### Thuật toán tìm kiếm quay lui (Backtracking)

Backtracking là một kỹ thuật giải quyết bài toán bằng cách thử nghiệm từng bước. Khi một lựa chọn không dẫn đến giải pháp hợp lệ, thuật toán sẽ quay lại bước trước đó và thử một lựa chọn khác. Kỹ thuật này thường được sử dụng trong các bài toán tổ hợp, như bài toán nữ hoàng cờ vua hoặc Sudoku, nơi có thể phải thử nhiều phương án khác nhau và quay lại khi gặp phải một tình huống không hợp lệ.

|  |
| --- |
| Backtracking(State, Goal, P)  Input:  State: Trạng thái hiện tại  Goal: Tập trạng thái đích  P: Tập các hành động  Output:  Chuỗi hành động hoặc trạng thái đích, hoặc thông báo "Không có lời giải"  1. If State ∈ Goal then:  Return (đường đi tới State)  2. For mỗi hành động a thuộc P(State) do:  Next ← Apply(State, a) // Sinh trạng thái mới  If Next hợp lệ then:  Result ← Backtracking(Next, Goal, P)  If Result ≠ "Không có lời giải" then:  Return Result  3. Return "Không có lời giải" |

## GIỚI THIỆU CÁC THƯ VIỆN VÀ CÔNG CỤ THỰC HIỆN

Dự án *GIẢI VÀ MÔ PHỎNG TRÒ CHƠI SOKOBAN BẰNG CÁC THUẬT TOÁN TÌM KIẾM* được phát triển bằng Python, sử dụng nhiều thư viện và công cụ hỗ trợ để xây dựng trò chơi và phân tích dữ liệu. Các thư viện và công cụ chính bao gồm:

**1. Pygame:**

* **Chức năng**: Pygame là một thư viện mạnh mẽ dành cho lập trình trò chơi 2D trong Python, hỗ trợ xử lý đồ họa, âm thanh, và tương tác sự kiện.
* **Vai trò trong dự án**: Pygame được sử dụng để xây dựng giao diện đồ họa của trò chơi Sokoban, bao gồm việc hiển thị các ô vuông, nhân vật, và các vật cản. Thư viện này cũng quản lý các sự kiện từ bàn phím, đảm bảo trò chơi phản hồi nhanh chóng với hành động của người chơi.

**2. Pandas:**

* **Chức năng**: Pandas cung cấp các công cụ để xử lý và phân tích dữ liệu có cấu trúc hoặc bán cấu trúc. Nó hỗ trợ đọc, ghi, và thao tác dữ liệu từ các nguồn như CSV.
* **Vai trò trong dự án**: Pandas được sử dụng để lưu trữ và phân tích kết quả của các thuật toán tìm kiếm được áp dụng trong trò chơi. Thông qua Pandas, nhóm có thể dễ dàng đánh giá hiệu suất của các thuật toán như BFS, DFS, A\*, SIMULATED ANNEALING và BACKTRACKING.

**3. Matplotlib:**

* **Chức năng:** Matplotlib là thư viện vẽ đồ thị phổ biến trong Python, cho phép trực quan hóa dữ liệu dưới dạng biểu đồ và hình ảnh.
* **Vai trò trong dự án:** Nhóm đã sử dụng Matplotlib để tạo các biểu đồ so sánh hiệu suất của các thuật toán tìm kiếm, giúp minh họa rõ ràng về số bước đi, thời gian thực hiện, và kết quả tối ưu.

**4. Jupyter Notebook:**

* **Chức năng:** Jupyter Notebook là một công cụ lập trình tương tác, hỗ trợ kết hợp mã Python, văn bản mô tả, và đồ thị trực quan trên cùng một tài liệu.
* **Vai trò trong dự án:** Jupyter Notebook được sử dụng để phân tích dữ liệu thu được từ các thuật toán tìm kiếm, trình bày kết quả và tạo báo cáo trực quan, giúp quá trình phân tích hiệu quả và dễ hiểu hơn.

**5. Tkinter:**

* **Chức năng:** Tkinter là thư viện GUI tiêu chuẩn của Python, giúp tạo ra các giao diện người dùng đơn giản.
* **Vai trò trong dự án:** Tkinter hỗ trợ trong việc xây dựng các cửa sổ giao diện bổ sung, chẳng hạn như bảng chọn mức độ hoặc bảng hiển thị kết quả, tăng trải nghiệm người dùng.

**6. PyCharm:**

* **Chức năng:** PyCharm là môi trường phát triển tích hợp (IDE) dành riêng cho Python, cung cấp các công cụ mạnh mẽ như gỡ lỗi, kiểm tra mã, và hỗ trợ lập trình.
* **Vai trò trong dự án:** PyCharm được sử dụng làm IDE chính để phát triển mã nguồn, quản lý dự án, và gỡ lỗi, giúp nhóm làm việc hiệu quả hơn trong suốt quá trình phát triển.

**7. Visual Studio Code:**

* **Chức năng:** Visual Studio Code là một IDE nhẹ, hỗ trợ nhiều ngôn ngữ lập trình, tích hợp Git, và cung cấp các tiện ích mở rộng phong phú.
* **Vai trò trong dự án:** Visual Studio Code được sử dụng song song với PyCharm, đặc biệt trong các tác vụ như quản lý mã nguồn trên GitHub và chỉnh sửa nhanh mã Python.

**8. GitHub:**

* **Chức năng**: Nền tảng quản lý mã nguồn trực tuyến hỗ trợ cộng tác và lưu trữ dự án.
* **Vai trò trong dự án**:
  + **Lưu trữ mã nguồn**: Toàn bộ mã nguồn và tài liệu của dự án được lưu trên GitHub, đảm bảo việc quản lý phiên bản và theo dõi thay đổi một cách hiệu quả.
  + **Chia sẻ và cộng tác**: Nhóm phát triển có thể làm việc cùng nhau và dễ dàng chia sẻ dự án với cộng đồng hoặc người dùng quan tâm.
  + **URL dự án**: Dự án được lưu tại [GitHub - MySokoban](https://github.com/ToiLaBao2004/MySokoban).

1. **PHÂN TÍCH VÀ THIẾT KẾ GIẢI PHÁP**
2. **THỰC NGHIỆM, ĐÁNH GIÁ, PHÂN TÍCH KẾT QUẢ**
3. **KẾT LUẬN**

**TÀI LIỆU THAM KHẢO**