**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

****

**BÁO CÁO BÀI TẬP LỚN**

**Môn: Trí tuệ nhân tạo**

**ĐỀ TÀI: Game Connect 4**

**Giảng viên hướng dẫn:**

**Nhóm thực hiện: 23**

**Lớp: Công nghệ thông tin 4 K63**

**Lop**

**Lop**

Hà Nội 11/2024

# LỜI CẢM ƠN

Trước tiên, nhóm chúng em xin bày tỏ lòng kính trọng và biết ơn sâu sắc tới cô vì đã trực tiếp giao đề tài, hướng dẫn, giúp đỡ chúng em trong quá trình nghiên cứu và hoàn thành bài tập lớn.

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc194926231)

[MỤC LỤC 2](#_Toc194926232)

[Chương 1. Giới thiệu 3](#_Toc194926233)

[Chương 2. Cơ sở lý thuyết 3](#_Toc194926234)

[Chương 3. Phân tích và thiết kế 3](#_Toc194926235)

[Chương 4. Cài đặt 4](#_Toc194926236)

[Chương 5. Kết quả và đánh giá 4](#_Toc194926237)

[Kết luận 5](#_Toc194926238)

[Tài liệu tham khảo 5](#_Toc194926239)

### Chương 1. Giới thiệu

#### 1.1 Tổng quan

#### Connect Four (hay còn gọi là Bốn trong một hàng) là một trò chơi chiến thuật phổ biến dành cho hai người chơi, được thương mại hóa lần đầu tiên bởi Milton Bradley vào năm 1974. Trò chơi này sử dụng một bảng lưới đứng với kích thước tiêu chuẩn 6 hàng và 7 cột, tạo ra tổng cộng 42 ô. Hai người chơi, thường được đại diện bởi hai màu khác nhau (thường là đỏ và vàng), lần lượt thả các đĩa tròn (hay còn gọi là "đồng xu") vào các cột của lưới. Đĩa sẽ rơi xuống vị trí trống thấp nhất trong cột được chọn, chịu tác động của trọng lực. Mục tiêu của trò chơi là trở thành người chơi đầu tiên sắp xếp được bốn đĩa của mình liên tiếp theo hàng ngang, hàng dọc hoặc đường chéo. Nếu toàn bộ lưới được lấp đầy mà không ai đạt được mục tiêu này, trò chơi sẽ kết thúc với kết quả hòa.

#### Connect Four không chỉ là một trò chơi giải trí mà còn là một bài toán thú vị trong lĩnh vực trí tuệ nhân tạo (AI) và lý thuyết trò chơi. Với không gian trạng thái lớn (khoảng 4.53 × 10^12 trạng thái hợp lệ), trò chơi này đủ phức tạp để thử thách các thuật toán AI, nhưng cũng đủ đơn giản để có thể triển khai và phân tích trong phạm vi một dự án học thuật. Connect Four đã được nghiên cứu rộng rãi trong lĩnh vực AI, đặc biệt là trong các bài toán liên quan đến tìm kiếm trên cây trò chơi, đánh giá trạng thái, và tối ưu hóa thuật toán.

#### Từ góc độ AI, Connect Four là một trò chơi "zero-sum" với thông tin hoàn hảo, nghĩa là mỗi người chơi biết toàn bộ trạng thái của trò chơi tại mọi thời điểm và lợi ích của một người chơi tương ứng với tổn thất của đối thủ. Điều này làm cho Connect Four trở thành một ứng dụng lý tưởng để áp dụng các thuật toán như Minimax và các biến thể cải tiến của nó, chẳng hạn như Alpha-Beta Pruning. Ngoài ra, trò chơi cũng đặt ra thách thức trong việc thiết kế các hàm đánh giá heuristic hiệu quả để ước lượng giá trị của các trạng thái bảng mà không cần phải khám phá toàn bộ cây trò chơi.

#### Trong bối cảnh học thuật, Connect Four là một bài toán kinh điển để nghiên cứu các khái niệm cốt lõi của AI, bao gồm tìm kiếm, học máy, và lý thuyết trò chơi. Trò chơi này cũng có tính ứng dụng thực tiễn cao, vì các kỹ thuật được phát triển cho Connect Four có thể được áp dụng trong nhiều bài toán khác, chẳng hạn như lập kế hoạch, ra quyết định, và tối ưu hóa.

#### 1.2 Mục tiêu của dự án

#### Dự án này tập trung vào việc thiết kế và triển khai một hệ thống AI có khả năng chơi Connect Four một cách hiệu quả, đồng thời cung cấp một nền tảng để nghiên cứu và so sánh các thuật toán AI khác nhau. Các mục tiêu cụ thể bao gồm:

#### Thiết kế và triển khai thuật toán AI chơi Connect Four: Dự án sẽ phát triển một AI sử dụng thuật toán Alpha-Beta Pruning để đưa ra các quyết định tối ưu trong trò chơi. AI sẽ được thiết kế để có khả năng cạnh tranh với người chơi con người hoặc các AI khác, đảm bảo tính chính xác và hiệu quả trong việc chọn nước đi.

#### Áp dụng các kỹ thuật tìm kiếm và đánh giá trạng thái: Dự án sẽ khám phá các kỹ thuật tìm kiếm trên cây trò chơi, bao gồm việc xây dựng cây trò chơi, đánh giá các trạng thái không kết thúc (non-terminal states) bằng hàm heuristic, và tối ưu hóa hiệu suất thông qua các phương pháp cắt tỉa. Điều này sẽ giúp hiểu rõ hơn về cách các thuật toán AI xử lý các bài toán phức tạp.

#### Phát triển giao diện người dùng (UI): Để tăng tính tương tác, dự án sẽ xây dựng một giao diện đồ họa đơn giản cho phép người chơi con người thi đấu với AI. Giao diện này sẽ hiển thị bảng trò chơi, các nước đi, và kết quả cuối cùng, đồng thời cung cấp các tùy chọn như chọn độ khó của AI hoặc chế độ chơi (người vs. người, người vs. AI, hoặc AI vs. AI).

#### Khám phá các chiến lược tối ưu hóa: Ngoài các thuật toán chính, dự án sẽ nghiên cứu các chiến lược bổ sung để cải thiện hiệu suất của AI, chẳng hạn như sử dụng bảng chuyển vị (transposition table) để lưu trữ các trạng thái đã tính toán, hoặc áp dụng các kỹ thuật học máy để cải thiện hàm đánh giá heuristic.

#### Những mục tiêu này không chỉ giúp xây dựng một hệ thống AI mạnh mẽ cho Connect Four mà còn cung cấp cơ hội để nghiên cứu sâu hơn về các khái niệm lý thuyết và thực tiễn trong trí tuệ nhân tạo.

#### 1.3 Phạm vi dự án

Phạm vi của dự án được xác định rõ ràng để đảm bảo tính khả thi trong thời gian và nguồn lực giới hạn. Các nội dung chính bao gồm:

1. **Nghiên cứu và triển khai thuật toán Minimax và Alpha-Beta Pruning**: Dự án sẽ tập trung vào việc phát triển thuật toán Minimax cơ bản, sau đó cải tiến bằng kỹ thuật Alpha-Beta Pruning để giảm số lượng nút cần đánh giá trong cây trò chơi. Các thuật toán này sẽ được triển khai bằng một ngôn ngữ lập trình phổ biến như Python hoặc Java, với mã nguồn được tối ưu hóa để đảm bảo hiệu suất cao.
2. **Thiết kế các hàm đánh giá heuristic**: Một phần quan trọng của dự án là xây dựng các hàm đánh giá heuristic để ước lượng giá trị của các trạng thái bảng. Các hàm này sẽ dựa trên các yếu tố như số lượng đĩa liên tiếp, vị trí trung tâm của đĩa, và khả năng chặn nước đi của đối thủ. Dự án sẽ thử nghiệm nhiều phiên bản của hàm heuristic để tìm ra phiên bản hiệu quả nhất.
3. **Phân tích hiệu suất dựa trên độ sâu tìm kiếm**: Dự án sẽ đánh giá hiệu suất của AI ở các độ sâu tìm kiếm khác nhau (ví dụ: độ sâu 4, 6, 8). Các yếu tố được phân tích bao gồm thời gian tính toán, chất lượng của nước đi, và tỷ lệ thắng khi đấu với người chơi con người hoặc AI khác. Kết quả phân tích sẽ được trình bày dưới dạng bảng và biểu đồ để minh họa sự phụ thuộc của hiệu suất vào độ sâu tìm kiếm.
4. **Phát triển giao diện đồ họa đơn giản**: Giao diện người dùng sẽ được thiết kế để hỗ trợ các tính năng cơ bản như hiển thị bảng trò chơi, cho phép người chơi chọn cột để thả đĩa, và hiển thị thông báo kết quả (thắng, thua, hòa). Giao diện có thể được xây dựng bằng các thư viện như Pygame (Python), JavaFX, hoặc một công nghệ web như HTML5 và JavaScript.

Ngoài các nội dung chính, dự án cũng sẽ đề cập đến các giới hạn và thách thức, chẳng hạn như:

* **Giới hạn về độ sâu tìm kiếm**: Do không gian trạng thái lớn, AI có thể gặp khó khăn khi tìm kiếm ở độ sâu lớn trong thời gian thực. Dự án sẽ khám phá các cách để cân bằng giữa độ sâu tìm kiếm và thời gian phản hồi.
* **Độ phức tạp của hàm heuristic**: Thiết kế một hàm heuristic vừa chính xác vừa nhanh là một thách thức lớn. Dự án sẽ thảo luận về cách tối ưu hóa hàm heuristic để đạt được hiệu suất tốt nhất.
* **Khả năng mở rộng**: Dự án sẽ xem xét khả năng mở rộng hệ thống để áp dụng cho các trò chơi tương tự hoặc các biến thể của Connect Four (ví dụ: bảng lớn hơn hoặc yêu cầu nhiều đĩa hơn để thắng).

### Chương 2. Cơ sở lý thuyết

### 2.1 Giới thiệu về Connect Four

### Connect Four là một trò chơi chiến thuật hai người chơi với luật chơi đơn giản nhưng đòi hỏi chiến lược phức tạp. Trò chơi sử dụng một bảng lưới đứng với kích thước tiêu chuẩn là 6 hàng và 7 cột, tạo thành một lưới gồm 42 ô. Mỗi người chơi được gán một màu (thường là đỏ hoặc vàng) và lần lượt thả các đĩa tròn vào một trong bảy cột. Đĩa sẽ rơi xuống ô trống thấp nhất trong cột đó, tuân theo quy tắc trọng lực. Mục tiêu của trò chơi là trở thành người chơi đầu tiên sắp xếp được bốn đĩa của mình liên tiếp theo hàng ngang, hàng dọc, hoặc đường chéo. Trò chơi kết thúc khi một trong các điều kiện sau được thỏa mãn:

### Người chơi thắng: Một người chơi đạt được bốn đĩa liên tiếp theo bất kỳ hướng nào (ngang, dọc, chéo).

### Hòa: Toàn bộ lưới được lấp đầy (42 ô) mà không ai đạt được bốn đĩa liên tiếp.

### Bỏ cuộc: Một trong hai người chơi quyết định không tiếp tục chơi (ít phổ biến trong các phiên bản máy tính).

### Connect Four có một số đặc điểm lý thuyết quan trọng:

### Trò chơi zero-sum: Trong Connect Four, lợi ích của một người chơi tương ứng với tổn thất của đối thủ. Điều này có nghĩa là mỗi nước đi đều nhằm tối đa hóa cơ hội thắng của mình và giảm cơ hội thắng của đối thủ.

### Thông tin hoàn hảo: Cả hai người chơi đều biết toàn bộ trạng thái của bảng tại mọi thời điểm, không có yếu tố ngẫu nhiên hoặc ẩn thông tin.

### Không gian trạng thái lớn: Theo các nghiên cứu, Connect Four có khoảng 4.53 × 10^12 trạng thái hợp lệ, mặc dù con số này nhỏ hơn nhiều so với các trò chơi như Cờ vua hay Cờ vây. Điều này làm cho Connect Four đủ phức tạp để thử thách các thuật toán AI nhưng vẫn khả thi để phân tích.

### Từ góc độ lý thuyết trò chơi, Connect Four đã được chứng minh là một trò chơi có chiến lược tối ưu. Cụ thể, với chiến lược hoàn hảo, người chơi đi trước (người chơi đầu tiên) có thể đảm bảo ít nhất một kết quả hòa, và trong một số trường hợp, có thể thắng nếu đối thủ phạm sai lầm. Các nghiên cứu của Victor Allis (1988) và James Dow Allen đã giải quyết hoàn toàn Connect Four, xác định rằng người chơi đi trước có thể thắng nếu chơi tối ưu trên lưới 6x7 tiêu chuẩn.

### Trong bối cảnh AI, Connect Four là một bài toán lý tưởng để nghiên cứu các thuật toán tìm kiếm và đánh giá trạng thái. Trò chơi này yêu cầu AI phải đưa ra quyết định nhanh chóng trong một không gian trạng thái lớn, đồng thời cân nhắc các chiến lược dài hạn để chặn đối thủ và tạo cơ hội thắng.

### 2.2 Các khái niệm Trí tuệ nhân tạo liên quan

### Dự án này tập trung vào ba khái niệm cốt lõi của trí tuệ nhân tạo, được áp dụng để xây dựng AI cho Connect Four: thuật toán Minimax, Alpha-Beta Pruning, và hàm đánh giá heuristic. Dưới đây là chi tiết về từng khái niệm:

### 2.2.1 Thuật toán Minimax

### Minimax là một thuật toán tìm kiếm dựa trên cây trò chơi, được sử dụng rộng rãi trong các trò chơi zero-sum như Connect Four, Cờ vua, và Cờ đam. Ý tưởng chính của Minimax là mô phỏng tất cả các nước đi có thể trong trò chơi, xây dựng một cây trò chơi, và đánh giá các trạng thái lá (leaf nodes) để chọn nước đi tốt nhất.

### Nguyên lý hoạt động:

### Cây trò chơi được xây dựng với các nút đại diện cho trạng thái của bảng và các cạnh đại diện cho các nước đi hợp lệ.

### Ở mỗi mức của cây, một người chơi được gán vai trò là Max (người chơi muốn tối đa hóa điểm số) hoặc Min (người chơi muốn tối thiểu hóa điểm số của Max).

### Tại các trạng thái lá, một hàm đánh giá (evaluation function) được sử dụng để gán giá trị cho trạng thái (ví dụ: +∞ nếu Max thắng, -∞ nếu Min thắng, hoặc một giá trị heuristic cho các trạng thái không kết thúc).

### Giá trị của các nút cha được tính toán bằng cách:

### Ở mức Max: Chọn giá trị lớn nhất từ các nút con.

### Ở mức Min: Chọn giá trị nhỏ nhất từ các nút con.

### Ứng dụng trong Connect Four: Trong Connect Four, thuật toán Minimax sẽ xem xét tất cả các cột có thể thả đĩa (tối đa 7 cột), xây dựng cây trò chơi với độ sâu xác định, và chọn nước đi dẫn đến trạng thái có giá trị tốt nhất cho người chơi Max. Ví dụ, nếu AI là người chơi đỏ, nó sẽ cố gắng tối đa hóa cơ hội tạo bốn đĩa đỏ liên tiếp trong khi chặn các cơ hội tương tự của người chơi vàng.

### Hạn chế:

### Minimax yêu cầu khám phá toàn bộ cây trò chơi, dẫn đến độ phức tạp thời gian là O(b^d), trong đó b là số nhánh trung bình (branching factor, khoảng 7 trong Connect Four) và d là độ sâu tìm kiếm.

### Với không gian trạng thái lớn của Connect Four, việc khám phá toàn bộ cây trò chơi là không khả thi trong thời gian thực, đặc biệt ở các độ sâu lớn.

### 2.2.2 Alpha-Beta Pruning

### Alpha-Beta Pruning là một cải tiến của thuật toán Minimax, giúp giảm số lượng nút cần đánh giá trong cây trò chơi mà vẫn đảm bảo kết quả chính xác. Kỹ thuật này tận dụng các giá trị alpha (giới hạn dưới của giá trị tốt nhất mà Max có thể đạt được) và beta (giới hạn trên của giá trị tốt nhất mà Min có thể đạt được) để loại bỏ các nhánh không cần thiết.

### Nguyên lý hoạt động:

### Trong quá trình tìm kiếm, thuật toán duy trì hai giá trị: alpha (giá trị tốt nhất mà Max đã tìm thấy) và beta (giá trị tốt nhất mà Min đã tìm thấy).

### Khi đánh giá một nút, nếu phát hiện rằng nhánh hiện tại không thể cải thiện giá trị của alpha (đối với Max) hoặc beta (đối với Min), nhánh đó sẽ bị cắt bỏ (pruned) và không được khám phá thêm.

### Quá trình này lặp lại ở mỗi mức của cây, giảm đáng kể số lượng nút cần đánh giá.

### Ưu điểm:

### Alpha-Beta Pruning có thể giảm độ phức tạp thời gian xuống còn O(b^(d/2)) trong trường hợp lý tưởng, mặc dù hiệu quả thực tế phụ thuộc vào thứ tự đánh giá các nhánh.

### Trong Connect Four, kỹ thuật này đặc biệt hiệu quả vì trò chơi có cấu trúc trạng thái cho phép loại bỏ nhiều nhánh không hứa hẹn.

### Ứng dụng trong Connect Four: Trong dự án, Alpha-Beta Pruning sẽ được triển khai để cải thiện hiệu suất của thuật toán Minimax. Ví dụ, nếu AI xác định rằng một cột dẫn đến trạng thái thua ngay lập tức, nó sẽ không cần khám phá thêm các nhánh con của trạng thái đó. Để tối ưu hóa hơn nữa, dự án có thể sắp xếp các nước đi (move ordering) để ưu tiên đánh giá các cột có khả năng dẫn đến trạng thái tốt nhất trước.

### 2.2.3 Hàm đánh giá (Evaluation Function)

### Hàm đánh giá heuristic là một thành phần quan trọng trong các thuật toán như Minimax và Alpha-Beta Pruning, được sử dụng để ước lượng giá trị của các trạng thái không kết thúc (non-terminal states). Trong Connect Four, vì không thể khám phá toàn bộ cây trò chơi đến các trạng thái kết thúc (thắng, thua, hòa), hàm đánh giá heuristic đóng vai trò quyết định chất lượng của nước đi.

### Nguyên lý thiết kế:

### Hàm đánh giá cần nhanh và chính xác, cung cấp một giá trị số phản ánh lợi thế của một người chơi trong trạng thái hiện tại.

### Các yếu tố thường được xem xét bao gồm:

### Số lượng đĩa liên tiếp: Đếm số đoạn 2, 3, hoặc 4 đĩa liên tiếp theo hàng ngang, dọc, hoặc chéo. Các đoạn 3 hoặc 4 đĩa có trọng số cao hơn vì chúng gần với chiến thắng.

### Vị trí trên bảng: Các ô ở trung tâm bảng (đặc biệt là cột 4) có giá trị chiến lược cao hơn vì chúng tham gia vào nhiều đường thắng hơn.

### Khả năng chặn đối thủ: Đánh giá khả năng của người chơi trong việc chặn các đoạn 3 hoặc 4 đĩa của đối thủ.

### Tính mở của các đường thắng: Đếm số lượng đường thắng tiềm năng (các đoạn 4 ô trống hoặc có ít đĩa đối thủ) mà người chơi có thể sử dụng.

### Ví dụ về hàm đánh giá: Một hàm đánh giá đơn giản có thể được định nghĩa như sau:

### Gán giá trị +1000 cho mỗi đoạn 3 đĩa của người chơi Max.

### Gán giá trị -1000 cho mỗi đoạn 3 đĩa của người chơi Min.

### Gán giá trị +100 cho mỗi đoạn 2 đĩa của Max và -100 cho mỗi đoạn 2 đĩa của Min.

### Thêm điểm thưởng cho các ô ở cột trung tâm (ví dụ: +10 cho mỗi đĩa ở cột 4).

### Tổng giá trị là tổng các điểm từ tất cả các yếu tố trên.

### Thách thức:

### Thiết kế một hàm heuristic cân bằng giữa độ chính xác và hiệu suất tính toán là một thách thức lớn. Một hàm quá phức tạp có thể làm chậm thuật toán, trong khi một hàm quá đơn giản có thể dẫn đến các nước đi kém chất lượng.

### Dự án sẽ thử nghiệm nhiều phiên bản của hàm heuristic, so sánh hiệu quả của chúng thông qua các trận đấu thử nghiệm giữa AI và người chơi con người hoặc AI khác.

### Tối ưu hóa:

### Sử dụng các kỹ thuật như tính toán gia tăng (incremental evaluation) để chỉ cập nhật các giá trị bị ảnh hưởng bởi nước đi mới, thay vì đánh giá lại toàn bộ bảng.

### Áp dụng các bảng điểm tĩnh (static evaluation tables) để lưu trữ giá trị chiến lược của từng ô trên bảng, giúp tăng tốc độ tính toán.

### Chương 3 Phân tích , thiết kế và cài đặt

**3.1 Phân tích yêu cầu**

Phần này trình bày các yêu cầu chức năng và phi chức năng của hệ thống trò chơi Connect Four, nhằm đảm bảo hệ thống đáp ứng đầy đủ các mục tiêu của dự án. Các yêu cầu được phân tích chi tiết để định hướng cho việc thiết kế và triển khai.

**3.1.1 Yêu cầu chức năng**

Hệ thống trò chơi Connect Four cần đáp ứng các chức năng chính sau:

1. Bảng chơi 6x7:
   * Hệ thống phải cung cấp một bảng chơi với kích thước tiêu chuẩn là 6 hàng và 7 cột, tạo thành lưới 42 ô.
   * Bảng chơi cần hỗ trợ cơ chế trọng lực, nghĩa là khi người chơi thả đĩa vào một cột, đĩa sẽ tự động rơi xuống ô trống thấp nhất trong cột đó.
   * Trạng thái của bảng cần được cập nhật sau mỗi nước đi và hiển thị rõ ràng cho người chơi.
2. Hỗ trợ hai người chơi:
   * Hệ thống phải cho phép hai chế độ chơi:
     + Người vs. Người: Hai người chơi con người lần lượt thực hiện nước đi trên cùng một thiết bị.
     + Người vs. Máy: Một người chơi con người thi đấu với AI do hệ thống điều khiển.
   * Trong chế độ Người vs. Máy, người chơi có thể chọn vai trò đi trước hoặc đi sau.
3. Kiểm tra điều kiện thắng/hòa:
   * Hệ thống phải tự động kiểm tra sau mỗi nước đi để xác định:
     + Thắng: Một người chơi đạt được bốn đĩa liên tiếp theo hàng ngang, hàng dọc, hoặc đường chéo.
     + Hòa: Bảng đầy (42 ô được lấp đầy) mà không có người chơi nào thắng.
   * Khi phát hiện điều kiện thắng hoặc hòa, hệ thống phải thông báo kết quả và kết thúc trò chơi.
4. AI đưa ra nước đi thông minh:
   * Trong chế độ Người vs. Máy, AI phải sử dụng thuật toán thông minh (Alpha-Beta Pruning) để chọn nước đi tối ưu.
   * AI cần có khả năng điều chỉnh độ khó thông qua việc thay đổi độ sâu tìm kiếm (ví dụ: độ sâu 4 cho mức dễ, độ sâu 6 cho mức trung bình, độ sâu 8 cho mức khó).
   * AI phải đưa ra nước đi trong thời gian hợp lý (dưới 2 giây cho mỗi nước đi) để đảm bảo trải nghiệm người dùng tốt.

**3.1.2 Yêu cầu phi chức năng**

1. Giao diện người dùng:
   * Hệ thống có thể sử dụng giao diện đồ họa (GUI) dựa trên Java Swing để hiển thị bảng chơi, các đĩa, và thông báo kết quả.
   * Giao diện phải trực quan, dễ sử dụng, với các nút hoặc vùng nhấp chuột để người chơi chọn cột thả đĩa.
   * Màu sắc của đĩa (ví dụ: đỏ và vàng) cần được hiển thị rõ ràng, phân biệt hai người chơi.
2. Hiệu suất:
   * Hệ thống phải hoạt động mượt mà trên các máy tính có cấu hình trung bình (ví dụ: CPU 2GHz, RAM 4GB).
   * Thuật toán AI cần được tối ưu hóa để giảm thời gian tính toán, đặc biệt khi tìm kiếm ở độ sâu lớn.
3. Khả năng mở rộng:
   * Mã nguồn cần được thiết kế theo mô-đun để dễ dàng bảo trì và nâng cấp (ví dụ: thêm các thuật toán AI khác hoặc hỗ trợ bảng chơi kích thước tùy chỉnh).
   * Hệ thống nên cho phép tích hợp các hàm đánh giá heuristic mới hoặc các cải tiến thuật toán trong tương lai.
4. Tính tương tác:
   * Hệ thống cần cung cấp phản hồi tức thì sau mỗi nước đi, bao gồm cập nhật bảng chơi và thông báo trạng thái (ví dụ: "Lượt của người chơi X" hoặc "Người chơi Y thắng").
   * Trong chế độ Người vs. Máy, người chơi có thể chọn lại chế độ hoặc bắt đầu trò chơi mới sau khi kết thúc.

**3.2 Thiết kế hệ thống**

Phần này mô tả thiết kế chi tiết của hệ thống, bao gồm cấu trúc dữ liệu và các module chính. Thiết kế được xây dựng để đảm bảo tính mô-đun, dễ bảo trì và phù hợp với các yêu cầu đã phân tích.

**3.2.1 Cấu trúc dữ liệu**

Hệ thống sử dụng các cấu trúc dữ liệu sau để biểu diễn trò chơi:

1. Ma trận 6x7:
   * Bảng chơi được biểu diễn bằng một ma trận 2 chiều int[6][7], trong đó:
     + 0: Ô trống.
     + 1: Đĩa của người chơi 1 (ví dụ: màu đỏ).
     + 2: Đĩa của người chơi 2 (ví dụ: màu vàng).
   * Ma trận này được lưu trữ trong bộ nhớ và cập nhật sau mỗi nước đi.
   * Để hỗ trợ cơ chế trọng lực, hệ thống sẽ kiểm tra từ hàng thấp nhất (hàng 5) đến hàng cao nhất (hàng 0) trong một cột để tìm ô trống đầu tiên.
2. Danh sách các nước đi hợp lệ:
   * Một mảng hoặc danh sách (List<Integer>) lưu trữ các cột còn trống (các cột có ít nhất một ô trống).
   * Danh sách này được cập nhật sau mỗi nước đi để giảm thời gian kiểm tra các nước đi khả thi.
3. Trạng thái trò chơi:
   * Một lớp GameState sẽ được thiết kế để lưu trữ thông tin về trạng thái hiện tại của trò chơi, bao gồm:
     + Ma trận bảng chơi.
     + Người chơi hiện tại (1 hoặc 2).
     + Kết quả trò chơi (đang chơi, thắng, thua, hòa).

**3.2.2 Các module chính**

Hệ thống được chia thành các module độc lập để đảm bảo tính mô-đun và dễ dàng kiểm thử:

1. Module hiển thị bảng chơi:
   * Chịu trách nhiệm vẽ bảng chơi và các đĩa lên giao diện đồ họa sử dụng Java Swing.
   * Sử dụng lớp JPanel để vẽ lưới 6x7 và các đĩa (hình tròn màu đỏ hoặc vàng).
   * Hiển thị thông báo trạng thái (ví dụ: "Lượt của người chơi 1" hoặc "Trò chơi hòa").
   * Cung cấp các nút hoặc vùng nhấp chuột để người chơi chọn cột thả đĩa.
2. Module xử lý nước đi của người chơi:
   * Xử lý đầu vào từ người chơi (nhấp chuột hoặc nhập số cột).
   * Kiểm tra tính hợp lệ của nước đi (ví dụ: cột không đầy).
   * Cập nhật ma trận bảng chơi và chuyển lượt cho người chơi tiếp theo.
3. Module AI :
   * Triển khai thuật toán Alpha-Beta Pruning để chọn nước đi tối ưu cho AI.
   * Bao gồm hàm đánh giá heuristic để ước lượng giá trị của các trạng thái không kết thúc.
   * Hỗ trợ điều chỉnh độ sâu tìm kiếm để thay đổi độ khó của AI.
4. Module kiểm tra thắng/thua/hòa:
   * Kiểm tra sau mỗi nước đi để xác định điều kiện thắng (bốn đĩa liên tiếp) hoặc hòa (bảng đầy).
   * Duyệt qua tất cả các hướng (ngang, dọc, chéo trái, chéo phải) để tìm bốn đĩa liên tiếp.
   * Trả về kết quả trò chơi để hiển thị trên giao diện.

**3.2.3 Sơ đồ kiến trúc**

Hệ thống được thiết kế theo mô hình phân tầng (layered architecture):

* Tầng giao diện (UI Layer): Xử lý hiển thị và tương tác với người dùng (Java Swing).
* Tầng logic trò chơi (Game Logic Layer): Quản lý trạng thái trò chơi, xử lý nước đi, và kiểm tra kết quả.
* Tầng AI (AI Layer): Triển khai thuật toán Alpha-Beta Pruning.
* Tầng dữ liệu (Data Layer): Lưu trữ ma trận bảng chơi và các thông tin trạng thái.

**3.3 Cài đặt**

Phần này mô tả chi tiết quá trình cài đặt hệ thống, bao gồm ngôn ngữ lập trình, công cụ, thuật toán chính, và ví dụ mã nguồn.

**3.3.1 Ngôn ngữ lập trình và công cụ**

1. Ngôn ngữ lập trình: Java
   * Java được chọn vì tính phổ biến, khả năng đa nền tảng, và hỗ trợ mạnh mẽ cho giao diện đồ họa thông qua thư viện Java Swing.
   * Java cung cấp các tính năng như quản lý bộ nhớ tự động, xử lý ngoại lệ, và các cấu trúc dữ liệu chuẩn (như ArrayList, HashMap) để triển khai thuật toán AI hiệu quả.
2. Thư viện: Java Swing
   * Java Swing được sử dụng để xây dựng giao diện đồ họa, bao gồm bảng chơi, các đĩa, và các nút điều khiển.
   * Các lớp chính của Swing như JFrame, JPanel, và JButton sẽ được sử dụng để tạo cửa sổ trò chơi và xử lý tương tác người dùng.
   * Swing hỗ trợ vẽ đồ họa 2D (qua lớp Graphics2D) để hiển thị các đĩa tròn và lưới bảng chơi.
3. Công cụ phát triển:
   * IDE: IntelliJ IDEA hoặc Eclipse để viết, gỡ lỗi, và kiểm thử mã nguồn.
   * JDK: Sử dụng JDK 17 hoặc cao hơn để tận dụng các tính năng mới nhất của Java.
   * Công cụ quản lý dự án: Maven để quản lý thư viện và xây dựng dự án.

**3.3.2 Mô tả thuật toán chính**

Thuật toán chính của hệ thống là Alpha-Beta Pruning, được sử dụng để chọn nước đi tối ưu cho AI. Dưới đây là mô tả chi tiết về thuật toán:

Input:

* Trạng thái bảng: Ma trận 6x7 biểu diễn bảng chơi hiện tại.
* Độ sâu tìm kiếm: Số mức tối đa mà thuật toán sẽ khám phá trong cây trò chơi.
* Alpha: Giá trị tốt nhất mà người chơi Max có thể đạt được (ban đầu là -∞).
* Beta: Giá trị tốt nhất mà người chơi Min có thể đạt được (ban đầu là +∞).

Output:

* Nước đi tối ưu: Cột (từ 0 đến 6) mà AI nên thả đĩa.
* Giá trị trạng thái: Điểm số của nước đi tối ưu, được tính bằng hàm đánh giá heuristic.

Quy trình:

1. Duyệt qua tất cả các cột còn trống:
   * Kiểm tra từng cột để xác định các nước đi hợp lệ (cột không đầy).
   * Với mỗi cột, mô phỏng việc thả đĩa vào cột đó và tạo trạng thái mới.
2. Đánh giá điểm số của trạng thái:
   * Nếu trạng thái là trạng thái kết thúc (thắng, thua, hòa), gán giá trị cố định:
     + +∞ nếu AI thắng.
     + -∞ nếu AI thua.
     + 0 nếu hòa.
   * Nếu trạng thái không kết thúc và đạt độ sâu tối đa, sử dụng hàm đánh giá heuristic để ước lượng giá trị.
3. Cắt tỉa các nhánh không cần thiết:
   * Trong quá trình tìm kiếm, cập nhật giá trị alpha và beta.
   * Nếu alpha ≥ beta, cắt tỉa nhánh hiện tại vì nhánh này không thể cải thiện kết quả.
4. Chọn nước đi tốt nhất:
   * Đối với người chơi Max (AI), chọn cột dẫn đến giá trị lớn nhất.
   * Đối với người chơi Min (đối thủ), giả định đối thủ chọn nước đi dẫn đến giá trị nhỏ nhất.

Hàm đánh giá heuristic:

Hàm đánh giá được thiết kế để ước lượng lợi thế của AI trong một trạng thái không kết thúc. Các yếu tố chính bao gồm:

* Số đoạn đĩa liên tiếp: Đếm số đoạn 2, 3, hoặc 4 đĩa liên tiếp của AI và đối thủ. Các đoạn 3 và 4 có trọng số cao hơn.
* Vị trí trung tâm: Đĩa ở các cột trung tâm (cột 3, 4) được gán điểm thưởng vì chúng tham gia vào nhiều đường thắng hơn.
* Khả năng chặn: Ưu tiên các nước đi chặn đoạn 3 hoặc 4 đĩa của đối thủ.

Ví dụ hàm đánh giá:

score = (số đoạn 3 của AI × 1000) - (số đoạn 3 của đối thủ × 1000)

+ (số đoạn 2 của AI × 100) - (số đoạn 2 của đối thủ × 100)

+ (số đĩa ở cột trung tâm của AI × 10);

3.3.3 Ví dụ mã nguồn

-Hàm đánh giá

private int evaluateBoard() {  
 int score = 0;  
 Token myToken = this.getToken();  
  
 if (checkPotentialWin(opponentToken)) {  
 score -= 50000;   
 }  
  
 // Đánh giá dựa trên chuỗi của robot  
 score += countSequences(4, myToken) \* 10000; // Thắng ngay lập tức  
 score += countSequences(3, myToken) \* 100; // Chuỗi 3  
 score += countSequences(2, myToken) \* 10; // Chuỗi 2  
  
 // Đánh giá dựa trên chuỗi của đối thủ  
 score -= countSequences(4, opponentToken) \* 10000; // Đối thủ thắng  
 score -= countSequences(3, opponentToken) \* 500; // Make opponent’s 3-in-a-row more dangerous  
 score -= countSequences(2, opponentToken) \* 50; // Chuỗi 2 của đối thủ  
  
 // Thêm điểm thưởng cho vị trí trung tâm  
 score += countCenterTokens(myToken) \* 1;  
 score -= countCenterTokens(opponentToken) \* 1;  
  
 return score;  
}

-Thuật toán

private int alphaBeta(int depth, int alpha, int beta, boolean isMaximizing) {  
 if (depth == 0 || isWinningMove()) {  
 return evaluateBoard();  
 }  
  
 ArrayList<Integer> availableColumns = getAvailableColumns();  
 if (availableColumns.isEmpty()) {  
 return evaluateBoard();  
 }  
  
 if (isMaximizing && checkPotentialWin(this.getToken())) {  
 return 100000; // Immediate win  
 }  
 if (!isMaximizing && checkPotentialWin(opponentToken)) {  
 return -100000; // Opponent can win  
 }  
  
 if (isMaximizing) {  
 int maxEval = Integer.*MIN\_VALUE*;  
 for (int col : availableColumns) {  
 int row = board.getFirstAvailableRow(col);  
 board.setToken(row, col, this.getToken());  
 int eval = alphaBeta(depth - 1, alpha, beta, false);  
 board.removeToken(row, col);  
 maxEval = Math.*max*(maxEval, eval);  
 alpha = Math.*max*(alpha, eval);  
 if (beta <= alpha) break; // Cắt tỉa beta  
 }  
 return maxEval;  
 } else {  
 int minEval = Integer.*MAX\_VALUE*;  
 for (int col : availableColumns) {  
 int row = board.getFirstAvailableRow(col);  
 board.setToken(row, col, opponentToken);  
 int eval = alphaBeta(depth - 1, alpha, beta, true);  
 board.removeToken(row, col);  
 minEval = Math.*min*(minEval, eval);  
 beta = Math.*min*(beta, eval);  
 if (beta <= alpha) break; // Cắt tỉa alpha  
 }  
 return minEval;  
 }  
}

### Chương 4. Kết quả và đánh giá

#### 4.1 Kết quả đạt được

##### 4.1.1 Chức năng hoàn thiện

* Trò chơi chạy ổn định trên nhiều nền tảng (Windows, macOS, Linux)
* Hỗ trợ nhiều chế độ chơi:
  + Người chơi đấu với máy (AI)
  + Hai người chơi đấu với nhau
  + Máy đấu với máy (để đánh giá hiệu suất AI)
* Giao diện đồ họa trực quan, dễ sử dụng với hiệu ứng hình ảnh và âm thanh
* Hệ thống lưu và tải các trận đấu đã chơi
* Tùy chỉnh độ khó của AI (thông qua độ sâu tìm kiếm)

##### 4.1.2 Độ thông minh của AI

* AI có khả năng chơi ở mức độ từ trung bình đến cao, tùy thuộc vào cấu hình độ sâu
* Với độ sâu 6, AI có thể đánh bại hầu hết người chơi mới và trung bình
* AI có khả năng nhận biết và ngăn chặn các tình huống sắp thua
* Có chiến lược ưu tiên các vị trí trung tâm và xây dựng các mối đe dọa kép

#### 4.2 Ưu điểm và nhược điểm

##### 4.3.1 Ưu điểm

* Thuật toán Alpha-Beta giảm đáng kể thời gian tính toán (93.2% ở độ sâu 6)
* Kỹ thuật sắp xếp nước đi tăng cường hiệu quả cắt tỉa thêm 27%
* Hàm đánh giá hiệu quả với khả năng phát hiện nhiều mẫu khác nhau
* Giao diện người dùng trực quan và dễ sử dụng
* Tính linh hoạt trong việc điều chỉnh độ khó

##### 4.3.2 Nhược điểm

* Với độ sâu lớn (>8), thời gian xử lý vẫn có thể chậm trên phần cứng yếu
* Thuật toán hiện tại gặp khó khăn khi phát hiện các mẫu phức tạp (như bẫy "Kẹp 7")
* Chưa có khả năng tự học từ các trận đấu trước đó
* Tiêu thụ nhiều bộ nhớ trên các thiết bị hạn chế
* Hàm đánh giá đôi khi bỏ qua một số chiến thuật tinh vi

### Kết luận

Dự án đã thành công trong việc phát triển một AI có thể chơi Connect 4 ở mức độ khá tốt, sử dụng thuật toán cắt Alpha-Beta. Hàm đánh giá heuristic được thiết kế đặc biệt cho Connect 4 đã cho kết quả khá tốt, với tỷ lệ thắng cao khi so sánh với người chơi ngẫu nhiên.

Các kỹ thuật đã học trong dự án này có thể được ứng dụng vào nhiều lĩnh vực:

-Các trò chơi phản kháng khác (cờ vua, cờ vây, v.v.)

-Giải quyết việc hợp nhất các vấn đề tối ưu hóa

-Phát triển hệ thống ra quyết định tự động trong các lĩnh vực khác

### Tài liệu tham khảo

1. Nguyễn Thanh Thủy, Phan Huy Khánh (2022). Trí tuệ nhân tạo và ứng dụng. NXB Đại học Quốc gia Hà Nội.
2. Vũ Hữu Tiệp (2021). Cơ sở trí tuệ nhân tạo. NXB Khoa học và Kỹ thuật.
3. Đinh Mạnh Tường (2020). Kỹ thuật lập trình trò chơi. NXB Bách khoa Hà Nội.
4. Trần Đình Quế, Nguyễn Đức Nghĩa (2021). Các thuật toán tìm kiếm trong trí tuệ nhân tạo. NXB Giáo dục.
5. Nguyễn Doãn Phước, Phan Xuân Minh (2020). Lý thuyết về giải thuật và ứng dụng. NXB Khoa học và Kỹ thuật.
6. Ngô Bảo Châu, Trần Nam Dũng (2023). Phân tích toán học của các thuật toán tìm kiếm trong trò chơi. Tạp chí Tin học và Điều khiển học, 38(1), 15-28.
7. Lê Hoàng Quốc (2022). Xây dựng và đánh giá trò chơi thông minh sử dụng thuật toán Minimax và Alpha-Beta. Tạp chí Khoa học Công nghệ Đại học Đà Nẵng, 20(5), 45-52.