Connect Four AI

Vũ Minh Quang – 18521304 và Nguyễn Đức Quốc - 18521309

*Abstract*— Mục đích chúng tôi làm bài báo cáo này là để tìm hiểu và áp dụng thuật toán Minimax và Alpha-Beta Pruning vào một game đơn giản đó chính là game Connect 4 bằng Python. Từ đó tạo ra được một game Connect 4 chơi một mình với AI hoàn hảo, với độ sâu nhìn trước của thuật toán càng lớn thì tỉ lệ càng cao, nhưng thời gian chạy cũng sẽ rất chậm.

# INTRODUCTION

C

onnect 4 được xuất bản lần đầu tiên bởi Milton Bradly vào tháng 2 năm 1974. Connect 4 là trò chơi board-game kết nối trong đó người chơi thả một đĩa màu từ trên xuống vào một lưới treo thẳng đứng bảy cột, sáu hàng, các mảnh rơi thẳng xuống, chiếm phần thấp nhất không gian trống trong cột, sau đó người chơi khác lần lượt làm như vậy. Mục tiêu thắng của trò chơi là trở thành người đầu tiên tạo thành một đường ngang, dọc hoặc chéo của bốn đĩa của riêng mình. Người chơi đầu tiên luôn có thể giành chiến thắng bằng cách chơi đúng nước đi.



Nguồn: google hình ảnh.

Chúng tôi chọn trò chơi này để làm nghiên cứu với những lí do là: trò chơi này khá đơn giản, không giống như cờ vua, cờ tướng phức tạp hơn nhiều về các nước, bước đi của những quân cờ khác nhau. Và Connect 4 tuy đơn giản về số nước đi nhưng nó vẫn có những chiến thuật hay mà chúng ta có thể cài đặt cho AI.

Có nhiều cách để giải quyết game Connect 4, và cũng có nhiều cấp độ khó của AI khác nhau. Như AI dễ nhất là AI chơi một cách ngẫu nhiên, mức độ khó hơn là một AI chơi theo cách phòng thủ, nó sẽ chặn tất cả nhưng bước đi mà đối phương có thể thắng được, hoặc AI chơi với phong cách tấn công bằng cách tìm ra những nước đi thắng đặt lên hàng đầu.

Nhưng theo chúng tôi nghĩ là thuật toán tốt nhất cho AI có thể giải quyết được trò chơi Connect 4 là sự kết hợp vừa phòng thủ vừa tấn công. Và có một thuật toán có thể thõa mãn cả 2 chiến lược trên đó là Minimax và Alpha-Beta Pruning. Minimax là một thuật toán đệ quy sử dụng kĩ thuật quay lui để tìm ra bước đi tối ưu, và là cách khó nhât cho đối phương có thể đánh bại. Để hiểu rõ hơn, chúng ta giả sử đối thủ là Min, còn AI là Max. Min sẽ cố gắng tối thiểu hóa số điểm của Max và ngược lại Max sẽ phải tối đa số điểm của mình. Thuật toán Minimax sẽ sử dụng luật của trò chơi và nhìn trước tất cả những bước đi để tìm ra nước đi tốt nhất về phòng thủ lẫn tấn công. Đó là lí do mà thuật toán Minimax có thể được cho là phù hợp nhất với trò chơi này, tuy nhiên thuật toán này đòi hỏi rất nhiều quá trình và bước đi để giải quyết hay dự đoán, chính vì vậy làm cho thời gian chạy trở nên chậm chạp. Chính vì từ thuật toán Minimax, chúng ta có thể kĩ thuật cắt tỉa hay còn gọi là Alpha-Beta Pruning để tối ưu thuật toán Minimax này. Bởi vì Minimax sẽ dự đoán các bước đi nhiều nhất có thể cho nên làm tăng độ phức tạp của thuật toán, cho nên sử dụng Alpha-Beta Pruning cắt bỏ đi những bước đi không quan trọng, không làm ảnh hưởng tới kết quả. Chính vì vậy làm tăng độ hiệu quả của thuật toán, giảm độ phức tạp và thuật toán sẽ chạy nhanh hơn.

1. RELATED WORKS

Thuật toán Minimax bắt nguồn “định lí Minimax”, được đề xuất bởi nhà toán học người Mỹ gốc Hungary, John von Nẻumann vào năm 1928. Ban đầu, định lí Minimax được sử dụng trong trò chơi có tổng bằng không (Zero-sum game) với hai người chơi biết tất cả các bước đi được diễn ra sau đó. Tuy nhiên, John von Neumann đã cải tiến và mở rộng định lí Minimax với những trò chơi thông tin không hoàn hảo với nhiều người chơi thay vì là 2 người chơi trong Zero-sum game, và đã được công bố kết quả vào năm 1944 (bài báo cáo được viết cùng với nhà kinh tế học người Đức Oskar Morgenstern).

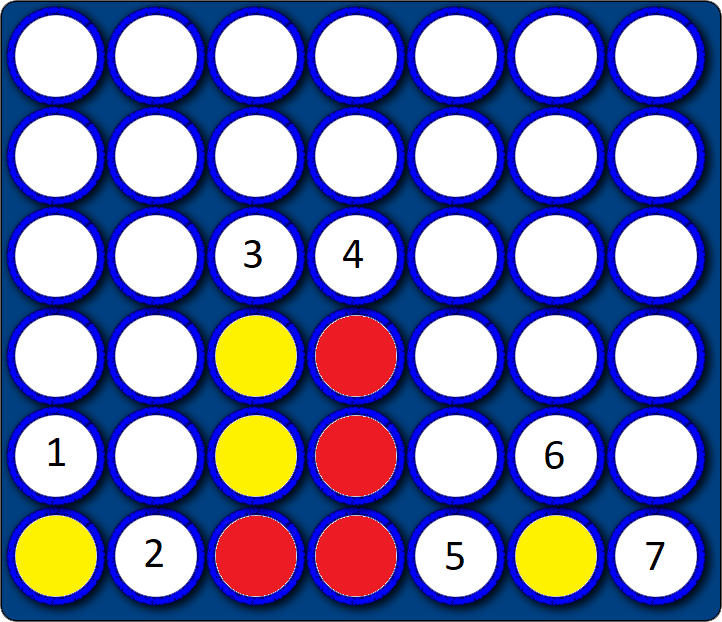
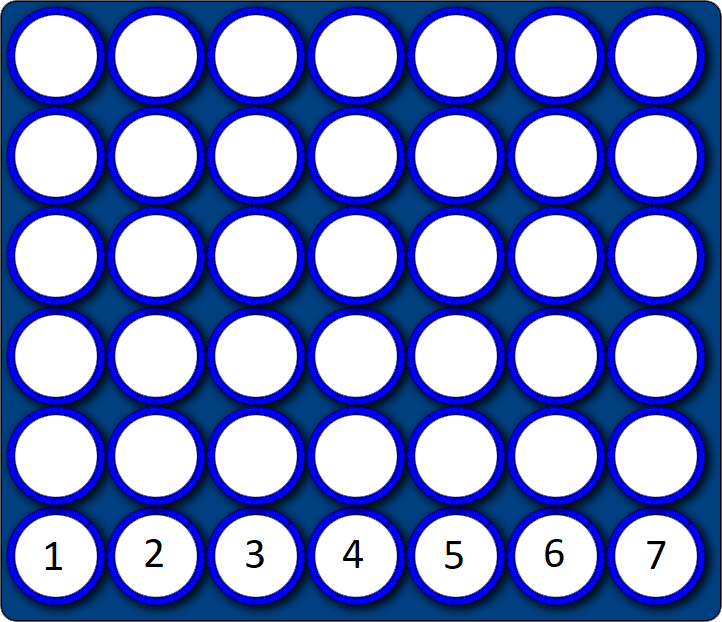
Về cơ bản thì, định lí Minimax được lần đầu sự dụng vào trong lĩnh vực trò chơi zero-sum game và vẫn đóng một vai trò quan trọng để giải quyết các trò chơi hiện đại ngày nay vẫn phổ biến ngày nay như Tic-tac-toe, Connect 4. Năm 1950, nhà toán học người Mỹ, được biết đến là “cha đẻ của lý thuyết thông tin, Claude Shannon đã xuất bản một bài báo cáo lần đầu đưa ra ý tưởng về một hàm để đánh giá hiệu quả của một bước đi cụ thể và thuật toán “minimax” đã tận dụng chức năng đánh giá này bằng cách tính đến hiệu quả của các bước đi sẽ được thực hiện trong tương lai có sẵn bằng bất kỳ bước đi cụ thể nào đó. Sau đó, sau nhiều năm cố gắng và nỗ lực để cải thiện thuật toán Minimax, máy tính chơi cờ vua Deep Blue do IBM phát triển sử dụng Minimax để lập trình đã xuất sắc đánh bại siêu đại kiện tướng cờ vua của Nga và được coi là kỳ thủ cờ vua mạnh nhất lịch sử, Garry Kasparov. Ngoài ra, thuật toán Minimax còn được ứng dụng cho nhiều lĩnh vực, vấn đề khác nhau. Trong trò chơi có tổng bằng không với hai người chơi, luôn luôn có một chiến lược cụ thể và hợp lí để cho người chơi đó đạt được lợi tức tối ưu, đó cũng là điều cốt lõi trong lí thuyết trò chơi. Chiến lược này còn được sử dụng trong lĩnh vực tài chính và kinh tế, khi mà người ra quyết định sẽ cố gắng tối ưu hóa lợi nhuận của chính mình Bên cạnh đó, kết hợp với các thuật toán và phương pháp toán học khác, minimax cũng tham gia vào thiết kế mạng, lĩnh vực robot và những cuộc bầu cử công khai.

# Materials

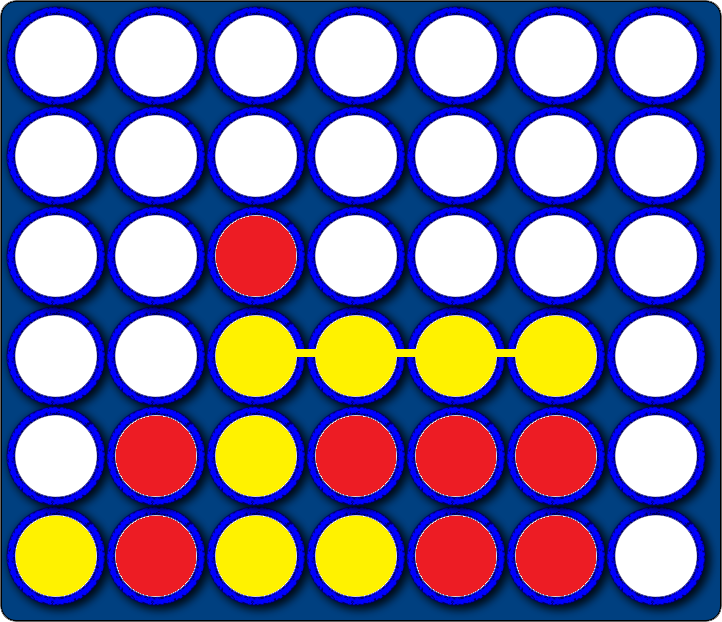
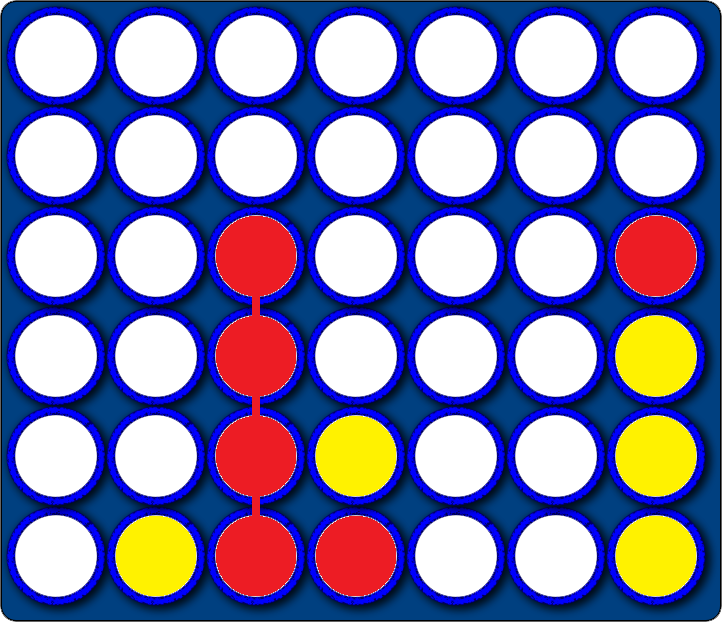
Chúng tôi chia bài cáo làm 2 phần. Phần đầu là giới thiệu cách tạo ra trò chơi Connect 4. Phần thứ hai, chúng tôi tạo ra AI sử dụng thuật toán Minimax và Alpha-Beta Pruning cho việc đấu với người chơi

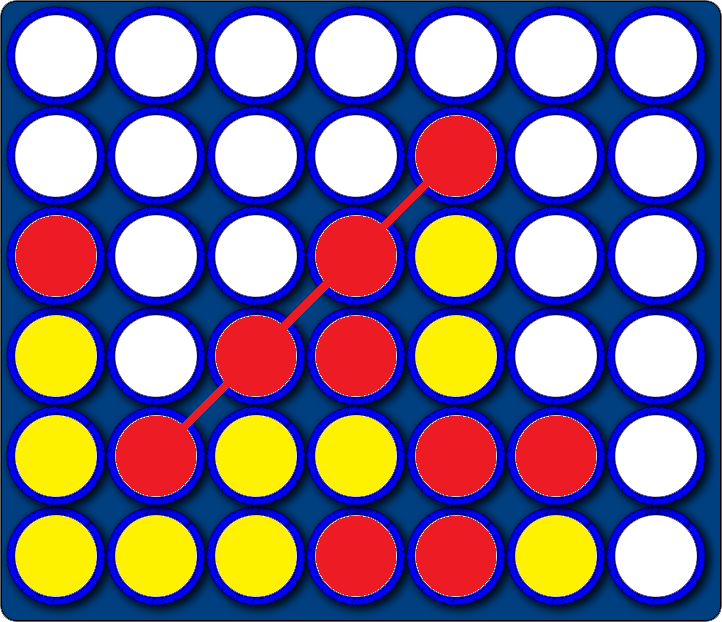
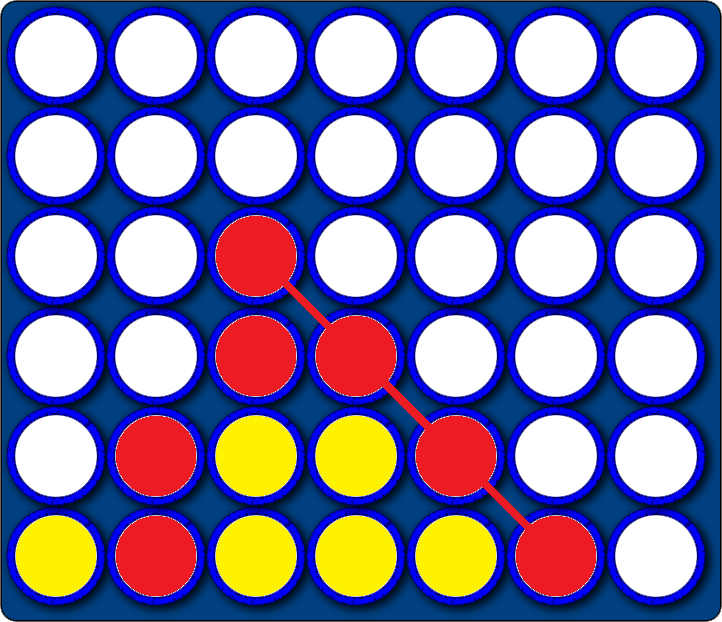
## Rules of the game

Connect 4 là một biến thể của cờ caro, sẽ có hai người chơi đối đầu nhau trên một bảng với 7 ô dọc chứa 6 vị trí của mỗi ô đó. Giả sử người chơi thứ nhất chọn màu vàng và người chơi thứ hai chọn màu đỏ. Mỗi người sẽ đặt một mảnh màu đã chọn trước đó vào một trong những ô trống và thả nó xuống dưới tác dụng của trọng lực vào vị trí thấp nhất của ô trống trong cột. Mỗi người chơi sẽ cố gắng làm sao đó mà có thể tạo thành được các đoạn thẳng ngang, dọc hoặc chéo có độ dài là 4. Hình ở dưới đây cho chúng ta một ví dụ về vị trí mà người chơi chọn vàng sẽ thắng nếu người chọn đỏ không chặn điều này bằng cách đặt mảnh màu đỏ vào ô bên phải.



Người chơi thắng trò chơi bằng cách là người đầu tiên xếp thành 4 đoạn theo bất kỳ hướng nào đó.





Trò chơi Connect-Four mà chúng tôi đã tạo chạy bằng cách kiểm tra chuyển động của con trỏ chuột để tìm vị trí của cột mà một mảnh có thể thả xuống. Điều này có thể được thực hiện bằng cách sử dụng thư viện pygame trong Python, và nếu có một cú nhấp chuột trong chương trình, nó sẽ thả mảnh vào cột đó nơi đặt con trỏ chuột. Tiếp theo, trò chơi sẽ kiểm tra trên bàn cờ xem có nước đi thắng hay không, nếu không, đối thủ sẽ đến lượt. Cuối cùng, nếu có nước đi thắng thì trò chơi sẽ kết thúc và thoát trò chơi.

## Functions

create\_board() : Tạo ra một ma trận với 6 hàng và 7 cột cho bàn cờ của trò chơi.

drop\_piece(): thả 1 mảnh vào trong bàn cờ.

is\_valid\_location(): Kiểm tra mảnh đó có hợp lệ, hợp vị trí hay không.

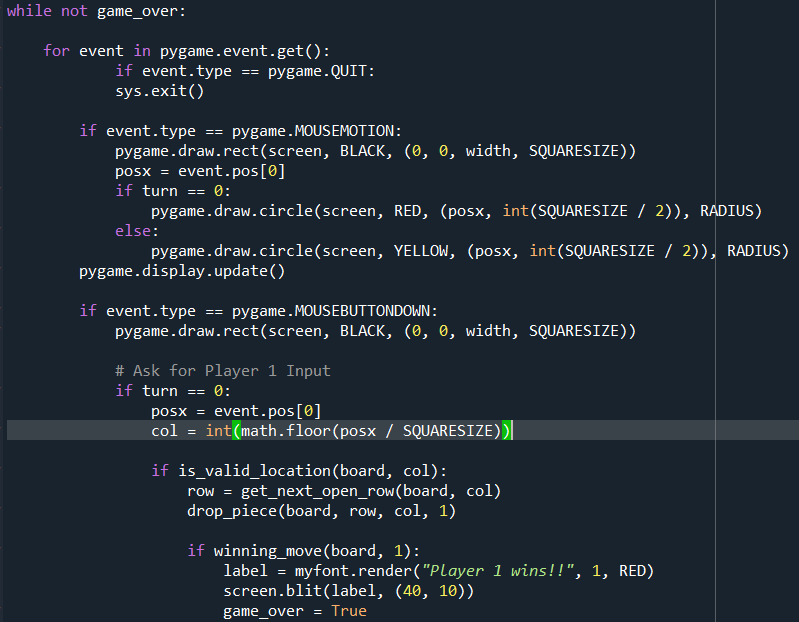
get\_next\_open\_row(): Làm cho một mảnh vào vị trí hàng còn trống tiếp theo của cùng một cột.

print\_board():Lật ngược bàn cờ lại để cho trò chơi có ảo giác như là có trọng lực thả mảnh xuống.

winning\_move(): Kiểm tra toàn bộ mảnh trên bàn cờ, nếu có một nước đi thắng mà được các mảnh kết nối thành một đoạn thẳng theo chiều dọc, ngang hoặc chéo.

draw\_board(): Sử dụng thư viện pygame để vẽ GUI cho Connect 4 game.

pygame.init(): Khởi tạo lại tất cả modules pygame





# Methods

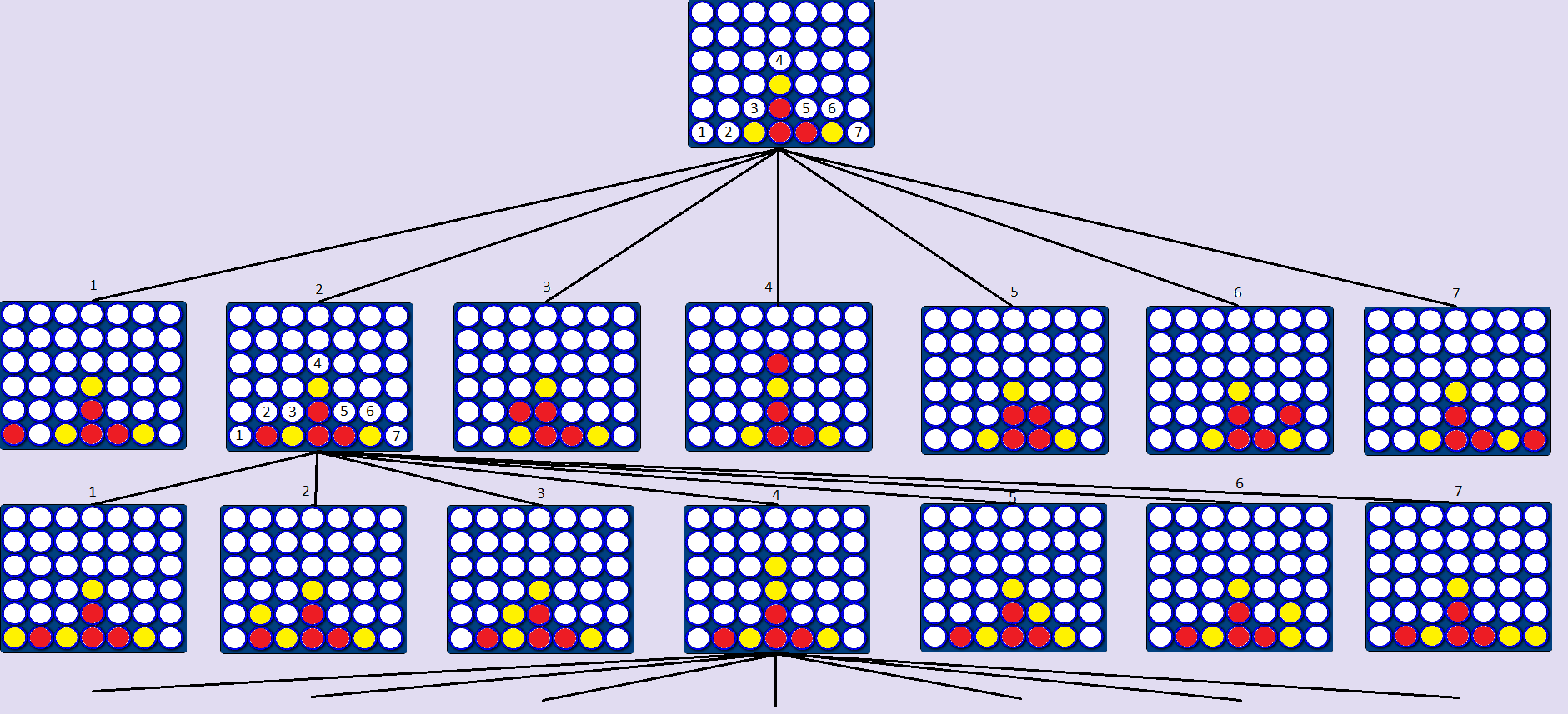
Bây giờ chúng ta sẽ nói về cách cài đặt một AI (các kĩ thuật trí tuệ nhân tạo và bàn luận cách cài đặt nó như thế nào).

1. *GAME TREES*

Đối với những trò chơi được lập trình để chơi với người hay chơi với chính nó. Nói một cách đơn giản thì thỏa mãn 2 tính chất sau:

* 2 người chơi – Chúng ta không thể liên minh mà chỉ có thể đối đầu nhau.
* Tổng bằng không (Zero sum) - một người chơi thắng hoặc là người kia thua, cũng không có hợp tác chiến thắng.

Đó chính là trò chơi dạng bảng tiêu biểu là Connect 4 và giống như nó là cờ vua, tic-tac-toe, cờ vây,... Với những loại trò chơi này chúng ta có thể cấu trúc bằng cây trò chơi (Game Tree). Trong lí thuyết trò chơi nổi tiếng, cây trò chơi là một đồ thị có hướng, có các nút là các vị trí trong trò chơi đó và các các cạnh của nó chính là các bước đi được dự đoán. Các node bên trong ở độ sâu là số lẻ được đại diện cho trạng thái trò chơi ban đầu (node gốc), hoặc trạng thái trò chơi mà kết quả của nó là bước đi của đối thủ. Các node bên trong ở độ sâu được đánh số chẵn thì lại thể hiện các trạng thái mà người chơi là chính chúng ta đang thực hiện. Trạng thái trò chơi kết thúc là lúc mà 4 mảnh trong bảng được kết nối thành đoạn thẳng dọc, ngang hoặc chéo, đó chính là node lá. Mỗi node lá thể hiện số điểm thưởng nhất định tùy thuộc vào kết quả trạng thái trò chơi kết thúc, và node lá không thể mở rộng thêm. Dưới đây là ví dụ minh họa về 1 cây trò chơi cho game Connect 4:



Cây trò chơi rất quan trọng trong trí tuệ nhân tạo vì nó đề xuất tất cả các trường hợp để rồi chọn cách để chọn nước đi tốt nhất trong trò chơi, tìm kiếm cây trò chơi bằng cách sử dụng bất kỳ thuật toán tìm kiếm cây nào, kết hợp với các quy tắc giống như minimax để cắt tỉa cây. Số nút lá trong cây trò chơi hoàn chỉnh là số cách chơi khác nhau có thể có của trò chơi. Cây trò chơi cho tic-tac-toe có thể dễ dàng tìm kiếm, nhưng cây trò chơi hoàn chỉnh cho các trò chơi lớn hơn như cờ vua, Connect 4 có thể quá lớn để tìm kiếm. Ví dụ ở đây, cây trò chơi của tic-tac-toe là 255,168 node lá. Và cây trò chơi của Connect 4 là 4,531,985,219,092 node lá có thể có được. Nói tóm lại, trò chơi có hai người có thể được biểu diễn dưới dạng cây. Để người chơi đầu tiên thắng một trò chơi, phải tồn tại một nước đi thắng cho tất cả các nước đi của người chơi thứ hai. Điều này được thể hiện trong cây và-hoặc bằng cách sử dụng phép kết hợp để thể hiện các nước đi thay thế của người chơi thứ nhất và sử dụng phép kết hợp để đại diện cho tất cả các nước đi của người chơi thứ hai.

1. *MINIMAX*

Như chúng ta đã biết thì có rất nhiều thuật toán tìm kiếm để làm AI trong game như A\*, Heuristic... Mỗi thuật toán thì sẽ phù hợp với từng loại game khác nhau. Những game đối kháng trong đối người chơi luân phiên đánh như cờ vua, cờ tường, caro... Khi chơi, người chơi có thể khai triển hết không gian trạng thái nhưng khó khăn chủ yếu là phải tính toán được phản ứng và nước đi của đối thủ mình như thế nào? Cách xử lý đơn giản là giả sử đối thủ của chúng ta cũng sử dụng kiến thức về không gian trạng thái giống chúng ta. Đối thủ đó sẽ tìm bước đi tốt nhất có thể, để giành chiến thắng. Và ngược lại, chúng ta sẽ ngăn đối phương tới con đường chiến thắng đồng thời cũng tìm cho mình ra bước đi đúng đắn để chiến thắng trò chơi. Đối với trò chơi Connect 4, một con đường nào đó sẽ tồn tại nếu bạn là người bắt đầu trò chơi. Trường hợp này thuật toán minimax sẽ đáp ứng những gì chúng ta cần.

Trước khi chúng ta áp dụng thuật toán này với cây trò chơi, chúng ta phải định nghĩa được hàm số điểm đạt cho mỗi node trên cây. Đây là cấu hình số điểm đạt được trong trò chơi:

\* +4 cho việc đặt quân ở trung tâm bàn cờ vì trung tâm sẽ cho nhiều lựa chọn hơn để có vị trí di chuyển có thể thắng tốt nhất.

\* +2 nếu có một dòng gồm 2 mảnh

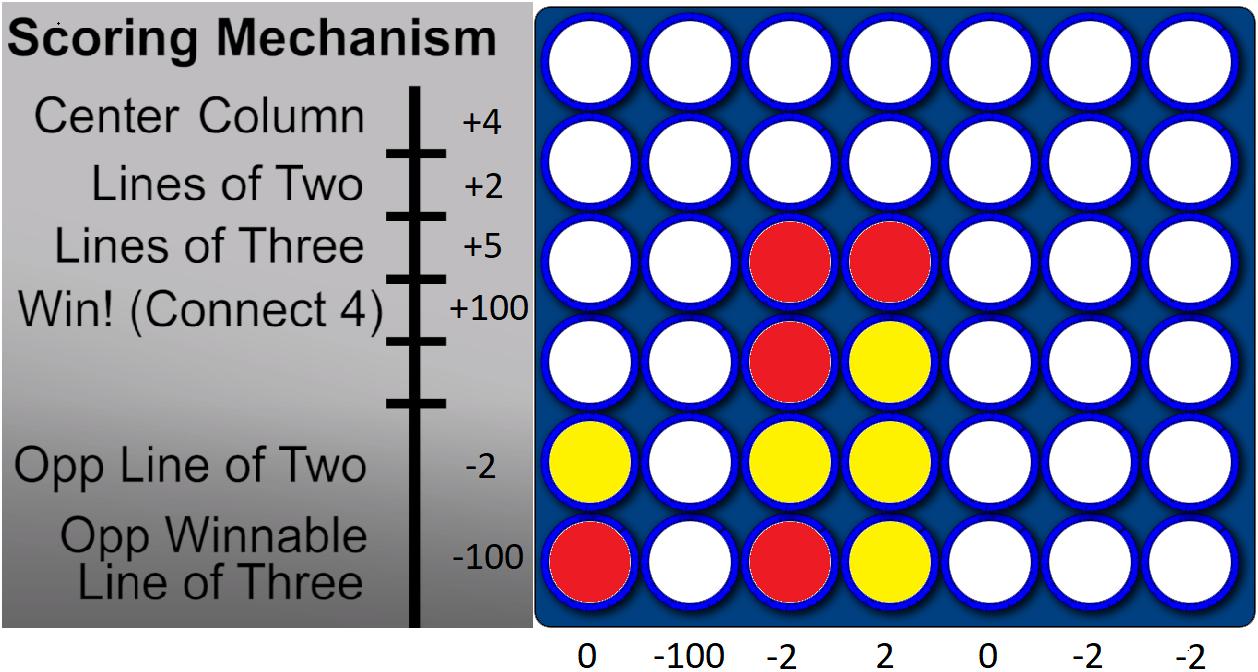
\* +5 nếu có một dòng gồm 3 mảnh

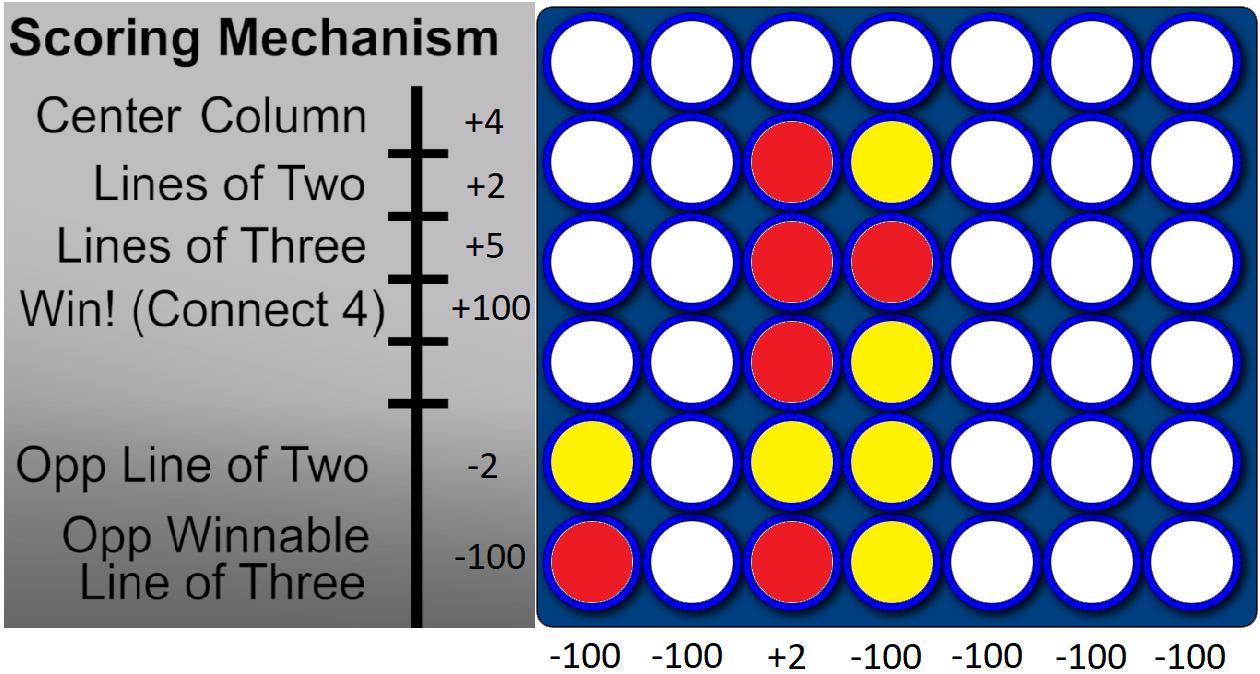
\* +1000000 nếu có nước đi thắng (một hàng 4 mảnh)

\* -2 nếu đối thủ có đường 2 mảnh

\* -5 nếu đối phương có 3 mảnh

\* -1000000 nếu đối thủ có nước đi thắng





Tóm lại, AI sẽ thực hiện:

* Nếu trên bàn cờ có nước đi thắng cuộc, thì nó sẽ đi nước đó
* Nếu đối phương có nước đi thắng cuộc thì nó sẽ đi nước đó để đối phương không đi được
* Luôn luôn đi vào cột chính giữa hơn là đi ở cột sát mép
* Nó sẽ đi sát mép bàn cờ khi hết những nước được cộng điểm.

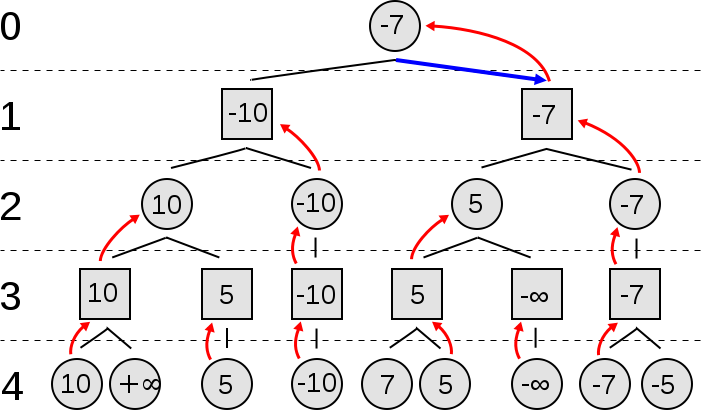
Trong trò chơi, chúng tôi lập trình cơ chế tính điểm ở trên bằng cách sử dụng các function này:

score\_position (): Tìm trên bảng, liệu có một hàng nào đó có các mảnh để ghi điểm hay không

eval\_window (): Cho điểm từng hàng của bảng phụ thuộc vào vị trí của số điểm.

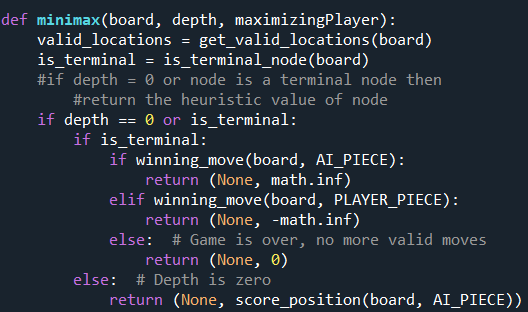
Giả dụ như trò chơi của chúng ta chỉ có hai nước đi tối đa cho mỗi người chơi với một lượt như vậy. Thuật toán này tạo ra cây trò chơi ở hình dưới đây, với các hình tròn đại diện cho người chơi có những nước đi khi đang chạy thuật toán sẽ cố gắng tối đa hóa số điểm. Ngược lại, đối thủ của người chơi sẽ được đại diện là nhưng hình vuông cố gắng tối thiểu hóa số điểm. Tùy thuộc vào người cài đặt thuật toán muốn nhìn trước bao nhiêu bước có thể, ở đây tài nguyên giới hạn của cây chỉ có thể nhìn trước được 4 bước mà thôi.

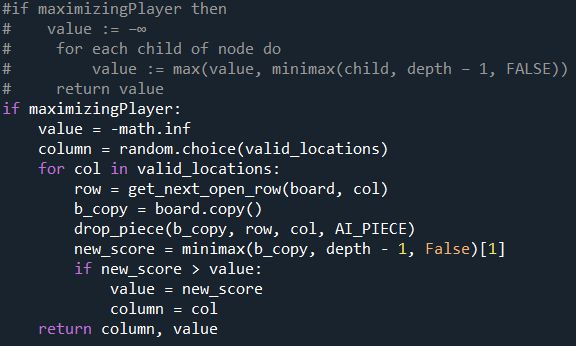
Thuật toán này đầu tiên chúng ta sẽ bắt đầu từ nút gốc ở cấp độ thấp và bắt đầu đi xuống nút con ở phía bên trái nhất. Chúng ta thấy mức độ 4 có node cha là 3(tùy thuộc vào người lập trình theo chúng tôi thì mặc định là nếu node cha ở mức độ chẵn thì sẽ tối đa số điểm của những node lá, là nếu node cha ở mức độ lẻ thì sẽ tối thiểu số điểm của những node lá). Vậy nên ví dụ ở đây giá trị của node cha ở mức độ 3 phía bên trái sẽ là 10 bởi vì 10 bé hơn dương vô cực, tiếp theo những node còn lại cũng được so sánh như vậy. Và nếu như node cha chỉ có một node con thì node cha sẽ được gán giá trị của chính node con đó. Và cứ từng bước như vậy thuật toán tiếp tục một cách tuần tự cho đến khi đạt đến được node gốc, nơi nó chọn giá trị lớn nhất được vễ bằng mũi tên màu xanh riêng biệt so với các mũi tên màu đỏ của các node khác. Đây chính là những bước đi mà người chơi cố gắng giảm tối thiểu tổn thất tối đa có thể xảy ra trong trò chơi.

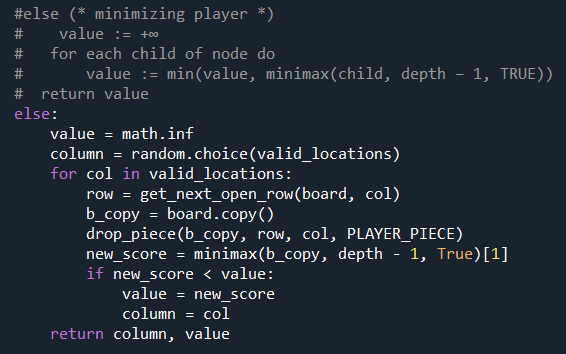


Nguồn:Wikipedia

Bây giờ chúng ta đã hiểu cách sử dụng thuật toán Minimax để giải quyết trò chơi Connect-4 , bằng cách chọn bước đi tốt nhất để đạt số điểm cao nhất. Chúng ta cài đặt thuật toán Minimax bằng những hàm sau đây:







Bước tiếp theo, chúng tôi sẽ để người chơi thứ 2 là AI , sử dụng hàm minmax để chơi trò chơi này



Minimax được dùng để tạo nên một AI hoàn hảo cho trò chơi Connect-Four và dường như không thể chiến thắng được khi chơi với AI biết trước 5 bước trong mỗi bước đi mà chúng tôi thực hiện. Nhưng vấn đề của Minimax là tốn quá nhiều thời gian để xử lý thuật toán, khiến trò chơi trở nên nhàm chán, xử lí lâu. Đó là lý do tại sao tính năng cắt tỉa Alpha-Beta có tác dụng.

1. *ALPHA-BETA PRUNING*

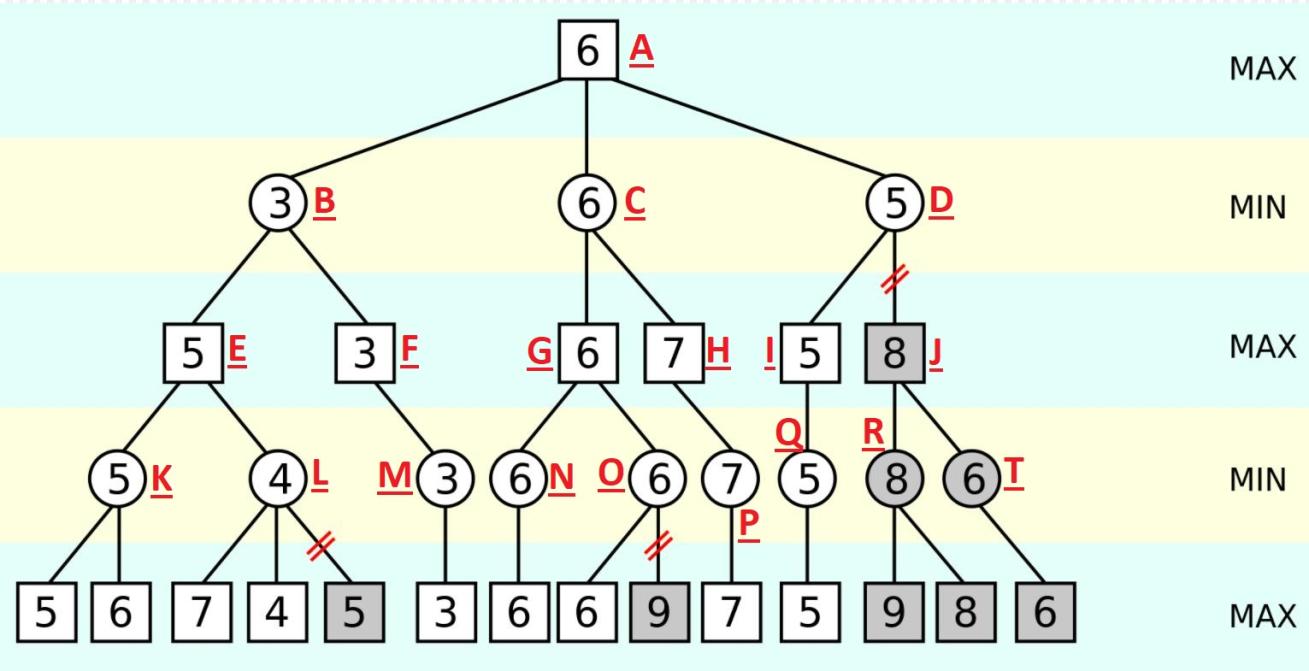
Chúng ta đã công nhận thuật toán Minimax là thuật toán hay nhất mà AI có thể sử dụng để chơi trò chơi Connect 4 khá hoàn hảo, nhưng nó xuất hiện ra một vấn đề lớn đó là thuật toán Minimax mất nhiều thời gian để chạy từng node lá một. Đó là lí do, Alpha-Beta Pruning là thuật toán tối ưu hơn, được cải tiến từ Minimax, vì sử dụng kĩ thuật cắt tỉa những node lá không cần thiết, không ảnh hưởng tới kết quả của thuật toán tìm nước đi tốt nhất. Giải thuật này thường sử dụng chung với thuật toán tìm kiếm Minimax nhằm hỗ trợ giảm bớt các không gian trạng thái trong cây trò chơi, giúp thuật toán Minimax có thể tìm kiếm sâu và nhanh hơn.

Ý tưởng của Alpha-Beta là thay vì tìm kiếm toàn bộ không gian đến một độ sâu cố định nào đó, Alpha-Beta Pruning thực hiện tìm kiếm theo kiểu tìm kiếm sâu. Có hai giá trị, gọi là Alpha và Beta được tạo trong quá trình tìm kiếm, cắt tỉa đó:

– Giá trị Alpha liên quan với các nút Max và có khuynh hướng không bao giờ giảm.

– Giá trị Beta liên quan đến các nút Min và có khuynh hướng không bao giờ tăng.

Có nghĩa là chúng ta lưu lại giá trị của Alpha-Beta, rồi xét các giá trị của node con , nếu giá trị node con mà nhỏ hơn Alpha hoặc là lớn hơn Be ta thì chúng ta sẽ bỏ các nhánh cây hơn node giá trị đó trở xuống. Hình dưới sau đây là một ví dụ của cắt tỉa Alpha-Beta:



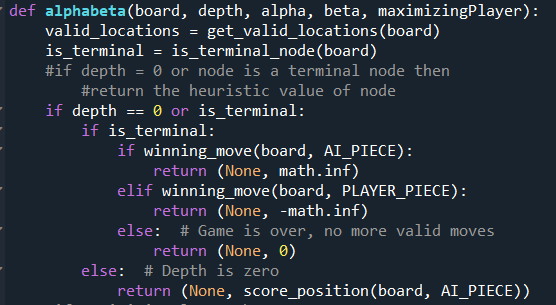
Nguồn: Wikipedia

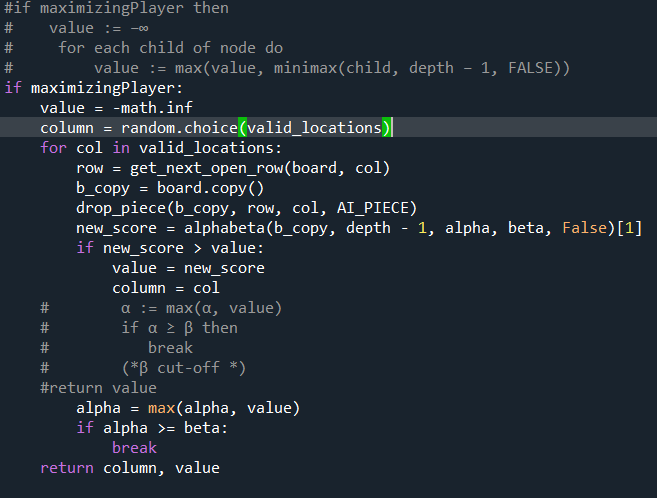
Ở đây giống như thuật toán Minimax chúng ta cũng xét từ trái qua phải bắt đầu từ nút gốc và nút con bên trái sẽ được ưu tiên duyệt trước. Xét duyệt từ trên gốc xuống sâu (vì ban đầu chưa hề tồn tại giá trị alpha hay beta của các nút). Nút chúng ta duyệt đầu tiên là nút sẽ gặp giá trị 5 (Beta <=5), khi đó ở trên chưa có giá trị Alpha để ta có thể so sánh nên sẽ bắt đầu duyệt nút con tiếp theo của nút K đó và chúng ta thấy nó có giá trị 6, nên sẽ chọn cho Beta = 5 (Min).

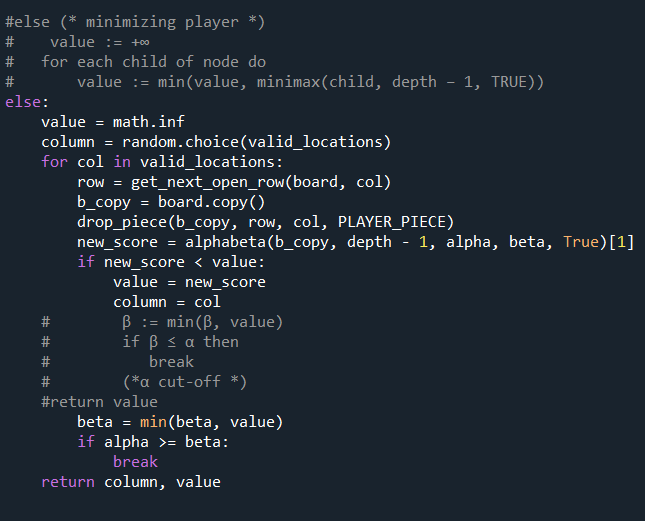
Luôn luôn duyệt từ trái sang phải và nó phải lần lượt từng nhánh một, sau đó sang nhánh tiếp theo cùng gốc. Vậy nên tiếp theo chúng ta đưa giá trị Beta này lên nút E (Max) và nút E trở thành E-Alpha >= 5.

Như vậy duyệt tiếp ở nút L có có nút con đầu tiên có giá trị 7 vậy Beta của L - Beta <= 7. Rồi duyệt ở nút kế tiếp thấy giá trị 4 nên Beta của L cập nhật lại giá trị Beta <=4. Chúng ta nhận thấy rằng giá trị Alpha hiện tại ở nút E là 5, mà tại L – Beta <=4, như vậy chúng ta không cần xét duyệt ở các nút con còn lại của L vì cái giá trị cần ở đây chỉ là khoảng >=5 nên cắt toàn bộ các nút còn lại trong cây. Cứ tiếp tục xét nút B sẽ có giá trị Alpha là 5. Và duyệt sang nhánh tiếp theo nhận thấy rằng cả nút M và E chỉ có một giá trị là 3, nên suy ra được giá trị Alpha tại E là 3. Sau đó lấy Beta ở nút B có giá trị Beta - <=3. Và các quá trình này cứ tiếp tục duyệt như vậy cho đến khi hết cây.

Tất cả những điều trên tuy có vẻ khó hiểu, nhưng một khi hiểu được thuật toán này thì có thể dễ dàng cài đặt thuật toán AI cho trò chơi Connect 4 này.



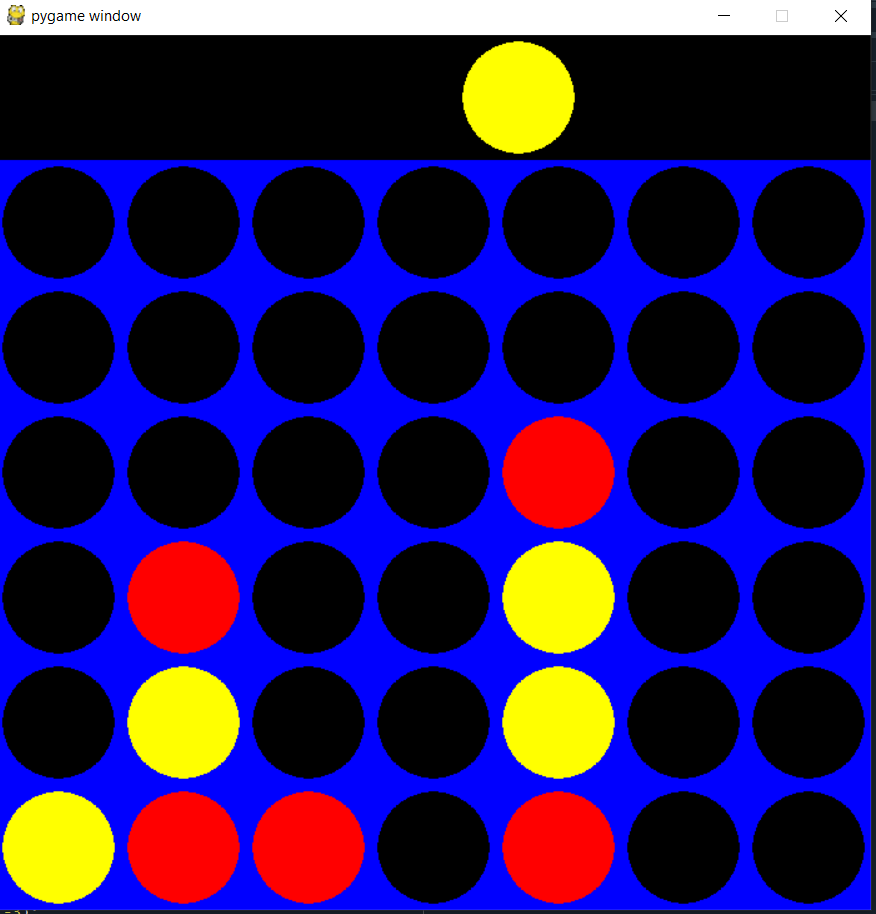




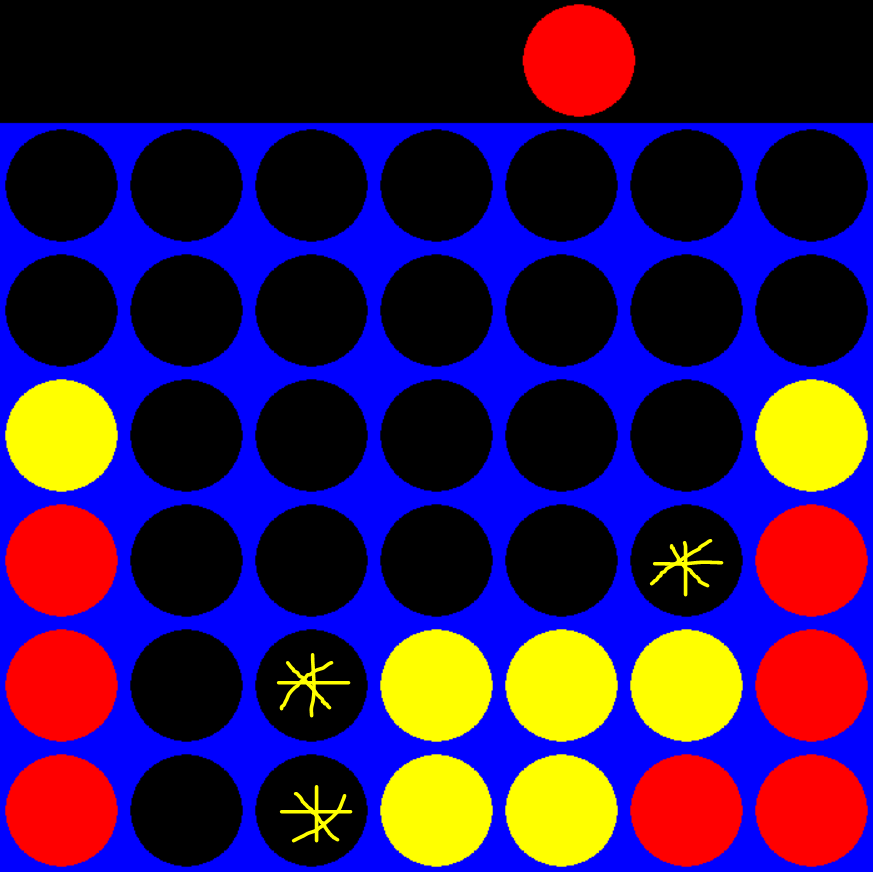
1. RESULT

Sau khi cài đặt Minimax và Alpha-Beta Pruning vào trò chơi Connect 4. Chúng ta có thể thấy rằng AI có thể chơi game một cách xuất sắc.

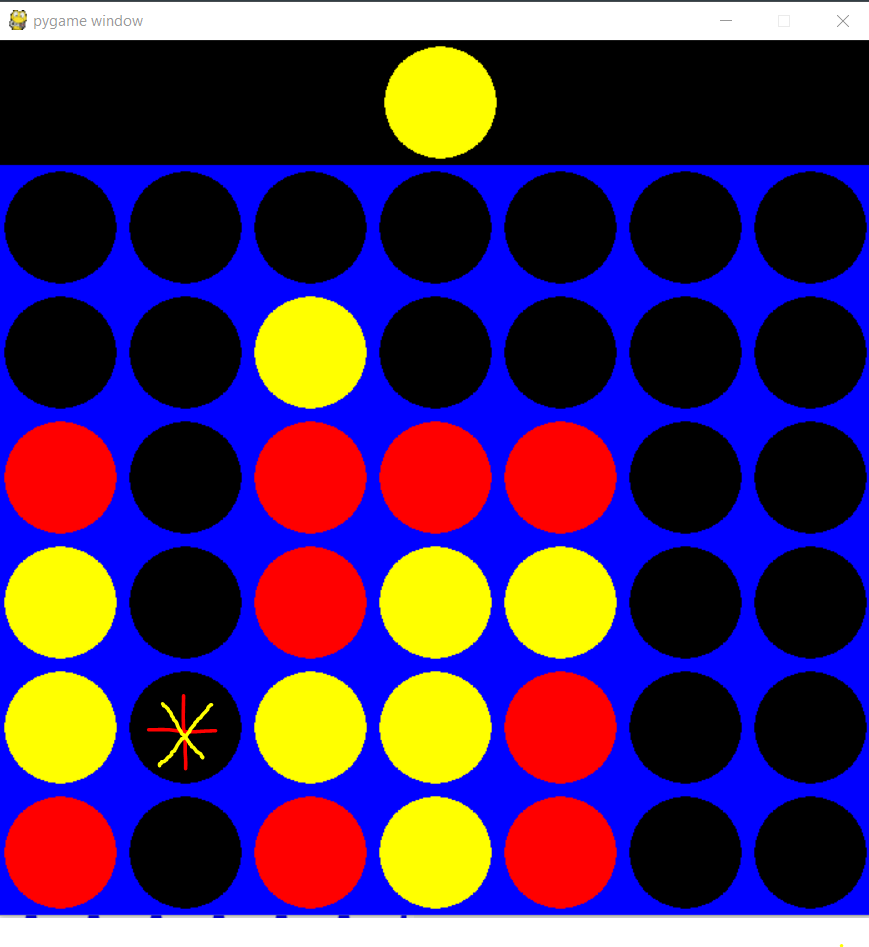
( Người chơi là mảnh màu đỏ, còn AI là mảnh màu vàng)



Bức hình ở trên cho chúng ta thấy rằng AI sẽ phải thả mảnh màu vàng của nó vào cột thứ 4, nếu không làm như vậy AI sẽ mất số điểm rất lớn, và Minimax sẽ không cho phép điều đó xảy ra.



Bức hình ở trên đây thì chỉ ra cách mà Minimax AI tấn công, có 3 khả năng mà AI có thể thắng, nếu người chơi thả 1 mảnh xuống vào cột thứ 6 dể chặn máy có thể thắng được, Minimax sẽ bảo AI thả vài cột thứ 3, bời vì nó sẽ cho AI thêm 2 khả năng dành bước đi thắng, đạt điểm cao nhất và dành chiến thắng



Đây là một tình huống rất thú vị, đặc biệt là cột thứ 2, nếu người chơi thả mảnh xuống cột thứ 2, AI thắng. Nếu AI thả mảnh vào cột thứ 2, người chơi sẽ thắng. Vì vậy trò chơi sẽ trở thành là ai thả mảnh vào cột thứ 2 xuống trước, sẽ thua, cho dù có chưa hoàn hảo đi nữa.

Chúng tôi chạy thử 10 lần mỗi trường hợp đặc biệt và rồi nhận xét về kết quả:

|  |  |  |
| --- | --- | --- |
| Trường hợp 1 | Đỏ  Minimax 1 | Vàng Minimax 5 |
| 0 | 10 |
| Trường hợp 2 | Đỏ  Minimax 2 | Vàng Minimax 5 |
| 0 | 10 |
| Trường hợp 3 | Đỏ  Minimax 3 | Vàng Minimax 5 |
| 0 | 7 |
| Trường hợp 4 | Đỏ  Minimax 3 | Vàng Minimax 3 |
| 4 | 6 |
| Trường hợp 5 | Đỏ  Minimax 2 | Vàng Minimax 4 |
| 4 | 6 |

- Trường hợp 1: Chúng ta thấy rằng là cho dù màu vàng hay màu đỏ đi trước đi nữa, màu vàng luôn thắng trò chơi vì là nó được nhìn trước sâu hơn nhiều so với đỏ, trong lúc đỏ chỉ nhìn trước 1 nước và như trên theo lý thuyết về Minimax thì 1 nước của nó chỉ nhìn được tới max, nghĩa là nó chỉ biết tấn công thôi, chứ không biết phòng thủ.

- Trường hợp 2: Bậy giờ thì AI đỏ thông minh hơn được 1 chút, nhưng vẫn chưa đủ để có thể thắng được trò chơi cho dù có được đi trước đi nữa.

- Trường hợp 3: Trong trường hợp này xảy ra 1 điều đặc biệt là trò chơi được hòa, không ai thắng, điều này xảy ra chỉ khi AI đỏ đi trước với độ nhìn trước 3 đánh với AI vàng với độ nhìn trước là 5.

- Trường hợp 4: Khi 2 AI có cùng một độ sâu nhìn trước của Minimax, thì trò chơi sẽ trở thành ai đi trước, người đó sẽ thắng.

- Trường hợp 5: Đây là trường hợp tiểu điểm để chứng minh độ quan trọng của việc đi trước trong trò chơi Connect – 4, máy được ngẫu nhiên chọn màu nào đi trước, nhưng khi màu nào đi trước, màu đó sẽ luôn thắng vì lợi thể của việc đi trước trong trò chơi Connect – 4 này.

\* Nhận xét tổng quan:

+ Qua nhiều lần thử nghiệm cho 2 AI với thuật toán Minimax hoặc Alpha-Beta pruning chơi với nhau, thi ta nhận ra được là 2 AI, chúng chơi với nhau một cách rất khô khan, nghĩa là kết quả lần này sẽ sẽ giống với lần trước nếu cùng đi trước, hoặc là cũng là đi sau, nó sẽ không thay đổi chiến lược chơi kể cả khi thua trận.

+ Nước đi đầu tiên là rất quan trọng, đa phần là AI nào mà được đi đầu tiên, kể cả chỉ nhìn trước được ít chiều sâu trong thuật toán Minimax, vẫn sẽ là người thắng cuộc, tệ nhất là hòa với AI sau, nên vì vậy, nếu ta là người chơi hoàn hảo đánh với 1 AI cố độ nhìn trước với chiều sâu là 100 cho đi nữa, thì tệ nhất thì ta chỉ có hòa, chứ không bao giờ có thể thua được, điều này cũng sẽ đúng nếu ngược lại.

+ Chiều sâu nhìn trước của thuật toán Minimax mà AI sử dụng để giải cho trò chơi Connect – 4 này không quan trong lắm như ban đầu chúng tôi nghĩ.

1. FUTURE WORK

Đối với nhưng việc làm sau này có thể cải hiện và nâng cao hơn cho trò chơi Connect 4 có AI. Chúng tôi có thể cải thiện phần code của game bằng cách loại bỏ nhưng dòng code lặp lại, tạo ra nhiều hàm có thể đọc dể hiểu code một cách rõ ràng hơn. Và có thể thử một số thuật toán khác ngoài Minimax, ví dụ như là Expectiminimax, Negamax,.. và so sánh độ hiệu quả của chúng với lại Minimax.

Ngoài ra, thêm một số cải tiến nhất định có thể được thực hiện cho chính bản thân nó là AI trong thực tế, để làm cho AI thông minh hơn. Ví dụ có thể kể đến: khi bật chế độ phòng thủ hoặc tấn công, trong đó AI sẽ thực hiện các bước tấn công hoặc phòng thủ tùy thuộc nó đang rơi vào thế thắng hay là thua. Bên cạnh đó, có thể triển khai nhiều tính năng giao diện người dùng, cải thiện chất lượng về độ dễ tiếp cận, chẳng hạn như là xây dựng GUI để cho phép người chơi có thể dễ dàng nhập hành động của mình, hoặc là nhận một tin nhắn một khi trò chơi kết thúc.

# Conclusion

Kết quả của chúng tôi cho thấy rằng bằng việc gia tăng chiều sâu của thuật toán Minimax có thể nhìn trước được, làm tăng tỉ lệ thắng cho AI, và khi đi trước cũng tăng tỉ lệ cơ hội thắng nhưng không được nhiều cho lắm.

Và chúng ta cũng không quên rằng tầm quan trọng của Alpha-Beta Pruning là biết bao nhiêu, bởi vì hiệu suất sẽ được tăng lên rất đáng kể khi sử dụng Alpha-Beta, làm tăng tốc độ xử lí của thuật toán.

Trong khi làm bài báo cáo này, chúng tôi đã có thể cải thiện thêm được nhiều kiến thức về Python, và đã học hỏi được nhiều chiến lược chiến thắng trong trò chơi Connect 4. Chúng tôi cũng nhận thấy rằng phải cần rất nhiều công sức để thực hiện một bài báo cáo khoa học, và có thể cải thiện được kĩ năng này trong tương lai.

1. Acknowledgment

Chúng tôi muốn cảm ơn Keith Galli từ Youtbe đã giải thích cho chúng tôi hiểu Connect-Four AI hoạt động bằng cách nào và Gilles Vandewile với những thông tin về Connect-Four aI

1. Reference

Thông tin về Connect-Four từ các trang web sau:

<http://en.wikipedia.org/wiki/Connect_Four>

<https://en.wikipedia.org/wiki/Minimax>

<https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning>

Thuật toán Minimax:

<https://en.wikipedia.org/wiki/Minimax>

Thuật toán Alpha-Beta Pruning :

<https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning>

Problem-Solving Methods in Artificial Intelligence by Nils J. Nilson. McGraw-Hill, 1971.

Gilles Vandewiele, SemWeb/ML PhD student @ IDLab

<https://towardsdatascience.com/creating-the-perfect-connect-four-ai-bot-c165115557b0>

1. Biography



**First A. Author, Vũ Minh Quang.** Tác giả đã trở thành sinh viên của Trường Đại học Công nghệ Thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh, năm 2018, hiện là năm thứ 2, KHCL2018.3, Khoa học Máy tính. Tác giả tốt nghiệp từ trường Trung học Tân Phú



**Second A. Author, Nguyễn Đức Quốc**. Tác giả đã trở thành sinh viên của Trường Đại học Công nghệ Thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh, năm 2018, hiện là năm thứ 2, KHCL2018.3, Khoa học Máy tính. Tác giả tốt nghiệp từ trường Trung học Tân Phú