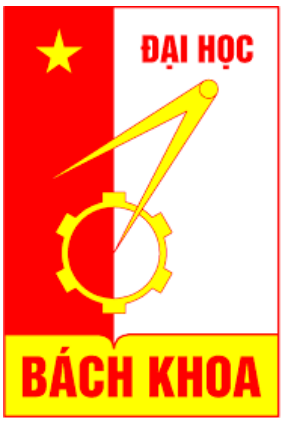
□□□□□

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO BÀI TẬP LỚN MÔN HỌC

TRÍ TUỆ NHÂN TẠO

Đề tài:

XÂY DỰNG TRÒ CHƠI CỜ TƯỚNG AI

Giảng viên hướng dẫn: Đỗ Tiến Dũng

Nhóm sinh viên thực hiện:

Hoàng Văn Kiên - 20205089

Nguyễn Thế Hưng - 20200295

Tạ Đức Tiến - 20205132

Nguyễn Chí Thành - 20205127

Mục lục

Trong một thời đại mà nền công nghiệp tự động hóa chiếm một vị trí không thể thiếu của một nền kinh tế phát triển, máy móc càng chứng tỏ những ưu thế vượt trội nhờ sự chính xác, độ ổn định cao,… Nhưng đằng sau đó, mỗi chiếc máy là do con người điều khiển, trí tuệ của máy móc cũng là do con người cài đặt mà có được. Vì vậy, tri thức con người vẫn luôn là yếu tố then chốt.

Từ những năm 50 của thế kỉ XX, trí tuệ nhân tạo mới bắt đầu nhen nhóm, vậy mà giờ đây, ta có thể thấy nó đã có những bước phát triển vượt bậc, không ngừng. Một trong những ứng dụng có thể kể đến của AI đó là việc xây dựng, phát triển các trò chơi đối kháng. Cờ tướng cũng là một trong số đó. Cờ tướng có nguồn gốc từ Trung Quốc,là trò chơi trí tuệ dành cho hai người,cùng với cờ vua trở thành hai loại cờ phổ biến nhất trên thế giới.

Để hiểu rõ hơn về AI nói chung cũng như các kỹ thuật tìm kiếm nói riêng chúng em đã quyết định lựa chọn đề tài: “Chương trình cờ tướng ”, một trò chơi cổ điển nhưng là một trong những trò chơi biểu tượng của trí tuệ để hoàn thành bài tập lớn môn học này. Dưới đây là phần trình bày báo cáo môn học thông qua đề tài đã nêu ở trên. Hiện tại, chương trình của chúng em còn nhiều thiếu sót, rất mong nhận được những ý kiến góp ý và tư vấn từ thầy và các bạn để chúng em có thể trau dồi thêm cho bản thân những kiến thức hữu ích.

Chúng em xin chân thành cảm ơn !

I. GIỚI THIỆU VÀ MÔ TẢ VỀ BÀI TOÁN

1. Bài toán thực tế cần giải quyết

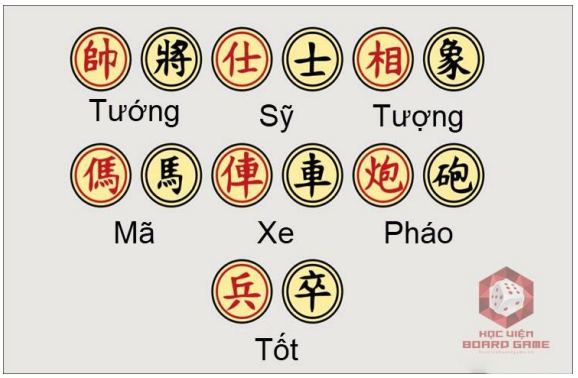
1.1. Giới thiệu trò chơi

Cờ tướng là boardgame trí tuệ, 1 trong những game được chơi nhiều nhất ở Việt Nam hiện nay cùng với bộ môn cờ vua.
Trò chơi này đã du nhập vào Việt Nam rất lâu và được chơi bởi mọi lứa tuổi, từ các bậc bô lão – cao niên đến các trẻ nhỏ đều có thể bị cuốn hút với những nước cờ đầy tính toán này. Vì được người Trung Quốc phát triển và hoàn thiện nên cờ tướng đã xây dựng nên hình ảnh của một quốc gia thu nhỏ với bàn cờ là một trận địa sinh động và 32 quân cờ đầy đủ các binh chủng. Có đầy đủ các tầng lớp: công, thủ, sông, cung các quân được chia thành ba lớp xen kẽ hài hoà.

- BIỂU TƯỢNG VÀ TÊN GỌI CÁC QUÂN CỜ:

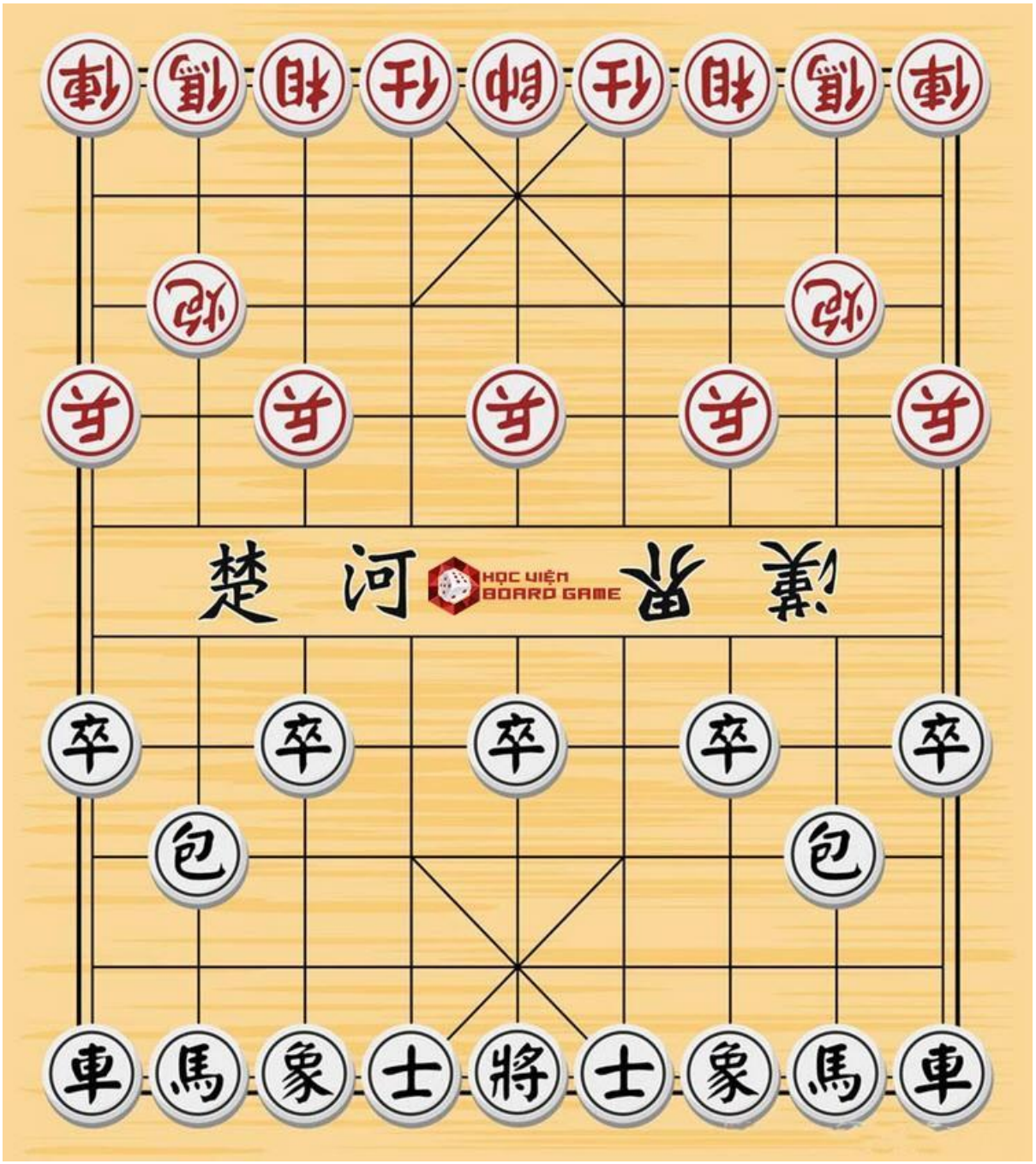
Tổng cộng có 32 quân cờ trên bàn cờ chia làm 2 bên, mỗi bên 16 quân với màu sắc đen, đỏ (hoặc trắng), với số lượng mỗi quân cờ là:

- 5 quân Tốt
- 2 quân Pháo
- 2 quân Xe
- 2 quân Mã
- 2 quân Tượng
- 2 quân Sỹ
- 1 quân Tướng.



- CÁCH SẮP XẾP CÁC QUÂN CỜ TRONG BÀN CỜ TƯỚNG

Để sắp xếp bàn cờ tướng bạn chỉ cần thuộc các quân cờ được mô tả ở dưới sau đó sắp xếp như hình mẫu bên dưới là được.



1.2. Các quy tắc của trò chơi

Bàn cờ được chia làm 2 đội, là đội đen (black, ký hiệu b) và đội đỏ (red , ký hiệu r), mỗi đội gồm 16 quân, bao gồm 1 con tướng (General hoặc king , ký hiệu k), 2 con sỹ (Advisor hoặc guards, ministers, ký hiệu là a), 2 con tượng (Elephants hoặc bishops - ký hiệu là b), 2 con mã (Horses hoặc knights - ký hiệu là n, do chữ k trùng với king là con tướng, nên người ta xài chữ n), 2 con xe (Chariot hoặc rooks - ký hiệu là r), 2 con pháo (canons, ký hiệu là c), 5 con chốt (Soldiers , ký hiệu là p (do con chốt ở cờ đen và cờ đỏ có phiên âm tiếng trung khác nhau, chốt cờ đen đọc gần giống chữ "zú" ("pawn" hoặc "private" - tiếng anh), còn chốt cờ đỏ đọc là bing ("soldier" - tiếng anh))).

Tổng cộng, ta có tướng, sỹ, tượng, mã, xe, pháo, chốt, 7 loại quân, tương đương với 7 ký hiệu, tổ hợp với 2 đội là đỏ và đen, tổ hợp với nhau, ta xác định được

Tham khảo thêm quy tắc tại:
https://hocvienboardgame.vn/huong-dan-cach-choi-co-tuong/

II. Một số phương pháp và khái niệm liên quan

II.1.1. Dạng trò chơi:

Các trò chơi có dạng như cờ Vua, cờ Tướng, cờ Vây, cờ Caro,… là những trò chơi đối kháng, diễn ra giữa hai đấu thủ. Nói chung, các trò chơi này đều có thể chuyển về một dạng bài toán tìm kiếm đặc biệt: tìm đường đi đến các điểm cao nhất giữa hai đấu thủ. Đặc điểm của loại trò chơi này như sau:

Có hai đấu thủ, mỗi người chỉ đi một nước khi tới lượt.

Các đấu thủ đều biết mọi thông tin về tình trạng trận đấu.

Trận đấu không kéo dài vô tận, phải diễn ra hài hoà, hoặc một bên thắng và bên kia thua.

II.1.2. Cây trò chơi

Các trạng thái bàn cờ khác nhau trong quá trình chơi có thể biểu diễn thành một cây tìm kiếm và ta sẽ tiến hành tìm kiếm trên cây để tìm được nước đi tốt nhất. Ở cây này, các nút của cây là các tình huống khác nhau của bàn cờ, các nhánh nối sẽ cho ta biết từ một tình huống cờ thế này chuyển sang tình huống cờ như thế khác thông qua một nước đi đơn nào đó. Các nước đi này diễn ra theo cặp do hai đấu thủ lần lượt tiến hành. Độ sâu của cây trò chơi là số tầng của cây.

II.1.3. Vét cạn

Thuật toán vét cạn có thể hiểu đơn giản là sinh ra hết tất cả mọi khả năng có thể xảy ra trong trò chơi. Sau đó tiến hành lựa chọn đánh giá trên từng khá năng, từ đó chọn ra phương án tối ưu nhất. Trong cờ tướng, nếu bạn áp dụng thuật toán này để tính toán nước đi, kết quả trả về sẽ rất chính xác.

Giả sử ta coi độ sâu của cây là d , hệ số nhánh là b thì nếu dùng thuật toán vét cạn thì số trường hợp có thể là b^d trường hợp (nút) ⇒ Bùng nổ tổ hợp , không phù hợp với trò chơi bài toán thực tế.

Qua phân tích trên ta thấy không thể áp dụng hoàn toàn thuật toán vét cạn vào trò chơi được. Để tạo ra một thuật toán đánh cờ tướng tối ưu cho máy tính, chúng ta phải xác định được chiến lược tìm kiếm trong trò chơi rồi dựa vào đó xây dựng nên những thuật toán tối ưu cho máy tính.

II.1.4 Chiến lược tìm kiếm

Khi chơi cờ , luôn phải xét hữu hạn các nước đi trước để tự tìm chiến thuật cho mình .

⇒ Cần phải có chiến lược tìm kiếm sao cho cả người lẫn máy có thể phân tích thế cờ của đối phương dựa trên một số nước đi trước đó.

Chiến lược có thể đánh giá độ tốt xấu của thế cờ nhận được. Tiếp theo, ta sẽ chọn nước đi sẽ dẫn tới một thế cờ tốt nhất trong đó số đó có cân nhắc đến cách đi của cả hai bên (thường chúng ta xem như cả hai người đều đưa ra những lựa chọn tối ưu). Với máy thế cờ này được đánh giá tối hơn thế cờ kia nhờ so sánh điểm của thế đó do bộ lượng giá đáp trả

lại.(Việc lượng giá quân cờ có thể hiểu đơn giản rằng, gán mỗi quân cờ một số điểm nhất định).

Vậy nếu càng xét sâu, nước đi càng chắc chắn và cơ hội chiến thắng càng cao.

II.2. Thuật toán và cấu trúc dữ liệu

II.2.1. Lượng giá thế cờ

- Mỗi quân cờ trên bàn cờ có một chức năng quan trọng khác nhau trên bàn cờ nên sẽ có mức độ lượng giá khác nhau. Ở bài lần này chúng em tham khảo lượng giá bàn cờ như sau:

Tướng của ta là 900 điểm, tướng của đối thủ là -900 điểm

Sỹ của ta là 20 điểm, sỹ của đối thủ là -20 điểm

Tượng của ta là 20 điểm, tượng của đối thủ là -20 điểm

Mã của ta là 40 điểm, mã của đối thủ là -40 điểm

Xe của ta là 90 điểm, xe của đối thủ là -90 điểm

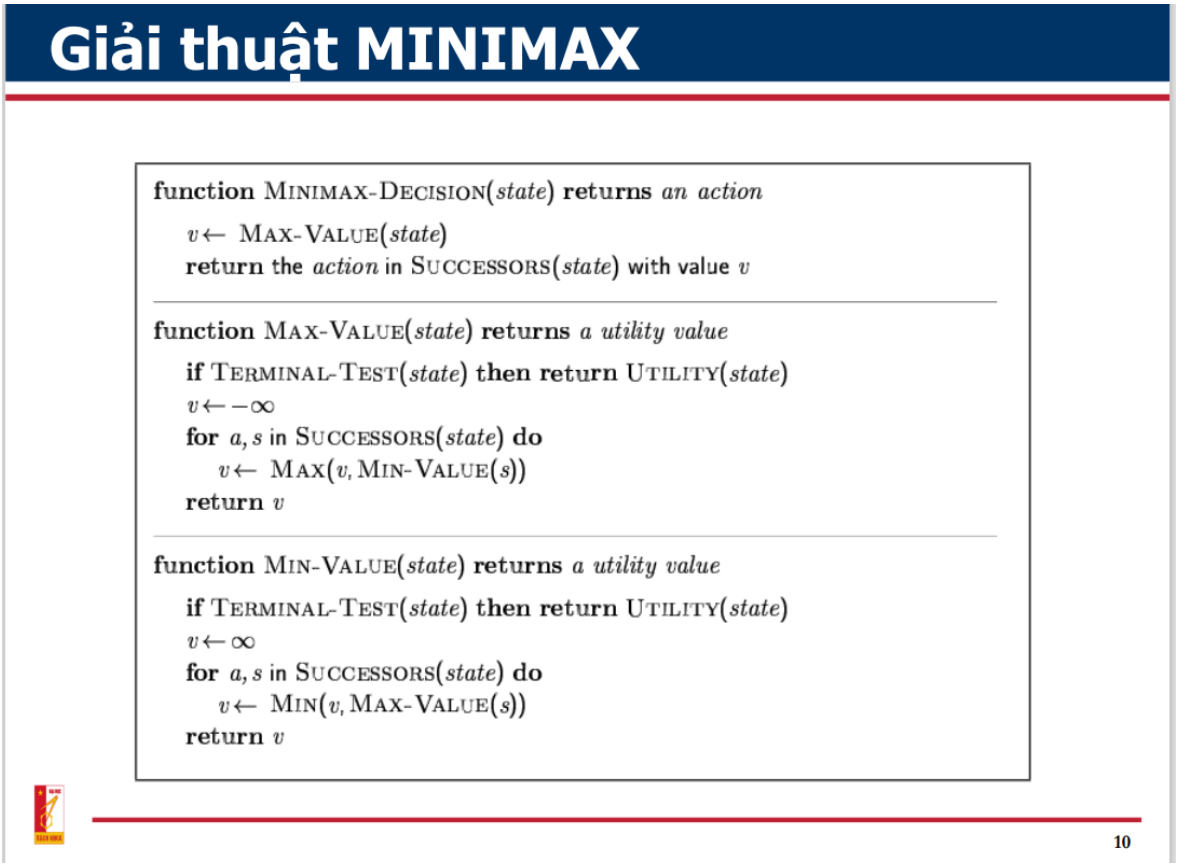
Pháo của ta là 45 điểm, pháo của đối thủ là -45 điểm

Tốt của ta là 15 điểm, tốt của đối thủ là -15 điểm

II.2.2 Thuật toán Minimax

Thuật toán minimax thuộc nhóm duyệt theo chiều sâu (depth first search). Hai người chơi, một người được gọi là MAX, người còn lại gọi là MIN. Thuật toán được thiết kế để tìm nước đi tối ưu cho người MAX. Người MAX sẽ giữ nút gốc, lần lượt duyệt đệ quy qua tất cả các nút con theo chiều sâu nhất định đến khi duyệt qua tất cả các nút hoặc là tìm được một đường đi mà đạt MAX.

- Áp dụng thuật toán MiniMax trong việc tìm kiếm nước đi cho máy bằng cách là duyệt qua toàn bộ các nước đi có thể của người chơi max và min rồi đưa ra lựa chọn tốt nhất cho người đi



Tham khảo thuật toán Minimax tại đây :
https://vi.wikipedia.org/wiki/Minimax

Nhưng vì duyệt toàn bộ nước đi nên thời gian duyệt toàn bộ sẽ rất lâu nên có thể dùng thuật toán Cắt tỉa Alpha - Beta để giảm thời gian thuật toán.

II.2.3. Cải tiến Alpha – Beta:

Cải tiến Alpha – Beta là một cải tiến từ thuật toán Minimax nhằm tỉa bớt nhánh cây của cây trò chơi từ đó làm giảm số lượng nút phải sinh và thao tác lượng giá, do đó có thể tăng độ sâu tìm kiếm để thuật toán trở nên hiệu quả hơn. Ta dễ dàng nhận thấy: khi đến một nút, thuật toán sẽ xét hết toàn bộ các nút con của cây trò chơi bất chấp nút đó là tốt hay xấu. Chính điều này đã làm cho Minimax chạy chậm đi rất nhiều lần. Từ đây, ta hình thành nên một cải tiến mới: **Cải tiến Alpha – Beta.**

Tham khảo thêm tại :
https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning

Giải thuật cắt tỉa α-β (1)

```
function ALPHA-BETA-SEARCH(state) returns an action
  inputs: state, current state in game
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for a, s in SUCCESSORS(state) do
    v ← MAX(v, MIN-VALUE(s, α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v
```

19

```
function MIN-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s, α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Cải tiến Alpha - Beta tiết kiệm lượng lớn thời gian nhưng vẫn không tránh được việc bùng nổ tổ hợp nên nhóm đã nghĩ đến hướng cải tiến thuật toán Alpha - Beta vào bài toán của mình .

II.2.4. Hướng cải thiện việc tỉa nhánh của thuật toán AlphaBeta

Thuật toán AlphaBeta nói chung giúp chúng ta tiết kiệm nhiều thời gian so với Minimax mà vẫn đảm bảo kết quả tìm kiếm chính xác. Tuy nhiên lượng tiết kiệm này không ổn định - phụ thuộc vào số nút mà nó cắt bỏ. Trong trường hợp xấu nhất thuật toán không cắt được một nhánh nào và phải xét số nút đúng bằng Minimax. Ta cần đẩy mạnh việc cắt bỏ nhờ đẩy nhanh sự thu hẹp của cửa sổ tìm kiếm alpha - beta. Cửa sổ này được thu hẹp một bước khi gặp một giá trị mới tốt hơn giá trị cũ. Khi gặp giá trị tốt nhất thì cửa sổ này thu hẹp nhất. Do đó nếu càng sớm gặp giá trị tốt nhất thì cửa sổ càng chóng thu hẹp. Như vậy phải làm sao cho các nút ở lá được sắp xếp theo trật tự từ cao xuống thấp. Trật tự ngày càng tốt bao nhiêu thì thuật toán chạy càng nhanh bấy nhiêu (các công thức về số nút phải lượng giá trong điều kiện lý tưởng ở trên tính được với trật tự là tốt nhất).

III. THỰC HIỆN XÂY DỰNG CHƯƠNG TRÌNH

III.1.Mô tả chương trình

III.1.1 Biểu diễn bàn cờ và quân cờ:

- Sử dụng thư viện javascript xiangqiboard có sẵn cung cấp các API cần thiết để lập trình giao diện cho bàn cờ.

```
463   let config = {
464     draggable: true,
465     position: 'start',
466     onDragStart: onDragStart,
467     onDrop: onDrop,
468     onMouseoutSquare: onMouseoutSquare,
469     onMouseoverSquare: onMouseoverSquare,
470     onSnapEnd: onSnapEnd
471   };
472
473   board = Xiangqiboard('myBoard', config);
```

Hình: thiết lập config cho bàn cờ và gọi đến hàm tạo bàn cờ của thư viện xiangqiboard.

III.1.2 Sinh nước đi:

- Như đã trình bày ở II.2, chúng em sẽ dùng giải thuật minimax có áp dụng cắt tỉa alpha - beta để sinh nước đi cho máy. Các node trên cây trò

chơi sẽ được sinh bởi thư viện xiangqiboard. Cụ thể triển khai trong hàm minimax() như sau:

```
function minimax(depth, game, alpha, beta, isMaximisingPlayer) {

  if (depth === 0) {
    return -evaluateBoard(game.board());
  }

  var newGameMoves = game.moves();  // Sinh cac node trong cay tro choi

  if (isMaximisingPlayer) {
    var bestMove = -9999;
    for (var i = 0; i < newGameMoves.length; i++) {
      game.move(newGameMoves[i]);
      bestMove = Math.max(bestMove, minimax(depth - 1, game, alpha, beta, !isMaximisingPlayer));
      game.undo();
      alpha = Math.max(alpha, bestMove);
      if (beta <= alpha) {
        return bestMove;
      }
    }
    return bestMove;
  } else {
    var bestMove = 9999;
    for (var i = 0; i < newGameMoves.length; i++) {
      game.move(newGameMoves[i]);
      bestMove = Math.min(bestMove, minimax(depth - 1, game, alpha, beta, !isMaximisingPlayer));
      game.undo();
      beta = Math.min(beta, bestMove);
      if (beta <= alpha) {
        return bestMove;
      }
    }
    return bestMove;
  }
}
```

- Thực hiện nước đi bằng 2 hàm getBestMove() và makeBestMove() như sau:

```
function makeBestMove() {
  var bestMove = getBestMoveMaxi(game);
  game.move(bestMove);
  board.position(game.fen());
 // nuocCo.play();
  updateStatus();
}

function getBestMoveMaxi(game) {
  updateStatus();
  var depth = 4;
  var bestMove = minimaxRoot(depth, game, true);
  return bestMove;
}
```

III.1.3 Đánh giá một thế cờ

- Để lượng giá một thế cờ, ta cần lượng giá hết tất cả các quân trên bàn cờ theo mức độ quan trọng và vị trí của chúng.
- Về mức độ quan trọng, nhóm để số điểm của các loại quân như sau: tướng 6000, sĩ 120, tượng 120, mã 270, pháo 285, xe 600, tốt 30.

- Về vị trí, với mỗi loại quân, nhóm sẽ có 1 ma trận integer 10x9 thể hiện giá trị của loại quân đó tại các vị trí trên bàn cờ.

```
const pEvalRed =
[
        [ 0,  3,  6,  9, 12,  9,  6,  3,  0],
        [18, 36, 56, 80,120, 80, 56, 36, 18],
        [14, 26, 42, 60, 80, 60, 42, 26, 14],
        [10, 20, 30, 34, 40, 34, 30, 20, 10],
        [ 6, 12, 18, 18, 20, 18, 18, 12,  6],
        [ 2,  0,  8,  0,  8,  0,  8,  0,  2],
        [ 0,  0, -2,  0,  4,  0, -2,  0,  0],
        [ 0,  0,  0,  0,  0,  0,  0,  0,  0],
        [ 0,  0,  0,  0,  0,  0,  0,  0,  0],
        [ 0,  0,  0,  0,  0,  0,  0,  0,  0],
];
```

Hình: ma trận thể hiện điểm tùy vào vị trí của quân tốt đỏ.

- Kết hợp cả 2 yếu tố trên, hàm tính giá trị quân cờ được viết như sau:

```
var getAbsoluteValue = function (piece, isRed, x ,y) {
  if (piece.type === 'p') { //chốt
    return 30 + ( isRed ? pEvalRed[x][y] : pEvalBlack[x][y] );
  } else if (piece.type === 'r') { //Xe
    return 600 +( isRed ? rEvalRed[x][y] : rEvalBlack[x][y] );
  } else if (piece.type === 'c') { //pháo
    return 285 +( isRed ? cEvalRed[x][y] : cEvalBlack[x][y] );
  } else if (piece.type === 'n') { // mã
    return 270 +( isRed ? nEvalRed[x][y] : nEvalBlack[x][y] );
  } else if (piece.type === 'b') { // tượng
    return 120 +( isRed ? bEvalRed[x][y] : bEvalBlack[x][y] );
  } else if (piece.type === 'a') { // sỹ
    return 120 +( isRed ? aEvalRed[x][y] : aEvalBlack[x][y] );
  } else if (piece.type === 'k') { // tướng
    return 6000 +( isRed ? kEvalRed[x][y] : kEvalBlack[x][y] );
  }
  throw "Unknown piece type: " + piece.type;
};
```

- Cuối cùng là hàm lượng giá thế cờ:

```
function evaluateBoard(board) {
  var totalEvaluation = 0;
  for (var i = 0; i < 10; i++) {
    for (var j = 0; j < 9; j++) {
      totalEvaluation = totalEvaluation + getPieceValue(board[i][j], i ,j);
    }
  }
  return totalEvaluation;
}
```

III.1.4 Xử lý điều khiển của người chơi:

- Chúng em cung cấp cho người chơi 1 số thao tác với bàn cờ như: đưa chuột vào vị trí quân cờ để hiện các nước có thể đi, kéo thả quân cờ để di chuyển.

```
400   function onMouseoverSquare (square, piece) {
401     // get list of possible moves for this square
402     let moves = game.moves({
403       square: square,
404       verbose: true
405     });
406
407     // exit if there are no moves available for this square
408     if (moves.length === 0) return;
409
410     // highlight the square they moused over
411     greySquare(square);
412
413     // highlight the possible squares for this piece
414     for (let i = 0; i < moves.length; i++) {
415       greySquare(moves[i].to);
416     }
417   }
```

Hình: hàm onMouseOverSquare() hiện các nước có thể đi khi di chuột vào quân cờ.

```
379   function onDrop (source, target) {
380     removeGreySquares();
381
382     // see if the move is legal
383     let move = game.move({
384       from: source,
385       to: target,
386       promotion: 'q' // NOTE: always promote to a queen for example simplicity
387     });
388
389     // illegal move
390     if (move === null) return 'snapback';
391
392    // updateStatus();
393
394     // make random legal move for black
395     // window.setTimeout(makeRandomMove, 250);
396     window.setTimeout(makeBestMove, 250);
397   }
```

Hình: hàm onDrop() thực hiện di chuyển quân cờ khi thả chuột.

III.1.5 Vòng lặp chính xử lý trò chơi

- Vòng lặp xử lý trò chơi cần xác định được đến lượt đi của ai, và xác định trạng thái kết thúc (hòa hoặc chiếu hết cờ) để tránh việc lặp vô hạn:

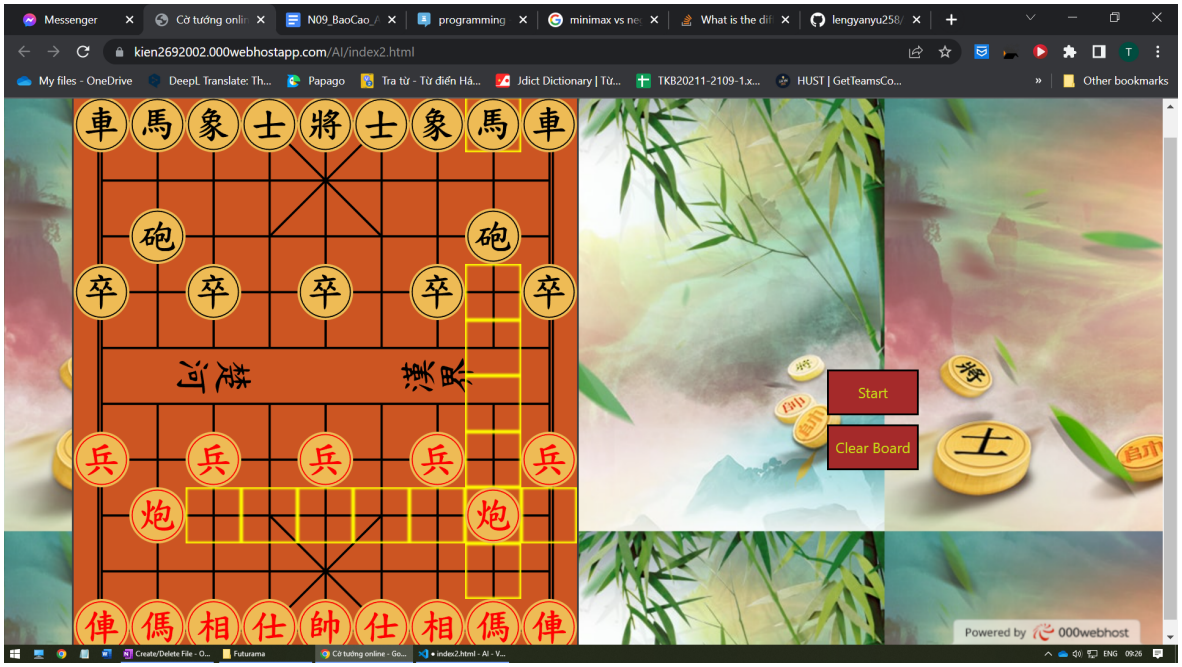
```
function updateStatus () {
  let status = '';

  let moveColor = 'Red';
  if (game.turn() === 'b') {
    moveColor = 'Black';
  }

  // checkmate?
  if (game.in_checkmate()) {
    status = 'Game over, ' + moveColor + ' is in checkmate.';
  }

  // draw?
  else if (game.in_draw()) {
    status = 'Game over, drawn position';
  }

  // game still on
  else {
    status = moveColor + ' to move';

    // check?
    if (game.in_check()) {
      status += ', ' + moveColor + ' is in check';
    }
  }
```

Hình: hàm updateStatus() thực hiện xử lý trò chơi.

IV. KẾT LUẬN

1. Kết quả và so sánh với mục tiêu đặt ra:

- Hình ảnh về chương trình:



- Chương trình đã có khả năng sinh ra các nước đi khá thông minh khi đặt độ sâu của thuật toán minimax là 4. Tuy vậy, do sự bùng nổ tổ hợp các nước đi trong trò chơi nên thời gian chạy thuật toán minimax kết hợp cắt tỉa alpha - beta là vẫn khá lâu.

2. Hướng phát triển

- Cải thiện thuật toán sinh nước đi minimax hiện tại để chương trình đạt tốc độ nhanh hơn.

V. CÁC NỘI DUNG KHÁC

1.Phân công công việc trong đề tài

1. Hoàng Văn Kiên: tìm tài liệu, hiểu thuật toán, cài đặt thử nghiệm.
2. Tạ Đức Tiến: tìm hiểu thuật toán minimax, làm báo cáo, slide.
3. Nguyễn Chí Thành: tìm hiểu hàm đánh giá, thuyết trình.
4. Nguyễn Thế Hưng: Tìm hiểu thuật toán cắt tỉa alpha-beta, làm slide

2. Gói phần mềm , dữ liệu có sẵn được sử dụng:

Thư viện tạo giao diện bàn cờ xiangqiboard

https://github.com/lengyanyu258/xiangqiboardjs

3. **Tham khảo**

https://www.phamduytung.com/blog/2021-08-12-china_chess_alpha_beta_ai/