



Ayudantía 6

Búsqueda adversaria

Por Blanca Romero y Daniel Toribio

29 de septiembre 2023



Antes de comenzar...

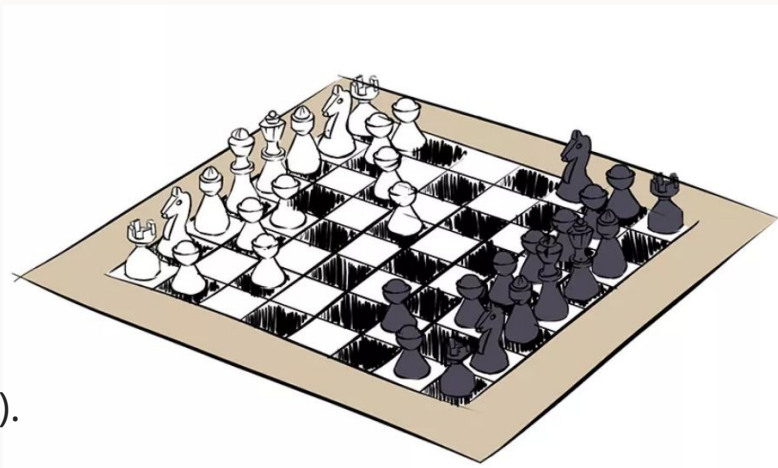
Menti!



Tipos de problemas (Juegos)

Veremos juegos de **dos jugadores**, con **turnos**, **información perfecta** y **suma cero**.

- **Información perfecta:** Tenemos toda la información del tablero (estado) del juego.
- **Suma cero:** Lo que es bueno para mí, es malo para mi oponente en igual medida (y vice versa).





Búsqueda

Podemos considerar un juego como un **espacio de búsqueda**:

- **Tablero** = nodo/estado
- **Jugadas** = conexiones/acciones

El **objetivo** será un **tablero donde ganamos** (estado objetivo):

- Gato: Tres fichas (nuestras) en línea
- Conecta-4: Cuatro fichas (nuestras) en línea
- Ajedrez: Jaque-mate al rey del oponente



Adversario

Para cada estado que revisemos en la búsqueda, tenemos que considerar si es **mi turno, o el turno de mi adversario.**

La búsqueda puede presentarse como una simulación del juego entre dos jugadores: Max y Min:

Si es mi turno: **Max**

- Elijo la mejor jugada que tengo disponible, para **max**imizar el puntaje

Si es el turno de mi oponente: **Min**

- Asumo que mi oponente siempre toma las decisiones óptimas (para él) entonces elige la opción que **min**imiza mi puntaje (la mejor jugada para mi oponente es la que me “hace peor”).



Adversario

Observación

Yo **no sé** cómo va a jugar mi adversario, pero si **asumo que es un genio** (siempre juega lo que es mejor para él/peor para mí) estaré preparada para lo peor.

Luego si mi adversario toma decisiones subóptimas durante el juego, sólo es mejor para mí.



¿Cómo defino la mejor jugada desde un tablero?

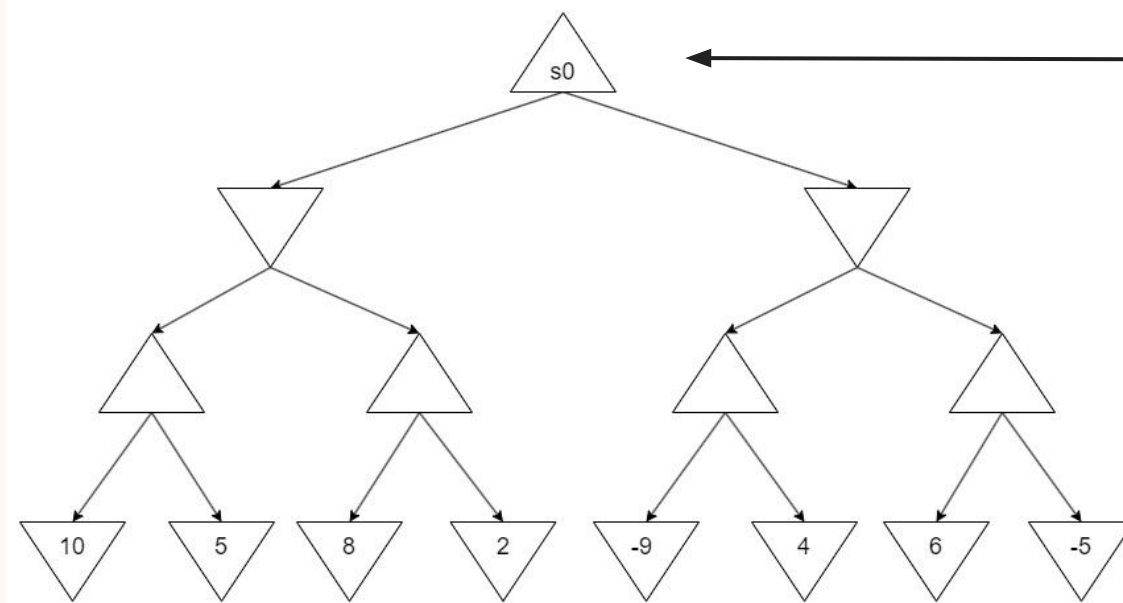
Puntuamos el tablero en base al valor *minimax*.

Valor del tablero para Max

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MIN} \end{cases}$$

Para jugar de manera óptima, elegimos la acción asociada al valor *minimax*

Cálculo del valor MiniMax



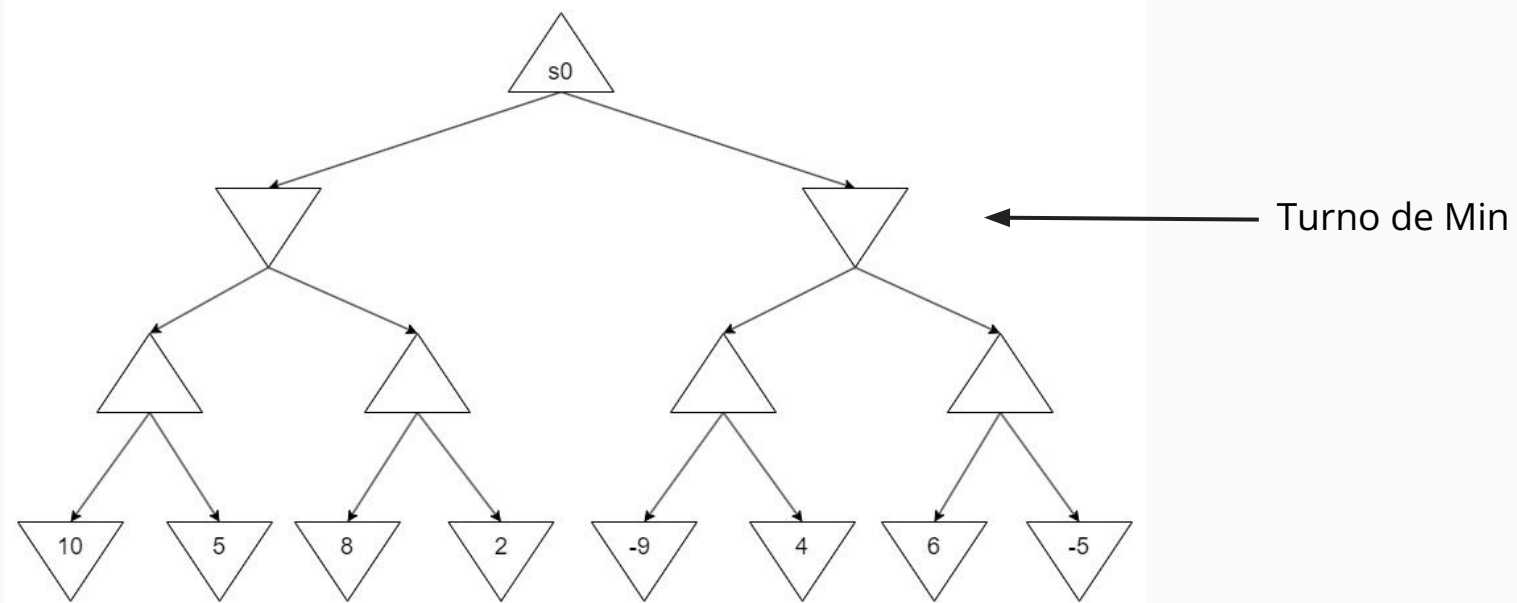
Turno de Max

$\text{MINIMAX}(s) =$

$$\begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MIN} \end{cases}$$



Cálculo del valor MiniMax

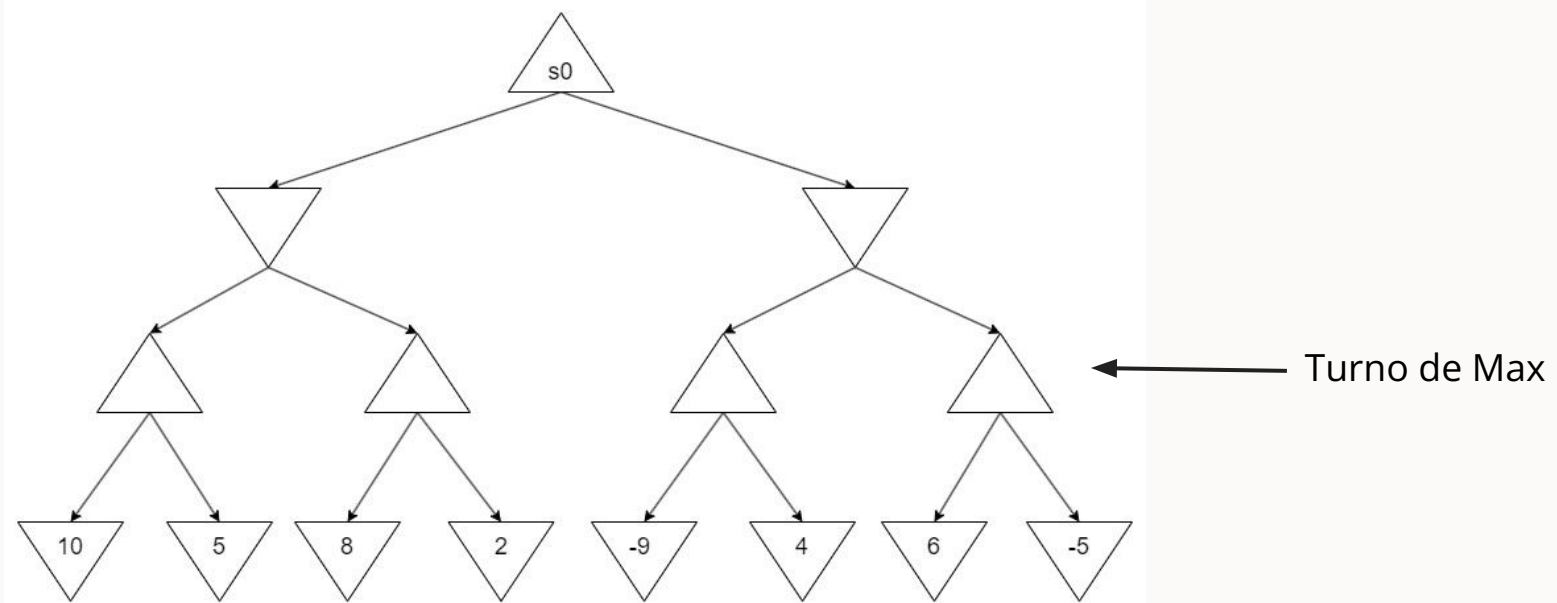


$\text{MINIMAX}(s) =$

$$\begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MIN} \end{cases}$$



Cálculo del valor MiniMax

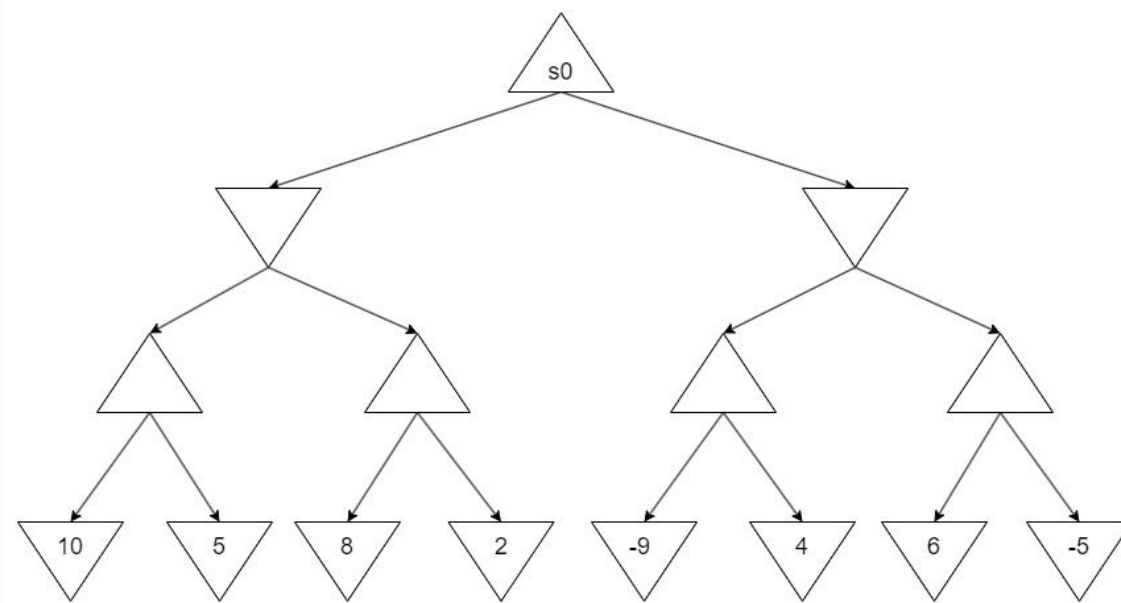


$\text{MINIMAX}(s) =$

$$\begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MIN} \end{cases}$$



Cálculo del valor MiniMax

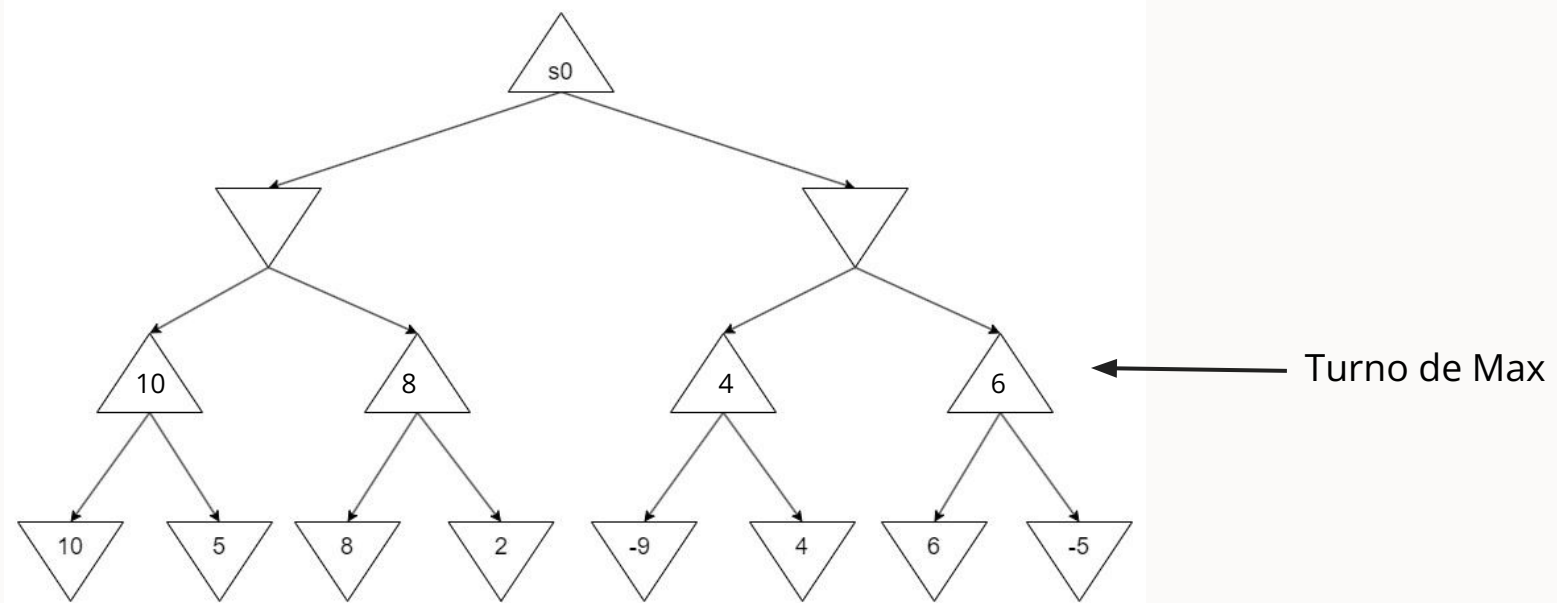


Estado terminal

$\text{MINIMAX}(s) =$

$$\begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MIN} \end{cases}$$

Cálculo del valor MiniMax

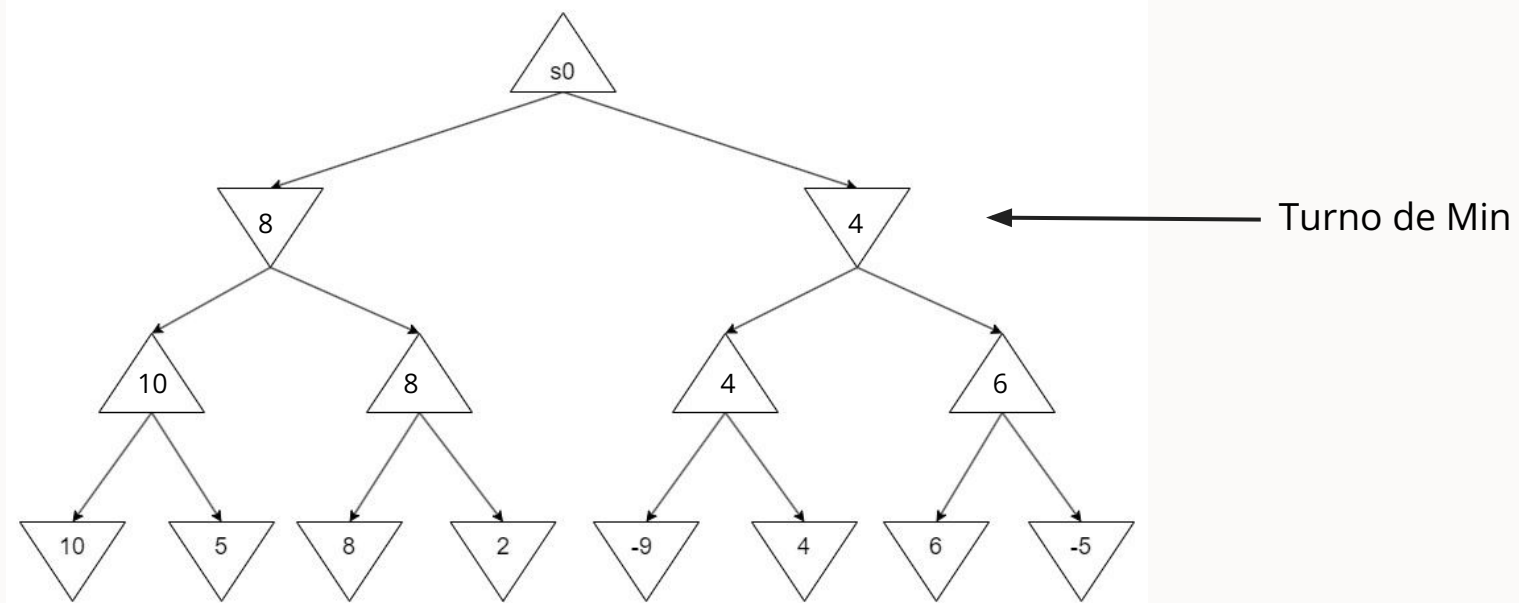


$\text{MINIMAX}(s) =$

$$\begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MIN} \end{cases}$$



Cálculo del valor MiniMax

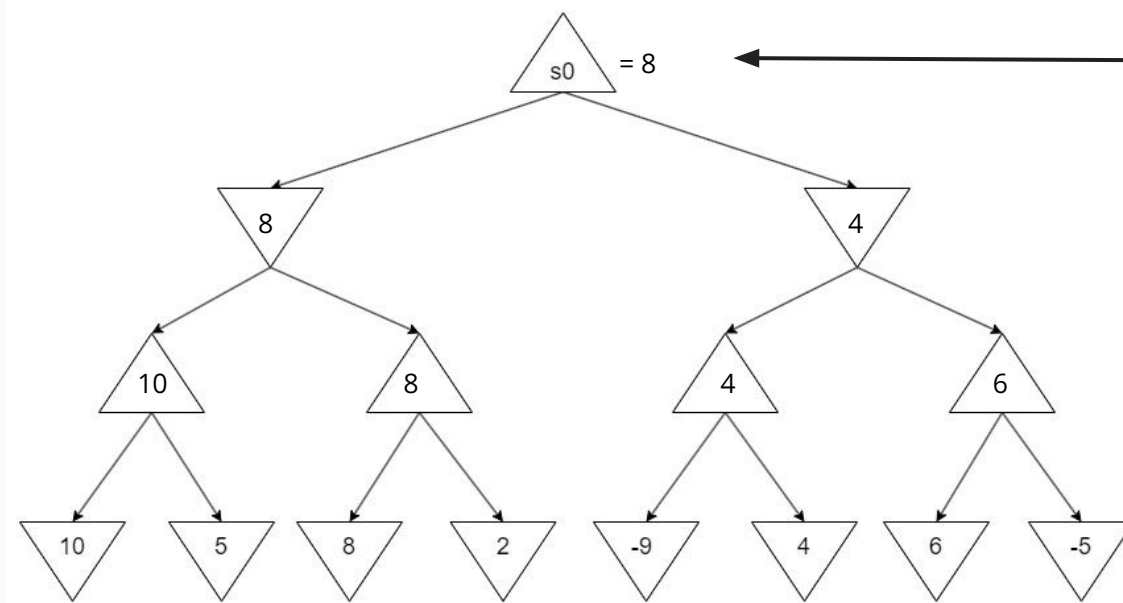


$\text{MINIMAX}(s) =$

$$\begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MIN} \end{cases}$$



Cálculo del valor MiniMax



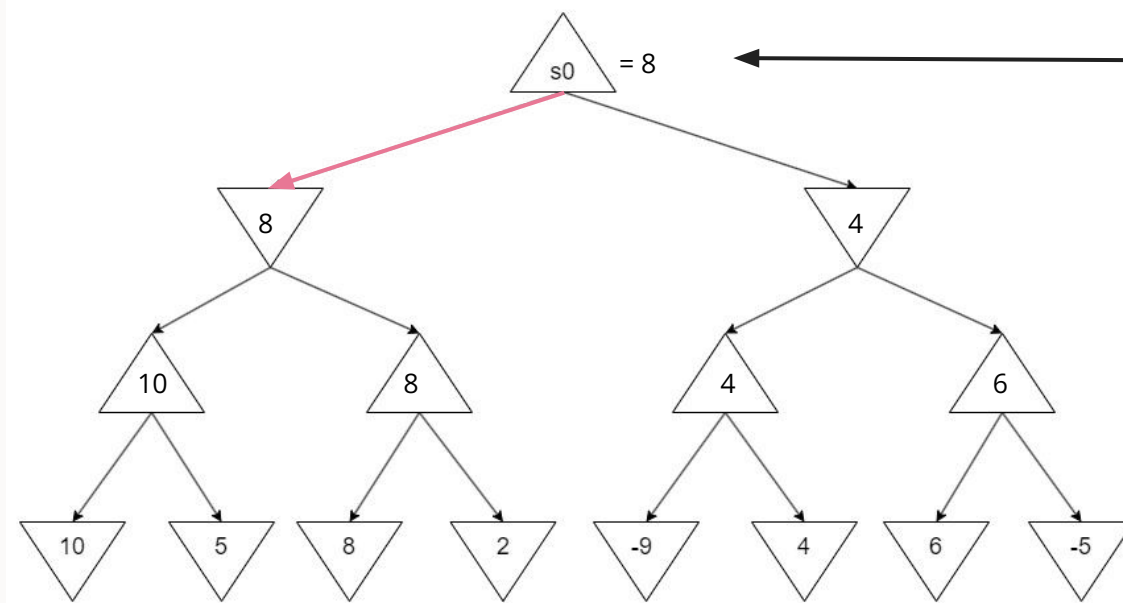
Turno de Max
(retorno de MiniMax)

$\text{MINIMAX}(s) =$

$$\begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MIN} \end{cases}$$



Cálculo del valor MiniMax



Elijo la acción que me lleva al estado asociado al 8 (hijo izquierdo)

$\text{MINIMAX}(s) =$

$$\begin{cases} \text{UTILITY}(s, \text{MAX}) & \text{if IS-TERMINAL}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if TO-MOVE}(s) = \text{MIN} \end{cases}$$

Búsqueda MiniMax



Es un algoritmo que recibe un tablero y retorna la acción asociada al valor minimax de dicho tablero

```
function MAX-VALUE(game, state) returns a (utility, move) pair
  if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
  v, move  $\leftarrow -\infty$ 
  for each a in game.ACTIONS(state) do
    v2, a2  $\leftarrow$  MIN-VALUE(game, game.RESULT(state, a))
    if v2 > v then
      v, move  $\leftarrow$  v2, a
  return v, move
```

```
function MIN-VALUE(game, state) returns a (utility, move) pair
  if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
  v, move  $\leftarrow +\infty$ 
  for each a in game.ACTIONS(state) do
    v2, a2  $\leftarrow$  MAX-VALUE(game, game.RESULT(state, a))
    if v2 < v then
      v, move  $\leftarrow$  v2, a
  return v, move
```




Menti!



Rendimiento MiniMax

La búsqueda MiniMax corre DFS **sobre el árbol de estados completo:**

→ Para el ajedrez, esto es explorar 10^{44} estados

Para mejorar el rendimiento podemos:

- **Acotar** la profundidad del árbol de búsqueda
- **Podar** ramas del árbol de búsqueda



Acotar profundidad del árbol de búsqueda

Función de evaluación

- Es como una **heurística** pero en el contexto de búsqueda adversaria
- **Asigna puntajes a estados no terminales**, lo que permite limitar a un máximo la altura de un árbol

→ Básicamente, hace una estimación sobre “quién va ganando”



Poda Alpha-Beta

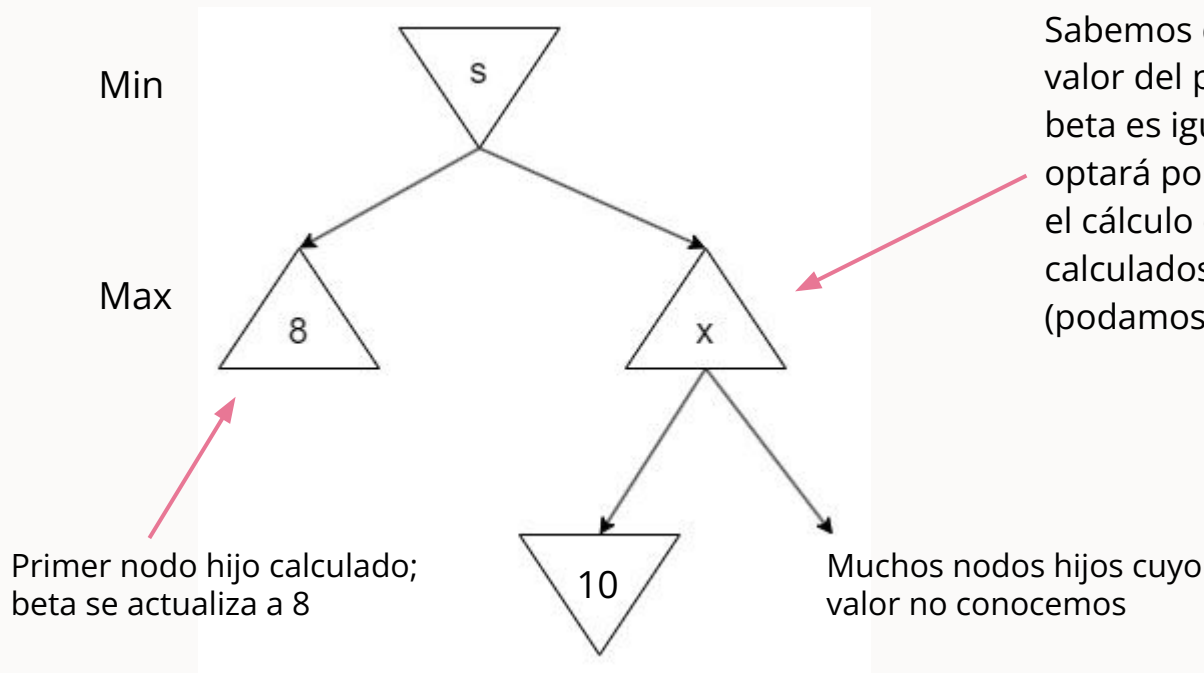
Podemos podar buena parte del árbol si guardamos dos parámetros adicionales en cada llamada a la búsqueda MiniMax:

- Alpha: cota **inferior** del valor de un nodo
- Beta: cota **superior** del valor de un nodo

Tener estas cotas nos indica si es necesario calcular un nodo.



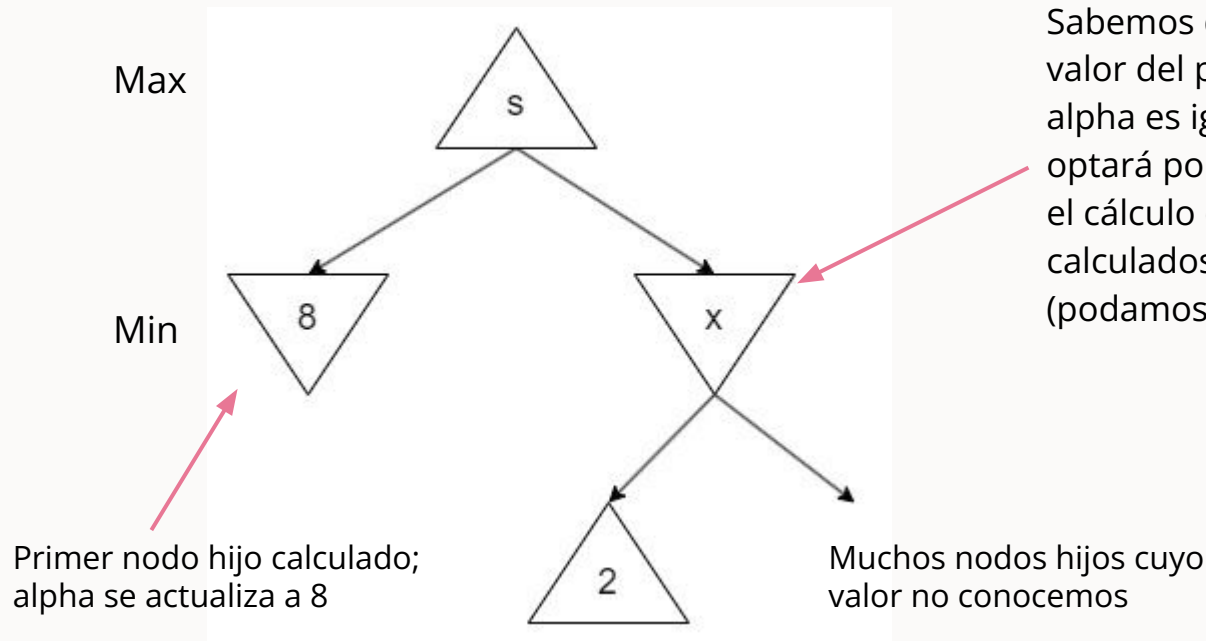
Poda Alpha-Beta - Cota superior Beta



Sabemos que $x \geq 10$ apenas verificamos el valor del primer nodo hijo. Como nuestro beta es igual a 8, sabemos que el nodo **s** **no** optará por el valor x . Luego podemos omitir el cálculo de los nodos hijos aún no calculados y llegar a la misma elección (podamos el sub-árbol)



Poda Alpha-Beta - Cota inferior Alpha



Sabemos que $x \leq 2$ apenas verificamos el valor del primer nodo hijo. Como nuestro alpha es igual a 8, sabemos que el nodo **s** **no** optará por el valor x . Luego podemos omitir el cálculo de los nodos hijos aún no calculados y llegar a la misma elección (podamos el sub-árbol)

Poda Alpha-Beta



```
function MAX-VALUE(game, state,  $\alpha$ ,  $\beta$ ) returns a (utility, move) pair
  if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
   $v \leftarrow -\infty$ 
  for each a in game.ACTIONS(state) do
     $v2, a2 \leftarrow$  MIN-VALUE(game, game.RESULT(state, a),  $\alpha$ ,  $\beta$ )
    if  $v2 > v$  then
       $v, move \leftarrow v2, a$ 
       $\alpha \leftarrow$  MAX( $\alpha$ ,  $v$ )
    if  $v \geq \beta$  then return  $v, move$ 
  return  $v, move$ 
```

```
function MIN-VALUE(game, state,  $\alpha$ ,  $\beta$ ) returns a (utility, move) pair
  if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
   $v \leftarrow +\infty$ 
  for each a in game.ACTIONS(state) do
     $v2, a2 \leftarrow$  MAX-VALUE(game, game.RESULT(state, a),  $\alpha$ ,  $\beta$ )
    if  $v2 < v$  then
       $v, move \leftarrow v2, a$ 
       $\beta \leftarrow$  MIN( $\beta$ ,  $v$ )
    if  $v \leq \alpha$  then return  $v, move$ 
  return  $v, move$ 
```



Menti!



Ayudantía 6

Búsqueda adversaria

Por Blanca Romero y Daniel Toribio

29 de septiembre 2023