ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ────────

A picture containing text, poster, font, graphics

Description automatically generated

BÀI TẬP LỚN

NHẬP MÔN TRÍ TUỆ NHÂN TẠO

**Xây dựng chương trình chơi Cờ vua**

|  |  |
| --- | --- |
| **Sinh viên thực hiện** | **Mã sinh viên** |
| Lê Anh Tuấn | 20205136 |
| Hoàng Như Nghĩa | 20205006 |
| Nguyễn Hùng Tiến | 20208001 |
| Trần Minh Tuấn | 20205137 |
| Đào Xuân Đạt | 20205065 |
| **Nhóm 28** | Mã lớp: **139313** |
| Giảng viên hướng dẫn | : TS. Đỗ Tiến Dũng |

*Hà Nội, tháng 7 năm 2023*

**Phân công nhiệm vụ**

|  |  |  |
| --- | --- | --- |
| **STT** | **Nhiệm vụ** | **Người thực hiện** |
| 1 | Thiết kế và code giao diện trò chơi | Nghĩa, Đạt |
| 2 | Mã hóa trò chơi, Nghiên cứu và cài đặt các giải thuật | Hùng Tiến, Anh Tuấn |
| 3 | So sánh, đánh giá các thuật toán đã áp dụng | Minh Tuấn |

**Mục lục**

[1. Giới thiệu bài toán: 4](#_Toc138627350)

[2. Các kỹ thuật tìm kiếm đối thủ 5](#_Toc138627351)

[2.1 Giải thuật Greedy: 6](#_Toc138627352)

[2.2 Giải thuật Minimax: 7](#_Toc138627353)

[2.3 Giải thuật Minimax kết hợp cắt tỉa alpha – beta: 9](#_Toc138627354)

[2.4 Giải thuật Negamax: 11](#_Toc138627355)

[2.5 Giải thuật Negascout: 13](#_Toc138627356)

[3. Cài đặt trò chơi: 14](#_Toc138627357)

[3.1 Biểu diễn bàn cờ và luật chơi: 14](#_Toc138627358)

[a) Lớp GameState: 14](#_Toc138627359)

[b) Lớp Move: 16](#_Toc138627360)

[3.2 Áp dụng các kỹ thuật tìm kiếm có đối thủ: 17](#_Toc138627361)

[a) Lớp AI: 17](#_Toc138627362)

[b) Greedy: 18](#_Toc138627363)

[c) Minimax, Negamax, Negascout: 18](#_Toc138627364)

[3.3 Giao diện trò chơi: 19](#_Toc138627365)

[4. Đánh giá 22](#_Toc138627366)

[4.1 Minimax và Negamax 22](#_Toc138627367)

[4.2 Negamax và Negamax cắt tỉa alpha – beta: 23](#_Toc138627368)

[4.3 Negamax có cắt tỉa alpha-beta và Negascout: 24](#_Toc138627369)

[5. Tài liệu tham khảo 25](#_Toc138627370)

1. **Giới thiệu bài toán:**

* Cờ vua là một trong những trò chơi bàn phổ biến nhất trên thế giới. Nó là một trò chơi chiến thuật giữa hai người, được chơi trên một bàn cờ vuông gồm 64 ô đen trắng xen kẽ. Mỗi người chơi điều khiển một bộ quân cờ gồm các quân cờ khác nhau và cố gắng chiếm lĩnh hoặc tấn công quân đối phương.
* Mục tiêu của trò chơi cờ vua là bắt hoặc chiếu bí quân Vua của đối phương. Người chơi phải sử dụng chiến thuật và kỹ năng chiếu quân để kiểm soát vị trí trên bàn cờ, tạo ra các đòn tấn công và phòng thủ, và đồng thời phải đề phòng các phản công từ đối thủ.
* Mỗi người chơi bắt đầu với 16 quân cờ, bao gồm Vua, Hậu, 2 Xe, 2 Tượng, 2 Mã và 8 Tốt.
* Cách di chuyển quân cờ:
  + Vua di chuyển 1 ô bất kỳ theo đường đi ngang, dọc hoặc chéo.
  + Hậu di chuyển bất kỳ số ô theo đường đi ngang, dọc hoặc chéo.
  + Xe di chuyển bất kỳ số ô theo đường đi ngang hoặc dọc.
  + Tượng di chuyển bất kỳ số ô theo đường đi chéo.
  + Mã di chuyển theo hình chữ L với 2 ô theo một chiều và 1 ô theo chiều khác.
  + Tốt di chuyển 1 ô về phía trước và có thể di chuyển 2 ô trong lần di chuyển đầu tiên. Khi tấn công, tốt di chuyển theo đường chéo.
* Tấn công và chiếu:
  + Chiếu Vua xảy ra khi Vua bị đối thủ tấn công và không thể tránh được.
  + Nếu Vua bị chiếu, người chơi phải thực hiện các nước đi để loại bỏ chiếu hoặc bảo vệ Vua.
  + Nếu không có cách nào để loại bỏ chiếu, lúc đó là chiếu hết và trò chơi kết thúc.

1. **Các kỹ thuật tìm kiếm đối thủ**

* Các ký hiệu sử dụng trong chương này:
  + node: nút hiện tại đang được tìm kiếm
  + depth: độ sâu mà chúng ta muốn tìm kiếm
  + alpha, beta: giá trị alpha, beta tại thời điểm đang xét
  + player: cho biến người chơi hiện tại
  + child: node con của node hiện tại
  + children: hàm trả về danh sách các nút con của nút hiện tại
  1. **Giải thuật Greedy:**
* Giải thuật Greedy (tham lam) là một phương pháp tìm kiếm trong lĩnh vực trí tuệ nhân tạo. Khi áp dụng Greedy vào cờ vua, người chơi sẽ chọn nước đi tốt nhất tại thời điểm hiện tại mà không xem xét tác động của nó tới các nước đi trong tương lai.
* Tuy nhiên, Greedy có thể được sử dụng trong một số trò chơi đơn giản hoặc trong các tình huống cụ thể trong cờ vua. Với Greedy, người chơi sẽ chọn nước đi dựa trên những ưu tiên cụ thể, chẳng hạn như cố gắng chiếm lấy quân cờ đối phương nhiều nhất trong tầm tấn công, tạo ra sự lợi thế ngắn hạn mà không xem xét đến các nước đi chiến lược hoặc tình huống chuyển đổi trong tương lai.
* Các bước thực hiện giải thuật Greedy:
  + Xây dựng biểu diễn trò chơi, mô tả tất cả các trạng thái có thể của trò chơi.
  + Duyệt qua tất cả các bước có thể của nước đi tiếp theo trong trò chơi. Với mỗi bước sẽ tính giá trị bàn cờ, từ đó tìm ra được bước nào là tối ưu nhất trong nước đi tiếp theo.
  + Trả về bước đi tốt nhất cho người chơi hiện tại.
* Mã giả giải thuật Greedy:

A picture containing text, screenshot, font

Description automatically generated

* 1. **Giải thuật Minimax:**
* Minimax là giải thuật cho lớp bài toán tìm kiếm có đối thủ (adversarial search). Nó được sử dụng rộng rãi trong trò chơi 2 người như cờ caro, cờ vua,... dùng để xác định nước đi tốt nhất trong các trò chơi 2 người.
* Ý tưởng chính của giải thuật Minimax là tạo ra một cây trò chơi đầy đủ, trong đó mỗi nút đại diện cho một trạng thái của trò chơi và các nút con đại diện cho các nước đi có thể được thực hiện từ trạng thái đó. Từ trạng thái gốc, giải thuật sẽ duyệt qua cây trò chơi, đánh giá các nút lá (trạng thái cuối cùng) và truy ngược lên các nút cha để xác định nước đi tốt nhất.
* Các bước thực hiện giải thuật Minimax:
  + Xây dựng cây biểu diễn trò chơi, mô tả tất cả các trạng thái có thể của trò chơi.
  + Đánh giá mỗi trạng thái bằng cách tính toán giá trị minimax đối với mỗi nút trong cây biểu diễn trò chơi. Giá trị minimax của 1 nút là giá trị hàm mục tiêu để đánh giá các trạng thái của trò chơi, giả sử rằng cả 2 người chơi đều đi nước đi tối ưu từ đó đến cuối trò chơi.
  + Duyệt cây trạng thái gốc đến từng nút con và chọn giá trị hàm mục tiêu tối đa hoặc tối thiểu tùy theo vai trò của mỗi người chơi.
  + Trả về quyết định tốt nhất cho người chơi hiện tại.
* Mã giả giải thuật Minimax:

A picture containing text, screenshot, font, document

Description automatically generated

* 1. **Giải thuật Minimax kết hợp cắt tỉa alpha – beta:**
* Vấn đề của giải thuật Minimax là số trạng thái trò chơi mà nó đánh giá tăng theo hàm mũ của độ sâu cây biểu diễn trò chơi. Do vậy, nó không phù hợp với nhiều bài toán trò chơi thực tế. Ta có thể cắt bỏ 1 số nhánh tìm kiếm không đem lại kết quả tối ưu cho giải thuật.
* Giải thuật Minimax kết hợp cắt tỉa alpha - beta là cải tiến để tối ưu hóa không gian tìm kiếm. Nó sử dụng các giá trị alpha, beta để giới hạn tìm kiếm, trả về nước đi tìm được bởi giải thuật Minimax, đồng thời cắt bỏ những nút không cần thiết mà không làm ảnh hưởng đến kết qủa cuối cùng.
* Trong mỗi nút, giá trị alpha là giá trị tối đa mà người chơi MAX có thể đạt được trên con đường từ nút gốc đến nút hiện tại, giá trị beta là giá trị tối thiểu mà người chơi MIN có thể đạt được trên con đường từ nút gốc đến nút hiện tại. Nếu giá trị alpha lớn hơn hoặc bằng beta, chúng ta có thể bỏ qua các nút con của nút hiện tại mà không cần tính toán tiếp.
* Mã giả thuật toán Minimax kết hợp cắt tỉa:

A picture containing text, screenshot, font, document

Description automatically generated

* 1. **Giải thuật Negamax:**
* Negamax là 1 biến thể của Minimax, trong đó giá trị của 2 người chơi được chuyển đổi thành số dương hoặc số âm.
* Giải thuật này sử dụng tính chất min(a,b) = -max(-b, -a) để đơn giản hóa việc triển khai giải thuật Minimax. Điều này giúp chúng ta tránh việc tính toán 2 TH riêng biệt cho mỗi người chơi.
* Giải thuật Negamax có thể kết hợp với cắt tỉa alpha-beta để cải thiện hiệu suất tìm kiếm. Alpha-beta pruning giúp loại bỏ các nhánh tìm kiếm không cần thiết, giúp giảm thời gian tính toán và tăng tốc độ tìm kiếm.
* Mã giả giải thuật Negamax:

A picture containing text, screenshot, font

Description automatically generated

* A screen shot of a computer program

  Description automatically generated with low confidenceDo Negamax hoạt động tương tự như Minimax, nên có thể áp dụng cắt tỉa Alpha- beta vào giải thuật Negamax. Mã giả giải thuật Negamax kết hợp cắt tỉa alpha - beta như sau:
  1. **Giải thuật Negascout:**
* Đối với các trò chơi có không gian trạng thái lớn như cờ vua thì phương pháp cắt tỉa alpha-beta vẫn không phù hợp vì không gian tìm kiếm khi đã kết hợp cắt tỉa vẫn rất lớn. Một phương pháp hiệu quả hơn giúp hạn chế không gian tìm kiếm của bài toán là giải thuật Negascout.
* Negascout là một giải thuật tìm kiếm alpha-beta cải tiến để tăng tốc độ tìm kiếm. Trong Negascout, khi tìm kiếm một nút, chúng ta sẽ tính toán giá trị của nút đó với một khoảng từ alpha đến beta (window). Nếu giá trị của nút nằm trong khoảng đó, chúng ta sẽ tiếp tục mở rộng nút đó để tìm kiếm giá trị chính xác hơn. Nếu giá trị nằm ngoài khoảng đó, chúng ta sẽ giảm khoảng tìm kiếm từ [alpha, beta] thành [−alpha − 1, −alpha] và tính toán lại giá trị của nút đó.
* Mã giả cho giải thuật Negascout:

A picture containing text, screenshot, font, document

Description automatically generated

**3. Cài đặt trò chơi:**

**3.1 Biểu diễn bàn cờ và luật chơi:**

Trạng thái bàn cờ được biểu diễn bởi lớp GameState, các nước đi có thể đi được biểu diễn bởi lớp Move.

a) Lớp GameState:

* Lớp GameState lưu trữ trạng thái của bàn cờ bởi các thuộc tính:
* board: một mảng 2 chiều có kích thước 8x8 trong đó mỗi phần tử đại diện cho ô vuông trên bàn cờ. Trong một ô vuông có thể không có quân cờ nào ở đó, kí hiệu: "- -". Một ô vuông có chứa một quân cờ sẽ được kí hiệu bởi màu và giá trị của quân cờ đó, ví dụ ô vuông có chứa quân hậu trắng được ký hiệu là "wQ".
* WhiteToMove: biến boolean cho biết hiện tại là lượt đi của quân nào, nếu là quân Trắng thì trả về True, ngược lại trả về False.
* MoveLog: 1 array lưu trữ tất cả nước đi đã được thực hiện.
* WhiteKingLocation: tọa độ của con vua trắng.
* BlackKingLocation: tọa độ của con vua đen.
* Lớp GameState có các phương thức như sau:
* makeMove(move): thực hiện 1 nước đi move được truyền vào, lưu nước đi đó vào moveLog, sau đó chuyển sang lượt của đối phương.
* undoMove: rollback lại nước đi gần nhất.
* getAllValidMoves: 1 array trả về các nước đi hợp lệ ở trạng thái hiện tại của trò chơi.
* inCheck: kiểm tra xem hiện tại quân vua có đang bị chiếu hay không, nếu có trả về True và ngược lại.
* getPawnMoves(r, c, moves): list ra các nước đi hợp lệ của quân Tốt đang ở vị trí (r, c) và thêm vào mảng moves.
* getRookMoves(r, c, moves): list ra các nước đi hợp lệ của quân Xe đang ở vị trí (r, c) và thêm vào mảng moves.
* getKnightMoves(r, c, moves): list ra các nước đi hợp lệ của quân Mã đang ở vị trí (r, c) và thêm vào mảng moves.
* getBishopMoves(r, c, moves): list ra các nước đi hợp lệ của quân Tượng đang ở vị trí (r, c) và thêm vào mảng moves.
* getQueenMoves(r, c, moves): list ra các nước đi hợp lệ của quân Hậu đang ở vị trí (r, c) và thêm vào mảng moves.
* getKingMoves(r, c, moves): list ra các nước đi hợp lệ của quân Vua đang ở vị trí (r, c) và thêm vào mảng moves.

b) Lớp Move:

Lớp Move biểu diễn cho các nước đi của quân cờ bởi các thuộc tính phục vụ việc ánh xạ từ ký hiệu cột (a,...,i) và hàng (1,...,11) trên bàn cờ sang chỉ số hàng và cột cho bàn cờ lưu trong chương trình. Mỗi vị trí trên bàn cờ sẽ được biểu diễn dưới dạng (hàng, cột). Mỗi nước đi của quân cờ sẽ được khởi tạo với các thông tin: vị trí bắt đầu, vị trí kết thúc, quân di chuyển, quân bị ăn và mã số (moveID) của nước đi đó phục vụ cho việc so sánh. Từ đó, ta mã hóa và lưu trữ được các nước đi trong trò chơi phục vụ việc chơi cờ của người dùng và triển khai các giải thuật tìm kiếm có đối thủ sau này.

**3.2 Áp dụng các kỹ thuật tìm kiếm có đối thủ:**

Các giải thuật tìm kiếm bao gồm 5 lớp sau:

* AI
* Greedy
* Minimax
* Negamax
* Negascout

a) Lớp AI:

* Lớp AI đại diện cho 1 đối tượng như 1 người máy AI với các thuộc tính CHECKMATE (kiểm tra xem có bị chiếu tướng không), DEPTH (độ sâu tìm kiếm của giải thuật), nextMove (nước đi tiếp theo AI sẽ thực hiện). Ở đây cũng định nghĩa hàm ước lượng bằng cách tính điểm cho bàn cờ.
* Đoạn mã tính hàm ước lượng:
* A picture containing text, screenshot, font, algebra

  Description automatically generatedHàm này duyệt qua từng ô trên bàn cờ và tính điểm tương ứng với quân cờ nằm trên mỗi ô. Ở đây, ta sẽ tính tổng điểm của bàn cờ ở trạng thái hiện tại, với quân trắng thì sẽ tính điểm số dương, quân đen sẽ tính điểm số âm. Cuối cùng, hàm sẽ trả về tổng số điểm hiện tại trên bàn cờ. Đánh giá hàm ước lượng nhằm mục đích tính xem trên bàn cờ, bên nào đang có lợi thế về chất lượng quân cờ hơn. Nếu điểm trả về từ hàm là một số dương càng lớn thì tức là quân trắng đang có lợi thế càng lớn và ngược lại.

b) Greedy:

* Lớp Greedy là lớp con của lớp AI, do đó có thể kế thừa và sử dụng các thuộc tính và phương thức của lớp AI. Lớp này xây dựng để triển khai giải thuật tham lam trong việc tìm kiếm nước đi tiếp theo trong trò chơi.
* Hàm findMove sử dụng giải thuật tham lam để tìm nước đi tiếp theo tốt nhất ở trạng thái hiện tại. Hàm này sẽ lấy tất cả các nước đi hợp lệ và đánh giá điểm cho mỗi nước đi. Nước đi có điểm cao nhất sẽ được trả về là nước đi tiếp theo tốt nhất cho AI.
* Giải thuật Greedy tương đối đơn giản, dễ hiểu, thời gian tìm kiếm nhanh. Nhưng có hạn chế là nó chỉ tìm kiếm nước đi tối ưu trong mỗi bước hiện tại mà không tính toán đến những nước đi sau.

c) Minimax, Negamax, Negascout:

* 3 lớp Minimax, Negamax và Negascout kế thừa từ lớp AI và override phương thức findMove để tìm kiếm nước đi tốt nhất tiếp theo cho trạng thái hiện tại của bàn cờ.
* Cách hoạt động chung của 3 lớp này là: Từ phương thức findMove, nhận giá trị tham số gồm trạng thái hiện tại của trò chơi và danh sách các nước hợp lệ, từ đó gọi đến phương thức tiếp theo (findMoveMinimax, findMoveNegamax và findMoveNegascout) triển khai các giải thuật riêng. Các giải thuật đều được tìm kiếm với 1 độ sâu đã xác định trước.

**3.3 Giao diện trò chơi:**

***Giao diện khi bắt đầu của trò chơi***

**Giao diện chọn thuật toán**

A picture containing chessman, board game, chess, chessboard

Description automatically generated**Giao diện bàn cờ**

**4. Đánh giá**

**4.1 Minimax và Negamax**

* Về lý thuyết, Negamax là cách triển khai khác của Miniax, nên bọn em kì vọng số trạng thái phải duyệt của hai giải thuật tại cùng một tình huống là như nhau.
* Dưới đây là kết quả đánh giá tại độ sâu bằng 3:
* A picture containing text, screenshot, plot, line

  Description automatically generatedNhư hình minh họa, số trạng thái của Minimax và Negamax tại mỗi nước đi luôn bằng nhau. Ngoài ra, kết quả nước đi tiếp theo được trả về của cả hai giải thuật cũng giống nhau. Điều này chứng tỏ Negamax là một triển khai khác của giải thuật Minimax là chính xác.

**4.2 Negamax và Negamax cắt tỉa alpha – beta:**

* Do Minimax và Negamax tương tự nhau về bản chất, nên việc so sánh giữa Negamax có cắt tỉa alpha-beta và Negamax tương đương với việc so sánh giữa Minimax có cắt tỉa alpha-beta và Minimax.
* Dưới đây là kết quả đánh giá của nhóm bọn em tại độ sâu 3:

A picture containing text, plot, line, screenshot

Description automatically generated

* Như hình vẽ ta có thể thấy, số nước đi tại mỗi bước đã giảm hơn rất nhiều lần khi sử dụng phương pháp cắt tỉa. Nhờ sửa dụng cắt tỉa, nó giúp loại bỏ các nhánh không tối ưu mà không ảnh hưởng đến kết quả cuối cùng. Điều này giúp làm giảm không gian tìm kiếm đi rất nhiều và từ đó tăng tốc độ chạy. Điều này hoàn toàn chính xác như lý thuyết.

**4.3 Negamax có cắt tỉa alpha-beta và Negascout**:

* Theo lý thuyết, Negascout cắt tỉa nhiều hơn so với Negamax sử dụng cắt tỉa alpha-beta.
* Dưới đây là kết quả đo đạc của nhóm tại độ sâu bằng 3:

A picture containing text, screenshot, line, plot

Description automatically generated

* Như hình vẽ, có thể thấy là thuật toán Negascout có chút cải tiến hơn so với khi sử dụng Negamax cắt tỉa alpha-beta. Đặc biệt ở nước đi 13 và 15, không gian tìm kiếm của Negascout đã được giảm đi rất nhiều. Đồng thời, nước đi tiếp theo được trả về của hai thuật toán là như nhau. Vì vậy, tại những độ sâu lớn, Negascout sẽ giảm không gian tìm kiếm, tăng tốc độ chạy và không làm ảnh hưởng đến nước đi tìm thấy.

**5. Tài liệu tham khảo**

* Stuart J. Russel and Peter Norvig. Artificial Intelligence: A Modern Approach. Pearson, 3 edition, 2010.
* Bài giảng IT 3160 Nhập môn Trí tuệ nhân tạo: PGS.TS Phạm Văn Hải & PGS.TS Lê Thanh Hương.
* Wikipedia Negmax. Negamax URL <https://en.wikipedia.org/wiki/Negamax>
* Chess Programming Wiki. Negascout URL <https://www.chessprogramming.org/NegaScout>
* [Chess Programming Wiki. Alpha – Beta](https://www.chessprogramming.org/NegaScout)

<https://www.chessprogramming.org/Alpha-Beta>

* Chess Engine Python

<https://www.youtube.com/watch?v=EnYui0e73Rs&list=PLBwF487qi8MGU81nDGaeNE1EnNEPYWKY_>