

## La Inteligencia Artificial en Juegos

# Objetivos de la clase

- Enumerar características que definen los juegos.
- Representar juegos no cooperativos en forma normal y en forma extensiva.
- Aplicar el algoritmo minimax.
- Aplicar el algoritmo minimax con poda alfa-beta.

# Teoría de juegos

Estudio de los métodos matemáticos de conflicto y cooperación entre agentes racionales deliberativos

Los juegos estudiados por la teoría de juegos **se definen matemáticamente en función de:**

- ☐ Número de jugadores
- ☐ Información disponible
- ☐ Acciones que cada jugador puede realizar
- ☐ Recompensa obtenida

# Teoría de juegos

Otras características relevantes:

- ☐ Secuencial/simultáneo → los jugadores juegan por turnos o todos a la vez
- ☐ Suma constante → la recompensa que recibe un jugador se resta de la de otros
- ☐ Cooperativo / no cooperativo → los jugadores comparten un mismo objetivo y deciden en consenso o no
- ☐ Simétrico / asimétrico → los jugadores pueden realizar idénticas acciones y recibir idénticas recompensas o no

# Teoría de juegos

Otras características relevantes:

- ☐ Duración finita/infinita
- ☐ Discreto / Continuo → número de estrategias y de estados finito o no
- ☐ Determinista / Estocástico → el resultado de una acción y su recompensa es determinista o fijo

**Juegos clásicos adversariales:** dos jugadores, información completa, recompensa infinita por victoria, secuencial, suma cero, no cooperativo, simétrico, duración finita, discreto y determinista

# Teoría de juegos

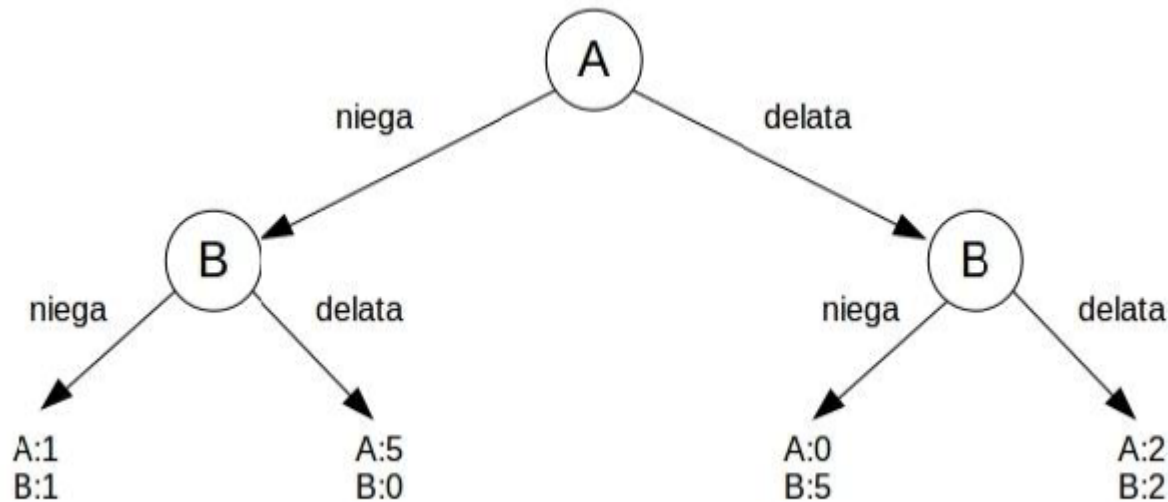
Los juegos no cooperativos se suelen describir de dos maneras:  
forma normal o forma extensiva

- ❑ La **forma normal** representa con una matriz  $n$ -dimensional el producto de las **estrategias** de los jugadores y la recompensa que cada jugador recibe.
- ❑ Juego del dilema del prisionero.

	Prisionero B niega	Prisionero B delata
Prisionero A niega	A: 1 años B: 1 años	A: 5 años B: Libre
Prisionero A delata	A: Libre B: 5 años	A: 2 años B: 2 años

# Teoría de juegos

- Los juegos en los que la recompensa se obtiene después de una **secuencia de acciones** se representan de forma extensiva.



# Teoría de juegos

**Equilibrio de Nash:** situación en un juego de 2 o más jugadores que asume que:

- Cada jugador conoce y adopta su mejor estrategia
- Todos los jugadores conocen las estrategias de los otros



Los jugadores aplican estrategias que maximizan sus ganancias dadas las estrategias de los otros jugadores.

Cada jugador no modificará su estrategia mientras los otros mantengan las suyas.



# Minimax (caso de un juego muy sencillo)

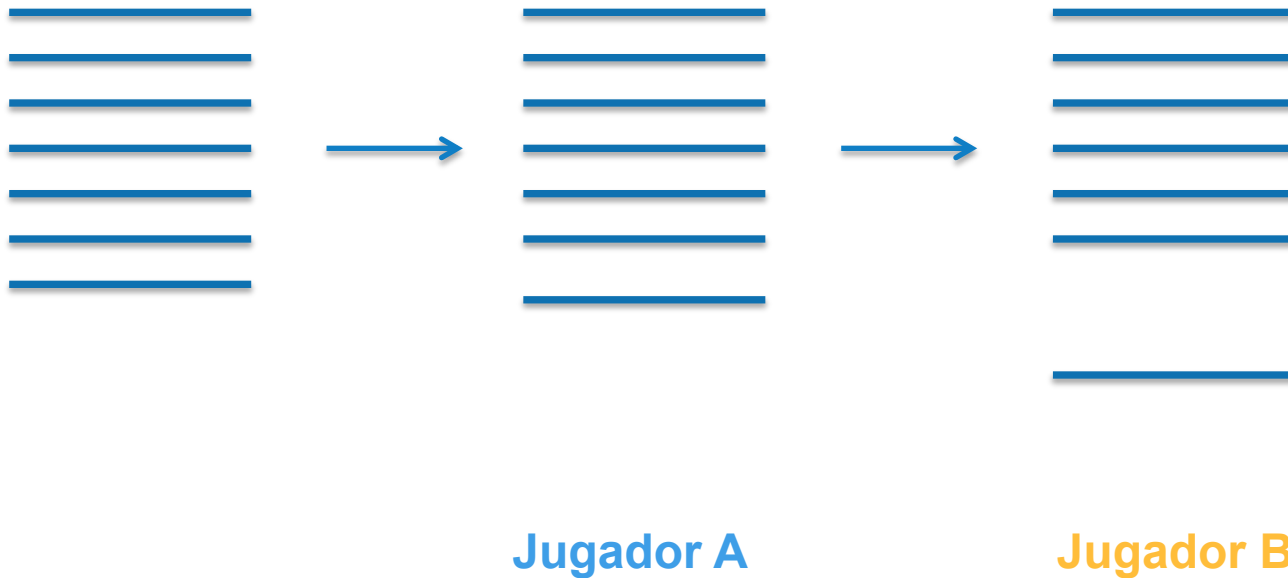
- Se consideran 2 jugadores: MIN y MAX.
- El objetivo es que gane MAX.
- MAX intentará maximizar su ventaja y minimizar la de MIN.
- MIN intentará hacer lo peor para MAX.

# Minimax (caso de un juego muy sencillo)

- Construye un árbol que representa todas las posibles jugadas con nodos MIN o MAX.
  - *Normalmente los problemas no son sencillos y se limita la profundidad del árbol de búsqueda*
- Asignar a los nodos finales un valor
  - $+\infty \rightarrow$  victoria MAX
  - $-\infty \rightarrow$  victoria MIN
- Propagar valores hacia la raíz
  - Nodo MAX  $\rightarrow$  valor máximo de hijos
  - Nodo MIN  $\rightarrow$  valor mínimo de hijos
- Camino de victoria: uniendo los nodos de valor  $+\infty$

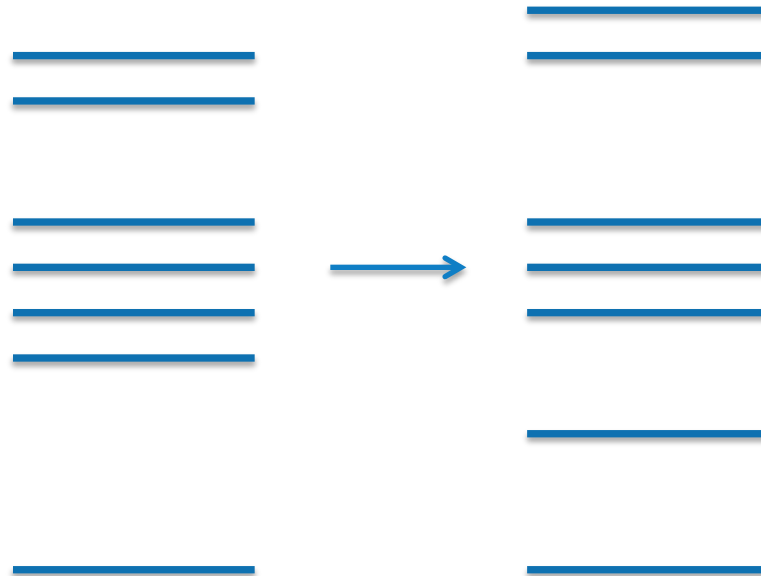
# Minimax (caso de un juego muy sencillo)

## EJEMPLO: juego de los palillos



# Minimax (caso de un juego muy sencillo)

## EJEMPLO: juego de los palillos

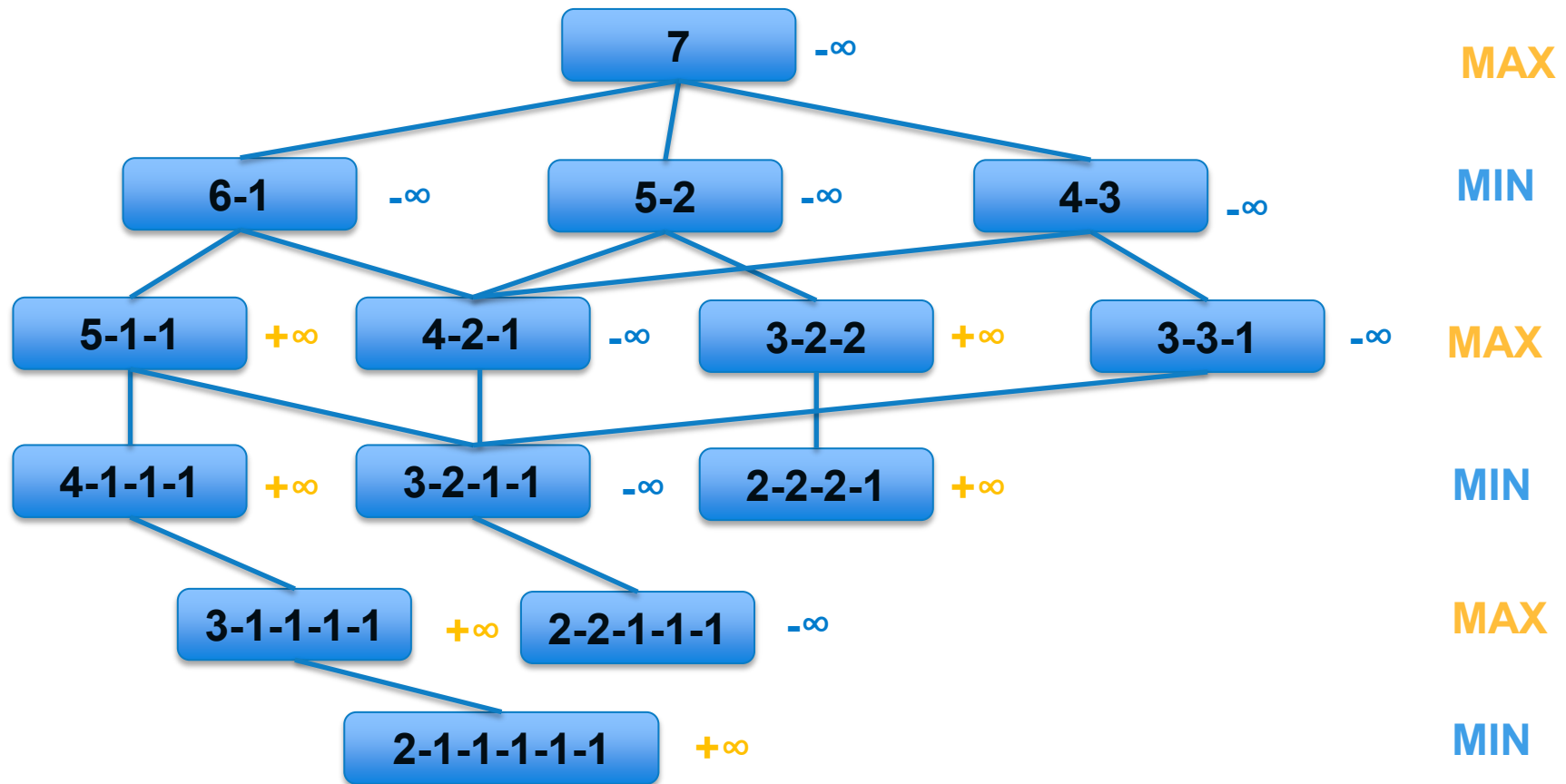


Jugador A

Jugador B

# Minimax (caso de un juego muy sencillo)

## EJEMPLO: juego de los palillos



# Minimax

Algoritmo de búsqueda en profundidad acotada diseñado para juegos:

- ☐ Secuenciales
- ☐ Deterministas
- ☐ Con información completa
- ☐ Discretos
- ☐ De suma cero
- ☐ De 2 jugadores

# Minimax

Se asume que:

- ☐ No se explora el árbol al completo
  - ☐ El rival siempre jugará la jugada óptima
- 
- ☐ Los valores de recompensa van de  $+\infty$  a  $-\infty$
  - ☐ Nodo hoja terminal toma el valor:  $+\infty$  o  $-\infty$
  - ☐ Nodo hoja no terminal: Aplicar función de evaluación
  - ☐ Estrategia jugador MAX es maximizar la recompensa
  - ☐ Estrategia jugador MIN es minimizar la recompensa

# Minimax

## Dos tipos de nodos

- ☐ Nodos en los que juega MAX: se elige la opción que maximiza la recompensa (de MAX)
  - ☐ Nodos en los que juega MIN (contrincante de MAX): se elige la opción que minimiza la recompensa (de MAX)
- 
- ☐ El árbol de búsqueda de Minimax se explora en profundidad hasta alcanzar el límite de jugadas o dar con un nodo terminal.
  - ☐ El valor de la recompensa se propaga hacia arriba.

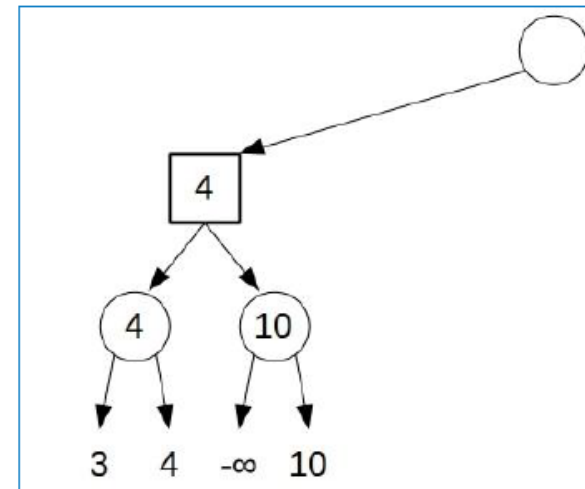
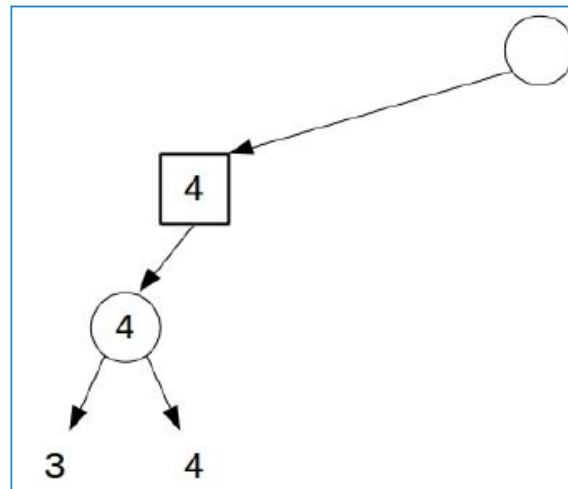
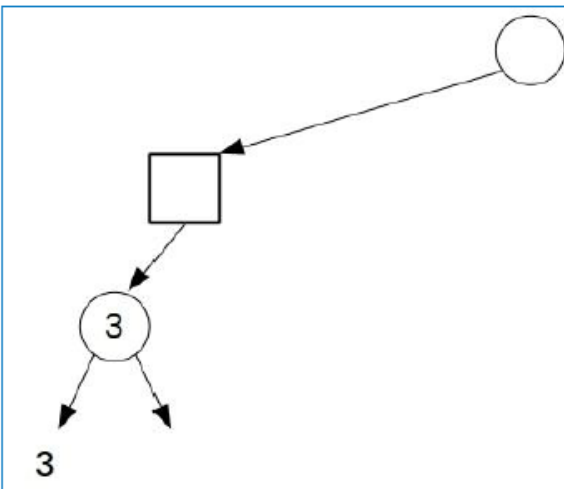


# Minimax

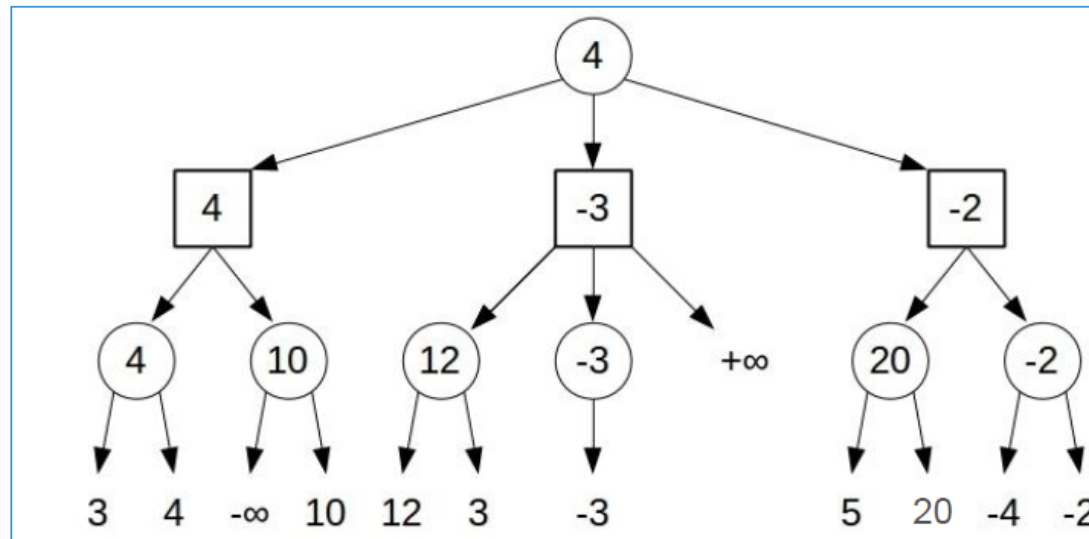
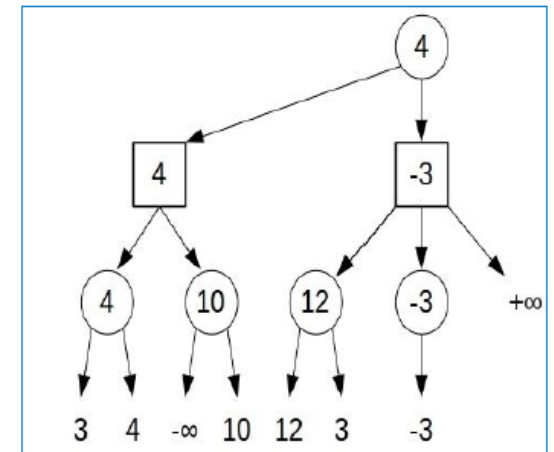
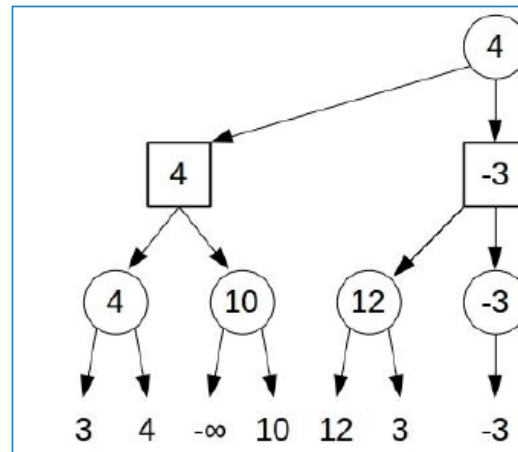
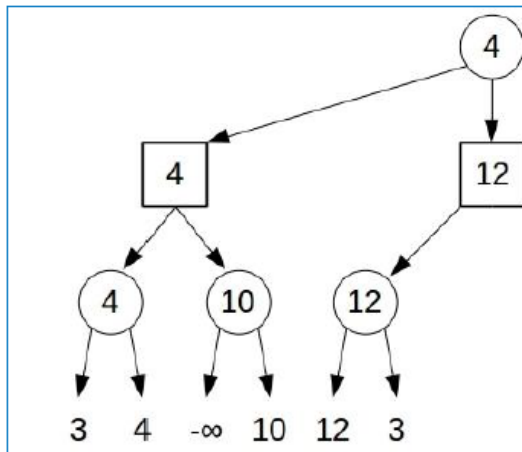
EJEMPLO: Tenemos un límite de 3 niveles (3 jugadas)

Jugador MAX – nodos redondos

Jugador MIN – nodos cuadrados



# Minimax



# Minimax y poda Alfa-Beta

Técnica que permite dejar de buscar en un subárbol cuando se prueba que la recompensa que se va a obtener va a ser menor que la de otro subárbol ya explorado.

$\alpha$  – cota asociada con nodos MAX que nunca decrece

- ☐ se actualiza en nodos MAX
- ☐ máximo de las cotas mínimas

$\beta$  – cota asociada con nodos MIN que nunca se incrementa

- ☐ se actualiza en nodos MIN
- ☐ mínimo de las cotas máximas

La recompensa actual de un nodo que sea relevante a la búsqueda tiene que estar entre  $\alpha$  y  $\beta$

# Minimax y poda Alfa-Beta

- Desde el nodo raíz **se llama recursivamente a los sucesores**
  - Nodo terminal: devuelve el valor de función de evaluación
  - **Nodo MAX:** devuelve alfa y poda si  $\alpha \geq \beta$
  - **Nodo MIN:** devuelve beta y poda si  $\alpha \geq \beta$
- Se poda por debajo de un nodo que tenga un valor de alfa mayor o igual que el valor de beta

```
minimax-alfabeta (nodo N, profundidad,  $\alpha$ ,  $\beta$ , MAX)
1. Si profundidad = 0 O N es terminal
   Entonces Devolver valor
2. Si MAX = Cierto
   Entonces Para cada sucesor S de N
      $\alpha \leftarrow \max(\alpha, \text{minimax-alfabeta}(S, \text{profundidad}-1, \alpha, \beta, \text{Falso}))$ 
     Si  $\beta \leq \alpha$ 
       Entonces Break
     Devolver  $\alpha$ 
3. Si No
   Entonces Para cada sucesor S de N
      $\beta \leftarrow \min(\beta, \text{minimax-alfabeta}(S, \text{profundidad}-1, \beta, \alpha, \text{Certo}))$ 
     Si  $\beta \leq \alpha$ 
       Entonces Break
   Devolver  $\beta$ 
```

# Minimax y poda Alfa-Beta

Si es nodo MAX:

-se actualiza alfa

-Se devuelve el valor de alfa

*Se llama con este mismo método a cada sucesor (y eso sucede recursivamente) y se les pasa alfa y beta .*

*Un nodo MAX al realizar esta llamada a un sucesor, su sucesor le devolverá el valor de su beta que se utilizará para actualizar el valor de alfa del nodo MAX (la beta quedará igual), siempre que supere el valor de alfa del nodo actual*

```
minimax-alfabeta (nodo N, profundidad,  $\alpha$ ,  $\beta$ , MAX)
1. Si profundidad = 0 O N es terminal
   Entonces Devolver valor
2. Si MAX = Cierto
   Entonces Para cada sucesor S de N
      $\alpha \leftarrow \max(\alpha, \text{minimax-alfabeta}(S, \text{profundidad}-1, \alpha, \beta, \text{Falso}))$ 
     Si  $\beta \leq \alpha$ 
       Entonces Break
     Devolver  $\alpha$ 
3. Si No
   Entonces Para cada sucesor S de N
      $\beta \leftarrow \min(\beta, \text{minimax-alfabeta}(S, \text{profundidad}-1, \alpha, \beta, \text{Cierto}))$ 
     Si  $\beta \leq \alpha$ 
       Entonces Break
   Devolver  $\beta$ 
```

Poda si alfa  
 $\geq$  beta

Si es nodo MIN:

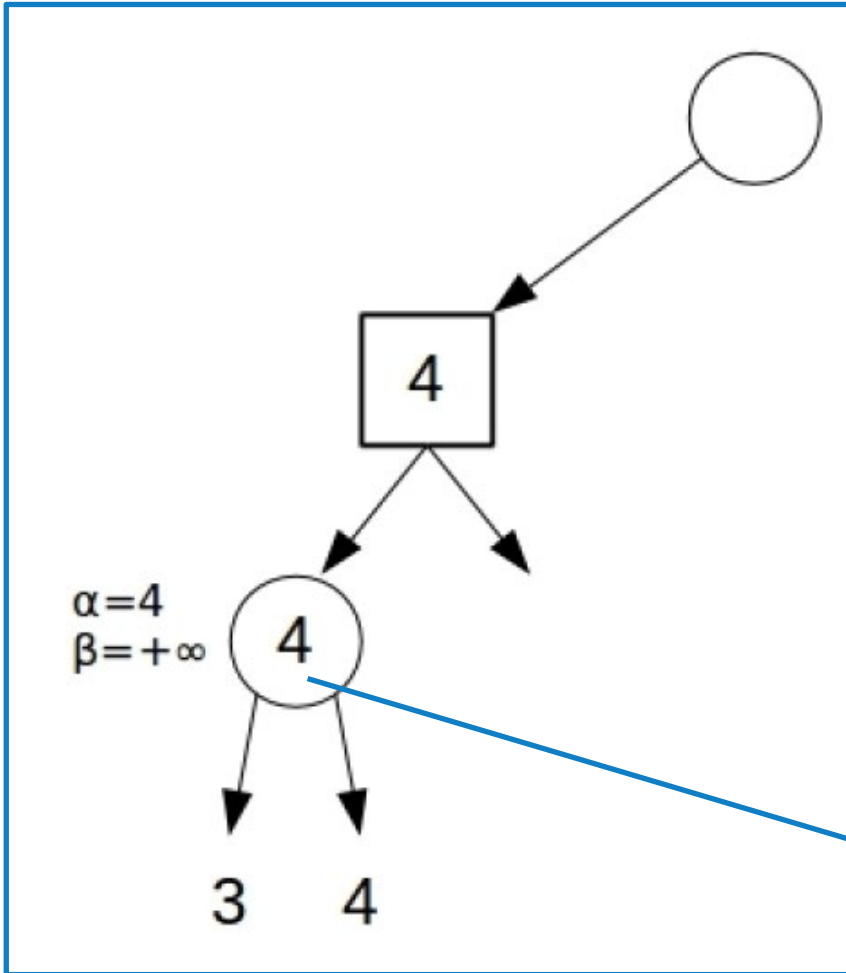
-se actualiza beta

-Se devuelve el valor de beta

*Beta se actualiza con el valor mínimo de los valores devueltos por sus sucesores (que son las alfas de los nodos MAX hijos)*

# Minimax y poda Alfa-Beta

- Inicialmente  $\alpha = -\infty$  y  $\beta = +\infty$



```
minimax-alfabeta (nodo N, profundidad,  $\alpha$ ,  $\beta$ , MAX)
1. Si profundidad = 0 O N es terminal
   Entonces Devolver valor
2. Si MAX = Cierto
   Entonces Para cada sucesor S de N
      $\alpha \leftarrow \max(\alpha, \text{minimax-alfabeta}(S, \text{profundidad}-1, \alpha, \beta, \text{Falso}))$ 
     Si  $\beta \leq \alpha$ 
       Entonces Break
     Devolver  $\alpha$ 
3. Si No
   Entonces Para cada sucesor S de N
      $\beta \leftarrow \min(\beta, \text{minimax-alfabeta}(S, \text{profundidad}-1, \beta, \text{Cierto}))$ 
     Si  $\beta \leq \alpha$ 
       Entonces Break
     Devolver  $\beta$ 
```

Nodo MAX  $\rightarrow$  Actualizamos Alfa  
 $\alpha = 4$  (máximo de las cotas mínimas)

# Minimax y poda Alfa-Beta

$\alpha$  y  $\beta$  se propagan desde nodos padres a sucesores

Nodo MAX: Actualiza  $\alpha$

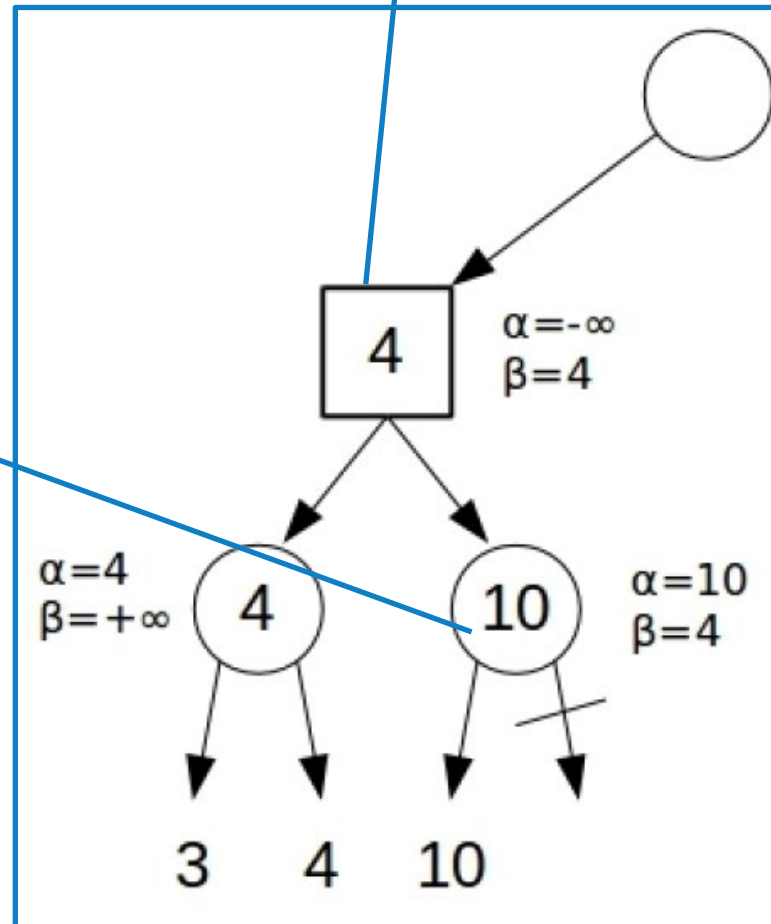
$\alpha = 10$  (máximo de las cotas mínimas)

$\beta = 4$  (valor propagado desde el padre)

Como  $\alpha$  es mayor que  $\beta$ , se poda.

*Ese nodo como mínimo vale 10 dado que es un nodo MAX, sabemos que el nodo padre MIN preferirá el nodo de valor 4 ( $\beta = 4$ ) y por tanto no hace falta seguir explorando este nodo*

Nodo MIN: Actualiza Beta  
 $\beta = 4$  (mínimo de las cotas máximas)



# Minimax y poda Alfa-Beta

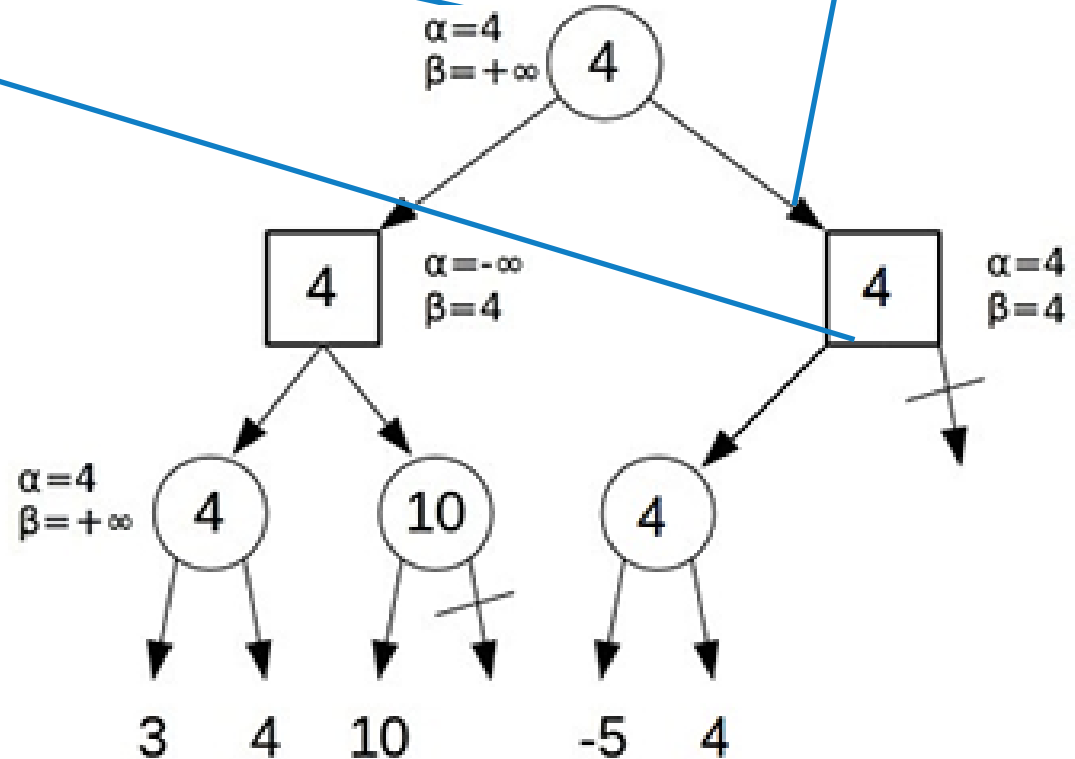
Nodo MAX: actualiza  $\alpha$

Nodo MIN que siempre tomará un valor menor o igual a  $\beta = 4$

$\alpha = 4$  nos indica que hay otro subárbol que será preferido por el nodo padre MAX a este, cuyo valor como mucho es 4.

Como  $\alpha$  es mayor o igual que  $\beta$ , se poda

Se propaga el valor de  $\alpha$  hacia sucesores



*Se poda por debajo de un nodo que tenga un valor de alfa mayor o igual que el valor de beta.*

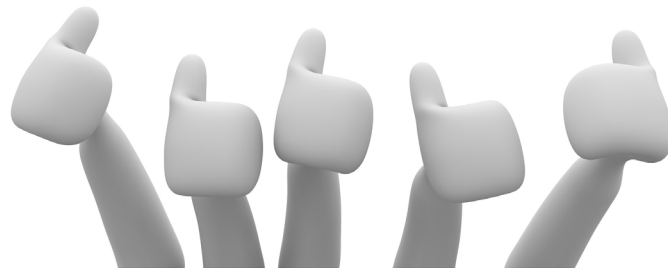


# ¿Dudas?



¡Muchas gracias por vuestra atención!

¡Feliz y provechosa semana!





[www.unir.net](http://www.unir.net)