Proyecto TA TE TI

Organización de Computadoras 2019

Juego TA TE TI

- En estas diapositivas veremos el algoritmo que vamos a utilizar para calcular la jugada más conveniente cuando le toque el turno a la computadora.
- Llegado el momento en que toque el turno de la computadora, se presenta el problema que se debe buscar la **mejor jugada**. Teniendo en cuenta los **movimientos** que puede llegar a hacer el **adversario**.
- Este tipo de problemas se denominan: **búsqueda con adversarios**.

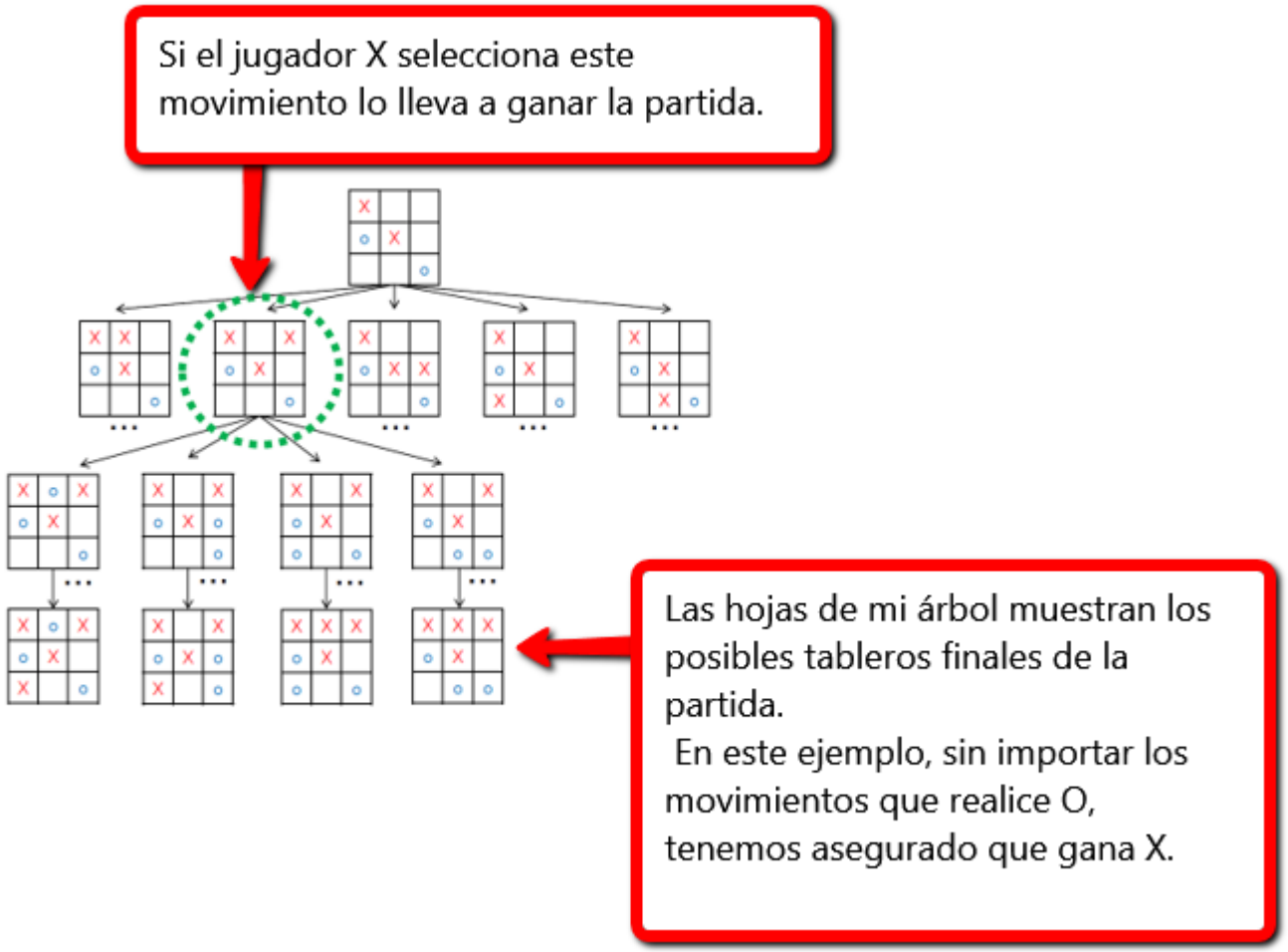
Búsqueda Adversaria

- Considerar una situación en la que tiene el **turno** el jugador **X**.
- Se podría definir una **estrategia**, para que
  - **NO GANE** el **adversario**.
  - o dar un **movimiento ganador.**

Búsqueda adversaria empleando árboles

- Consideremos entonces los **juegos adversarios** donde todos los **estados** del juego son **accesibles** por los dos jugadores (información perfecta).
- A partir de un **estado inicial** del juego, se pueden inferir todos los **estados alcanzables** aplicando algún **operador** hasta llegar a un **estado meta**.
- Todos los **estados alcanzables** se pueden visualizar a través de un **árbol**, que denominaremos **árbol de búsqueda**.
- Es claro que sobre dicho **árbol de búsqueda**, considerándolo como un **espacio de búsqueda** para un problema en particular, se pueden computar diferentes soluciones posibles a dicho problema.

Ejemplo

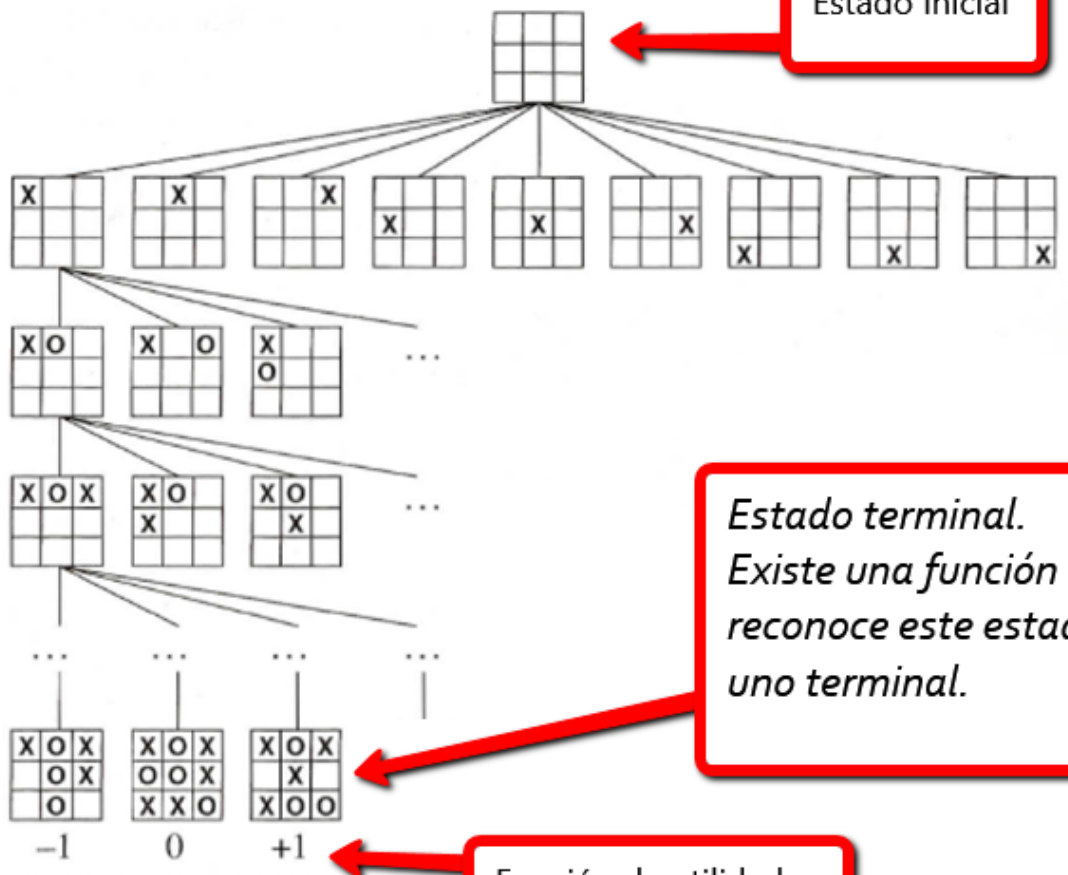

Si el jugador X selecciona este movimiento lo lleva a ganar la partida.
Las hojas de mi árbol muestran los posibles tableros finales de la partida.
En este ejemplo, sin importar los movimientos que realice O, tenemos asegurado que gana X.

MIN vs MAX, Objetivo

- Examinaremos problemas en los que existen dos jugadores a los que llamaremos MIN y MAX.
- MIN y MAX son *adversarios*. Cada uno juega planificando su estrategia en contra del otro.
- MAX siempre trata de *maximizar* su jugada.
- MIN trata de *minimizar* la jugada de MAX.

Cuáles son los componentes?

- **Estado inicial:** que incluye la composición del tablero y una indicación de quién debe realizar la movida.
- **Operadores:** definen las movidas legales a disposición de los jugadores.
- **Test de terminación:** establece cuando el juego ha terminado. Los estados donde el juego ha terminado se denominan **estados terminales**.
- **Función de utilidad o debeneficio:** asigna un valor numérico a un **estado final** del juego, usualmente **1** (gana**max**) y **–1** (gana**min**). Puede ser **0**(**empate**). Depende del juego, puede ser también cualquier valor real.



Estado Inicial

Operadores aplicables y válidos sobre un estado anterior:
- Fijar_X(3,2)
- Fijar_X(3,3) . . .

Estado terminal.
Existe una función que reconoce este estado como uno terminal.

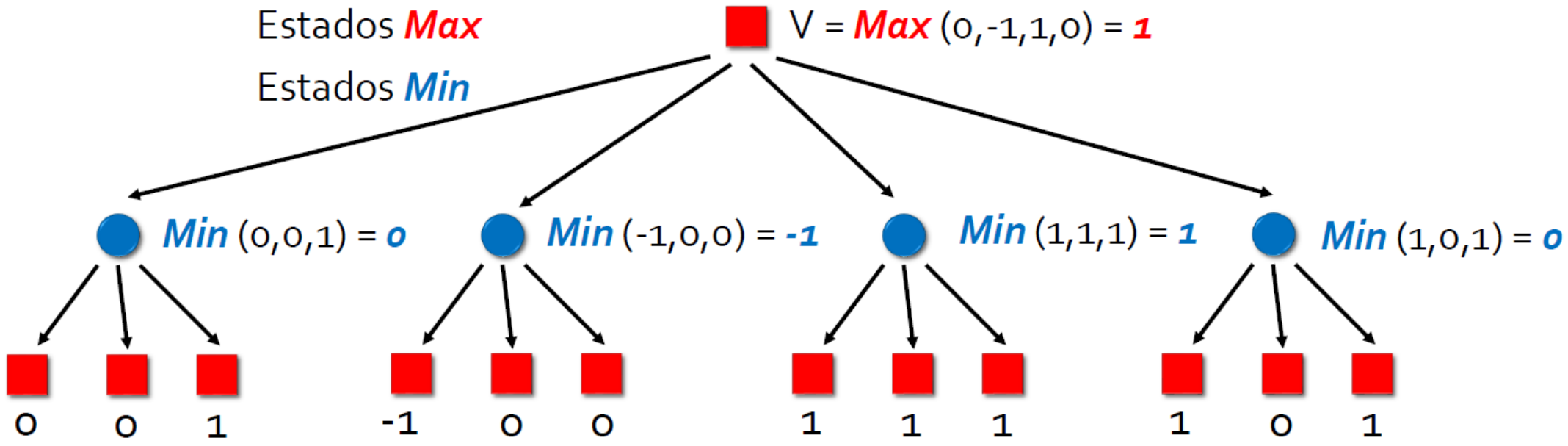
Función de utilidad

Algoritmo MINIMAX

- Este **algoritmo** está diseñado para determinar la **estrategia óptima** para que **Max** gane.
- Presupuestos:
  - **Max** va a realizar siempre **su mejor** jugada.
  - **Min** va a realizar siempre **la peor** jugada para **Max**.
- Una **estrategia optimal** para **Max**, se puede realizar examinando el valor de cada estado con la siguiente función.

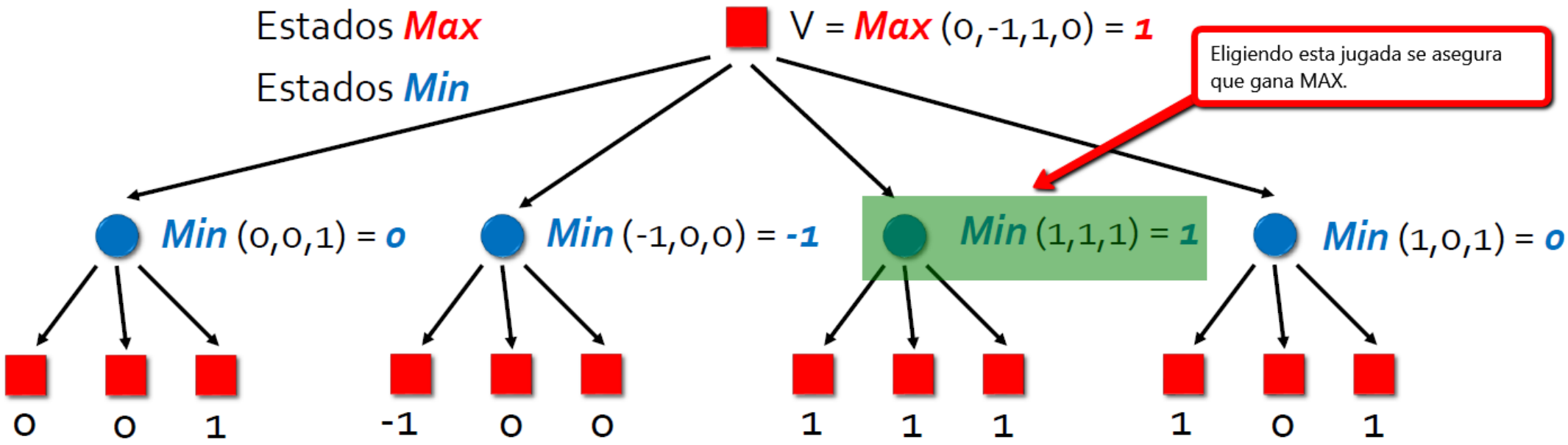
- **Valor(N:Estado) -> número**
  - Si N es **estado terminal:** retorna **utilidad** del estado N.
  - Si N es **estado** de **Max**, retorna **máximo valor** de sucesores de N.
  - Si N es **estado de Min**, retorna **mínimo valor** de sucesores de N.

Ejemplo:


Estados Max
Estados Min
V = Max (0,-1,1,0) = 1
Min (0,0,1) = 0
Min (-1,0,0) = -1
Min (1,1,1) = 1
Min (1,0,1) = 0
0 0 1
-1 0 0
1 1 1
1 0 1

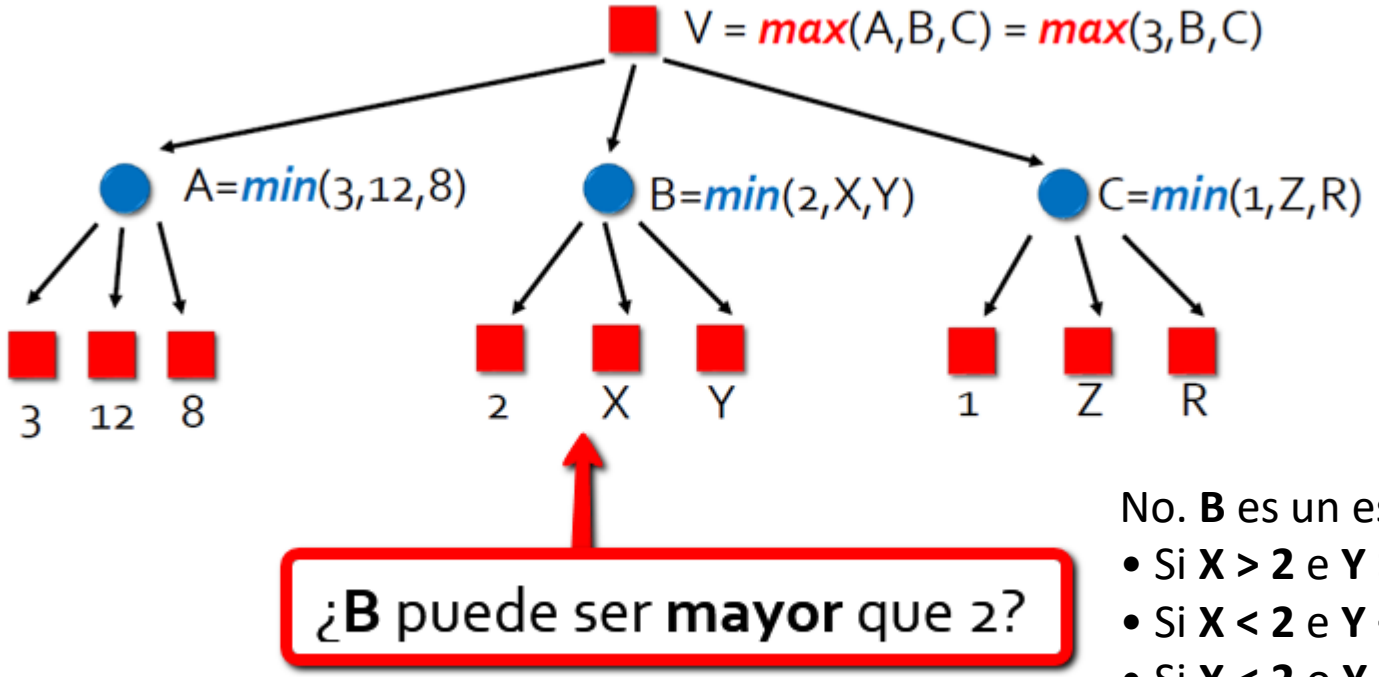
- El **algoritmo minimax** parte de un **estado inicial N (raíz del árbol)** y explora **en profundidad** hasta un **estado terminal (nodo hoja)** del cual obtiene un **valor de utilidad**.
- Luego propaga los valores desde los **estados terminales** hasta el **estado inicial N**, calculando el **máximo** o el **mínimo** de sus **estados sucesores (hijos),** según corresponda a una jugada propia (**max**) o del adversario (**min**).
- Con el valor calculado, **max** puede **decidir** su jugada (eligiendo el operador que lleve al **estado sucesor** de mayor valor).

Ejemplo


Estados Max
Estados Min
V = Max (0,-1,1,0) = 1
Eligiendo esta jugada se asegura que gana MAX.
Min (0,0,1) = 0
Min (-1,0,0) = -1
Min (1,1,1) = 1
Min (1,0,1) = 0
0 0 1
-1 0 0
1 1 1
1 0 1

Podas Alpha-Beta

- ¿Es necesario analizar todo el árbol?

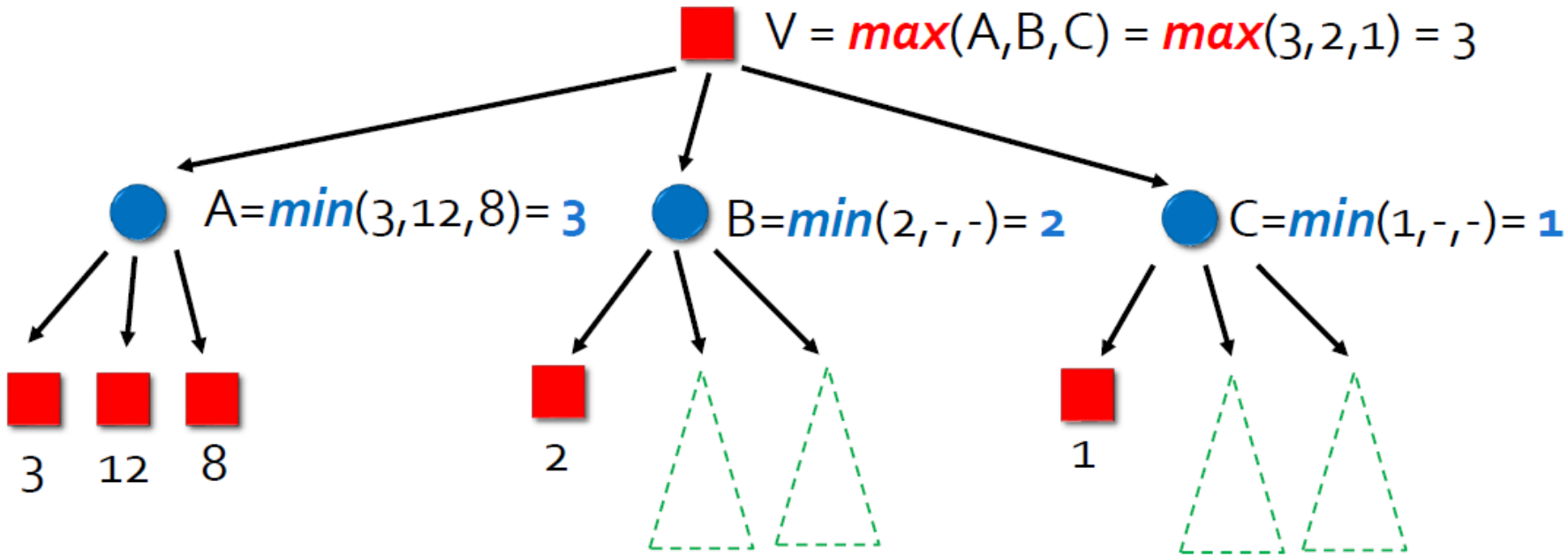


No. **B** es un estado **min**, y:

- Si **X > 2** e **Y > 2** entonces **B = 2**.
- Si **X < 2** e **Y < 2** entonces **B < 2**.
- Si **X < 2** o **Y < 2** entonces **B < 2**.

Podas

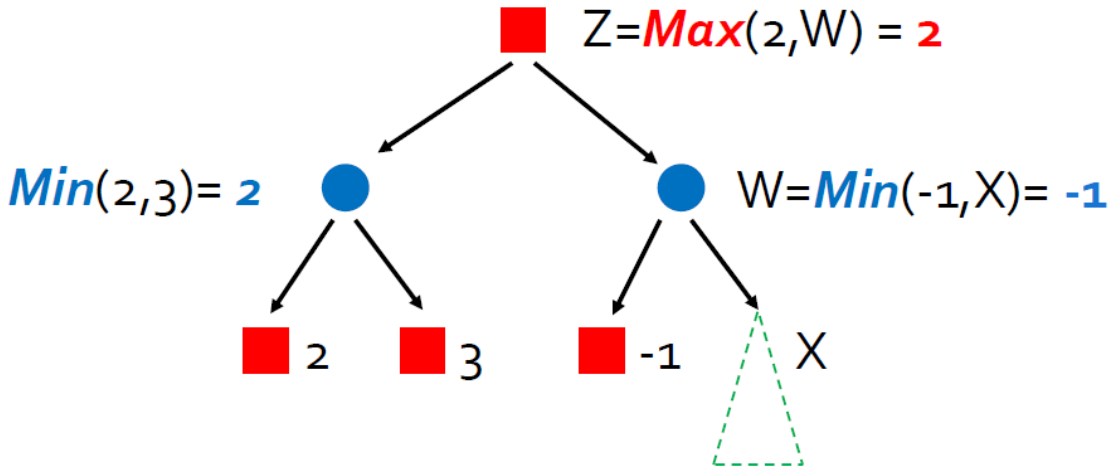
- Los subárboles verdes son podados, no es necesario evaluarlos. Con lo cual se descartan.



Reduciendo estados a evaluar

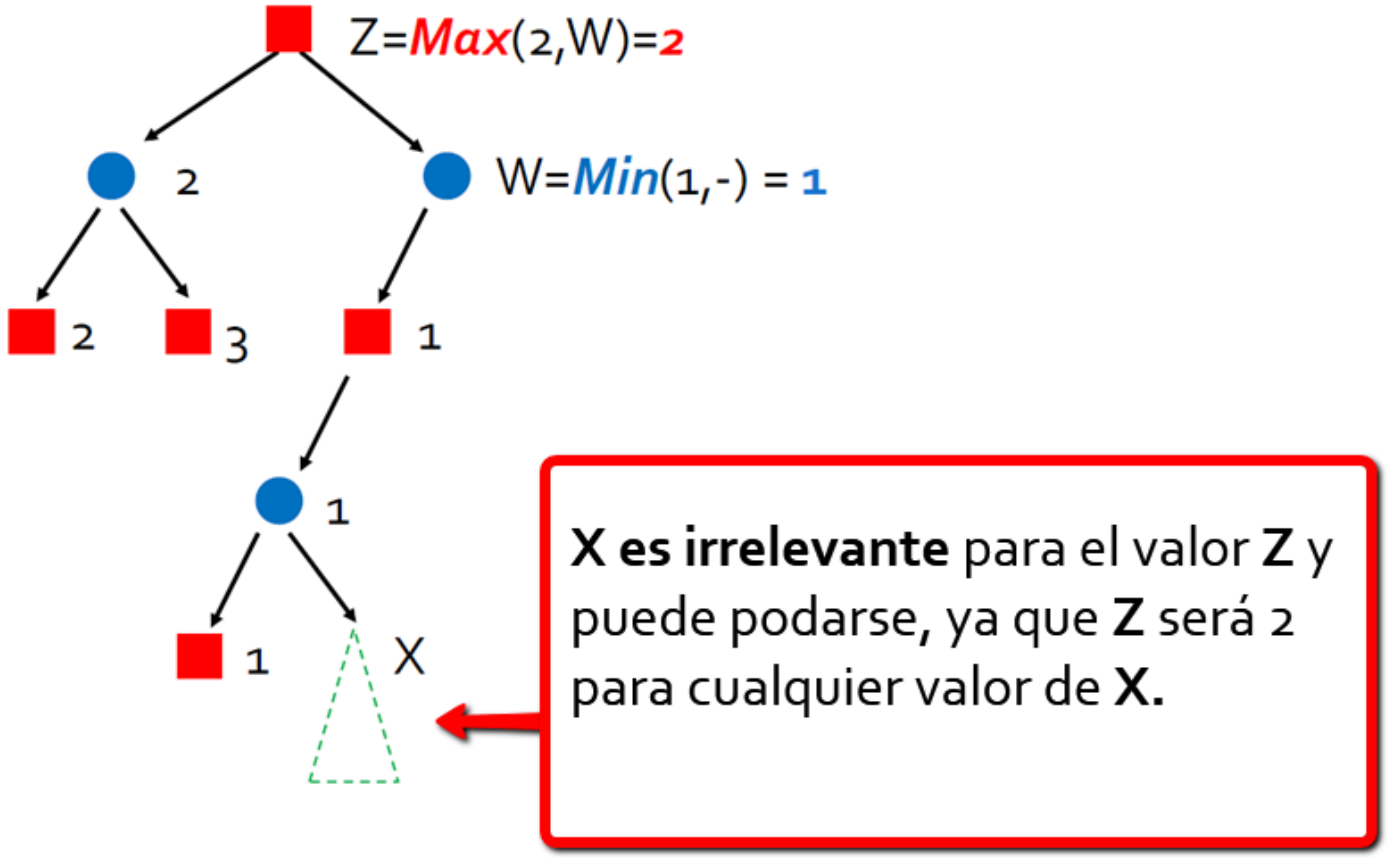
- La **poda alfa-beta** es una técnica que **reduce** el número de **estados** evaluados en el **árbol de búsqueda adversaria**.
- Permite **evitar la exploración completa del árbol de búsqueda adversaria** de forma tal que:
  - No existe **pérdida** de información.
  - Permite computar el procedimiento **minimax correctamente**.
- El proceso de **eliminar una rama del árbol de búsqueda** sin considerarla se denomina **poda** (**pruning**).
- El **sub-árbol** que se **poda** es **irrelevante** para el resultado del valor que el algoritmo **minimax** le asigna a la raíz del árbol.

Ejemplo

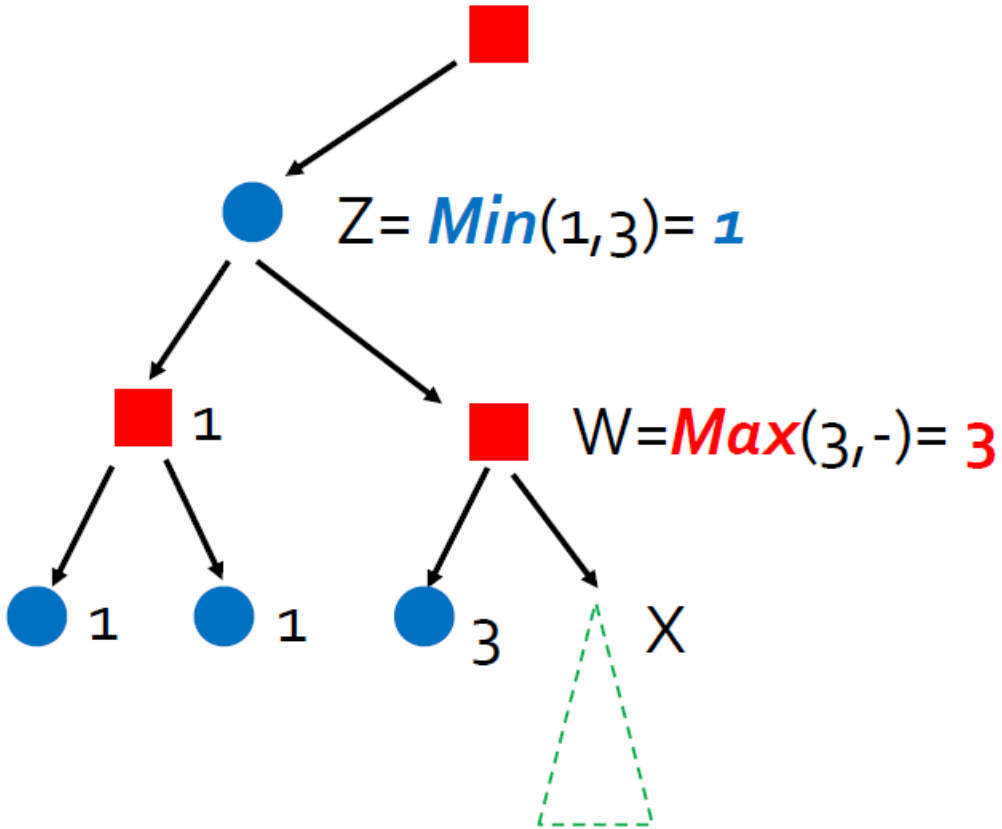


- En este ejemplo, el valor del nodo X resulta irrelevante para Z.
- Si **X>=-1** entonces **W=-1** y luego **Z=2**.
- Si **X<-1** entonces **W=X** y luego **Z=2(W<-1)**.

Ejemplo


Z=Max(2,W)=2
2
W=Min(1,-) = 1
2
3
1
1
1
X
X es irrelevante para el valor **Z** y puede podarse, ya que **Z** será 2 para cualquier valor de **X.**

Ejemplo



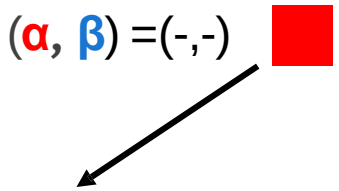
Técnica

- La idea de esta técnica es que **cada estado N** se analiza teniendo en cuenta:
    - el **valor de utilidad** que **por el momento** tiene el **estado N**.
    - el **valor de utilidad** que **por el momento** tiene el **padre** del **estado N**.
- Luego, con estos valores, un **intervalo (α,β)** de posibles valores que podría tomar el **estado N** puede establecerse.

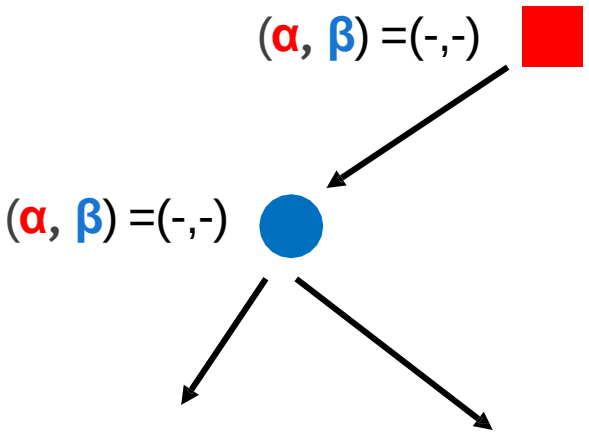
En resumen…

- El significado intuitivo de **α** y **β** en cada momento es:
  - Estados ***Max***: **α** es el **valor de utilidad actual** del **estado N** (que finalmente será igual o superior a **α**), y **β** es el **valor de utilidad actual** del padre del **estado N**(que finalmente será igual o inferior a **β**).
  - Estado ***Min***: **β** es el **valor de utilidad actual** del **estado N**(que finalmente será igual o inferior a **β**), y **α** es el **valor de utilidad actual** del padre del **estado N**(que finalmente será igual o superior a **α**).
- La poda se produce si en algún momento **α ≥ β**, y en ese momento no hace falta analizar los restantes sucesores del estado. En nodos ***Min***, se denomina **poda β**, y en nodos ***Max***, **poda α**.

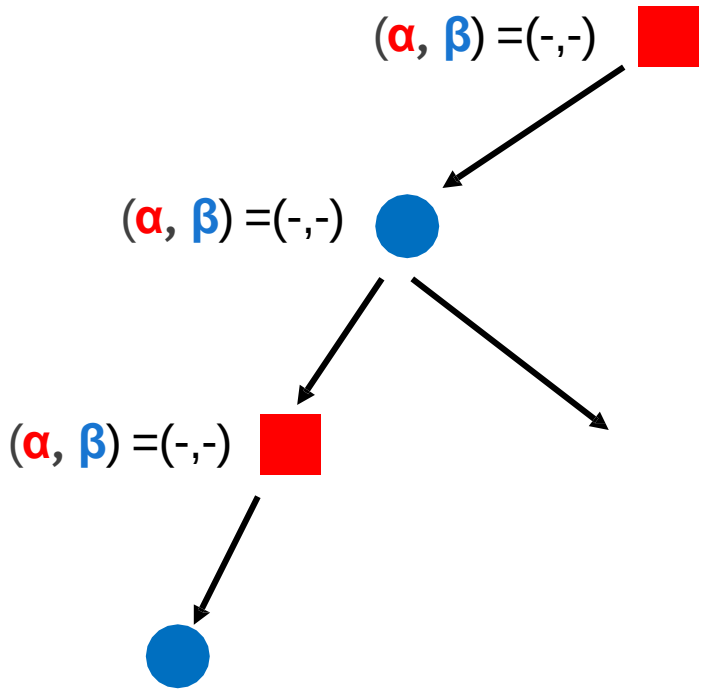
Ejemplo de poda



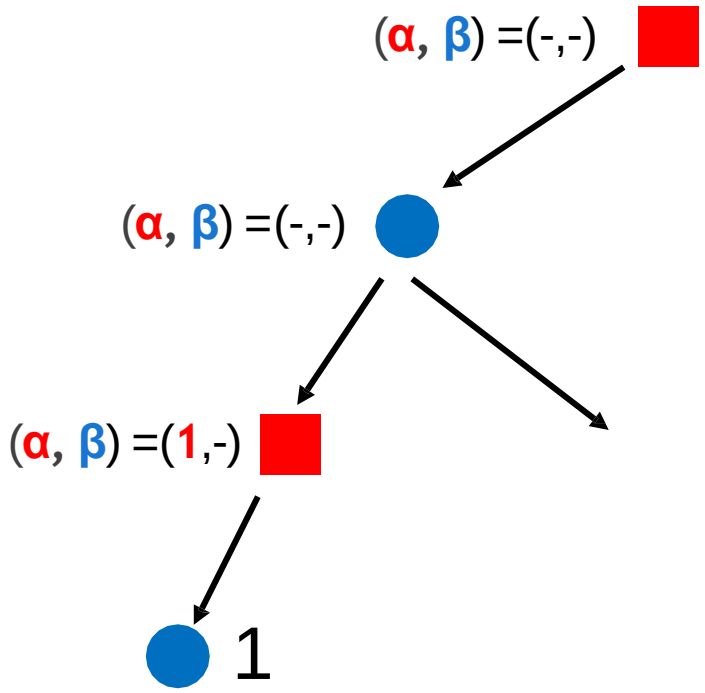
Ejemplo de poda



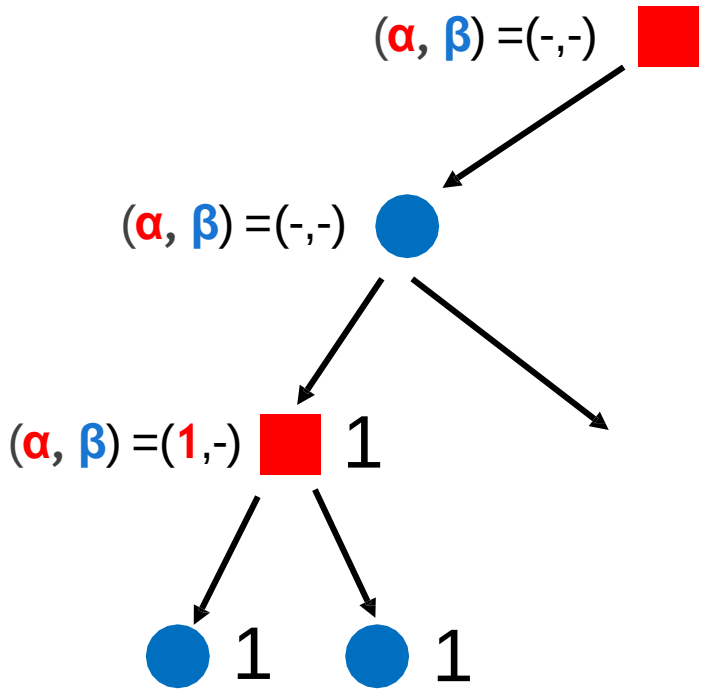
Ejemplo de poda



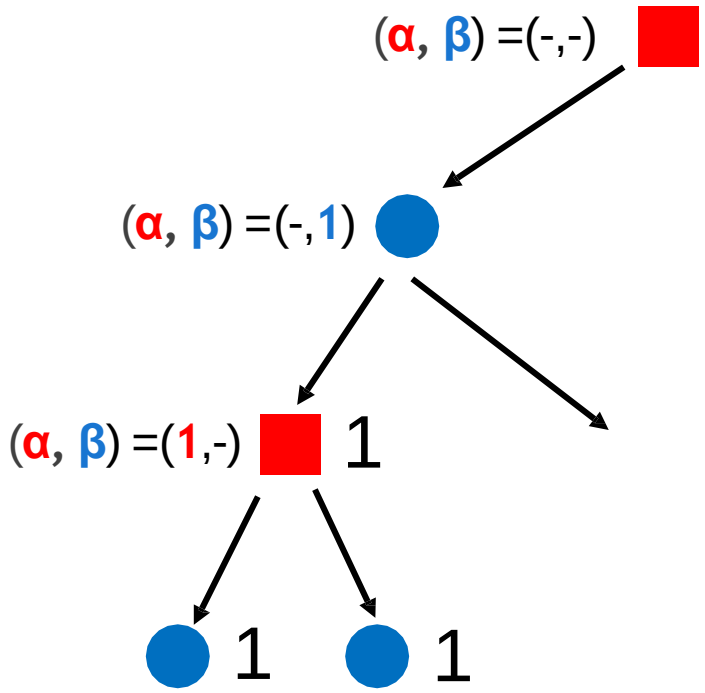
Ejemplo de poda



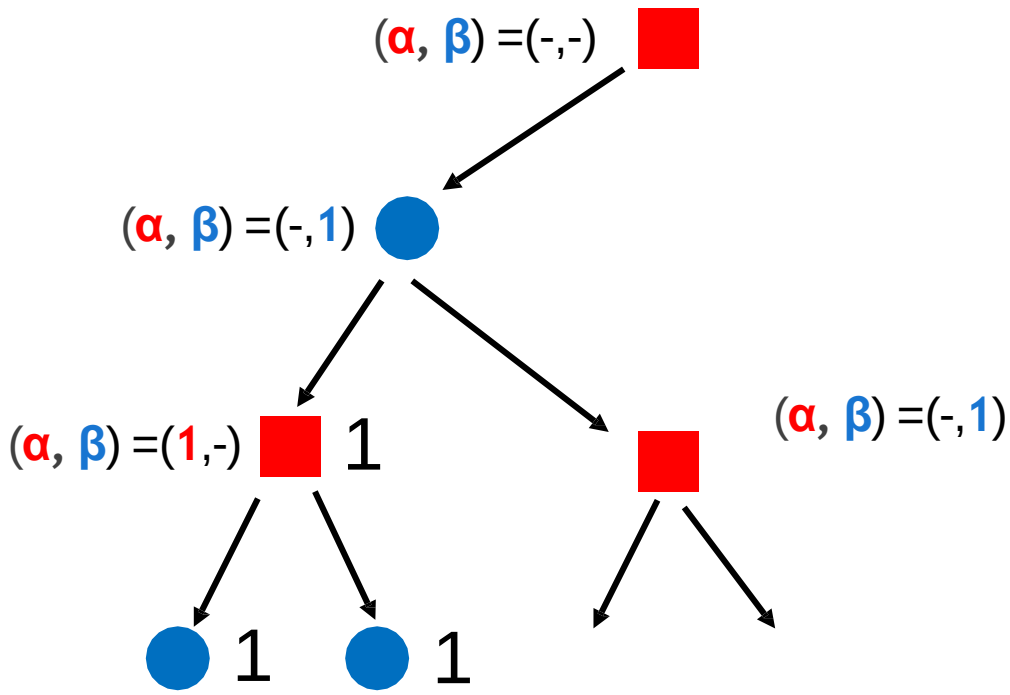
Ejemplo de poda



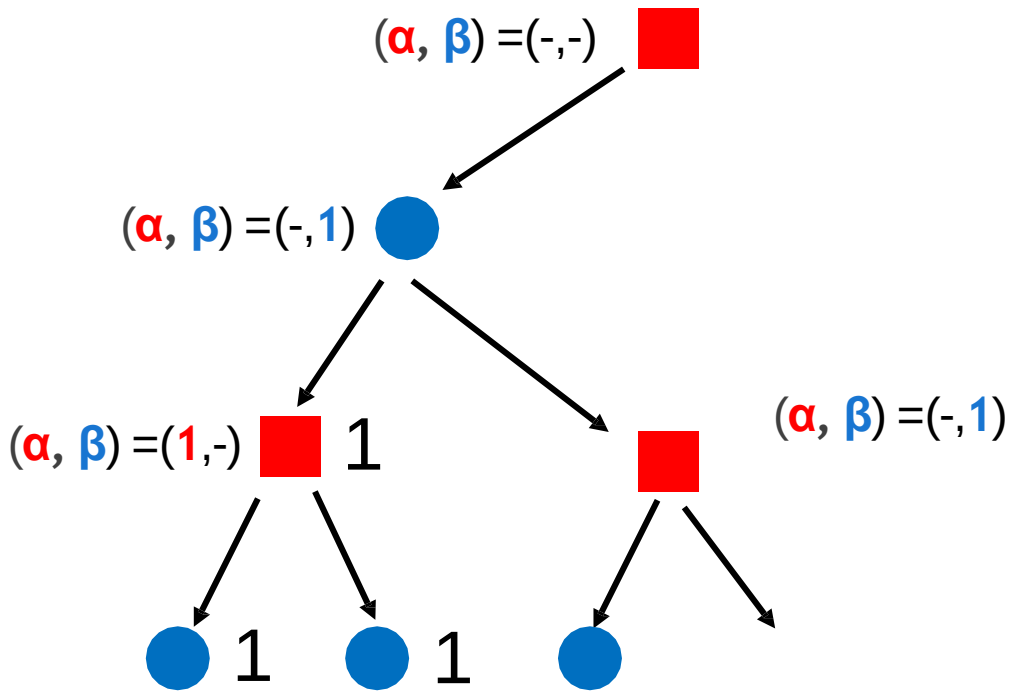
Ejemplo de poda



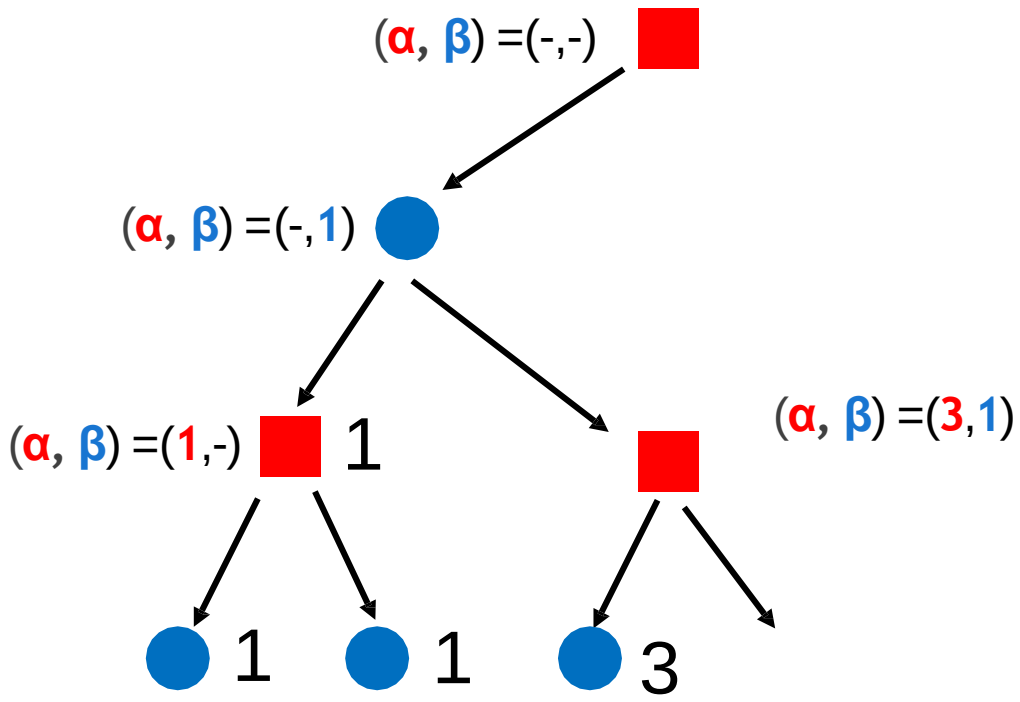
Ejemplo de poda



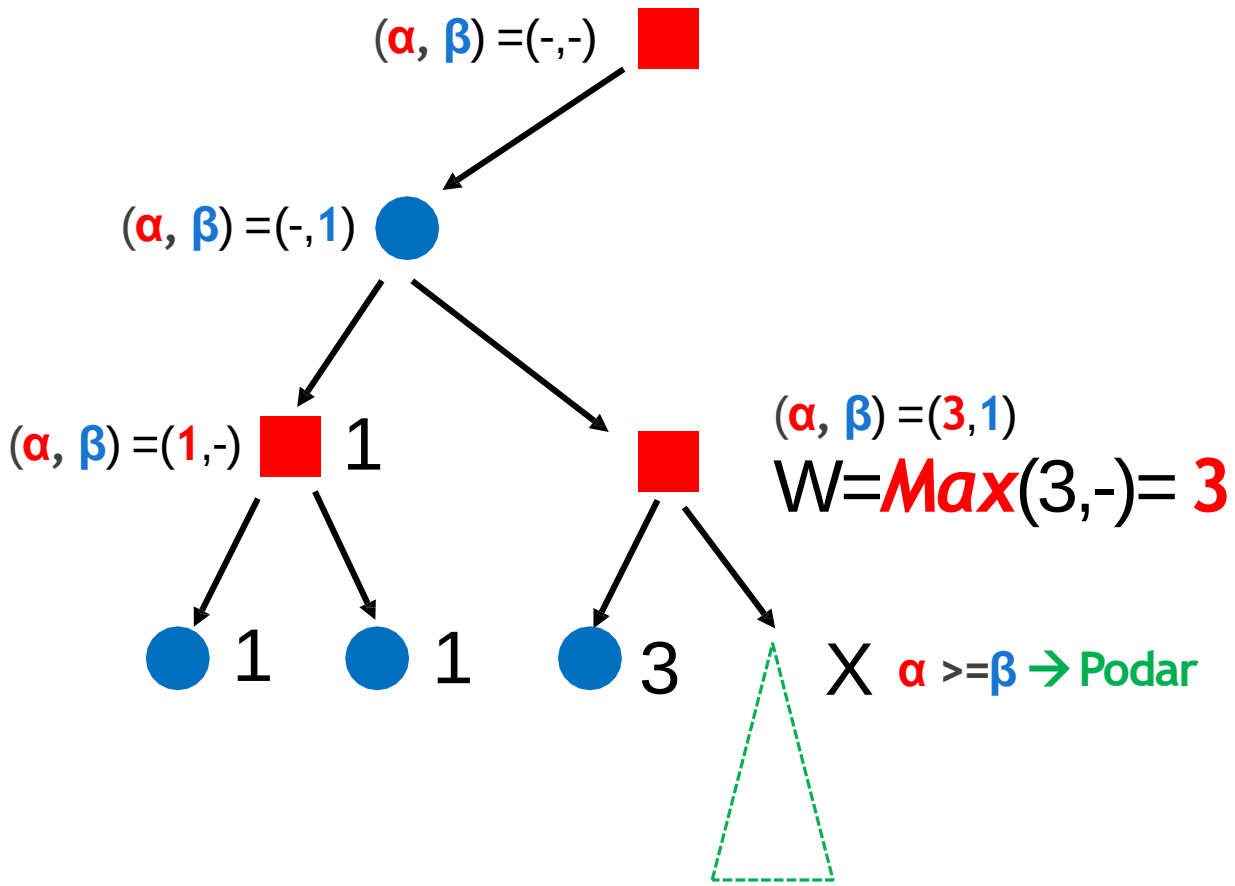
Ejemplo de poda



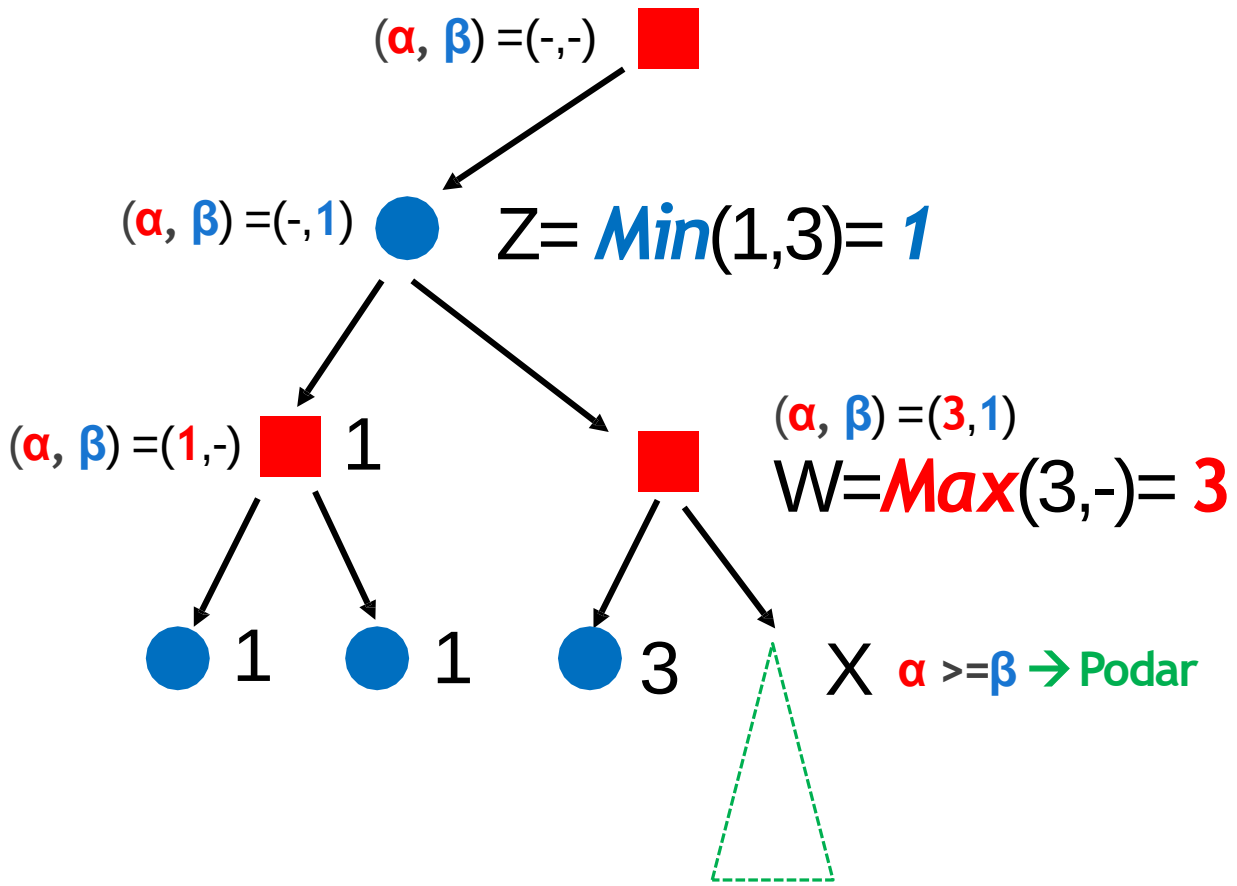
Ejemplo de poda



Ejemplo de poda



Ejemplo de poda



Pseudocódigo

```
función minimax (estado, es_jugador_max, alpha, beta)
    Si es_estado_terminal(estado)
        retornar valor_utilidad(estado)
    fin si

    Si (es_jugador_max)
        mejor_valor_sucesores = -INFINITO
        Para cada estado_sucesor de sucesores(estado)
            valor_sucesor = minimax (estado_sucesor, false, alpha, beta)
            mejor_valor_sucesores = max (mejor_valor_sucesores, valor_sucesor)
            alpha = max (alpha, mejor_valor_sucesores)
            Si (beta <= alpha)
                parar
           retornar mejor_valor_sucesores
    Si no
        mejor_valor_sucesores = +INFINITO
        Para cada estado_sucesor de sucesores(estado){
            valor_sucesor = minimax (estado_sucesor, true, alpha, beta)
            mejor_valor_sucesores = min (mejor_valor_sucesores, valor_sucesor)
            beta = min (beta, mejor_valor_sucesores)
            Si (beta <= alpha)
                parar
           retornar mejor_valor_sucesores
    fin si
fin función
```

```
// Considerando un estado ei que
// representa el estado inicial por
// donde comienza la búsqueda, la
// llamada al algoritmo sería:
minimax (ei, true, -INFINITY, +INFINITY)
```