ADVERSARIAL SEARCH (Tìm kiếm đối kháng)

Một trò chơi có thể được định nghĩa là một loại tìm kiếm trong AI, có thể được hình thức hóa các yếu tố sau:

- Initial state: xác định cách trò chơi được thiết lập khi bắt đầu.
- Player(s): xác định người chơi nào đã di chuyển trong không gian trạng thái.
- Action(s): trả về tập hợp các bước di chuyển hợp pháp trong không gian trạng thái.
- Result(s, a): đây là mô hình chuyển tiếp, chỉ định kết quả của các di chuyển trong không gian trạng thái.
- Terminal-Test (s): thử nghiệm cuối là đúng nếu trò chơi kết thúc, nếu không thì nó là sai trong mọi trường hợp. Trạng thái nơi trò chơi kết thúc được gọi là trạng thái cuối.
- Utility (s, p): Hàm tiện ích trả về giá trị số cuối cùng cho trò chơi kết thúc ở trạng thái cuối s cho người chơi p. Nó cũng được gọi là hàm hoàn trả. Đối với Cờ vua, kết quả là thắng, thua hoặc hòa và giá trị hoàn trả của nó là +1, 0, $\frac{1}{2}$ và đối với tic-tac-toe, các giá trị tiện ích là +1, -1 và 0.

Trong một cây trò chơi, chiến lược tối ưu có thể được xác định từ giá trị minimax của mỗi nút, có thể được viết là MINIMAX (n). MAX chuyển sang trạng thái giá trị tối đa và MIN chuyển sang trạng thái giá trị tối thiểu sau đó:

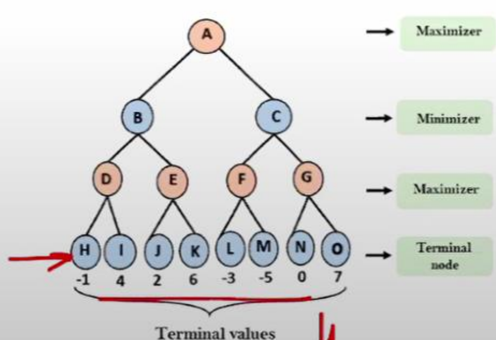
Cho một trạng thái s:

$$MINIMAX(s) = \begin{cases} UTILITY(s) & \text{if } TERMINAL-TEST(s) \\ \max_{a \in Action(s)} MINIMAX(RESULT(s,a)) & \text{if } PLAYER(s) = MAX \\ \min_{a \in Action(s)} MINIMAX(RESULT(s,a)) & \text{if } PLAYER(s) = MIN \end{cases}$$

Thuật toán Mini-Max trong AI

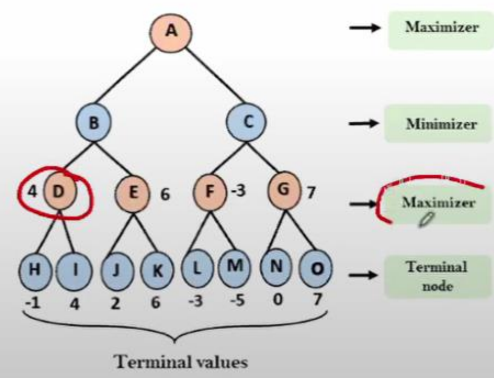
Bước 1:

- Trong bước đầu tiên, thuật toán tạo toàn bộ cây trò chơi và áp dụng hàm tiện ích để lấy các giá trị tiện ích cho các trạng thái đầu cuối.
- Trong sơ đồ cây bên dưới, hãy coi A là trạng thái ban đầu của cây.
- Giả sử Maximizer thực hiện lượt đầu tiên có giá trị ban đầu trong trường hợp xấu nhất = - ∞ và Minimizer sẽ thực hiện lượt tiếp theo có giá trị ban đầu trong trường hợp xấu nhất = + ∞



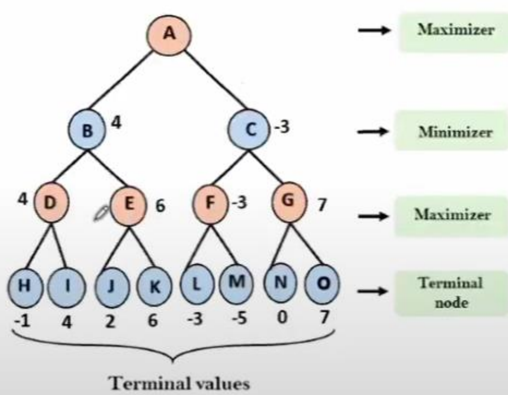
Bước 2: Bây giờ, đầu tiên chúng ta tìm giá trị tiện ích cho Maximizer, giá trị ban đầu của nó là -∞, vì vậy chúng ta sẽ so sánh từng giá trị ở trạng thái cuối với giá trị ban đầu của Maximizer và xác định các giá trị nút cao hơn. Sẽ tìm thấy giá trị lớn nhất trong số đó:

- Đối với nút D max (-1, - ∞) ⇒ max (-1, 4) = 4
- Đối với nút E max (2, - ∞) ⇒ max (2, 6) = 6
- Đối với nút F max (-3, - ∞) ⇒ max (-3, -5) = -3
- Đối với nút G max (0, - ∞) = max (0, 7) = 7



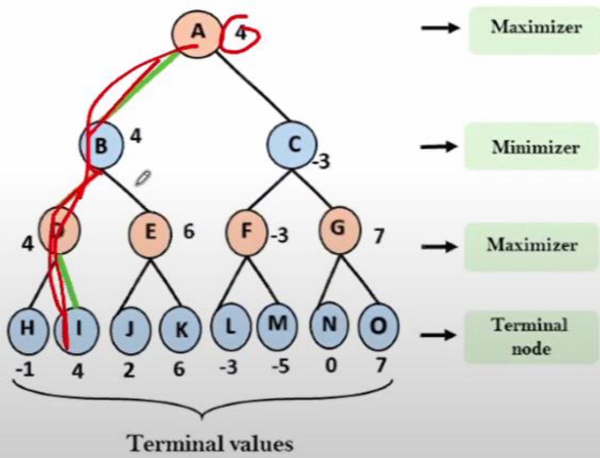
Bước 3: Trong bước tiếp theo, đến lượt Minimize, do đó, nó sẽ so sánh tất cả các giá trị nút với + ∞ và sẽ tìm thấy các giá trị nút của lớp thứ ba.

- Đối với nút B = min (4, 6) = 4
- Đối với nút C = min (-3, 7) = -3



Bước 4: Bây giờ đến lượt Maximizer và một lần nữa nó sẽ chọn tối đa của tất cả các giá trị nút và tìm giá trị tối đa cho nút gốc. Trong cây trò chơi này, chỉ có 4 lớp, do đó chúng ta tiếp cận ngay với nút gốc, nhưng trong các trò chơi thực tế, sẽ có hơn 4 lớp.

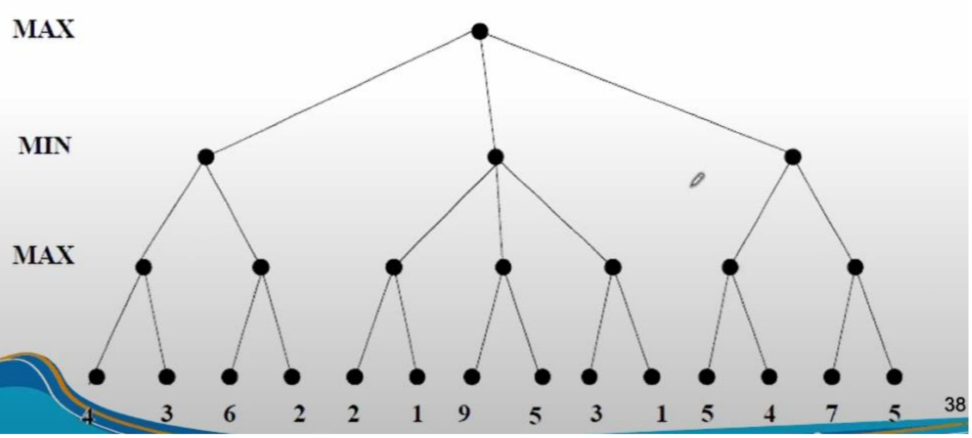
- Đối với nút A max (4, -3) = 4

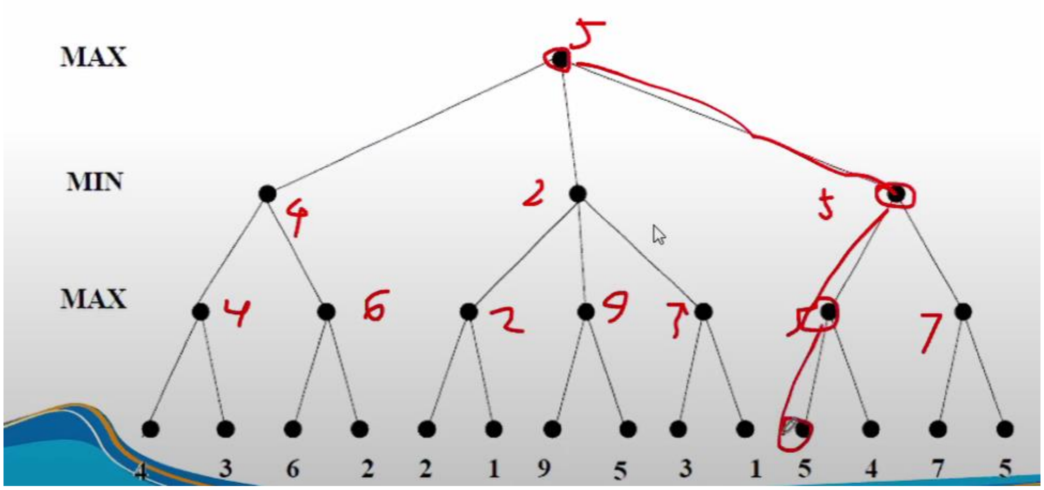


<u>Ví dụ minh họa 1:</u>

Cho cây trò chơi như bên dưới.

- Hãy tính giá trị tiện ích cho nước đi còn lại.
- Để đạt kết quả tốt nhất, MAX và MIN sẽ lần lượt đi những nước nào?





Cắt tỉa Alpha – Beta

Hai tham số có thể được định nghĩa:

- **Alpha**: Sự lựa chọn tốt nhất (giá trị cao nhất) mà chúng ta đã tìm thấy cho đến nay, tại bất kỳ điểm nào trên con đường của Maximizer. Giá trị ban đầu của alpha là -∞.
- **Beta**: Sự lựa chọn tốt nhất (giá trị thấp nhất) mà chúng ta đã tìm thấy cho đến nay, tại bất kỳ điểm nào trên con đường của Minimizer. Giá trị ban đầu của beta là + ∞.

Việc cắt tỉa Alpha-beta theo thuật toán Mini-Max chuẩn trả về kết quả giống như thuật toán tiêu chuẩn, nhưng nó loại bỏ tất cả các nút không thực sự ảnh hưởng đến quyết định cuối cùng nhưng làm cho thuật toán chậm. Do đó bằng cách cắt tỉa các nút này, sẽ làm cho thuật toán nhanh hơn

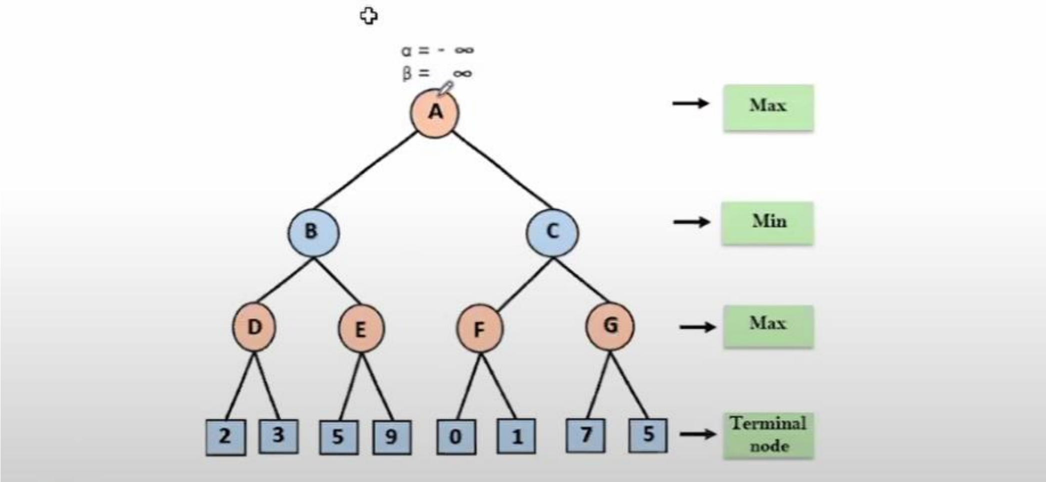
Điều kiện để cắt tỉa Alpha-beta:

- Điều kiện chính cần thiết cho việc cắt tỉa alpha-beta là: $\alpha \geq \beta$

Những điểm chính về cắt tỉa alpha-beta:

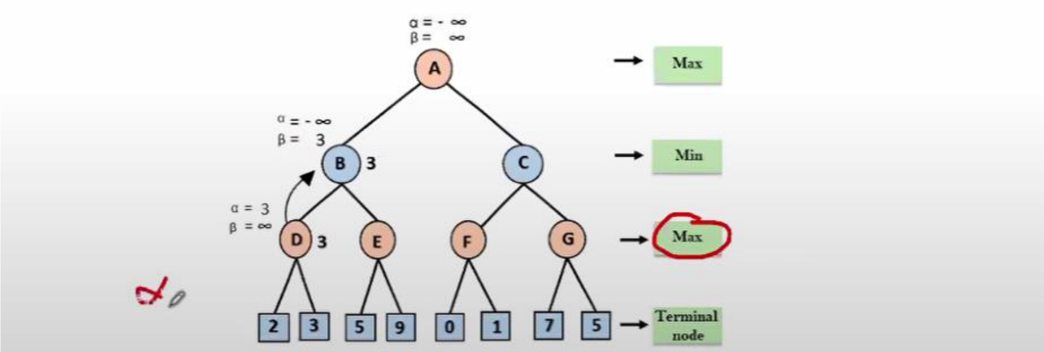
- Người chơi Max sẽ chỉ cập nhật giá trị của alpha.
- Người chơi Min sẽ chỉ cập nhật giá trị của beta.
- Trong khi quay lại cây, các giá trị nút sẽ được chuyển đến các nút trên thay vì giá trị của alpha và beta.
- Chúng ta sẽ chỉ chuyển các giá trị alpha, beta cho các nút con.

Bước 1: Ở bước đầu tiên, người chơi Max sẽ bắt đầu di chuyển đầu tiên từ nút A trong đó α = -∞ và β = +∞, các giá trị này của alpha và beta được truyền xuống nút B trong đó một lần nữa α = -∞ và β = +∞ và nút B truyền cùng giá trị cho con của nó D.



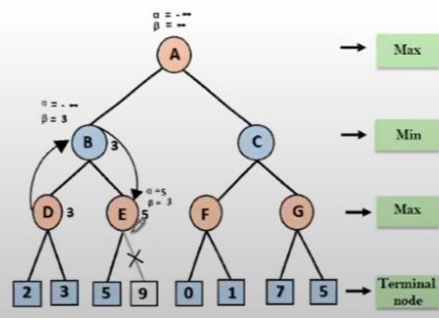
Bước 2: Tại Node D, giá trị của α sẽ được tính là lần lượt cho Max. Giá trị của α được so sánh với đầu tiên là 2 và sau đó là 3 và giá trị tối đa (2, 3) = 3 sẽ là giá trị của α tại nút D và giá trị nút cũng sẽ là 3.

Bước 3: Bây giờ thuật toán quay lại nút B, trong đó giá trị của β sẽ thay đổi vì đây là lượt của Min, Bây giờ β = +∞, sẽ so sánh với giá trị các nút tiếp theo có sẵn, tức là min (∞, 3) = 3, do đó tại nút B bây giờ là α = - ∞ và β = 3.



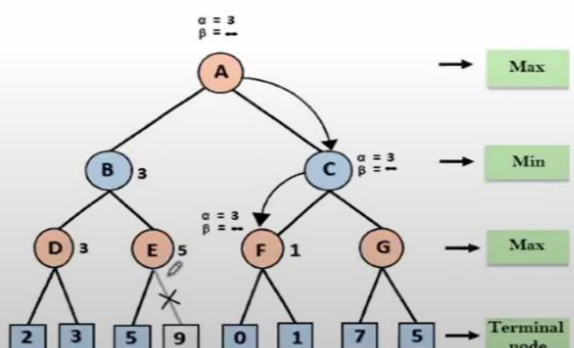
☐ Trong bước tiếp theo, thuật toán đi qua nút tiếp theo của nút B là nút E và các giá trị của α = -∞ và β= 3 cũng sẽ được thông qua.

- **Bước 4:** Tại nút E, Max sẽ thực hiện lần lượt và giá trị của alpha sẽ thay đổi. Giá trị hiện tại của alpha sẽ được so sánh với 5, do đó max (-∞, 5) = 5, khi đó tại nút E: α = 5 và β = 3, trong đó α ≥ β, do đó nút tiếp theo của E sẽ được cắt tỉa, thuật toán sẽ không đi qua nó và giá trị tại nút E sẽ là 5.

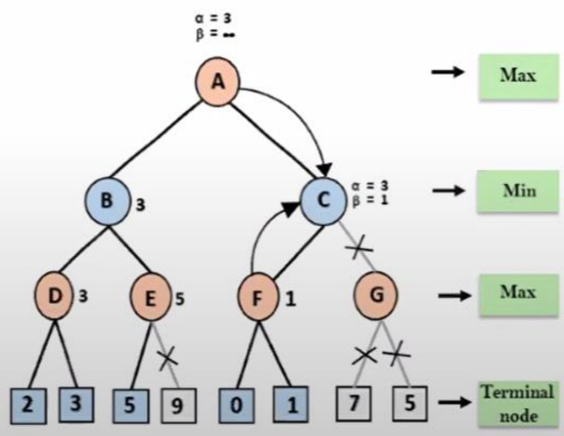


Bước 5: Ở bước tiếp theo, thuật toán lại quay lại cây, từ nút B đến nút A. Tại nút A, giá trị của alpha sẽ được thay đổi giá trị lớn nhất có sẵn là 3 là max (-∞, 3) = 3 và β= +∞, hai giá trị này bây giờ chuyển sang bên phải của A là nut C. Tại nút C, α = 3 và β = +∞ và các giá trị tương tự sẽ được truyền cho nút F.

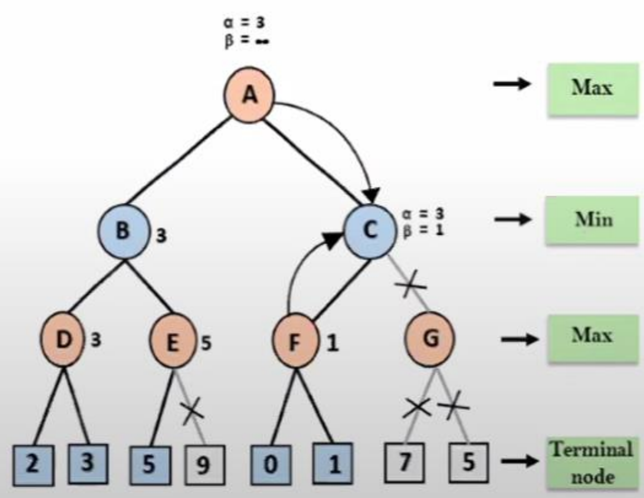
Bước 6: Tại nút F, một lần nữa giá trị của α sẽ được so sánh với con trái là 0 và max (3.0) = 3 và sau đó so sánh với con phải là 1 và max (3,1) = 3 vẫn là 3, nhưng giá trị nút của F sẽ trở thành 1.



Bước 7: Nút F trả về giá trị nút 1 cho nút C, tại C: α = 3 và β= +∞, ở đây giá trị của beta sẽ được thay đổi, nó sẽ so sánh với 1 nên min (+∞, 1) = 1. Bây giờ tại C, α = 3 và β = 1 và một lần nữa, nó thỏa mãn điều kiện α ≥ β, vì vậy nút con tiếp theo của C là G sẽ được cắt tỉa và thuật toán sẽ không tính toán toàn bộ cây con G.



Bước 7: Nút F trả về giá trị nút 1 cho nút C, tại C: α = 3 và β= +∞, ở đây giá trị của beta sẽ được thay đổi, nó sẽ so sánh với 1 nên min (+∞, 1) = 1. Bây giờ tại C, α = 3 và β = 1 và một lần nữa, nó thỏa mãn điều kiện α ≥ β, vì vậy nút con tiếp theo của C là G sẽ được cắt tỉa và thuật toán sẽ không tính toán toàn bộ cây con G.



<u>Ví dụ minh họa 1:</u>

Cho cây trò chơi như bên dưới. Hãy tỉa các nhánh không cần duyệt.