TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

TRÍ TUỆ NHÂN TẠO

*****

***ĐỒ ÁN* :**

CỜ TƯỚNG

(Đã chỉnh sửa theo góp ý của giáo viên)

Lớp: *CS106.I22*

GVLT*: Huỳnh Thị Thanh Thương*

NHÓM TH:

Nguyễn Văn Tài - 13520729

TP. Hồ Chí Minh, tháng 07 năm 2018

Mục Lục

CHƯƠNG 1: GIỚI THIỆU CỜ TƯỚNG

1.1. Nguồn gốc trò chơi

Cờ tướng (phiên âm Hán Việt là Tượng Kỳ), hay còn gọi là cờ Trung Hoa (Tiếng Anh: Chinese Chess) vì nó được phổ biến ra thế giới từ Trung (nhưng theo phương Tây thì nó có nguồn gốc từ Ấn Độ), là một trò chơi trí tuệ dành cho hai người, là loại cờ được chơi phổ biến nhất thế giới cùng với cờ vua.

Ván cờ được tiến hành giữa hai người, một người cầm quân Trắng (hay Đỏ), một người cầm quân Đen (hay Xanh lá cây). Mục đích của mỗi người là tìm mọi cách đi quân trên bàn cờ theo đúng luật để chiếu bí hay bắt Tướng (hay Soái, hoặc Suý) của đối phương và giành thắng lợi.

Bàn cờ tướng thật sự là một trận địa sinh động, có tầng có lớp và thật hoàn hảo: đủ các binh chủng trên chiến trường, công có, thủ có, các quân được chia thành ba lớp xen kẽ hài hoà. Lại còn có cả sông, cung cấm. Hình tượng quốc gia hoàn chỉnh, có vua tôi, có 5 binh chủng, có quan ở nhà, quân ra trận v.v..., vừa có ý nghĩa, vừa mang sắc thái phương Đông rõ nét.

Tại Việt Nam, cờ tướng được chơi rất phổ biến trong đời sống hàng ngày cũng như dịp lễ tết. Người chơi cờ tướng ở Việt Nam đã sáng tạo ra cách chơi mới, đó là "cờ người".

1.2. Mô tả trò chơi

+Cách chơi:

Ván cờ được tiến hành giữa hai người, một người cầm quân Trắng (hay Đỏ), một người cầm quân Đen (hay Xanh lá cây). Mục đích của mỗi người là tìm mọi cách đi quân trên bàn cờ theo đúng luật để chiếu bí hay bắt Tướng (hay Soái, hoặc Suý) của đối phương và giành thắng lợi.

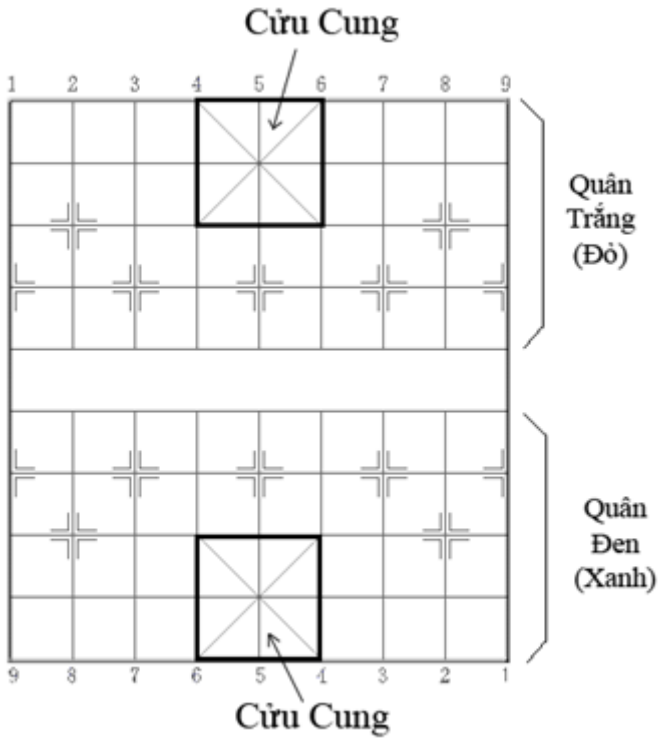
+ Sơ lược về luật chơi:

Bàn cờ: là một hình chữ nhật do 9 đường dọc và 10 đường ngang cắt nhau vuông góc tại 90 điểm hợp thành. Một khoảng trống gọi là sông (hay hà) nằm ngang giữa bàn cờ, chia bàn cờ thành hai phần đối xứng bằng nhau. Mỗi bên có một cung Tướng hình

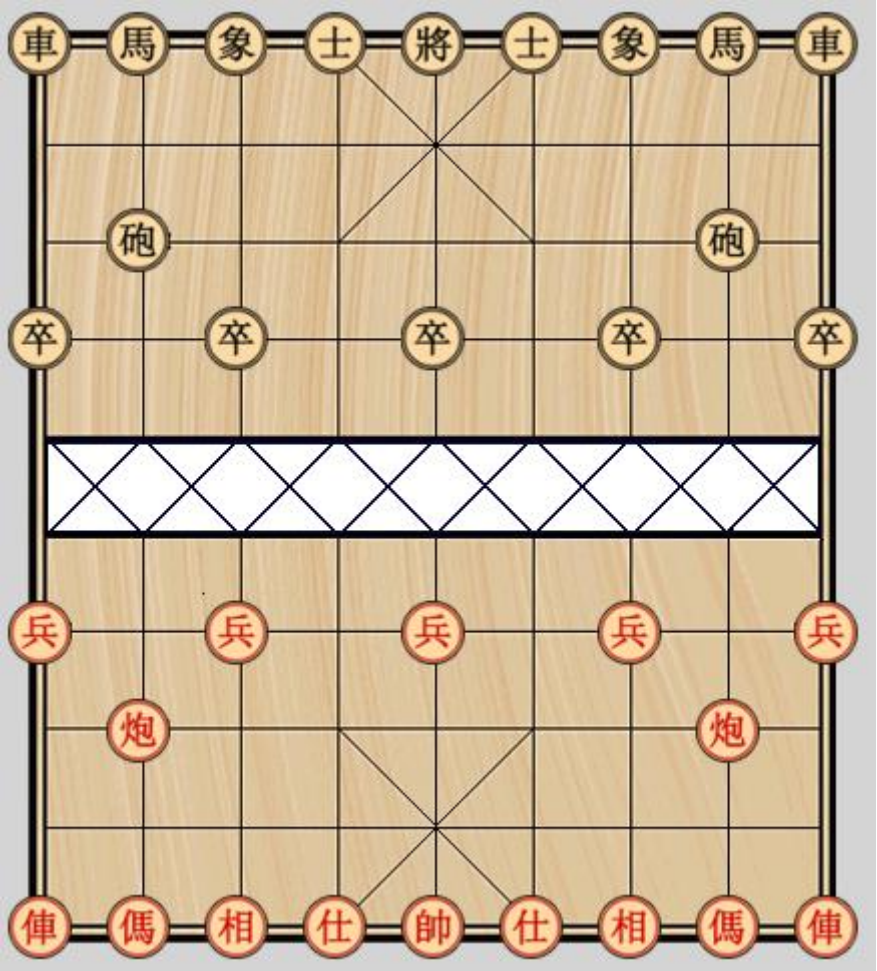
vuông (Cửu cung) do 4 ô hợp thành tại các đường dọc 4, 5, 6 kể từ đường ngang cuối của mỗi bên, trong 4 ô này có vẽ hai đường chéo xuyên qua.

Theo quy ước, khi bàn cờ được quan sát chính diện, phía dưới sẽ là quân Trắng (hoặc Đỏ), phía trên sẽ là quân Đen. Các đường dọc bên Trắng (Đỏ) được đánh số từ 1 đến 9 từ phải qua trái. Các đường dọc bên Đen được đánh số từ 9 tới 1 từ phải qua trái.

Ranh giới giữa hai bên là "sông" (hà). Con sông này có tên là "Sở hà Hán giới" - con sông định ra biên giới giữa nước Sở và nước Hán.



Sơ đồ bàn cờ tướng



Sắp xếp các quân cờ theo đúng vị trí

1.3. Quân cờ và nước đi

Mỗi ván cờ lúc bắt đầu phải có đủ 32 quân, chia đều cho mỗi bên gồm 16 quân Trắng (Đỏ) và 16 quân Đen, gồm 7 loại quân. Tuy tên quân cờ của mỗi bên có thể viết khác nhau (ký hiệu theo chữ Hán) nhưng giá trị và cách đi quân của chúng lại giống nhau hoàn toàn. Bảy loại quân có ký hiệu và số lượng cho mỗi bên như hình sau:

Quân	Ký hiệu	Số lượng
Tướng	帥 將	1
Sỹ	仕 士	2
Tượng	相 象	2
Xe	俥 車	2
Pháo	炮 砲	2
Mã	傌 馬	2
Tốt	兵 卒	5

Các quân cờ tướng

Tướng

Quân tướng

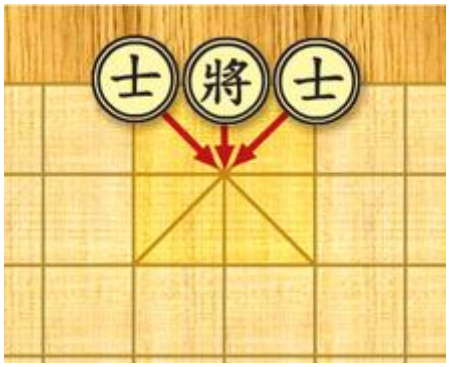
Tướng được chốt chặt trong cung và có tới 2 Sỹ và Tượng canh gác hai bên. Tướng chỉ được đi ngang hay đi dọc từng bước một trong phạm vi cung tướng. Tính theo khả năng chiến đấu thì Tướng là quân yếu nhất do chỉ đi nước một và bị giới hạn trong cung. Tuy nhiên trong nhiều tình huống, đặc biệt khi cờ tàn đòn "lộ mặt tướng" lại tỏ ra rất hiểm và mạnh. Lúc này Tướng mạnh ngang với Xe.

Sĩ

Quân sĩ

Trong cờ tướng, quân Sỹ có vai trò "hộ giá" cho Tướng (hoặc Soái). Chúng đứng ngay sát cạnh Tướng, chỉ đi từng bước một và đi theo đường chéo trong Cửu cung. Như vậy, chúng chỉ di chuyển và đứng tại 5 điểm và được coi là quân cờ yếu nhất vì bị hạn

chế nước đi. Sỹ có chức năng trong việc bảo vệ Tướng, mất Sỹ được cho là nguy hiểm khi đối phương còn đủ 2 Xe hoặc dùng Xe Mã Tốt tấn công.



Nước đi của quân Sĩ

Tượng (Tịnh, Bồ)

Hình 1: Quân Tượng

Quân Tượng đứng bên cạnh quân Sỹ và tương đương với Tượng trong cờ vua. Quân này đi theo đường chéo của hình vuông gồm 2 ô cờ. Chúng không được qua sông, chúng có nhiệm vụ ở lại bên này sông để bảo vệ vua. Chỉ có 7 điểm mà Tượng có thể di chuyển tới và đứng ở đó. Tượng sẽ không di chuyển được đến vị trí đã nêu nếu có 1 quân đặt tại vị trí giữa của hình vuông 2 ô. Khi đó ta gọi là Tượng bị cản và vị trí cản được gọi là "mắt Tượng".

Tượng được tính là mạnh hơn Sĩ một chút. Khả năng phòng thủ của Tượng cũng được tính nhỉnh hơn. Nói chung mất Tượng cờ dễ nguy hơn mất Sĩ.

Xe (Xa)

Quân Xe

Quân Xe đi và ăn theo một đường thẳng đứng hoặc ngang giống hệt quân Xe trong cờ tướng. Chúng bắt đầu nước đi từ phía góc của bàn cờ, chúng được coi là quân cờ mạnh nhất trong cờ tướng.

Pháo

Pháo

Quân Pháo đi giống quân Xe, theo chiều thẳng đứng hoặc ngang, nhưng ăn quân bằng cách nhảy qua 1 quân cờ khác. Có thể nói pháo là quân cờ lợi hại nhất trong cờ tướng.

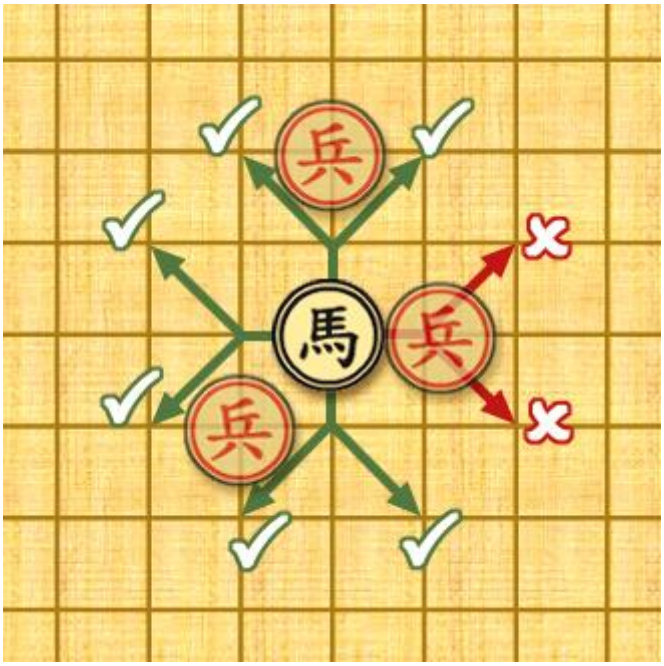
Cách ăn của quân Pháo

Mã

Quân mã

Với bàn cờ được cải tiến như hiện nay, đất rộng và có vô số đường để tung hoành, Mã sẽ phi nước đại trên khắp bàn cờ.

Quân Mã di chuyển theo hình chữ L hay nói cách khác quân Mã di chuyển theo đường chéo của hình chữ nhật 2x3 hoặc 3x2.



Cách đi của quân mã

Tốt (Binh, Chốt)

Quân Tốt

Quân Tốt ở đây tương tự như quân Tốt ở cờ vua, chúng đi thẳng theo chiều đứng và có thể ăn quân từng bước một. Khi Tốt qua được sông, chúng có thể đi và ăn theo chiều ngang.

Không giống như trong cờ vua, chúng không có luật phong Hậu, hay Xe,... khi đi đến hết bàn cờ, lúc này, chúng được gọi là Tốt lụt. Việc mất mát một vài Tốt ngay từ đầu cũng được xem như việc thí quân trong cờ.

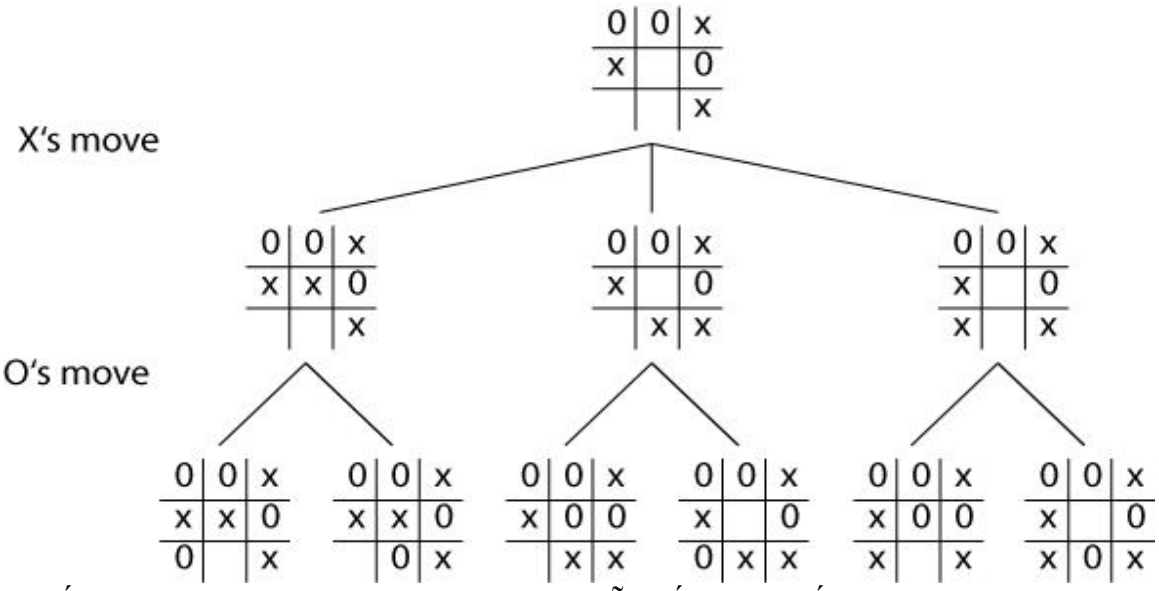
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Để giải quyết bài toán cờ tướng ta áp dụng chiếc lược Minimax - cắt tỉa AlphaBeta

2.1. Chiến lược Minimax

Minimax (còn gọi là minmax) là một phương pháp trong lý thuyết quyết định có mục đích là tối thiểu hóa (*mini*mize) tổn thất vốn được dự tính có thể là "tối đa" (*max*imize). Có thể hiểu ngược lại là, nó nhằm tối đa hóa lợi ích vốn được dự tính là *tối thiểu* (maximin). Nó bắt nguồn từ trò chơi có tổng bằng không. Nó cũng được mở rộng cho nhiều trò chơi phức tạp hơn và giúp đưa ra các quyết định chung khi có sự hiện diện của sự không chắc chắn.

Một phiên bản của giải thuật áp dụng cho các trò chơi như tic-tac-toe, khi mà mỗi người chơi có thể thắng, thua, hoặc hòa. Nếu người chơi A *có thể* thắng trong 1 nước đi, thì "nước đi tốt nhất" chính là nước đi để dẫn đến kết quả thắng đó. Nếu người B biết rằng có một nước đi mà dẫn đến tình huống người A có thể thắng ngay ở nước

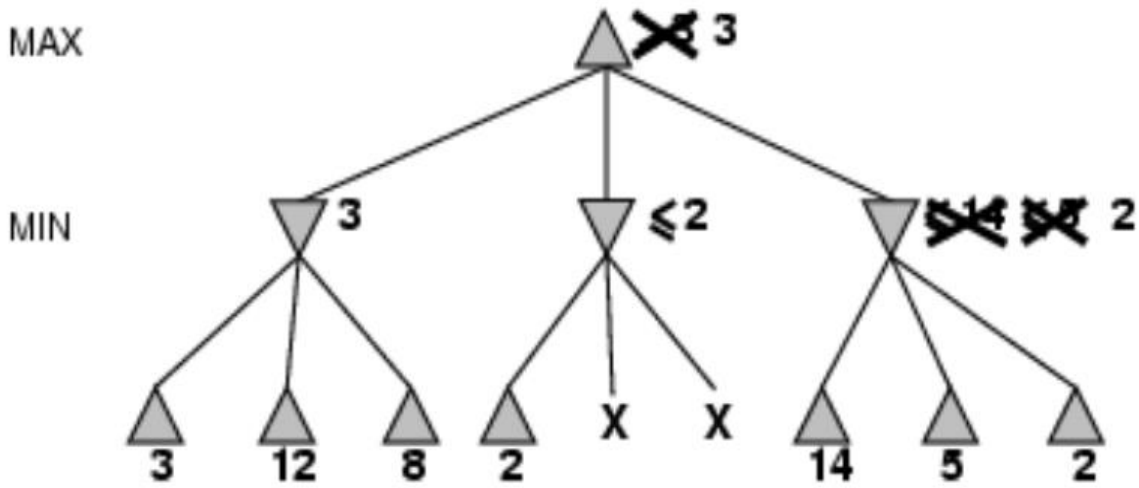


đi tiếp theo, trong khi nước đi khác thì sẽ dẫn đến tình huống mà người chơi A chỉ có thể, tốt nhất, là hòa thì nước đi tốt nhất của người B chính là nước đi sau.

Ta sẽ nắm rõ, thế nào là một nước đi "tốt nhất". Giải thuật Minimax giúp tìm ra nước đi tốt nhất, bằng cách đi ngược từ cuối trò chơi trở về đầu. Tại mỗi bước, nó sẽ ước định rằng người A đang cố gắng tối đa hóa cơ hội thắng của A khi đến phiên anh ta, còn ở nước đi kế tiếp thì người chơi B cố gắng để tối thiểu hóa cơ hội thắng của người A

(nghĩa là tối đa hóa cơ hội thắng của B).

2.2. Cắt tỉa AlphaBeta



Thuật toán cắt tỉa Alpha – Beta là cải tiến của thuật toán Min – Max với tư tưởng “Nếu đã thấy một việc làm là tệ thì không nên mất thời gian xem nó tệ đến mức nào ”.Nhằm mục đích tỉa bớt nhánh của cây trò chơi, làm giảm số lượng nút phải sinh và lượng giá, do đó có thể tăng độ sâu của cây tìm kiếm giúp tìm kiếm nước đi tốt nhất nhanh hơn.

Ví dụ về phương pháp cắt tỉa AlphaBeta

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ

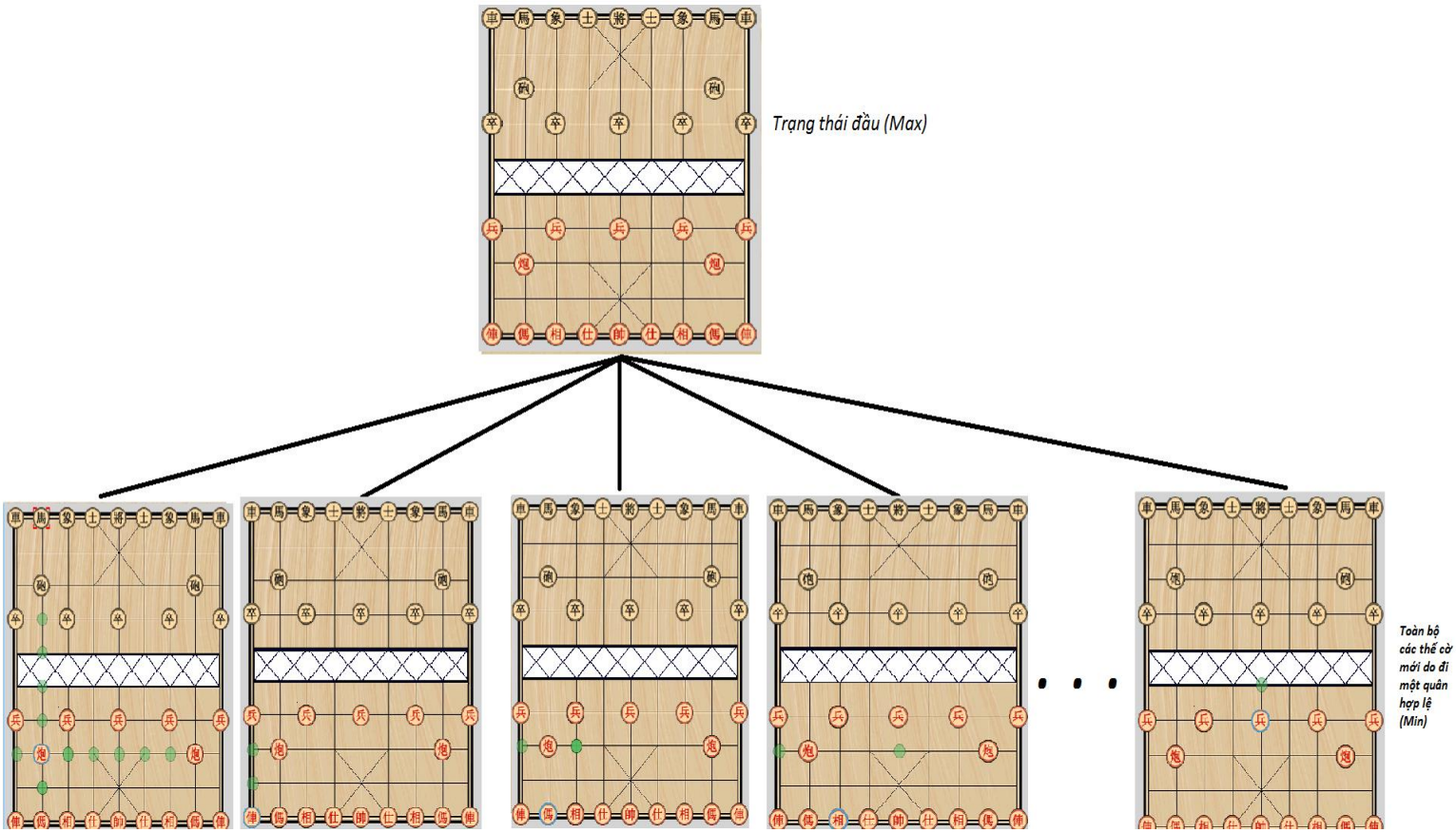
3.1. Phân tích bài toán cờ tướng

+ Quá trình chơi cờ tướng là quá trình bên Đen và Đỏ thay phiên nhau đưa ra quyết định với các nước đi hợp lệ. Để máy có thể đánh cờ với người, thì ta cần phải làm cho máy hiểu được cờ tướng.Vì vậy ta biểu diễn cờ tướng theo dạng cây tìm kiếm với mỗi node là một trạng thái đi. Để tìm được trạng thái tốt cho máy đi tốt nhất ta cần áp dụng chiếc lược Minimax – Cắt tỉa AlphaBeta để nhanh chóng tìm cho nó một nước đi tốt nhất từ cách xem xét lượng giá của từng node.

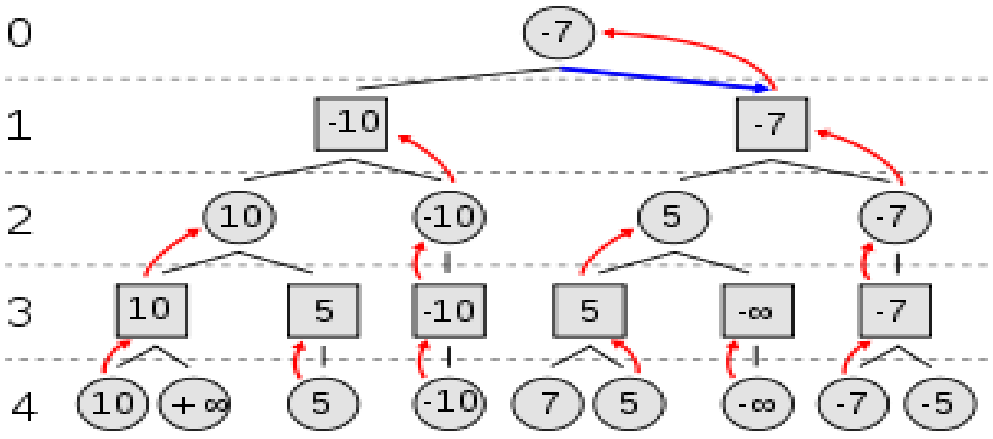
+ Để làm được ta cần phải đưa bàn cờ thực biểu diễn bằng một không gian trạng thái lên bộ nhớ thực máy tính để máy hiểu.Không gian trạng thái được biểu diễn dưới dạng Cây Trò Chơi với:

-Gốc của cây ứng với trạng thái ban đầu

-Gọi đỉnh ứng với trạng thái mà Đỏ(Đen) sẽ đưa ra nước đi là đỉnh Đỏ(Đen).



+Với mỗi nước đi (mỗi thay đổi) sẽ làm cho KGTT của bàn cờ thay đổi thành KGTT mới.Như vậy, đến nước đi của máy với chiến lược tìm kiếm nước đi Minimax sẽ được sử dụng để làm sao tìm ra nước đi tốt nhất cho máy qua lượng giá của các nước đi.



Ví dụ như hình:

Mức 0 2 4 là lượt đi của máy – mức MAX (tìm kết quả lượng giá càng cao càng tốt)

Mức 1 3 là lượt đi của người – mức MIN (tìm kết quả lượng giá càng nhỏ càng tốt)

Chiến lược Minimax(được thể hiện bằng giải thuật minimax) dựa trên 2 giả thiết sau:

Cả 2 đối thủ có cùng kiến thức như nhau về không gian trạng thái của trò chơi.

Cả 2 đối thủ có cùng mức cố gắng thắng như nhau.

Hai đối thủ trong trò chơi có tên MAX và MIN

MAX: biểu diễn cho mục đích đối thủ này là làm lớn tối đa lợi thế của mình.

MIN: biểu diễn cho mục đích của đối thủ này là làm nhỏ tối đa lợi thế của đối phương.

Trên cây tìm kiếm sẽ phân thành các lớp MAX và MIN

Tại một Node u bất kỳ:

Node lá có giá trị: 1 nếu MAX thắng và 0 nếu MIN thắng.

Thuộc lớp MAX-> gán giá trị lớn nhất của các node con

Thuộc lớp MIN-> gán giá trị nhỏ nhất của các node con.

Vì tại mỗi thời điểm trên bàn cờ có rất nhiều quân cờ và tại mỗi quân cờ lại có rất nhiều nước đi nên cây tìm kiếm sẽ rất lớn và thế ta sẽ cải tiến chiến lược Minimax

Minimax với đọ sâu giới hạn

Giải pháp:

Giới hạn không gian trạng thái theo một độ sâu nào đó và giới hạn các node con theo 1 qui tắc nào đó

Chiến lược thông thường của người chơi cờ: khả năng tính trước bao nhiêu nước cờ.

Ý tưởng:

Chỉ triển khai các nút con đến độ sâu giới hạn.

Đánh giá cho các nút này như là nút lá bằng 1 hàm lượng giá Heuristic:

u->f(u): “ đọ lợi thế” của trạng thái u

(u càng thuận lợi cho MAX thì f(u) là số dương càng lớn, thuận lợi cho MIN thì f(u) là số âm càng nhỏ)

Áp dụng chiến lược minimax cho việc đánh giá các nút cấp trên.

Kỹ thuật này gọi là nhìn trước K bước với K là độ sâu giới hạn.

Nhưng Minimax lại có nhược điểm phụ thuộc vào rất nhiều hàm đánh giá (lượng giá).Nếu hàm đánh giá cho ta sự đánh giá không chính xác về các trạng thái, nó có thể hướng dẫn ta đi tới trạng thái được xem là tốt, nhưng thực tế lại rất bất lợi cho ta.Thiết kế một một hàm đánh giá tốt là một việc khó, đòi hỏi ta phải quan tâm đén nhiều nhân tố: các quân còn lại của hai bên, sự bố trí của các quân đó,… ở đây có sự mâu thuẫn giữa độ chính xác của hàm đánh giá và thời gian tính của nó. Hàm đánh giá chính xác sẽ đòi hỏi rất nhiều thời gian tính toán, đôi khi tính toán rất lâu khiến người chơi không thể chờ được.Vì vậy ta chuyển sang kỹ thuật cải tiến là cắt tỉa AlphaBeta để tăng tốc độ suy nghĩ của máy.

Giải thuật cắt tỉa α - β

Ý tưởng:

Tìm kiếm theo kiểu depth – first.

Nút MAX có giá trị α (luôn tăng).

Nút MIN có giá trị β (luôn giảm).

Tìm kiếm có thể kết thúc dưới:

Nút MIN nào có $\beta \leq \alpha$ của một nút MAX bất kì

Nút MAX nào có $\alpha \geq \beta$ của một nút min bất kì

Giải thuật cắt tỉa α – β thế hiện mối quan hệ giữa các nút ở lớp n và n+2, mà tại đó toàn bộ cây có gốc tại lớp n+1 có thể cắt bỏ.

3.2. Cấu trúc dữ liệu và cách biểu diễn các trạng thái của bài toán

Cấu trúc dữ liệu:

Gồm có 4 Class: Class ChessDefinition,Class ChessStd , ChessThink, Class resource

-Class ChessDefinition: định nghĩa cờ tướng bao gồm:

+Khai báo người đánh (Man) và máy (Com) lần lượt giá trị là 0 và 1 với các phe tương tự là Đỏ và Đen.

+Khai báo các quân cờ thuộc phe đỏ tướng(King),sĩ(Advisor),Tượng(Elephant),… lần lượt là 0,1,2…Tương tự phe đen.

+Khai báo các hàm loadpieceicon để chèn hình ảnh quân cờ

+Khai báo Struct Movehistory để quản lý nước đi của quân cần đi gồm điểm xuất phát (from) và điểm đến(dest).

-Class ChessStd:

+Khai báo hàm bool CanGo để kiểm tra các nước đi bị cản của các quân cờ (Mã,Tượng) và xử lý cách ăn của quân Pháo.

+Khai báo hàm bool IsNormal để kiểm tra 1 quân cờ được phép đi vào ô nào đó không? Ví dụ:Tướng và sĩ không thể đi ra ngoài cung,Tượng không được qua sông …

+Hàm void FixManMap

-Class ChessThink:

Khai báo hàm bool Think với các điểm số cơ bản (lượng giá của mỗi quân cờ) để đánh giá thế cờ và các bước đi hợp lý.

-Class resource: Dùng để tạo giao diện icon và chức năng undo của chương trình.

Các trạng thái:

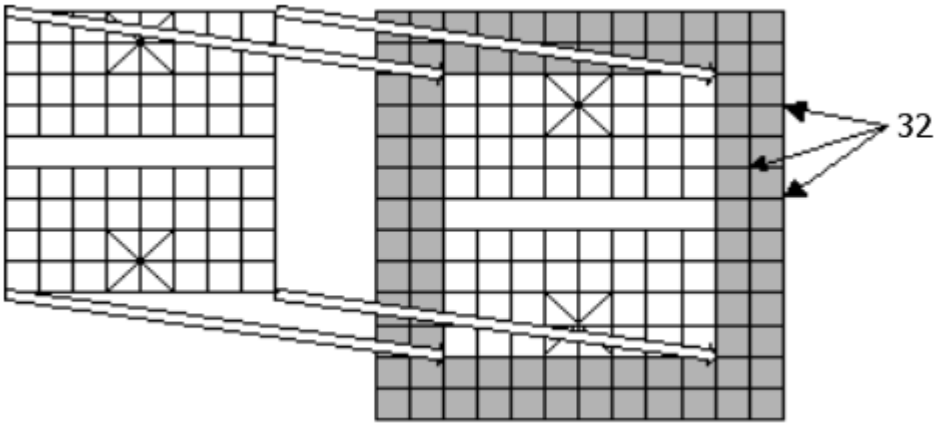
-Trạng thái ban đầu là trạng thái của các quân cờ ở trên bàn cờ khi chưa đi nước nào.

-Trạng thái kết thúc là trạng thái khi tướng của một bên bị ăn.

3.3. Các vấn đề thuật giải

3.3.1 Biểu diễn bàn cờ

Bàn cờ trong cờ tướng là một hình chữ nhật bao gồm 9 đường dọc và 10 đường ngang. Các quân cờ chia làm 2 bên đứng tại các giao điểm của các đường. Cách đơn giản nhất để biểu diễn bàn cờ trong máy tính là ta dùng một mảng 2 chiều, kích thước 9 x 10. Trong chương trình này, nhóm nghiên cứu sử dụng kĩ thuật có tên gọi là "Hộp thư". Mục đích chính của kĩ thuật này là nhằm giảm bớt các phép kiểm tra vượt giới hạn bàn cờ và làm đơn giản chương trình. Mấu chốt là thay cho bàn cờ có kích thước 9 x 10 = 90, ta dùng một bàn cờ mới có kích thước 11 x 12 = 121. Các ô ứng với đường bao mở rộng đều có giá trị 32, tức là các giá trị vượt biên. Sở dĩ có đến 2 đường biên chứ không phải 1 do quân Mã và Tượng có thể nhảy đến 2 ô.



Hình 2: Kĩ thuật Mailbox

Mảng trên hoạt động tốt nhưng có cái bất tiện là ta phải dùng tới hai chỉ số để truy cập vào mảng (ví dụ vị trí quân Pháo góc trên bên trái (cột 2, dòng 3) là piece[3, 2]). Một cải tiến nhỏ là ta dùng mảng một chiều như sau: int piece[1..90]; Truy nhập đến vị trí quân Pháo góc trên bên trái lúc này là piece[20]. 404 Not Found Các ô của mảng sẽ chứa những giá trị khác nhau cho biết đó là quân cờ gì. Mỗi quân cờ sẽ được gán một mã số khác nhau như bảng dưới đây. Các chỗ trống (không có quân cờ) sẽ được điền kí hiệu trống (EMPTY):

Quân cờ ------Kí hiệu ------- Giá trị

Tốt (Chốt)----PAWN----------0

Sĩ -----------BISHOP---------1

Tượng ------ELEPHANT -------2

Mã ----------KNIGHT --------3

Pháo --------CANNON --------4

Xe -----------ROOK----------5

Tướng --------KING----------6

Trống--------EMPTY---------7

Ngoài mục đích gán mỗi quân cờ một mã số để phân biệt, mã này còn giúp ta ước lượng sơ bộ tầm quan trọng của quân cờ đó. Như vậy, lúc khởi đầu, các ô trong mảng sẽ được gán các giá trị quân cờ nhờ khai báo const (trong Java) như dưới, trong đó BOARD_SIZE (kích thước bàn cờ = 90) là một hằng số đã được định nghĩa trước đó: char piece=

```
{  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
{  0,  5,  3,  2,  1,  6,  1,  2,  3,  5,  0},
{  0,  7,  7,  7,  7,  7,  7,  7,  7,  7,  0},
{  0,  7,  4,  7,  7,  7,  7,  7,  4,  7,  0},
{  0,  0,  7,  0,  7,  0,  7,  0,  7,  0,  0},
{  0,  7,  7,  7,  7,  7,  7,  7,  7,  7,  0},
{  0,  7,  7,  7,  7,  7,  7,  7,  7,  7,  0},
{  0,  0,  7,  0,  7,  0,  7,  0,  7,  0,  0},
{  0,  7,  4,  7,  7,  7,  7,  7,  4,  7,  0},
{  0,  7,  7,  7,  7,  7,  7,  7,  7,  7,  0},
{  0,  5,  3,  2,  1,  6,  1,  2,  3,  5,  0};
{  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
```

Đến đây, bàn cờ còn chưa có thông tin để phân biệt một quân cờ là của bên nào. Ta có thể cải tiến thay cho kiểu byte của mảng piece là một bản ghi nhằm lưu thêm

các thông tin này. Chúng tôi xin giới thiệu một phương pháp đơn giản là dùng thêm một mảng nữa - mảng color, để lưu các thông tin về bên. Hai bên được gán kí hiệu và mã như bảng dưới. Những chỗ trống sẽ dùng cùng kí hiệu trống EMPTY. Bên----------Đen----------Trắng---------Trống--- Kí hiệu---------DARK---------LIGHT----------EMPTY--- Giá trị----------0----------1----------7----- Ta lại có thông tin về bên được khai báo khởi đầu tương tự: char color=

```
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{ 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0},
{ 0, 7, 0, 7, 7, 7, 7, 7, 0, 7, 0},
{ 0, 0, 7, 0, 7, 0, 7, 0, 7, 0, 0},
{ 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0},
{ 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0},
{ 0, 1, 7, 1, 7, 1, 7, 1, 7, 1, 0},
{ 0, 7, 1, 7, 7, 7, 7, 7, 1, 7, 0},
{ 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0},
{ 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0};
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
```

Có thể tự phát triển giao diện đồ hoạ và cài chuột theo ý mình.Quân cờ được biểu diễn bằng một chữ cái viết hoa đứng đầu tên của nó. Quân hai bên sẽ có mầu khác nhau. Do quân Tướng và Tượng có cùng chữ cái đầu T nên để tránh nhầm lẫn ta dùng chữ V (Voi) biểu diễn quân Tượng. Tuy quân Tốt và Tướng cũng có cùng chữ cái đầu nhưng ta không sợ nhầm do Tốt không thể nhập cung bên mình được. Nó chỉ có thể "tiếp chuyện" Tướng đối phương, nhưng lúc này hai bên lại phân biệt được nhờ mầu (các phương pháp khác là dùng chữ cái đầu tên tiếng Anh: P-Tốt, E-Tượng, N-Mã, B-Sĩ, R-Xe, C-Pháo, K-Tướng; hoặc theo qui định của Liên đoàn cờ Việt Nam: Tg-Tướng, S-Sĩ, T-Tượng, X-Xe, P-Pháo, M-Mã, C-Chốt).

3.3.2 Tạo nước cờ hợp lệ

Một trong những việc quan trọng nhất để máy tính có thể chơi được cờ là phải chỉ cho nó biết mọi nước đi có thể đi được từ 1 thế cờ. Máy tính sẽ tính toán để chọn nước đi có lợi nhất cho nó. Các yêu cầu chính với thủ tục sinh nước đi là:

- Chính xác (rất quan trọng): Do số lượng nước đi sinh ra lớn, luật đi quân nhiều và phức tạp nên việc kiểm tra tính đúng đắn tương đối khó.
- Đầy đủ (quan trọng): Sinh ra được mọi nước đi có thể có từ một thế cờ.
- Nhanh: Do chức năng này phải sinh ra được hàng triệu nước đi mỗi khi máy đến lượt nên việc giảm thời gian sinh nước đi có ý nghĩa rất lớn.

Sinh nước đi là một thuật toán vét cạn. Máy sẽ tìm mọi nước đi hợp lệ có thể có. Máy phải sinh được từ những nước đi rất hay cho đến những nước đi rất ngớ ngẩn (như đẩy Tướng vào vị trí khống chế của đối phương). Ta hình dung ngay thủ tục sinh nước đi Gen sẽ có đầy những vòng lặp for, các câu lệnh kiểm tra if và rẽ nhánh case, trong đó các phép tính kiểm tra giới hạn chiếm một phần đáng kể. Thủ tục này luôn sinh nước

cho bên đang tới lượt chơi căn cứ vào nội dung của biến side. Đây là một trong những thủ tục phức tạp và dễ sai nhất.
Một nước đi có hai giá trị cần quan tâm. Đó là điểm xuất phát (from) và điểm đến (dest). Ta sẽ khai báo một cấu trúc move như sau để dùng những nơi cần đến dữ liệu nước đi.

```
struct MOVEHISTORY
{
  int count;
  int man[MAXMOVE];
  POINT from[MAXMOVE];
  POINT to[MAXMOVE];
  int betaken[MAXMOVE];
};
```

Ví dụ có một quân Xe nằm ở ô số 84 (trong mảng piece). Bây giờ ta sẽ sinh thử một nước đi sang trái một ô cho nó. Nước đi sang trái một ô được chuyển thành ô số 83 là một vị trí cùng hàng với ô xuất phát nên được chấp nhận. Một nước đi khác cần phải xem xét là sang trái ba ô - ô 81. Ô 81 tuy có trong bàn cờ nhưng khác hàng nên không được chấp nhận.

Như vậy, muốn biết ô của nước đi sang trái có được phép không ta phải kiểm tra xem nó có cùng hàng với ô xuất phát không. Việc kiểm tra thực hiện bằng cách chia cho 9 (kích thước của một dòng) và lấy phần nguyên (trước đó lại phải chuyển thứ tự ô về gốc là 0 bằng cách trừ đi 1). Ta thấy ((83-1) div 9) = ((84-1) div 9) nên ô 83 được chấp nhận; trong khi đó do ((81-1) div 9) <> ((84-1) div 9) nên kết luận là nước đi đến ô 81 không hợp lệ Các nước đi vừa sinh ra sẽ được đưa vào danh sách nước đi nhờ gọi thủ tục Gen_push. Thủ tục này có hai tham số là vị trí xuất phát của quân cờ sẽ đi và nơi đến dest của quân cờ đó.

Dưới đây là đoạn chương trình dùng để sinh những nước đi sang trái của một quân Xe, trong đó x là vị trí của quân Xe này.

```
 i = x - 1; //Nước sang trái đầu tiên while ((i-1) / 9) = ((x-1) / 9)
{
        if (ô thứ i là trống) or (ô thứ i có quân đối phương)
        gen_push(vị trí của Xe, vị trí ô đang xét);
        if ô thứ i không trống break; // Kết thúc quá trình sinh nước đi sang trái i
= i - 1; // Xét tiếp vị trí bên trái
        }
```

Việc sinh những nước đi theo chiều khác cũng tương tự (ví dụ để sinh nước đi theo chiều đứng ta chỉ cần cộng hoặc trừ một bội số của 9) nhưng điều kiện kiểm tra sẽ khác nhau.

Như vậy, chương trình sinh nước đi Gen có dạng như sau: for (Xét lần lượt từng quân cờ bên hiện tại) case quâncờ of Xe:

while (Xét lần lượt tất cả các ô từ bên phải Xe cho đến lề trái bàn cờ) { if (ô đang xét là ô trống) or (ô đang xét chứa quân đối phương)

gen_push(vị trí của Xe, vị trí ô đang xét)

if (ô đang xét không trống) break; }

while (Xét lần lượt tất cả các ô từ bên trái Xe cho đến lề phải bàn cờ) { ...}

while (Xét lần lượt tất cả các ô từ bên trên Xe cho đến lề trên bàn cờ) { ... }

while Xét lần lượt tất cả các ô từ bên dưới Xe cho đến lề dưới bàn cờ { ...} Break;

Pháo: ... Phương pháp này có nhược điểm là chương trình phải viết phức tạp, cồng kềnh, khó tìm lỗi nhưng khá nhanh. Trong chương trình mẫu VSCCP, chúng tôi giới thiệu một kĩ thuật khác có tên gọi là "hộp thư" (mail box - do các bảng của nó có dạng các hộp phân thư). Mục đích chính của kĩ thuật này là nhằm giảm bớt các phép kiểm tra vượt giới hạn bàn cờ và làm đơn giản chương trình. Mấu chốt là thay cho bàn cờ có kích thước bình thường 9x10 = 90, ta dùng một bàn cờ mở rộng, mỗi chiều có thêm 2 đường nữa (bàn cờ mới có kích thước 13x14 = 182). Các ô ứng với các đường bao mở rộng đều có giá trị -1, tức là các giá trị báo vượt biên. Các nước đi trên bàn cờ 9x10 được chuyển tương ứng sang bàn cờ này. Nếu một nước đi được sinh ra lại rơi vào một trong hai đường bao thì có nghĩa nó đã rơi ra ngoài bàn cờ rồi, phải bỏ đi và ngừng sinh nước về phía đó. Sở dĩ có đến hai đường biên (chứ không phải một) do quân Mã và Tượng có thể nhảy xa đến hai ô Việc chuyển đổi toạ độ thực hiện nhờ hai mảng. Mảng mailbox90 dùng để chuyển từ toạ độ bàn cờ thường sang toạ độ bàn cờ mới. Mảng ứng với bàn cờ mới - mailbox182 dùng để kiểm tra các nước đi vượt quá đường biên và dữ liệu để chuyển trở lại toạ độ bình thường. Ví dụ, nếu vị trí quân Xe nằm ở ô số 84 (trong mảng piece) như hình 2.5, thì khi đổi sang bàn cờ mở rộng sẽ thành vị trí mailbox90[84] = 148.

Có nghĩa là nó ứng với ô thứ 148 của mảng mailbox182. Bây giờ ta sẽ sinh thử một nước đi sang trái một ô cho quân Xe này. Việc sinh và kiểm tra sẽ được thực hiện

trong bàn cờ mở rộng, nước đi mới là ô số 148 - 1 = 147. Do mailbox182[147] = 83 ¹ - 1 nên nước đi này được chấp nhận. Số 83 cho biết kết quả sang trái một ô là vị trí 83 trong bàn cờ bình thường. Tuy nhiên, nước đi sang trái ba ô, có số 148 - 3 = 145 và mailbox182[145] = -1 cho biết đó là một nước đi không hợp lệ. Chương trình cũng cần kiểm tra số nước đi tối đa theo một hướng của quân cờ đang xét. Chỉ có quân Pháo và Xe có thể đi đến 9 nước đi, còn các quân khác có nhiều nhất là 1. Việc đi một quân sang vị trí mới thực chất là ta đã thay đổi toạ độ của nó bằng cách cộng với một hằng số (dương hoặc âm). Ở trên ta đã thấy để quân Xe sang trái một nước ta đã phải cộng vị trí hiện tại với -1.

Để sinh một ô mới theo hàng dọc, ta phải cộng với +13 hoặc -13 (chú ý, việc sinh nước và kiểm tra hợp lệ đều dựa vào mảng mailbox182 nên giá trị 13 là kích thước một dòng của mảng này).

3.3.3 Kiểm tra vị trí được phép đến:

Việc chuyển tọa độ từ bàn cờ thường sang bàn cờ mở rộng chỉ giúp ta loại bỏ được các nước đi vượt khỏi bàn cờ. Ta còn phải kiểm tra một số giới hạn khác. Ví dụ như Tướng và Sĩ không thể đi ra ngoài cung, Tượng không được phép đi qua sông, Tốt chỉ được đi tiến, có thể đi ngang trên đất đối phương, 2 quân Tướng không được đối mặt trực tiếp với nhau. Để kiểm tra một quân cờ có được phép đi vào ô nào đó không, ta sử dụng hàm ***IsNormal****(const int &piece, const POINT &point)*. Ngoài ra còn phải kiểm tra nước bị cản (đối với quân Tượng và Mã) và xử lý cách ăn quân của quân pháo như một trường hợp đặc biệt bằng hàm **CanGo**. Tổng thể ta có 2 hàm sau:

BOOL IsNormal(const int &piece, const POINT &poin)

```
Begin
    if(đi ra ngoài biên) return FALSE;
    case piece of:
        {xử lý các quân cờ Tướng, Sĩ, Tượng, Tốt)
end
return TRUE
End.
```

BOOL CanGo(int map[11][12], int piece, const POINT &from, const POINT &to)

Begin

if(!IsNormal(piece, to)) {xử lý các nước đi của Xe, Pháo, Mã)

case piece of:

{xử lý kiểm tra nước đi hợp lệ của tất cả các quân cờ}

end.

return TRUE

End.

3.3.4 Đánh giá thế cờ

Đánh giá một thế cờ là một trong những nhiệm vụ quyết định chương trình chơi cờ có là "cao thủ" hay không. Căn cứ vào một thế cờ máy sẽ gán cho nó một điểm số (lượng giá tĩnh) để đánh giá độ tốt - xấu. Nhờ điểm này máy mới có thể so sánh các thế cờ với nhau và biết chọn nước đi tốt nhất. Đây là một nhiệm vụ rất khó khăn và phức tạp do không tồn tại một thuật toán tổng quát và thống nhất nào để tính điểm cả. Điểm của một thế cờ dựa trên rất nhiều yếu tố mà khó có thể số hoá hết được như phụ thuộc vào số lượng và giá trị các quân cờ hiện tại, phụ thuộc vào tính hãm, tính biến, thế công, thế thủ của từng quân cờ cũng như cả cục diện trận đấu.

Trong chương trình của mình, nhóm nghiên cứu quyết định đưa ra đánh giá điểm số cho các quân cờ như sau:

Quân cờ	Điểm số cơ bản
Tướng	300
Sĩ	400
Tượng	300
Xe	1000
Pháo	600
Mã	600
Tốt	300

Bên cạnh đó cũng cần có cách tính lại điểm của quân Tốt khi đi sang sông bằng ma trận 3 chiều như sau:

```
const int Bonus[2][12][11]=
{
    {
        { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
```

```
                { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                { 0, 1, 2, 3, 4, 4, 4, 3, 2, 1, 0},
                { 0, 1, 2, 3, 4, 4, 4, 3, 2, 1, 0},
                { 0, 1, 2, 3, 3, 3, 3, 3, 2, 1, 0},
                { 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0},
                { 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0},
                { 0, 0, 0, 0, 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, 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, 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, 0, 0, 0, 0, 0, 0},
                { 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0},
                { 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0},
                { 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0},
                { 0, 1, 2, 3, 3, 3, 3, 3, 2, 1, 0},
                { 0, 1, 2, 3, 4, 4, 4, 3, 2, 1, 0},
                { 0, 1, 2, 3, 4, 4, 4, 3, 2, 1, 0},
                { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
        }
};
```

Sở dĩ là mảng 3 chiều vì ta cần tính cho quân tốt của cả 2 bên.

Trong chương này chỉ cài đặt phương pháp đơn giản nhưng cơ bản nhất: lượng giá dựa trên cơ sở giá trị của từng quân cờ. Cách tính này sẽ lấy tổng giá trị các quân cờ hiện có của bên mình trừ đi tổng giá trị các quân cờ hiện có của đối phương. Do đó, một thế cờ này hơn thế cờ kia ở chỗ nó còn nhiều quân bên mình hơn, nhiều quân giá trị cao hơn cũng như có bắt được nhiều quân và quân giá trị cao của đối phương hơn không.

Ưu điểm của cách tính này:

- Đơn giản, dễ thực hiện và tính toán
- Là cách tính nhanh nhất. Do tính nhanh nên có thể tăng độ sâu tìm kiếm. Việc tăng thêm độ sâu lại giúp máy có cái nhìn xa hơn, "cao cờ" hơn và nhiều khi khắc phục nhược điểm của cách tính đơn giản

Nhược điểm:

- Cách làm này đã bỏ qua mất những nghệ thuật, chiến lược chơi cờ

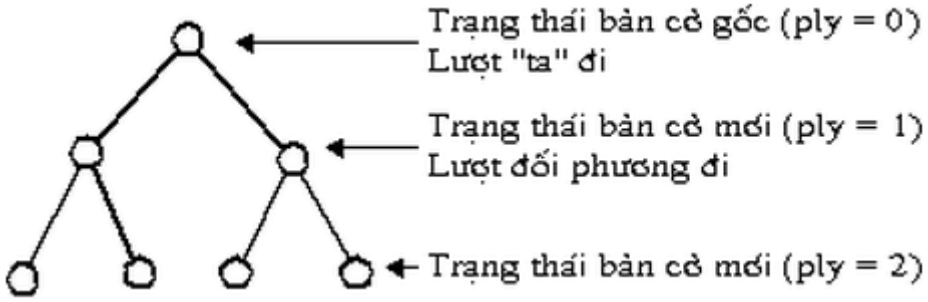
- Các quân cờ được triển khai không theo một chiến lược chung nào hết
- Chỉ chú trọng tới việc ăn quân đối phương nhanh nhất mà không xây dựng một thế trận có lợi cho mình sau này

Để cải thiện được nhược điểm trên, cần phải đánh giá thêm khả năng hỗ trợ nhau của các quân cờ. Các quân cờ nếu được hỗ trợ nhau thì sức mạnh sẽ được tăng thêm nhiều. . Để đánh giá việc hỗ trợ nhau; ta tìm ma trận thể hiện sự kiểm soát của phe mình (nếu vị trí nào là true thì vị trí đó có ít nhất một quân cờ cùng phe kiểm soát). Nếu một quân nằm trong vùng kiểm soát thì tức là nó ít nhất được hỗ trợ bởi một quân bên mình.

3.3.5 Thuật toán tìm nước đi tốt nhất

Cây trò chơi

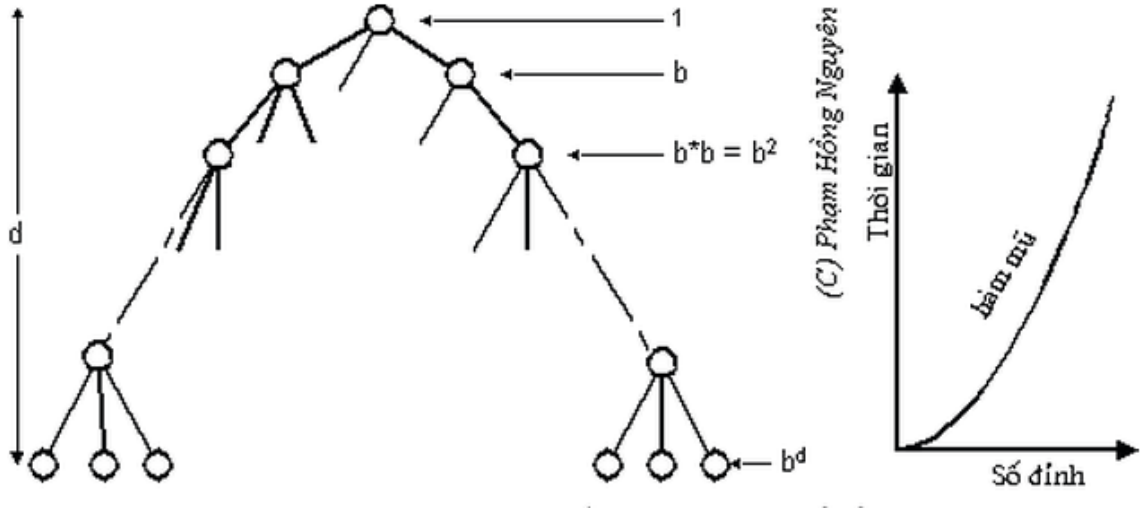
Các trạng thái bàn cờ khác nhau (hay còn gọi là một thế cờ, tình huống cờ) trong quá trình chơi có thể biểu diễn thành một cây tìm kiếm và ta sẽ tiến hành tìm kiếm trên cây để tìm được nước đi tốt nhất. Cây trò chơi có các nút của cây là các tình huống khác nhau của bàn cờ, các nhánh nối giữa các nút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



Hình 3: Minh họa độ sâu của cây trò chơi

Vét cạn

Dùng một thuật toán vét cạn để tìm kiếm trên cây trò chơi dường như là một ý tưởng đơn giản. Ta chỉ cần chọn nhánh cây sẽ dẫn tới nước thắng để đi quân là đảm bảo thắng lợi. Rất tiếc rằng, cách làm này lại không thể thực hiện nổi do bùng nổ tổ hợp. Ví dụ, nếu từ một thế cờ, trung bình có khả năng đi được 16 nước đi khác nhau (ta gọi đó là hệ số nhánh con tại mỗi nút là b = 16). Như vậy, sau một tầng ta sẽ có 16 nút con, mỗi nút này lại có thể có 16 con nữa. Tổng số nút con ở độ sâu thứ hai là 16x16 = b^2. Cứ như vậy ở độ sâu d sẽ có b^d nút.



Hình 4: Thuật toán vét cạn

Nếu giả sử độ sâu của cây là 100 (hệ số nhánh 16 và độ sâu 100 đều là những con số còn nhỏ hơn con số thường gặp trong trò chơi cờ), thì số nhánh phải duyệt lên đến 16^{100} hay xấp xỉ 10^{120} - một con số lớn khủng khiếp.

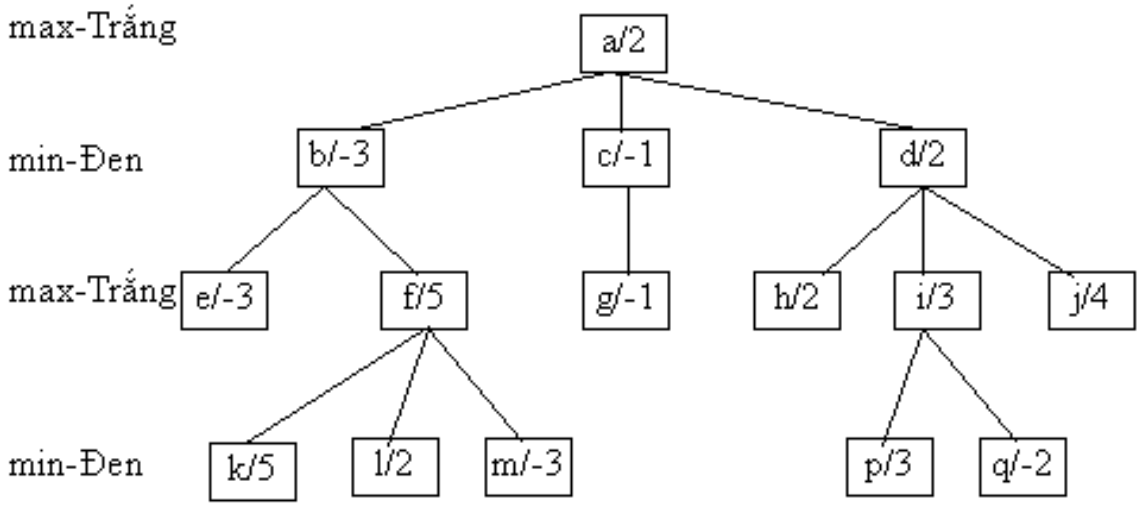
Vì số các khả năng tăng quá nhanh, chỉ có một số ít những vấn đề đơn giản là thích hợp với kiểu tìm kiếm vét hết mọi khả năng này (kiểu tìm kiếm vét cạn đòi hỏi phải kiểm tra tất cả các đỉnh). Do đó, các phương pháp tìm kiếm khác đã ra đời và phát triển. Ngược lại, nếu có một phương pháp luôn luôn chính xác nhằm đánh giá một thế cờ này là tốt hay kém so với thế kia, thì trò chơi trở thành đơn giản bằng cách chọn nước đi dẫn tới thế cờ tốt nhất. Do đó sẽ không cần phải tìm kiếm gì nữa. Rất tiếc, các thủ tục như vậy không hề có. Ta cần có chiến lược tìm kiếm trong trò chơi.

Thuật toán Minimax

Minimax (còn gọi là minmax) là một phương pháp trong lý thuyết quyết định có mục đích là tối thiểu hóa (minimize) tổn thất vốn được dự tính có thể là "tối đa" (maximize). Có thể hiểu ngược lại là, nó nhằm tối đa hóa lợi ích vốn được dự tính là tối thiểu (maximin). Nó bắt nguồn từ trò chơi có tổng bằng không. Nó cũng được mở rộng cho nhiều trò chơi phức tạp hơn và giúp đưa ra các quyết định chung khi có sự hiện diện của sự không chắc chắn.

Ta sẽ nắm rõ, thế nào là một nước đi "tốt nhất". Giải thuật Minimax giúp tìm ra nước đi tốt nhất, bằng cách đi ngược từ cuối trò chơi trở về đầu. Tại mỗi bước, nó sẽ ước định rằng người A đang cố gắng tối đa hóa cơ hội thắng của A khi đến phiên anh ta, còn ở nước đi kế tiếp thì người chơi B cố gắng để tối thiểu hóa cơ hội thắng của người A (nghĩa là tối đa hóa cơ hội thắng của B).

Ví dụ minh họa trên một cây trò chơi với Minimax



Hình 5: Thuật toán Minimax

Đánh giá thuật toán Minimax

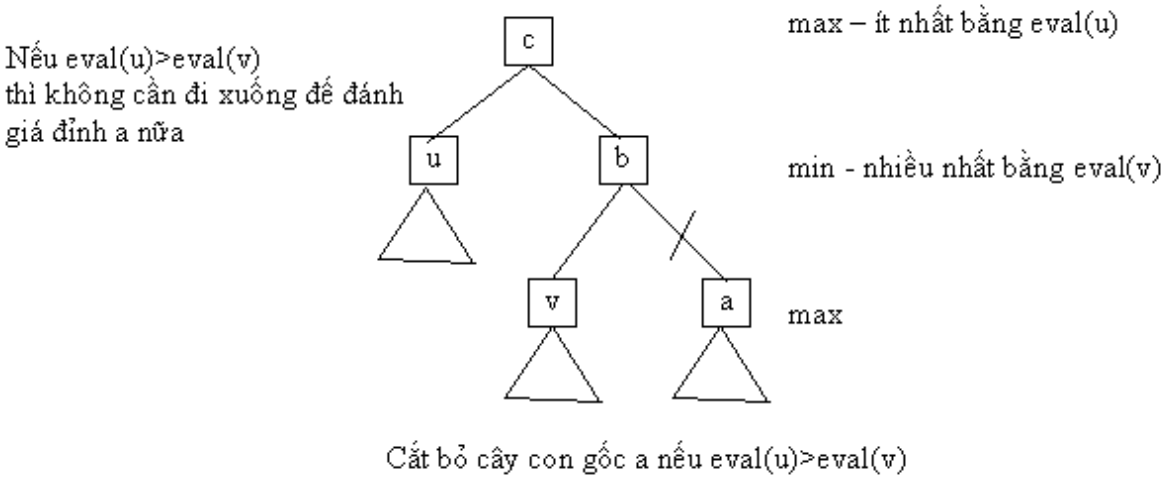
Nếu hệ số nhánh trung bình của cây là b và ta thực hiện tìm kiếm đến độ sâu d thì số nút phải lượng giá ở đáy cây như ta đã biết là bd. Đây chính là số đo độ phức tạp của thuật toán. Nếu b = 40, d = 4 (các con số thường gặp trong trò chơi cờ) thì số nút phải lượng giá là 40^4 = 2560000 (trên 2 triệu rưỡi nút). Còn với b = 40, d = 5 thì số nút phải lượng giá sẽ tăng 40 lần nữa thành 40^5 = 102400000 (trên 102 triệu nút).

Lưu ý: toàn bộ ý tưởng của thuật toán này là dựa trên việc chuyển đổi mỗi thế cờ thành một con số để đánh giá. Rất tiếc là các con số này thường không tốt và không đủ để đánh giá hết mọi điều. Mặt khác, thuật toán này có thể rất tốn kém (chạy chậm) do việc sinh các nước đi và lượng giá rất tốn thời gian tính toán, do vậy độ sâu của cây trò chơi cũng bị hạn chế nhiều. Ta cần có thêm những cải tiến để cải thiện tình hình.

Thủ tục AlphaBeta

Thủ tục AlphaBeta là một cải tiến thuật toán Minimax nhằm tỉa bớt nhánh của cây trò chơi, làm giảm số lượng nút phải sinh và lượng giá, do đó có thể tăng độ sâu của cây tìm kiếm.

Nguyên tắc Alpha-Beta: “Nếu biết điều đó thật sự tồi thì đừng mất thời gian tìm hiểu nó sẽ tồi tệ đến đâu”



Hình 6: Cắt tỉa AlphaBeta

Tại nút u là nút min. Thì b cũng là nút min.
Nếu eval(u) > eval(v) thì giả sử eval(a) > eval(v) đi chăng nữa, thì min(v, a) = eval(v)
Mặt khác eval(u) > eval(v) vậy thì max(u, b) = eval(u)
Trong trường hợp eval(a) <= eval(v) thì ta thấy eval(u) > eval(v) => eval(u) > eval(a) vậy thì max(u, b) = eval(u).
Vậy kết luận vị tìm giá trị cho nút a là vô nghĩa nên ta có thể cắt bỏ nó đi.

Trong điều kiện lí tưởng, thuật toán AlphaBeta chỉ phải xét số nút theo công thức:

$= 2b^{\frac{d}{2}} - 1$ với d chẵn

$= b^{\frac{d+1}{2}} + b^{\frac{d}{2}} - 1$ với d lẻ

Với b = 40 và d = 4 ta có số nút phải xét là $2x40^2 - 1 = 3199$. Như vậy trong điều kiện lí tưởng thì số nút phải xét nhờ AlphaBeta (chỉ khoảng 3 nghìn nút) ít hơn thuật toán Minimax (hơn 2,5 triệu nút) là 2560000 / 3199 khoảng 800 lần. Còn với b = 40 và d = 5 ta có số nút phải xét là $40^3 + 40^{(5/2)} - 1 = 64000+10119-1 = 74118$. Số nút phải xét nhờ AlphaBeta ít hơn thuật toán Minimax (hơn 102 triệu nút) là 102400000/74118 = 1382 lần.

Dưới đây là bảng so sánh số nút phải xét giữa hai thuật toán Minimax và AlphaBeta.

Độ sâu	Minimax		AlphaBeta		Tỉ lệ số nút
	Số nút	Số lần tăng	Số nút	Số lần tăng	Minimax / AlphaBeta
1	40		40	1	
2	1600	40	79	1.9	20
3	64000	40	1852	23.2	34
4	2560000	40	3199	1.7	800
5	102400000	40	74118	23.2	1381
6	4096000000	40	127999	1.7	32000
7	163840000000	40	2964770	23.2	55262
8	6553600000000	40	5120000	1.7	1280000

Hình 7: So sánh Minimax và AlphaBeta

3.4 Các hàm tính toán chiến lược MINIMAX- ALPHABETA

1.Tính giá trị cơ bản các quân cờ trên bàn cờ

```
const int contactpercent1 = 20; // emphasis on defend
const int contactpercent2 = 25; // emphasis on attack

const int BasicValue[7] = {
                        base[0] - base[0] * range[0] / 100,
                        base[1] - base[1] * range[1] / 100,
                        base[2] - base[2] * range[2] / 100,
                        base[3] - base[3] * range[3] / 100,
                        base[4] - base[4] * range[4] / 100,
                        base[5] - base[5] * range[5] / 100,
                        base[6] - base[6] * range[6] / 100
                    };
// value for Solider in different places
const int BasicValue_Soldier[5] =
                    {
                        0,
                        1 * 2 * base[6] * range[6] / 100 / 4,
                        2 * 2 * base[6] * range[6] / 100 / 4,
                        3 * 2 * base[6] * range[6] / 100 / 4,
                        4 * 2 * base[6] * range[6] / 100 / 4,
                    };
```

2.Tính giá trị của các nước đi

int Value(int map[11][12], POINT pieceCoordinate[32], int &side)

- Tính giá trị của các quân cờ
- Xét xem quân cờ đó có kết nối với quân cờ khác không (hỗ trợ tấn công hay phòng thủ)
- Tính maxvalue của thế cờ này Maxvalue = tổng giá trị quân cờ + giá trị kết nối của các quân cờ
- Đánh giá thế cờ của đối thủ trong nước đi này
- Hàm này trả về giá trị của nước đi

```
for (i = 0; i < 32; i++)
{
    k = PieceToType7[i];
    PieceBaseValue[i] = BasicValue[k] + PieceBaseValue[i];// * BV2[k];
    // if is soldier
    if (k == 6)
    {
        PieceBaseValue[i] += BasicValue_Soldier[BonusForSoldier[SideOfPiece[i]][pieceCoordinate[i].y][pieceCoordinate[i].x]];
    }
}

//xet xem co ket noi voi quan khac khong va tinh gia tri khi ket noi
for (i = 0; i < 32; i++)
{
    for (j = 0; j < 32; j++)
    {
        if (PieceContact[i][j] != 0)
        {
            if (SideOfPiece[i] == SideOfPiece[j])
            {
                BeAteCount[j]++;
                if (!OwnSee[j])
                {
                    //xet gia tri co ket noi theo phong thu
                    PieceExtValue[i] += PieceBaseValue[j] * contactpercent1 / 100;  //one's own
                    OwnSee[j] = TRUE;
                }
            }
            else
            {
                //xet gia tri co ket noi theo tan cong
                PieceExtValue[i] += PieceBaseValue[j] * contactpercent2 / 100;      //other side
                BeAteCount[j]--;
            }
        }
```

3.Sinh ra các nước đi

*BOOL EnumList(int map[11][12], POINT pieceCoordinate[32], int &side, int *piecesArray, POINT *move, int &count)*

- Xét từng nước đi của mỗi quân cờ
- Khi xét tướng, xét 2 con tướng có đối diện với nhau không. Và nếu chúng đối diện thì xét xem ở giữa 2 quân tướng có quân cờ nào không
- Tùy từng quân cờ xét xem nó có đi quân đó được không. Trả về True thì sinh ra nước đi đó còn false thì không sinh ra nước đi.

Ví dụ sinh nước đi cho quân Tướng:

```
        if (pieceCoordinate[0].x == pieceCoordinate[16].x)
        {
            flag = FALSE;
            for (j = pieceCoordinate[16].y + 1; j < pieceCoordinate[0].y; j++)
            {
                if (map[x][j] != 32)
                {
                    flag = TRUE;
                    break;
                }
            }
            if (!flag)
            {
                ADD(0, x, pieceCoordinate[16].y)
            }
        }
        j = y + 1; if (j <= 10 && NORED(x, j))  ADD(0, x, j)
        j = y - 1; if (j >=  8 && NORED(x, j))  ADD(0, x, j)
        i = x + 1; if (i <=  6 && NORED(i, y))  ADD(0, i, y)
        i = x - 1; if (i >=  4 && NORED(i, y))  ADD(0, i, y)
        break;
    case 16: // BLACK KING
        if (pieceCoordinate[0].x == pieceCoordinate[16].x)
        {
            flag = FALSE;
            for (j = pieceCoordinate[16].y + 1; j < pieceCoordinate[0].y; j++)
            {
                if(map[x][j] != 32)
                {
                    flag = TRUE;
                    break;
                }
            }
            if(!flag)
            {
                ADD(16, x, pieceCoordinate[0].y);
            }
```

4.Hàm search tính trước các nước đi

int Search(int map[11][12], POINT pieceCoordinate[32], int &side, int piece, POINT point, int upmax, int depth)

- Dùng hàm EnumList sinh nước đi
- Xét cây trò chơi với độ sâu là 6
- Xử lý di chuyển và việc ăn quân

Ví dụ: Đệ quy tính trước nước đi với độ sâu là 6:

```
POINT targetpoint[125];
if (EnumList(map, pieceCoordinate, side, piecesArray, targetpoint, count))
{
    if (depth >= 2 && count > S_WIDTH + 2)
    {
        cur = 0;
        maxvalue = -10000;
        int value[125];
        while(cur < count)
        {
            curvalue = Search(map, pieceCoordinate, side, piecesArray[cur], targetpoint[cur], -10000, depth - 2);
            value[cur] = curvalue;
            maxvalue = curvalue > maxvalue ? curvalue : maxvalue;
            cur++;
        }
        QuickSort(value, piecesArray, targetpoint, 0, count - 1);
        count = S_WIDTH;
    }

    cur = 0;
    maxvalue = -10000;
    while(cur < count)
    {
        curvalue = Search(map, pieceCoordinate, side, piecesArray[cur], targetpoint[cur], maxvalue, depth);
        maxvalue = curvalue > maxvalue ? curvalue : maxvalue;
        if(curvalue >= -upmax) goto _ENDSUB;
        cur++;
    }
}
else
    maxvalue = 9800;
```

5. Hàm Think tính toán việc lựa chọn nước đi

BOOL Think(int map[11][12],POINT pieceCoordinate[32],int &side,int &resultPiece, POINT &resultpoint)

- Sử dụng các hàm enumList và hàm search để sinh nước đi và tính toán giá trị của nước đi
- Tính toán việc đi quân cờ nào và vị trí của quân cờ sẽ đi

```
if (EnumList(map, pieceCoordinate, side, piecesArray, targetpoint, count))
{
    if (S_DEPTH >= 2 && count > S_WIDTH + 2)
    {
        int value[125];
        cur = 0;
        maxvalue = -10000;
        while(cur < count)
        {
            curvalue = Search(map, pieceCoordinate, side, piecesArray[cur], targetpoint[cur], -10000, S_DEPTH - 2);
            value[cur] = curvalue;
            maxvalue = curvalue > maxvalue ? curvalue : maxvalue;
            cur++;
        }
        QuickSort(value, piecesArray, targetpoint, 0, count - 1);
        count = S_WIDTH;
    }

    cur = 0;
    maxvalue = -10000;
    while(cur < count)
    {
        curvalue = Search(map, pieceCoordinate, side, piecesArray[cur], targetpoint[cur], maxvalue, S_DEPTH);
        if (curvalue > maxvalue)
        {
            maxvalue = curvalue;
            resultPiece = piecesArray[cur];
            resultpoint = targetpoint[cur];
        }
        cur++;
    }
    return TRUE;
}
```

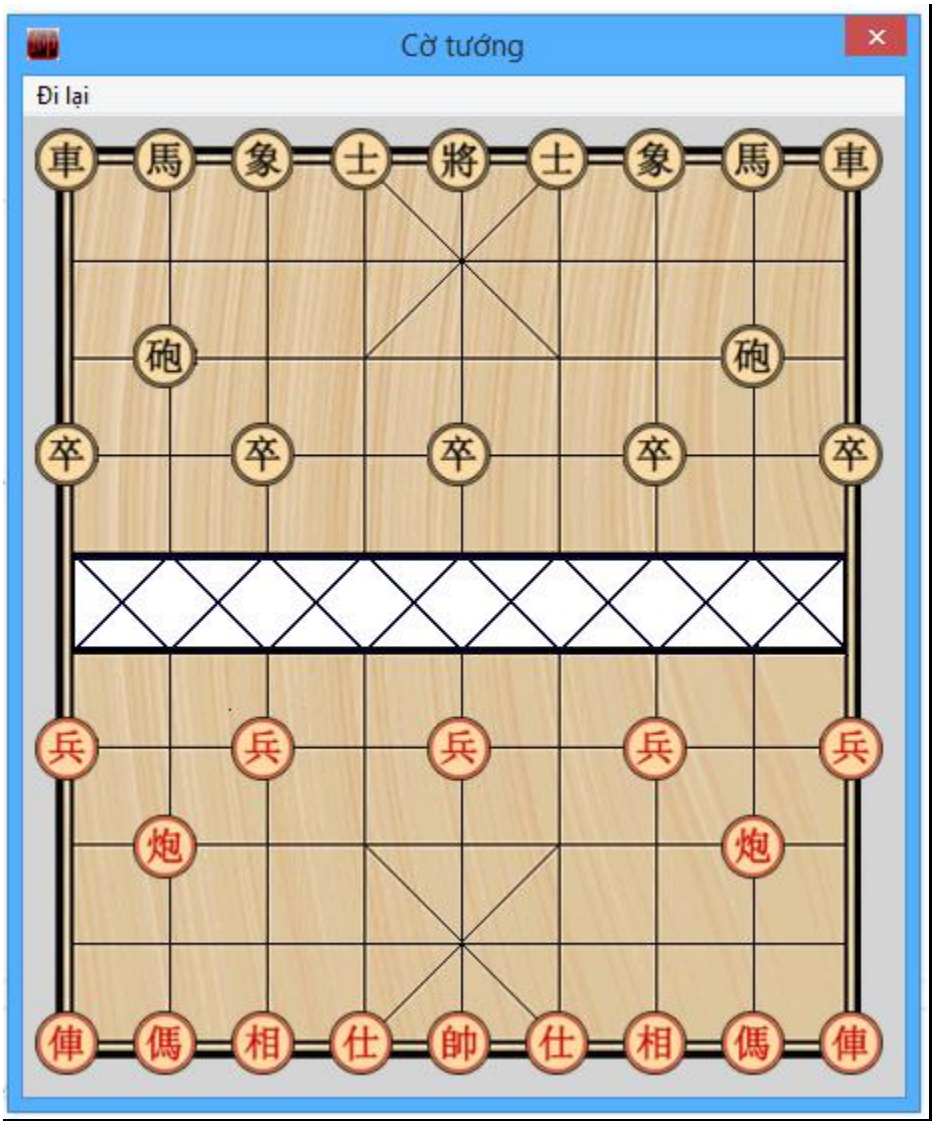
Chương 4: Ứng dụng

4.1 Giới thiệu chương trình cờ tướng

Chức năng

Chương trình có chức năng đánh lại , người chơi mặc định là Quân Đỏ, máy là Quân Đen.

Giao diện trò chơi



4.2 Cài đặt

Ngôn ngữ sử dụng C++ để lập trình trong môi trường Visual Studio 2013

4.3 Kết quả chạy chương trình (Minh họa trong file Video)

Nhận xét: Giao diện còn đơn giản. Các nước đi đôi khi chưa cho ra những nước đi tốt nhất.

Chương 5: KẾT LUẬN

Do thời gian có hạn nên chương trình cờ tướng chưa thực sự hoàn thiện nhưng trong tương lai nhóm sẽ cố gắng tăng độ sâu cho cây tìm kiếm, cải thiện thuật toán đồng thời làm giao diện tốt hơn.

Sau thời gian làm đồ án, em rút ra được nhiều kinh nghiệm cũng như kiến thức lập trình. Em xin cám ơn cô và thầy đã hướng dẫn chúng em trong quá trình làm đồ án.

TÀI LIỆU THAM KHẢO

TS Phạm Thanh Hà (2012), *Giáo trình học phần Trí tuệ nhân tạo*
Phạm Hồng Nguyên (2001), *Tự viết chương trình cờ tướng*
Bài viết Writing a chess program in 99 steps từ : http://www.sluijten.com
Code tham khảo : Phạm Hồng Nguyên http://xiang-qi.appspot.com/software2.html

Thêm nội dung các phần biểu diễn bàn cờ, tạo các nước cờ hợp lệ và thuật giải các phầ, ở trang 17,18,19,20