# ALPHA BETA SEARCH

Alpha beta search là một cải tiến tối ưu hóa của thuật toán minimax. Giống với minimax, đây là thuật toán đệ quy tìm kiếm đối kháng với hai người chơi là MAX và MIN, hai người này sẽ đi luân phiên nhau với MAX sẽ là giá trị lớn nhất và MIN sẽ là giá trị nhỏ nhất. Thuật toán lựa chọn nước đi tốt nhất dựa trên các giá trị trên.

Alpha beta search khác với minimax ở việc nó có thể loại bỏ, không cần xét những node và những node con của node đó khi nhánh của node đó thỏa điều kiện để cắt tỉa, loại bỏ. Từ đó làm cho cây tìm kiếm của thuật toán nhỏ hơn so với minimax do không cần phải đi hết tất cả các node của cây.

# Ý TƯỞNG

## Ý tưởng chung

Trong chương trình này, ta sẽ sử dụng thuật toán alpha beta search để tìm kiếm nước đi tốt nhất cho BOT nhằm giúp nó đánh thắng người chơi (con người) ở cả vị trí player 1 (đi trước) lẫn player 2 (đi sau).

Ta sẽ lập trình BOT để thử nghiệm hết tất cả các ô trống (nước đi) của bàn cờ và tìm ra nước đi tốt nhất để thắng người chơi.

Mỗi nước đi của cả 2 player đều được tính là 1 state, mỗi state sẽ tương ứng với một giá trị trả về (thắng – thua – hòa), ta sẽ dùng thuật toán alpha beta search để đệ quy đến state cuối cùng của đối với từng state trên để tìm giá trị của state đó. Việc đệ quy là hàm min gọi max và max gọi min vì vậy MIN MAX tương ứng với 2 player như đã nói ở trên. State cuối cùng là trường hợp mà bàn cờ đã hết ô trống hoặc một trong hai chiến thắng. Lúc này tương ứng với ba trường hợp trên sẽ là ba giá trị trả về khác nhau, và đối với nước đi tốt nhất thì tương ứng với giá trị lớn nhất, nghĩa là BOT thắng.

## Vấn đề

Thuật toán alpha beta search đối với bàn cờ kích thước 3x3 thì hoạt động rất tốt, nhưng khi kích thước của bàn cờ tăng lên đến 5x5 (có khoảng 325 state) và lớn hơn nữa là 7x7 (có khoảng 349 states) thì cây tìm kiếm của thuật toán này sẽ trở nên khổng lồ và rất rất là lớn. Điều này làm cho thời gian search của thuật toán sẽ rất lâu. Vì thế nên thuật toán sẽ có thêm một tham số là depth chỉ chiều sâu của cây mà thuật toán sẽ search tới.

Đối với bàn cờ 3x3 thì depth của cây là không cần thiết nhưng đối với 5x5 và 7x7 thì lại rất quan trọng. Chẳng hạn như 3x3 giả sử player 1 đã đi, tức là đã có 1 ô không còn trống thì tổng số state còn lại chỉ là 38(=6561 state) nhưng nếu 5x5 thì là 524 = 59,604,644,775,390,625) và nếu là 7x7 thì còn lớn hơn thế nữa. Vì vậy việc cài đặt depth để thuật toán search đến một chiều sâu nhất định là rất cần thiết.

Nhưng nếu cài đặt depth quá nhỏ thì thuật toán có thể sẽ không thể hoạt động chính xác do thuật toán chưa thể search đến trạng thái đó, từ đó đưa ra nước đi sai dẫn đến con BOT dễ bị thua. Và ngược lại, nếu depth quá cao thì thời gian BOT nghĩ nước đi sẽ lớn.

Qua vài lần thử nghiệm đánh với bot ở cả 5x5 và 7x7, ta rút ra được một chiều sâu hợp lý cho 2 trường hợp này để thời gian cho nước đi tiếp theo của bot không quá lâu và nhất định phải thắng hoặc chặn được người chơi.

# KHÁI QUÁT CHUNG VỀ CODE

* Chương trình sử dụng pygame để tạo UI
* Toàn bộ bàn cờ sẽ được lưu vào một ma trận 2D và ta chủ yếu sẽ thao tác, xử lý trên ma trận này
* Chương trình có một hàm chính là main\_game() để gọi tất cả các hàm.
* Phần xử lý thuật toán alpha beta search nằm trong các hàm:
  + Alpha\_beta\_search // gọi hàm min\_value
  + Min\_value // tìm giá trị khi MIN của thuật toán
  + Max\_value // tìm giá trị khi MAX của thuật toán
  + Cal\_state\_value // Tính giá trị của state hiện tại
  + isEmptyTable // Trường hợp bàn cờ còn ô trống
  + Best\_move // Tìm nước đi lớn nhất dựa trên thuật toán search
* Các hàm còn lại dùng để cài đặt hiển thị UI của game

# PSEUDOCODE ALPHA BETA PRUNE ĐỐI VỚI TIC TAC TOE

Ta sẽ dùng pseudocode từ slide của tiết Lý Thuyết để cài đặt thuật toán alpha beta search, thuật toán này sẽ có 3 hàm là alpha\_beta\_search, max\_value, min\_value và một hàm nữa là best\_move để gọi hàm alpha\_beta\_search và qui định depth cho các loại bàn cờ (3x3, 5x5, 7x7) và hai hàm phụ là Cal\_state\_value và isEmptyTable

Đối với 2 hàm max\_value và min\_value thì hai hàm này sẽ đệ quy gián tiếp, nghĩa là hàm này gọi hàm kia. Còn hàm alpha\_beta\_search chỉ gọi một hàm duy nhất là min\_value để tìm nước đi tốt nhất.

Hàm Cal\_state\_value tính toán các trường hợp mà player hoặc bot thắng theo từng dòng, từng cột, đường chéo chính và đường chéo phụ và các đường xiên đối với bàn cờ 7x7.

**Function alpha\_beta\_Search(board, depth)**

Return min\_value(board, depth, alpha, beta)

**Function max\_value(board, depth, alpha, beta, tranpositiontable)**

Tính giá trị hiện tại của state, nếu player thắng return 1, nếu BOT thắng return -1, nếu bàn cờ không còn vị trí trống return 0,

Nếu depth = 0 return giá trị hiện tại của state.

Best = -inf

Với mỗi vị trí trống trong bàn cờ

Thử từng vị trí và tìm giá trị tốt nhất (best) best = max(best, min\_value(depth -1))

Nếu best >= beta thì cắt (prune) 🡪 return best

Nếu khác trên, gán alpha = max(alpha, best)

**Function min\_value(board, depth, alpha, beta, tranpositiontable)**

Tính giá trị hiện tại của state, nếu player thắng return 1, nếu BOT thắng return -1, nếu bàn cờ không còn vị trí trống return 0,

Nếu depth = 0 return giá trị hiện tại của state.

Best = inf

Với mỗi vị trí trống trong bàn cờ

Thử từng vị trí và tìm giá trị tốt nhất (best) best = max(best, min\_value(depth -1))

Nếu best <= alpha thì cắt (prune) 🡪 return best

Nếu khác điều kiện trên, gán beta = min(beta, best)

**Function best\_move(board, count)**

gán bestValue = -Inf, bestMove = (-1, -1)

Gán depth theo size của board

Với mỗi ô trống của bàn cờ

Gán ô đang xét với giá trị tương ứng với kí hiệu bot

Dùng thuật toán alpha beta search tìm giá trị tốt nhất (new) cho state hiện tại

Nếu giá trị tốt nhất (mới) > giá trị tốt nhất hiện tại

Gán nước đi tốt nhất là vị trí của ô đang xét

Gán giá trị tốt nhất hiện tại là giá trị tốt nhất (new)

Gán ô tương ứng với giá trị tốt nhất là kí hiệu của bot

# COMPLETENESS

Minimax là một thuật toán search có tính completeness vì cây tìm kiếm của nó dù nhiều nhưng vẫn chỉ có hữu hạn các bước và thuật toán alpha beta search cũng kế thừa tính chất này của minimax.

Thuật toán luôn có thể trả về một kết quả là nước đi tốt nhất nếu nước đi đó tồn tại (nghĩa là ván cờ chưa có player chiến thắng hoặc vẫn còn ô trống)

# Time/Space complexity

## Time complexity

Độ phức tạp về thời gian của thuật toán alpha beta prune:

Worst case: O(bd) trường hợp này về cơ bản là alpha beta search của ta biến thành minimax bình thường. Đây là trường hợp mà không thể prune bất kì nhánh nào, đồng nghĩa với việc duyệt hết tất cả các nhánh của cây tìm kiếm.

Best case: O(bd/2) là trường hợp mà cây của ta có “good ordering” nên có thể dễ dàng prune các nhánh. Điều trên cũng đồng nghĩa với việc, với cùng một lượng thời gian thì alpha beta prune có thể tìm kiếm đến đến depth sâu hơn so với minimax và nếu cùng depth thì alpha beta prune sẽ nhanh hơn so với minimax.

## Space complexity

Space complexity của thuật toán alpha beta search là O(bd) tương ứng với b là branching factor và d là max depth của thuật toán.

# References