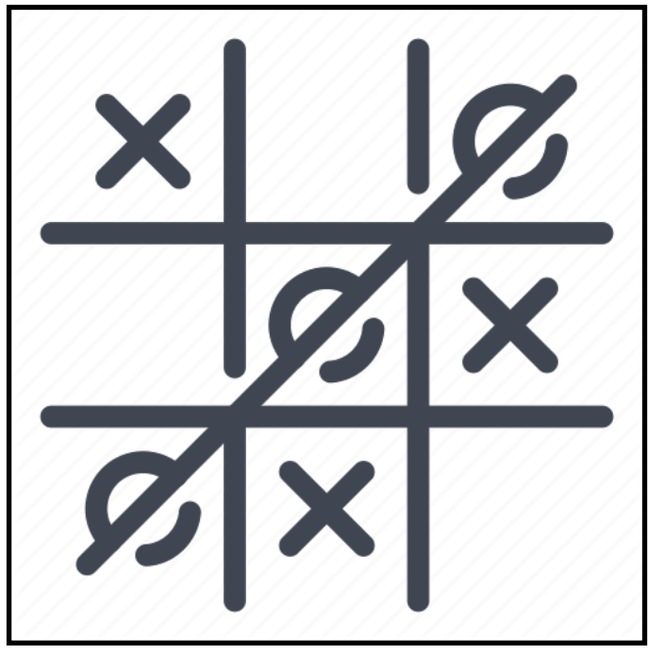
ALPHA BETA ALGORITHM
CARO - GAME

Người thực hiện: Trần Ngọc Bảo Duy - 51702091

1. MinMaxSearch Algorithm

1. Khái niệm

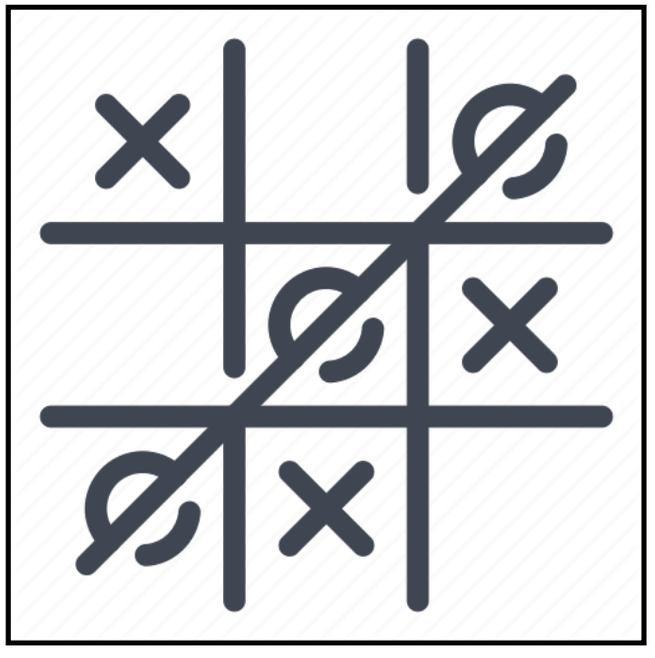
- Minimax search là một thuật toán nhằm tối thiểu hoá các tổn thất (giá trị mất mát), hoặc có thể hiểu là tối đa hoá lợi ích (giá trị tốt) trong các nước đi được tính toán trước
- Giải thuật Minimax giúp tìm ra nước đi tốt nhất, bằng cách đi ngược từ cuối trò chơi trở về đầu. Tại mỗi bước, nó sẽ ước 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 như trò tic tac toe, với mỗi bước đi player sẽ tính toán các bước đi để tối đa cơ hội thắng của mình và tối thiểu cơ hội thắng cuả đối phương



1. MinMaxSearch Algorithm

2. Áp dụng

- Ta sẽ cần một hàm evaluate (thường là hàm đệ quy) để đánh giá số điểm của 1 nước đi cho trước bới 1 trạng thái nhất định của người chơi
- Sau khi evaluate, ta sẽ chọn nước đi có cơ hội thắng và cơ hội để đối thủ thua cao nhất. Hàm này sẽ tự tạo ra các trạng thái kế tiếp của cả người chơi và đối thủ để tính toán nước đi mà đối thủ có thể chọn.
- Nếu các bước evaluate tiến dần về vô hạn đối với mỗi nước đi kế tiếp, thì người chơi có thể chọn ngẫu nhiên các nước đi tiên phong, tránh trường hợp vô cực.
- Ví dụ như trò tic tac toe, nếu không có nước đi trên bàn cờ người chơi có thể chọn ngẫu nhiên trước khi thuật toán evaluate được kích hoạt



1. MinMaxSearch Algorithm

3. Mã giả

```
MinMaxSearch(player)
        { // player là nút muốn tính điểm
        If  depth is 0: (độ sâu mà chúng ta muốn tính)
                return score of  player
        else:
                If  player in a MIN_PLAYER
                        For all_next_move of  player: v1,…,vn
                        Return   min { MinMaxSearch (v1),…, MinMaxSearch (vn)}
                Else (player is MAX_PLAYER)
                        For all_next_move of player: v1,…,vn
                        Return   max { MinMaxSearch (v1),…, MinMaxSearch (vn)}
}
```

2. Alpha beta Algorithm

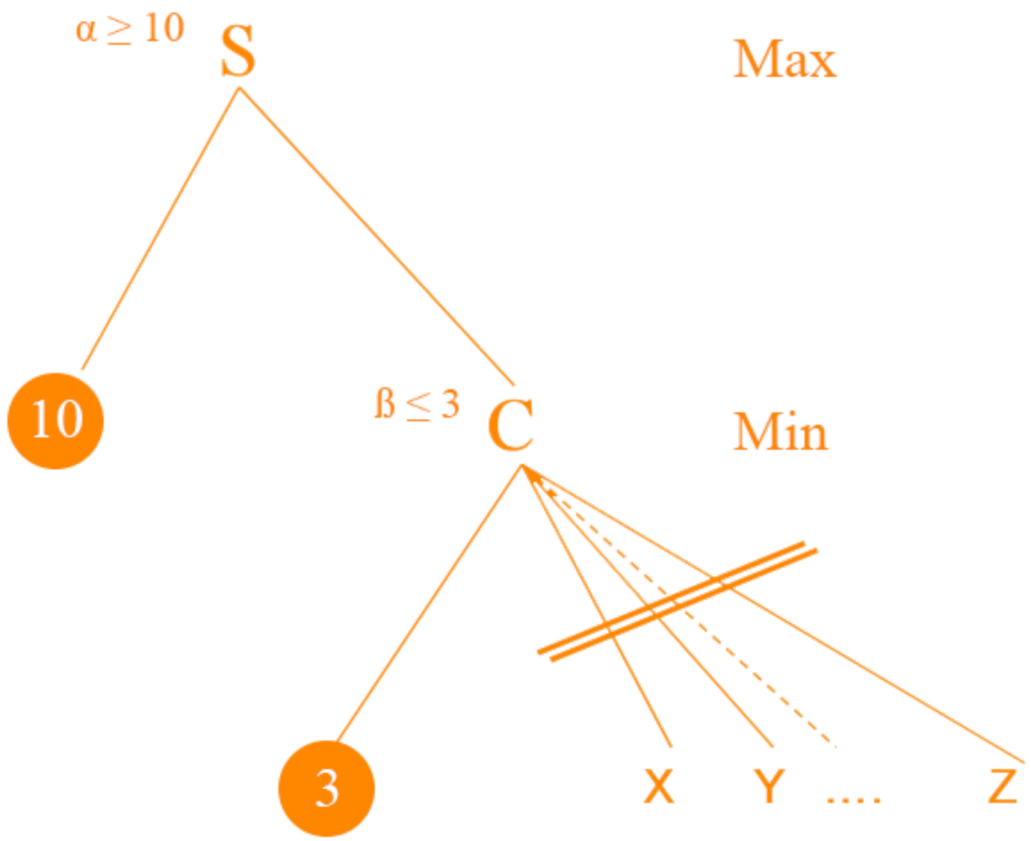
1. Khái niệm

- Alpha beta Algorithm là một thuật toán nâng cấp của Minmaxsearch thay vì chúng ta phải tìm tất cả các nước đi trên một bàn cờ cho đến độ sâu mà chúng ta muốn, điều này sẽ dẫn đến overload RAM vì không thể nào đệ quy hết được tất cả các nước đi, ở độ sâu 2 các nước đi đã dần chậm chạp so với các nước ban đầu, nếu tính lên độ sâu 5,6 thậm chí là 10 máy tính chúng ta sẽ không chạy được hoặc thời gian chờ đợi sẽ rất lâu, vì vậy thuật toán alpha beta đã xuất hiện để giải quyết vấn đề này
- Chúng ta đều biết rằng khi thuật toán minmax nó sẽ tìm ra được các ngưỡng trên và ngưỡng dưới của nước đi tuỳ theo người chơi là min hay max.
- Chính điều này có thể giúp chúng ta chặn trước để có thể làm giảm các tính toán của máy tính, ta sẽ dùng 2 biến ngưỡng trên và ngưỡng dưới [alpha, beta] tại mỗi node để tính toán và kiểm tra có nên đi evaluate tiếp hướng đi (tree structure) đó không hay quay lại.

2. Alpha beta Algorithm

2. Áp dụng

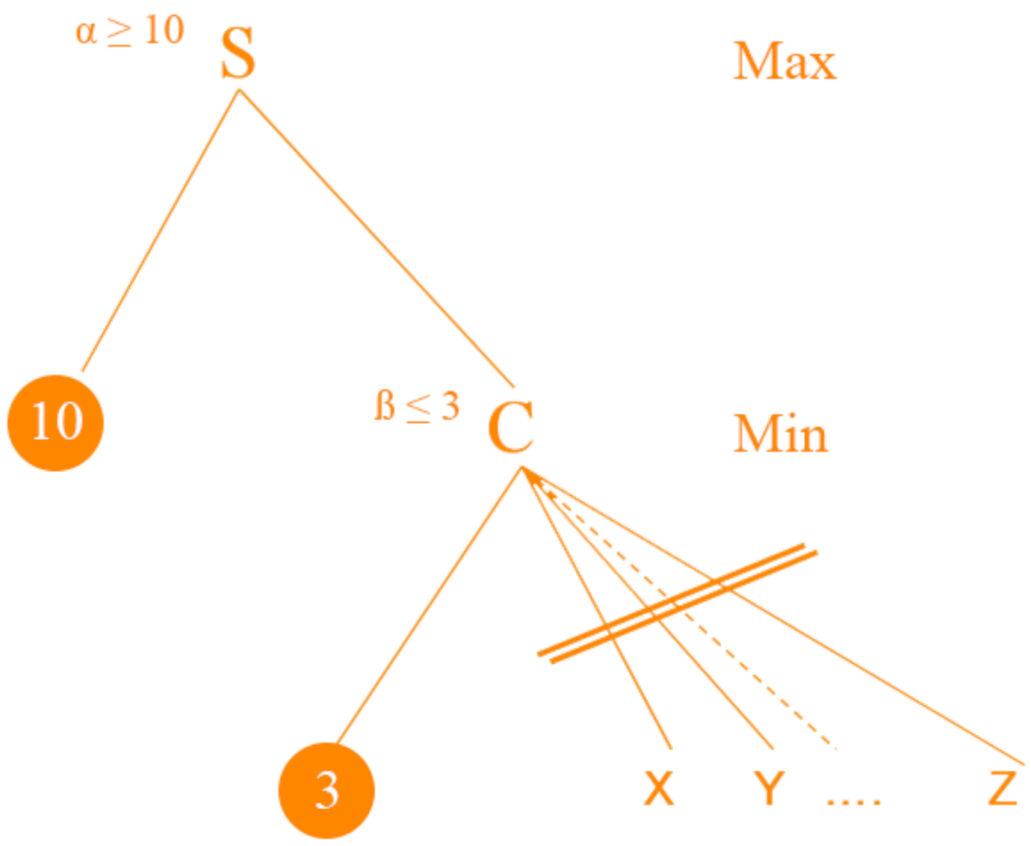
- Ta thấy người chơi cần đánh giá là Max cho nên các node kế tiếp phải lấy giá trị max
- Đi xuống node đầu tiên giá trị là 10, vì người chơi mà Max nên ta đặt lại khoảng [alpha, beta] là [10, beta]
- Tiếp tục đi node kế tiếp, vì node kế tiếp có con nên chúng ta phải tính tiếp, xuống tới node 3 vì đây là người chơi min nên chúng ta sẽ đặt lại khoảng [alpha, beta] là [alpha, 3]



2. Alpha beta Algorithm

2. Áp dụng

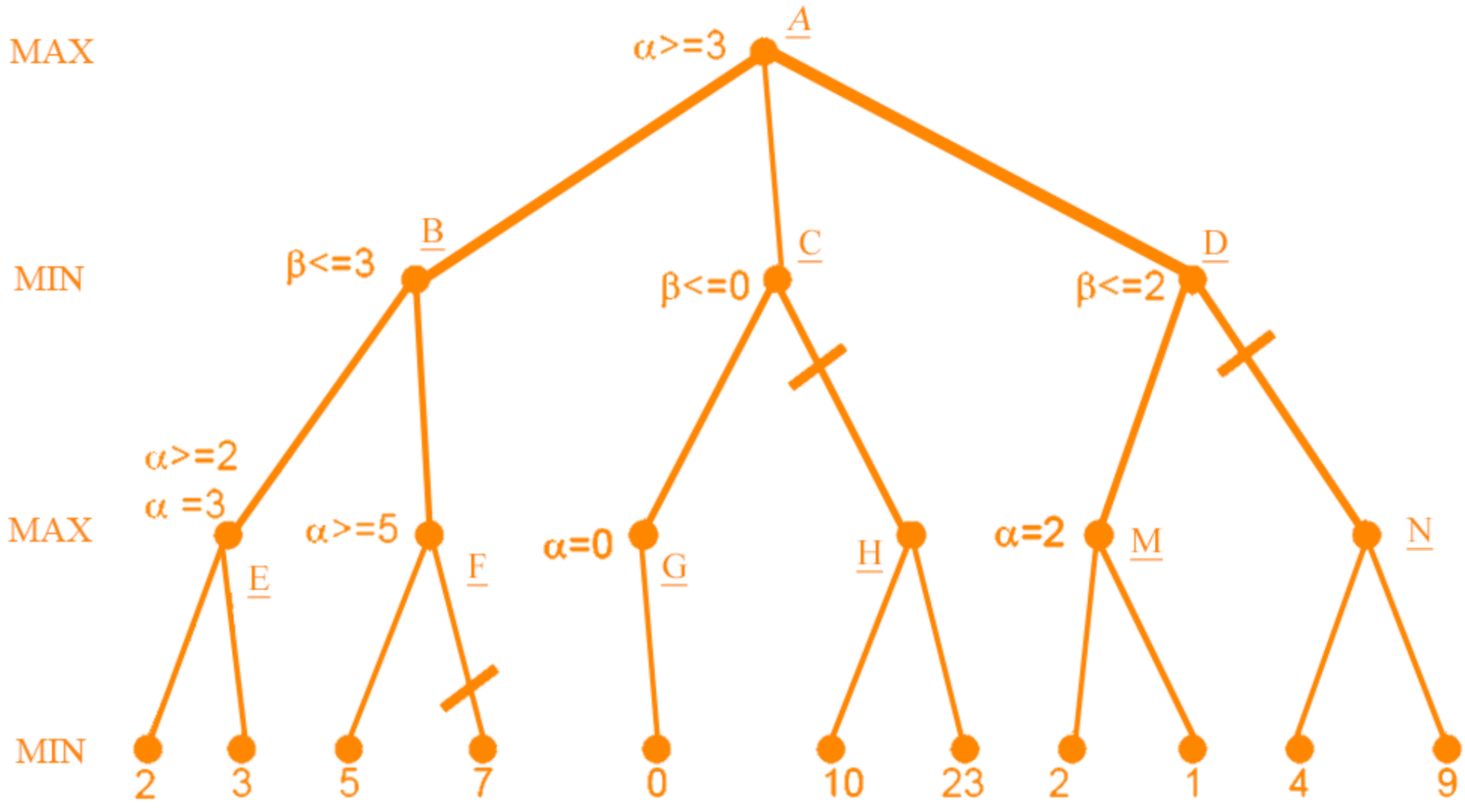
- Sau khi xác định được alpha và beta, chúng ta có thể dễ dàng xác định việc có cắt tỉa hay không. Ở nút S (Max), giá trị alpha luôn ≥ 10 (luôn tăng) nhưng ở C (Min) thì giá trị luôn luôn ≤ 3 (luôn giảm), nên việc xét các con còn lại ở C là không cần thiết. Vì ở S chúng ta sẽ nhận giá trị Max luôn phải lớn hơn bằng 10 cho nên giá trị nhỏ hơn bằng 3 kia không có ý nghĩa nữa



2. Alpha beta Algorithm

2. Áp dụng

- Ví dụ khác về cây sử dụng alpha beta để cắt nhánh, tránh tính trạng quá tải trong các nút



2. Alpha beta Algorithm

3. Mã giả

```
if depth = 0 then
        AlphaBeta = Eval  // Tính điểm tại vị trí node lá
else
        {
           best = -INFINITY; // cho best là âm vô cực để chặn các giá trị trên
           Generate;  //Sinh ra các node con có thể đi
                  while (còn lấy được node để đi) and (best < beta) do
                  {
                        if best > alpha then:
                              alpha = best;
                              đi node con kế tiếp để tính value dự đoán
                              value = -AlphaBeta(-beta, -alpha, depth-1);
                              lấy lại node ban đầu, lúc chưa đi node kế tiếp để tính value ngay tại
                              node đó
                       if value > best then:
                              best = value;
                  }
           AlphaBeta = best;
        }
}
```

3. Reference

Giải Thuật Cắt Tỉa Alpha-beta

https://www.stdio.vn/articles/giai-thuat-cat-tia-alpha-beta-564

Giải Thuật Minimax Search

https://vi.wikipedia.org/wiki/Minimax#Ti%C3%AAu_chu%E1%BA%A9n_Minimax_trong_l%C3%BD_thuy%E1%BA%BFt_quy%E1%BA%BFt_%C4%91%E1%BB%8Bnh_th%E1%BB%91ng_k%C3%AA