Bloque 1 – Representación del conocimiento y búsqueda

Tema 4:
Búsqueda con adversario

Tema 6- Índice

1. Juegos: definición del problema
2. Algoritmo MINIMAX
3. Poda Alfa-beta
4. Refinamientos adicionales

Bibliografía:

- **S. Russell, P. Norvig. *Artificial Intelligence . A modern approach.* Prentice Hall, 3rd edition, 2010 (Tema 5) http://aima.cs.berkeley.edu/**
- N. J. Nilsson**. *Artificial Intelligence: A New Synthesis*.** Morgan Kaufmann Publishers, 1998 (Tema 1)

Alternativamente:

- S. Russell, P. Norvig. ***Inteligencia Artificial. Una aproximación moderna.*** Prentice Hall, 2nd edition, 2004 (Tema 6) http://aima.cs.berkeley.edu/2nd-ed/

1. Definición del problema

Un juego es una situación conflictiva en la que uno debe tomar una decisión sabiendo que los demás también toman decisiones, y que el resultado del conflicto se determina, de algún modo, a partir de todas las decisiones realizadas. (John von Neumann)

Siempre existe una forma racional de actuar en juegos de dos participantes, si los intereses que los gobiernan son completamente opuestos. (John von Neumann)

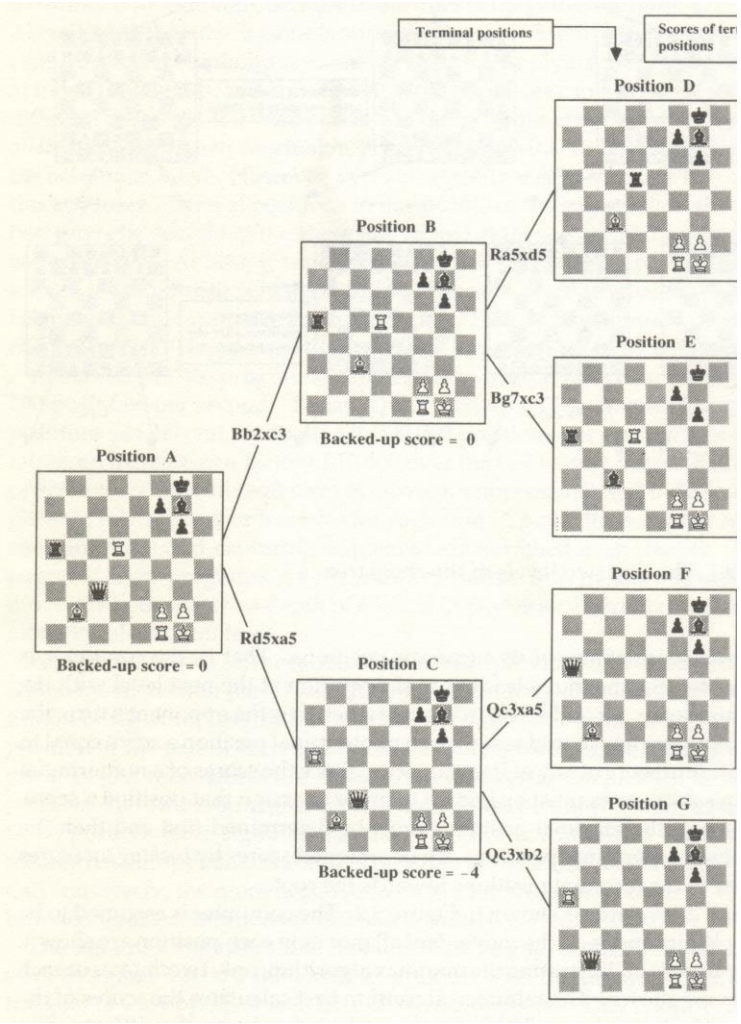
La demostración es la Teoría Minimax (1926).

1. Definición del problema

La mayoría de juegos que se estudian en IA son:

- **Deterministas** (pero estratégicos): la suerte no interviene
- Se juegan alternativamente **por turnos**
- **Dos jugadores**: persona-persona, persona-ordenador, ordenador-ordenador
- **Suma cero**: la ganancia (o pérdida) de un participante es exactamente la pérdida (o ganancia) del oponente. Esta oposición define los juegos como problemas de búsqueda con adversario.
- **Información perfecta**: los juegos son conocidos y limitados: cada jugador conoce perfectamente la evolución del juego y los movimientos que puede hacer su oponente (observable)

1. Definición del problema



El número de movimientos de los jugadores está generalmente restringido a un pequeño número de acciones

Los resultados de las acciones se definen con reglas precisas

Los juegos son problemas complejos. Por ejemplo, el ajedrez tiene un factor de ramificación medio de aproximadamente 35, y en un juego, cada jugador hace una media de 50 movimientos, así que el árbol de búsqueda del juego tendría unos 35^{100} nodos …

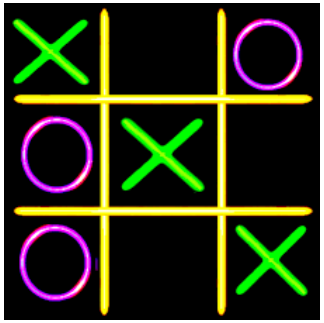
Los juegos son problemas que requieren tomar una decisión aunque la decisión óptima sea inviable

1. Definición del problema

1. Definición del problema

Dos personas quieren jugar al tres-en-raya

Eva

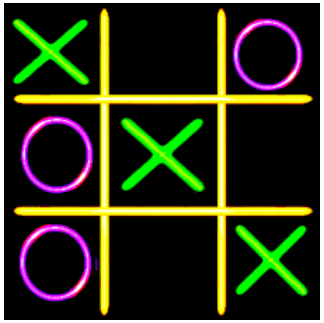


Paula

1. Definición del problema

Dos personas quieren jugar al tres-en-raya

Eva



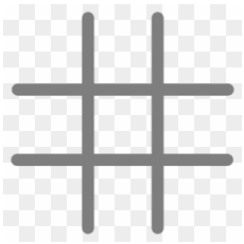
Paula

Se lanza una moneda para ver quien mueve primero … **es el turno de Eva**

1. Definición del problema

Eva

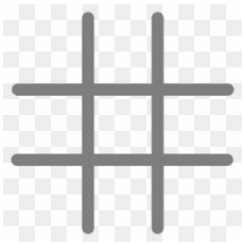
Tengo que decidir cuál va a ser mi primer movimiento

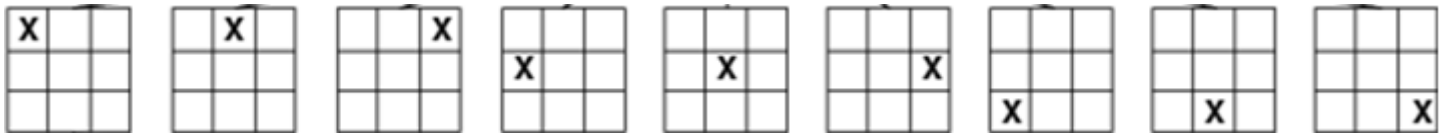


1. Definición del problema

Eva

Tengo que decidir cuál va a ser mi primer movimiento



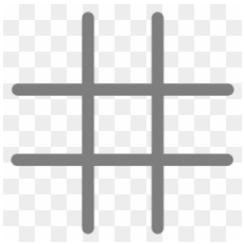


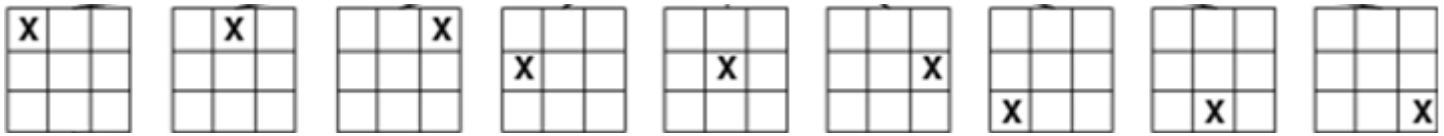
Tengo 9 posibiidades !!

1. Definición del problema

Eva

Tengo que decidir cuál va a ser mi primer movimiento





Tengo 9 posibiidades !!

¿Cómo puedo saber cuál es el **mejor movimiento**?

1. Definición del problema

Voy a simular el juego o parte de la evolución del juego
(dependiendo de cuánto tiempo tenga para realizar mi movimiento)

1. Definición del problema

Voy a simular el juego o parte de la evolución del juego
(dependiendo de cuánto tiempo tenga para realizar mi movimiento)

Voy a simular mis movimientos, luego los de Paula,
luego los míos, luego los de Paula
… y así sucesivamente

1. Definición del problema

jugador MAX

jugador MIN

1. Definición del problema

jugador MAX

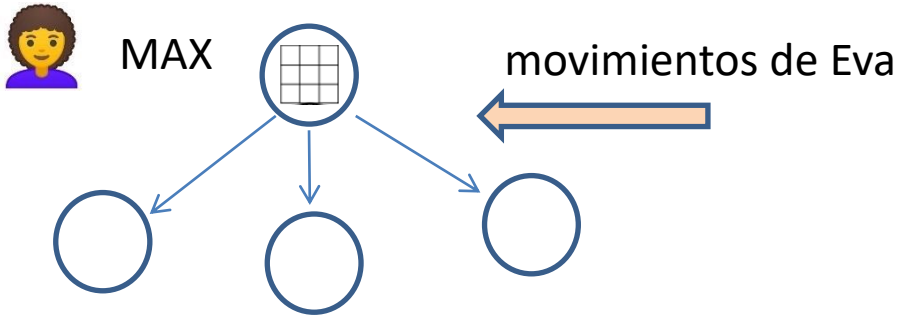
jugador MIN

MAX

1. Definición del problema

jugador MAX

jugador MIN



1. Definición del problema

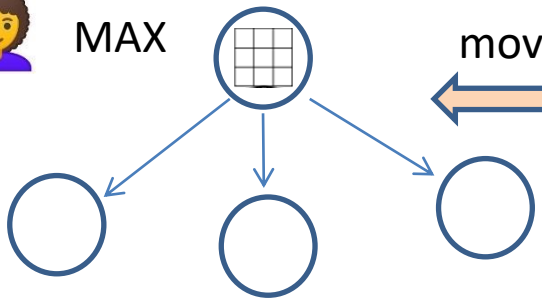
jugador MAX

jugador MIN

MAX

movimientos de Eva

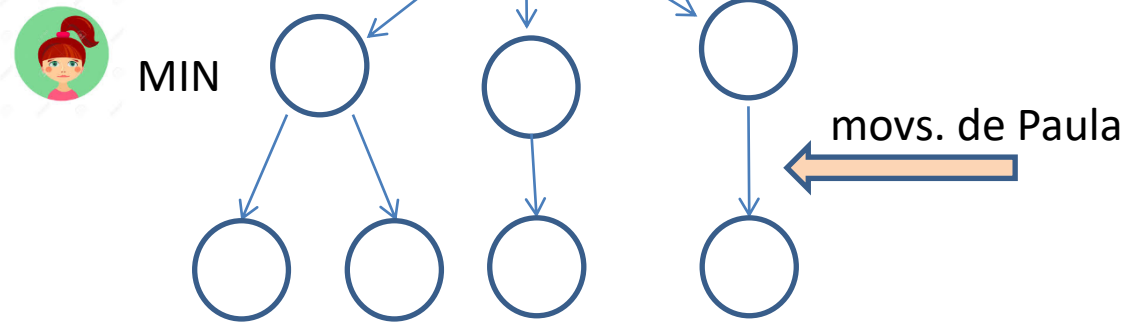
MIN



1. Definición del problema

jugador MAX

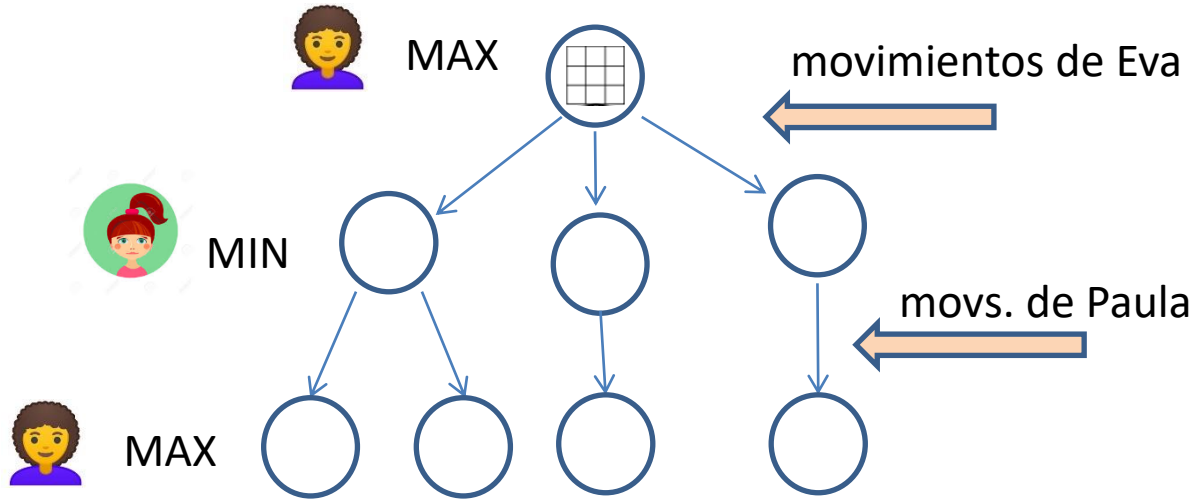
jugador MIN



1. Definición del problema

jugador MAX

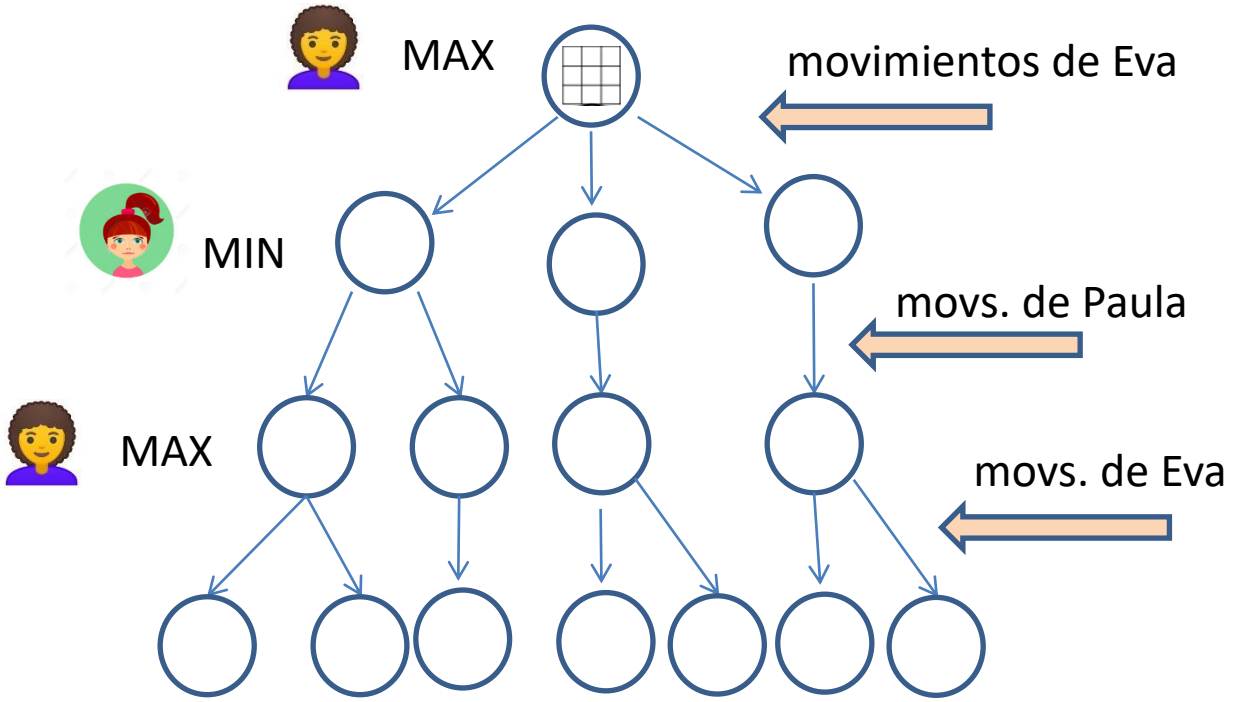
jugador MIN



1. Definición del problema

jugador MAX

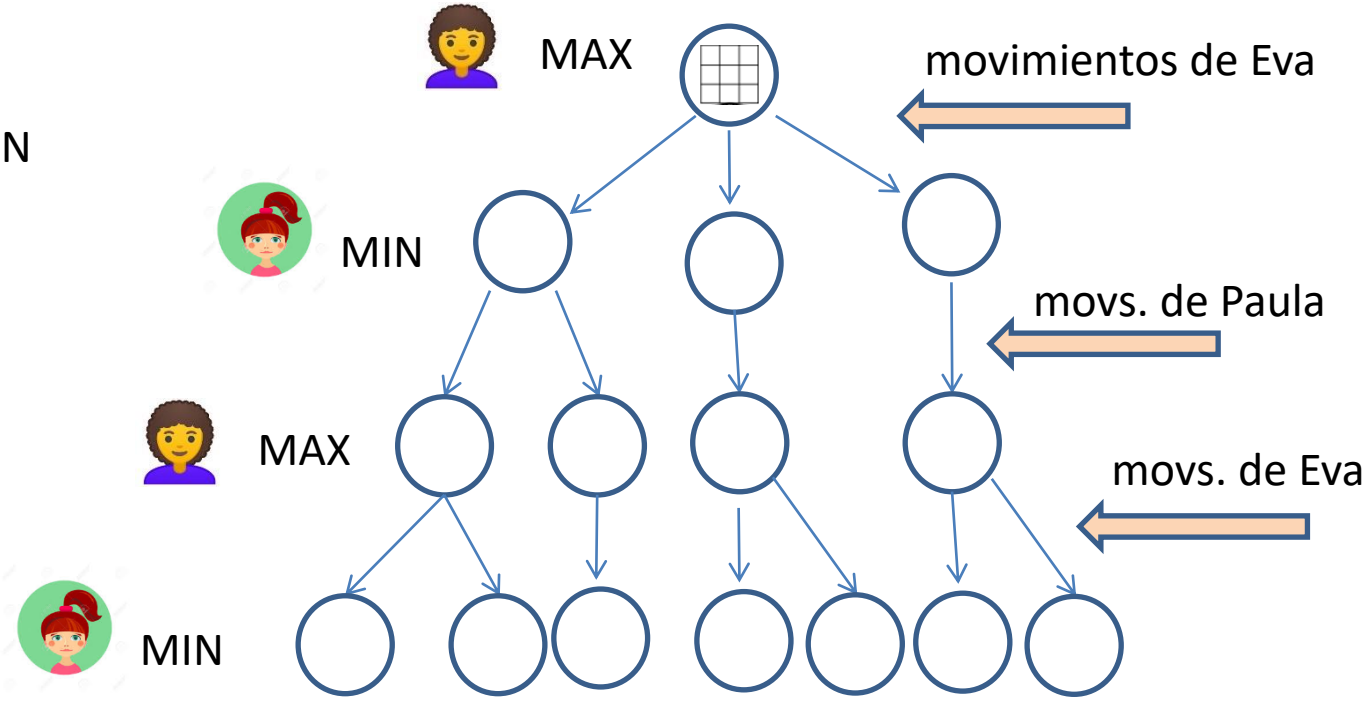
jugador MIN



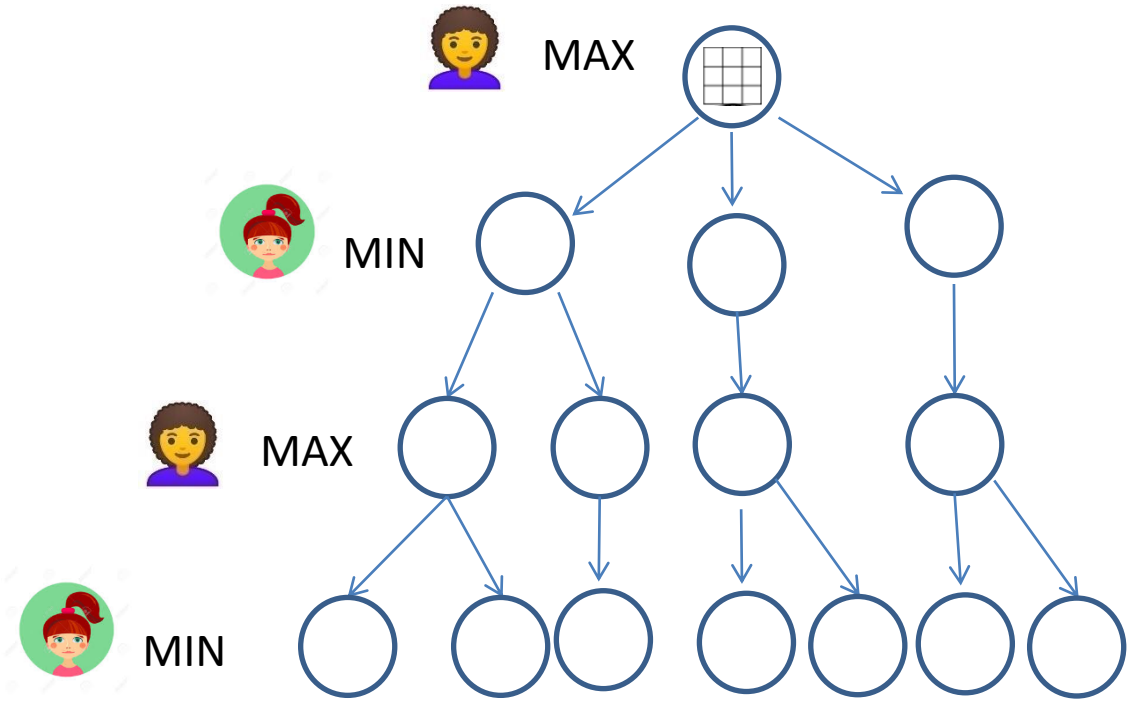
1. Definición del problema

jugador MAX

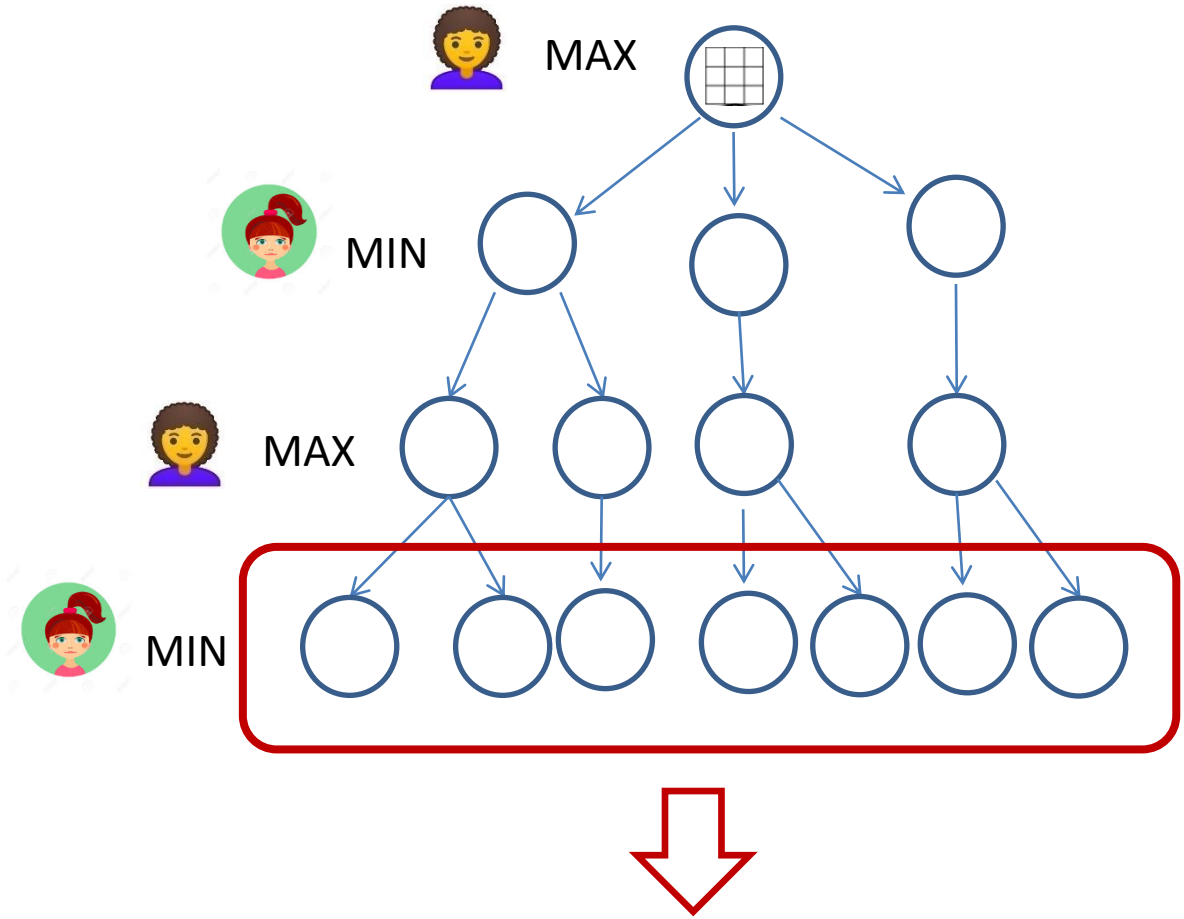
jugador MIN



1. Definición del problema

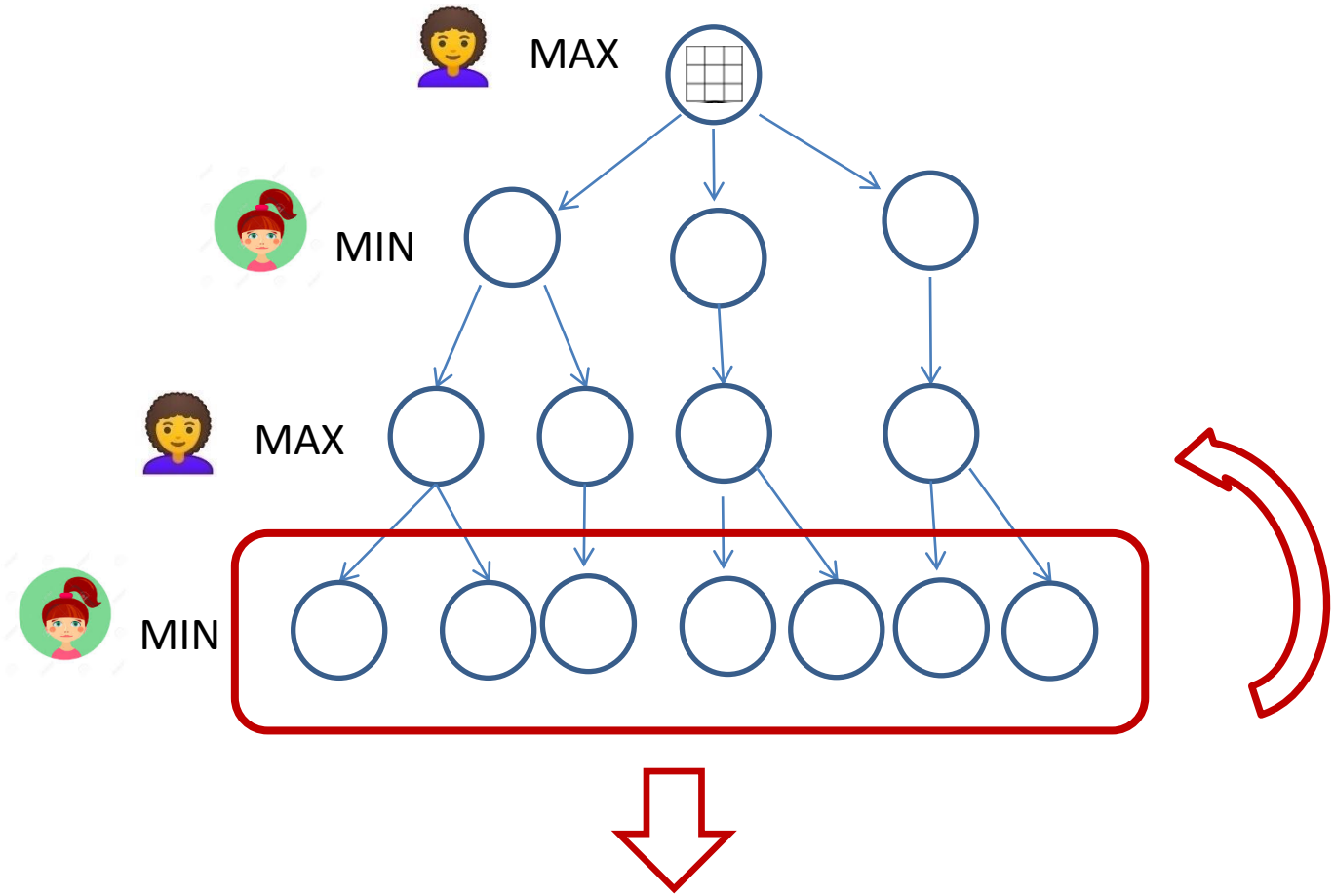


1. Definición del problema



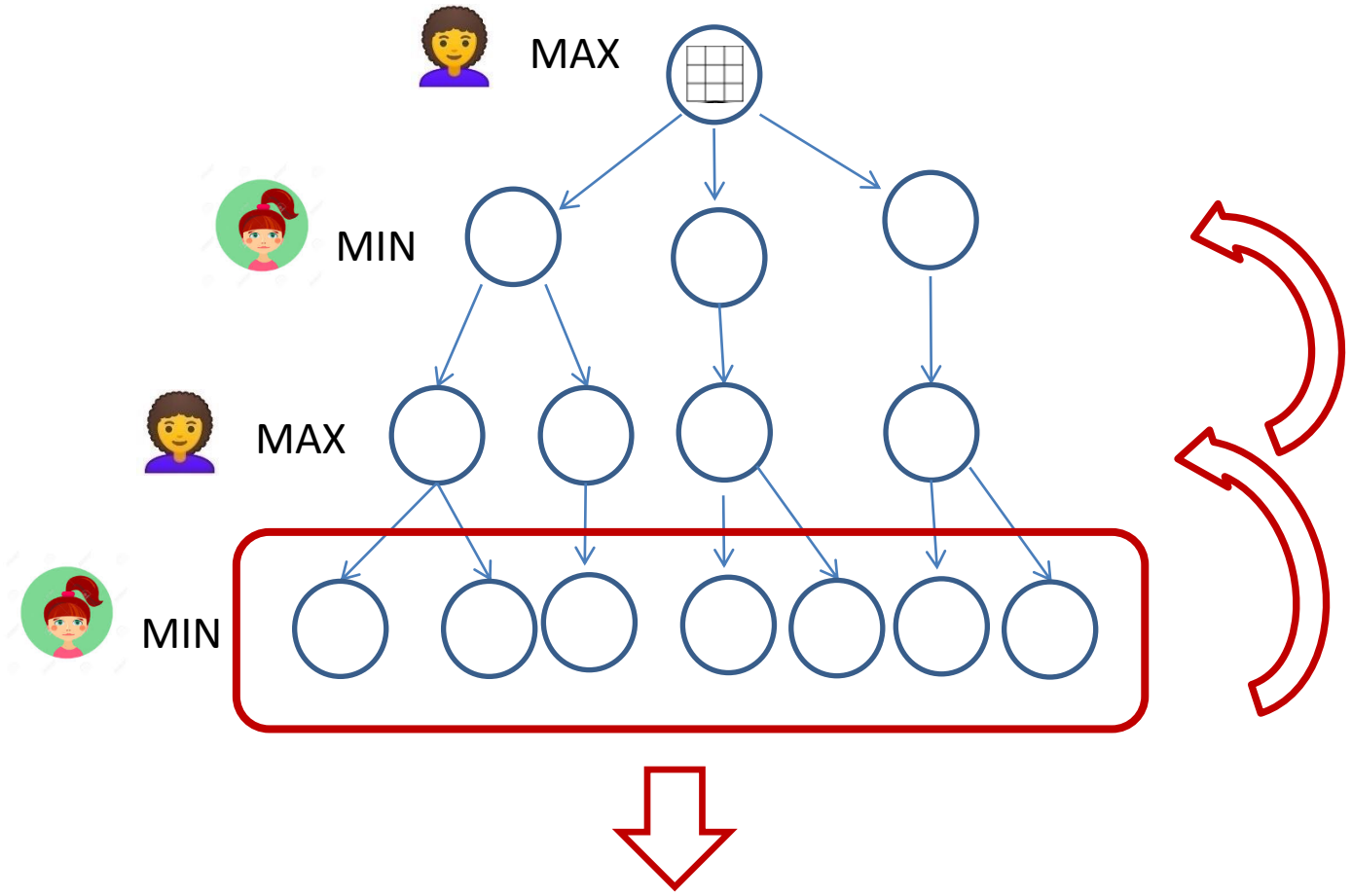
Aplicamos una función de evaluación o función de **utilidad** $f(n)$

1. Definición del problema



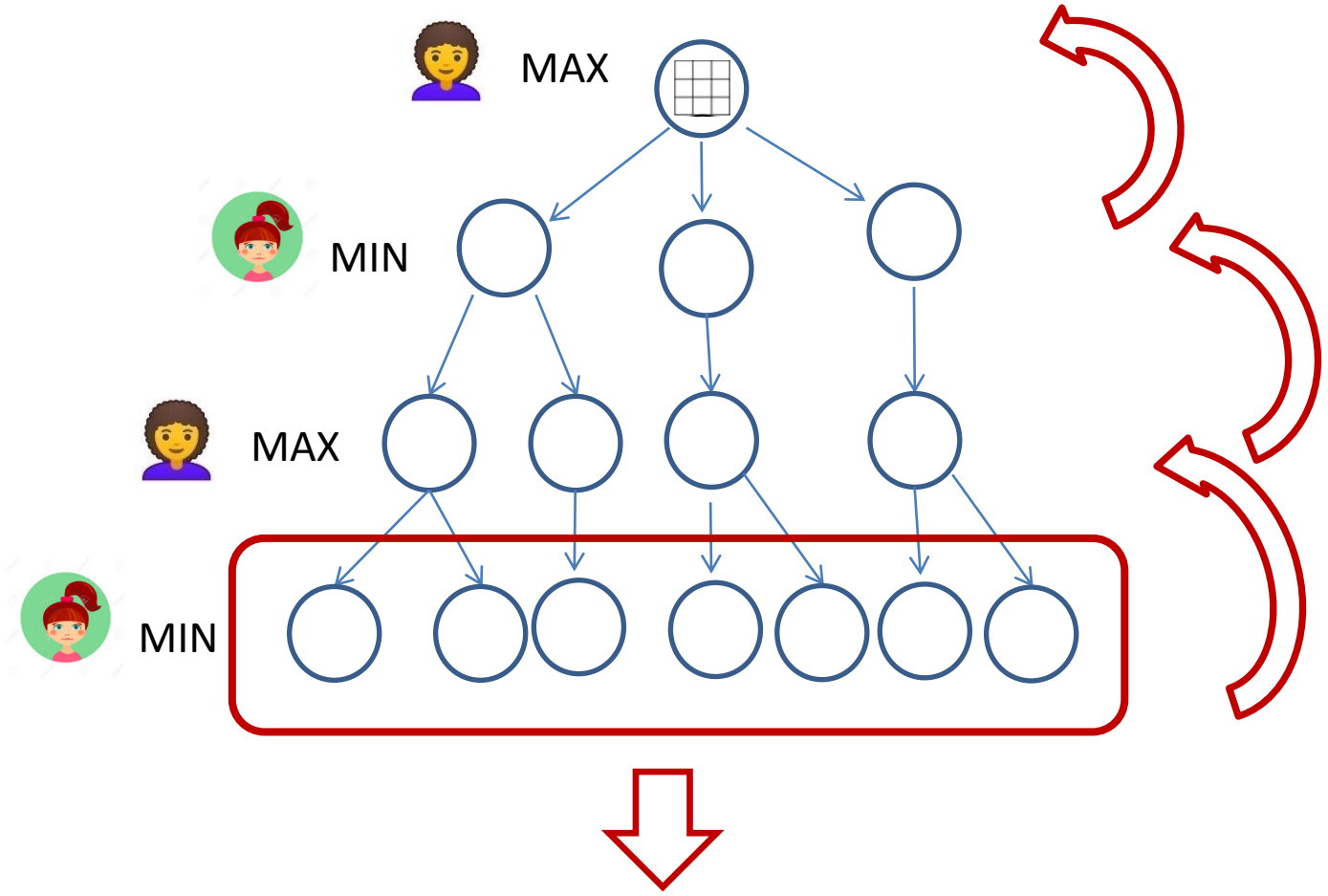
Aplicamos una función de evaluación o función de **utilidad** $f(n)$

1. Definición del problema



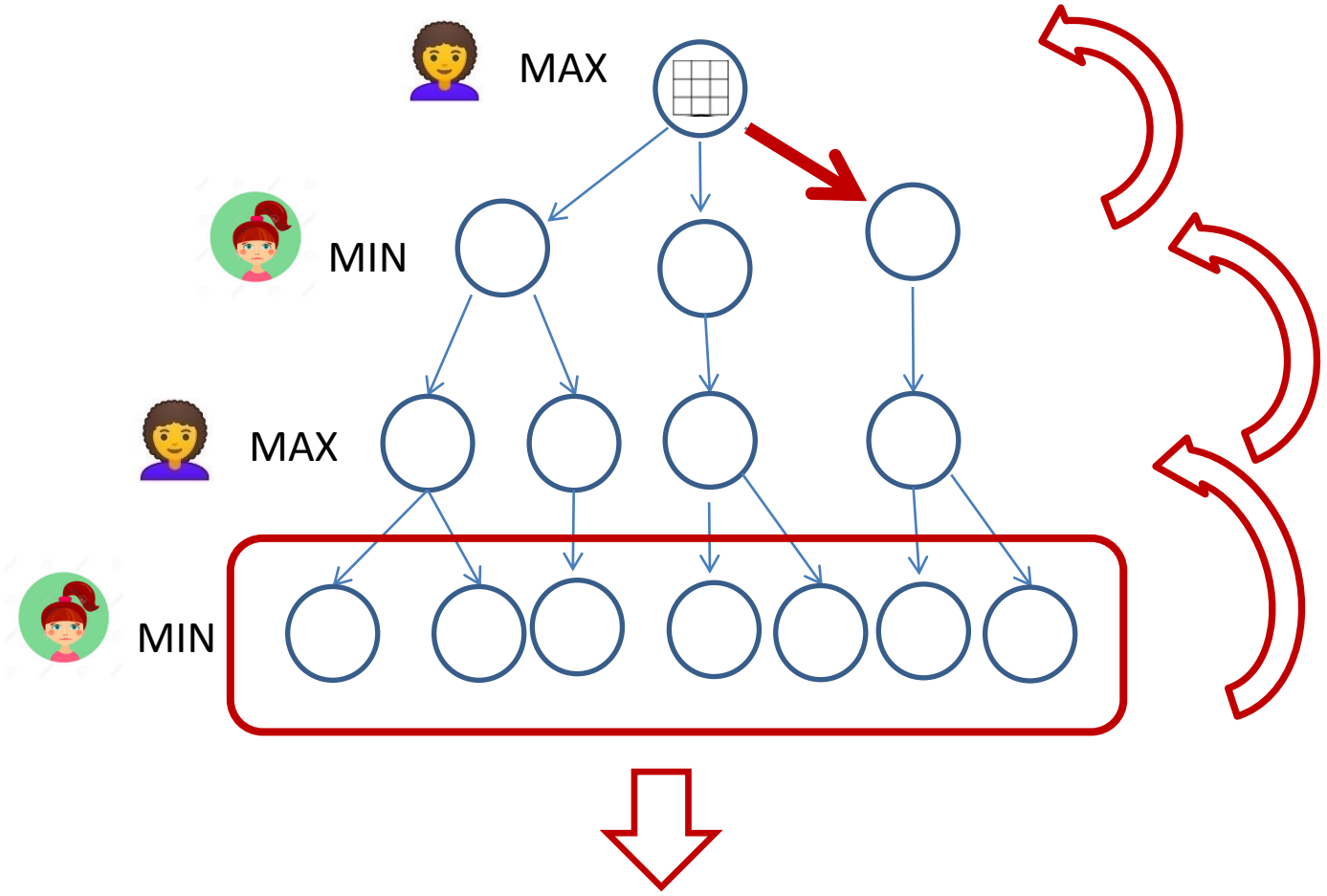
Aplicamos una función de evaluación o función de **utilidad** $f(n)$

1. Definición del problema



Aplicamos una función de evaluación o función de **utilidad** $f(n)$

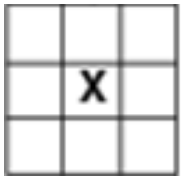
1. Definición del problema



Aplicamos una función de evaluación o función de **utilidad** $f(n)$

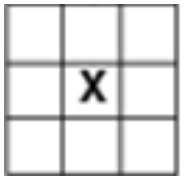
1. Definición del problema

Mi algoritmo me dice que el mejor movimiento es …



1. Definición del problema

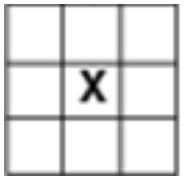
Mi algoritmo me dice que el mejor movimiento es …



Así que juego dicho movimiento ….

1. Definición del problema

Mi algoritmo me dice que el mejor movimiento es …

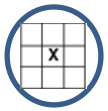


Así que juego dicho movimiento ….

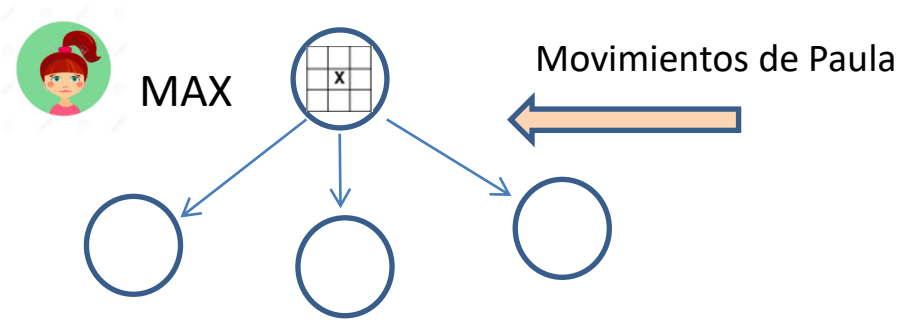
Ahora es el turno de Paula, y ella hace lo mismo que yo

1. Definición del problema

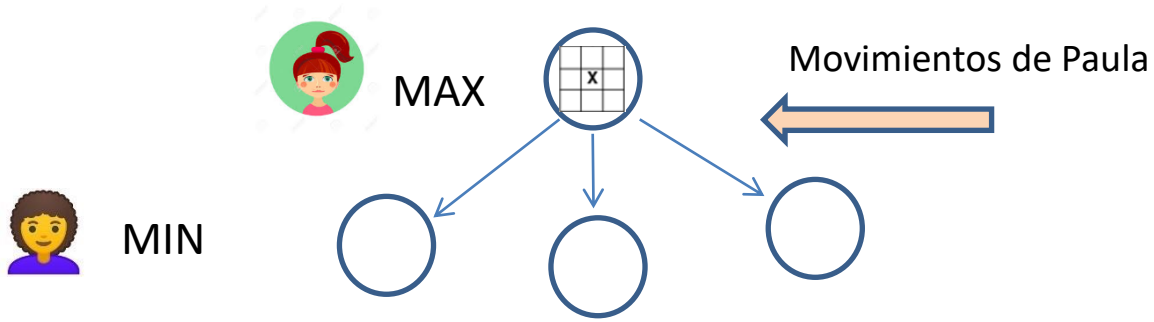
MAX



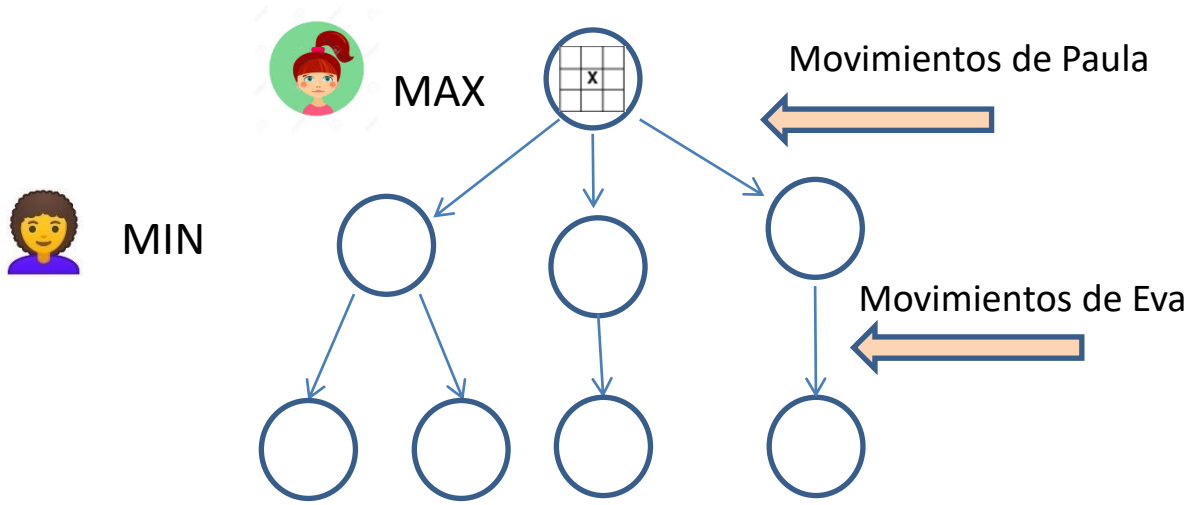
1. Definición del problema



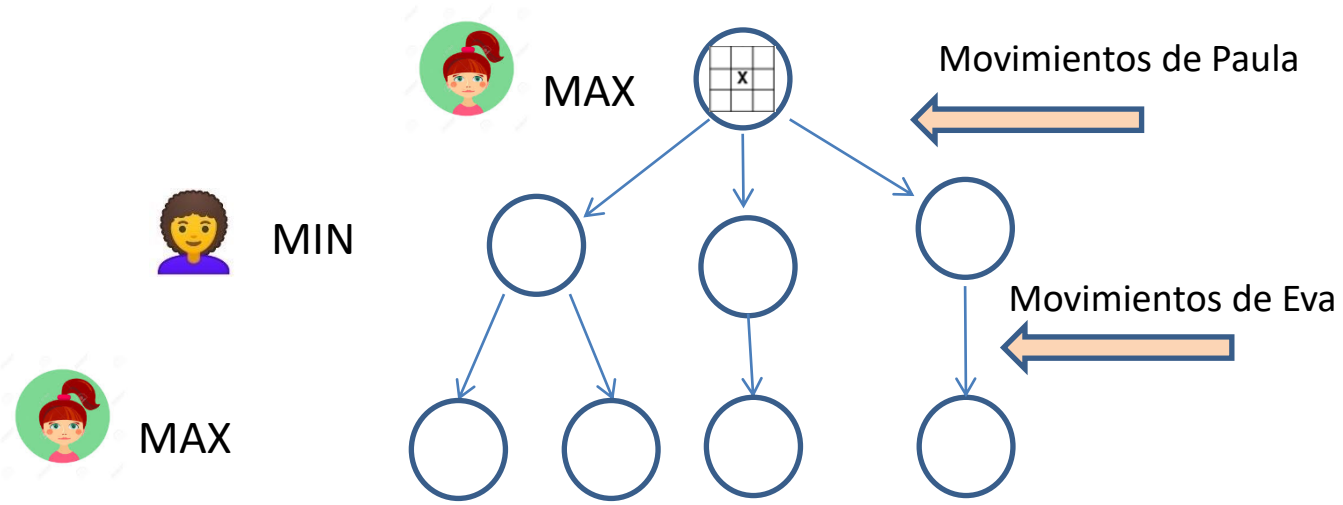
1. Definición del problema



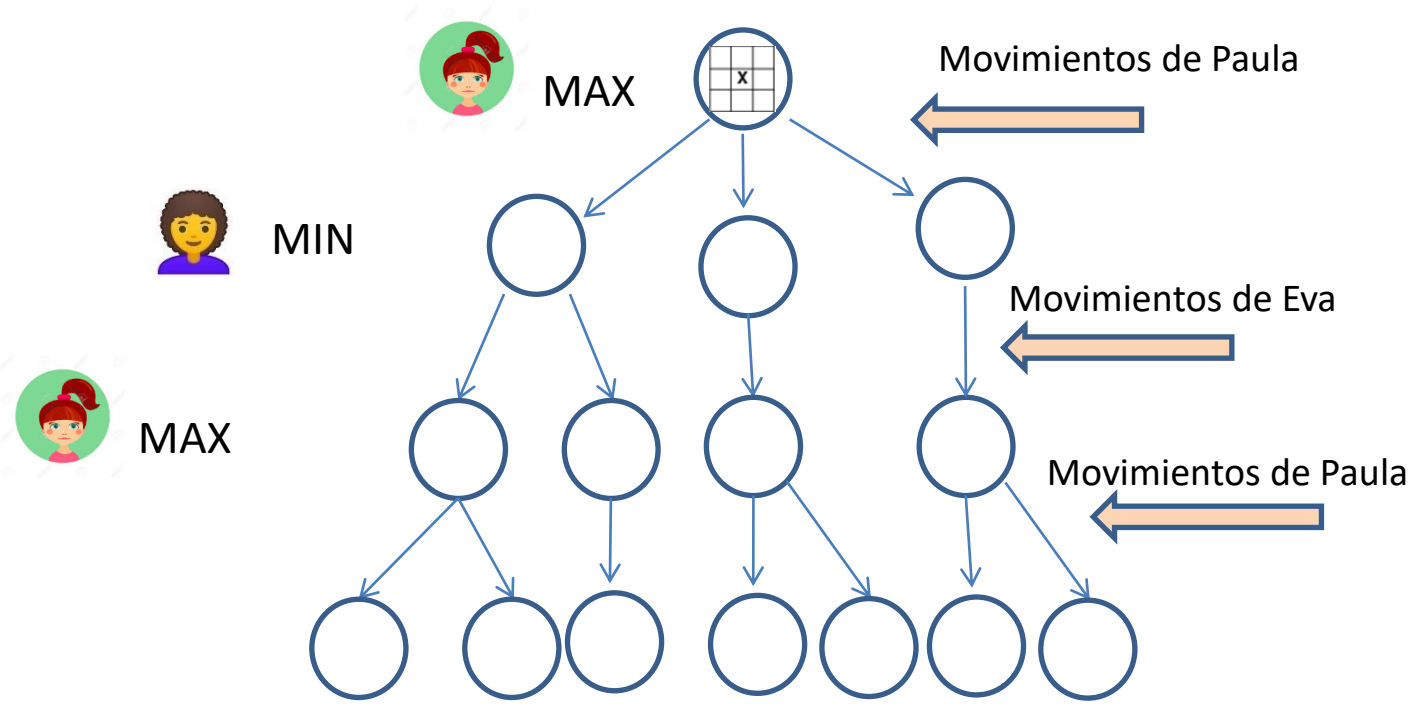
1. Definición del problema



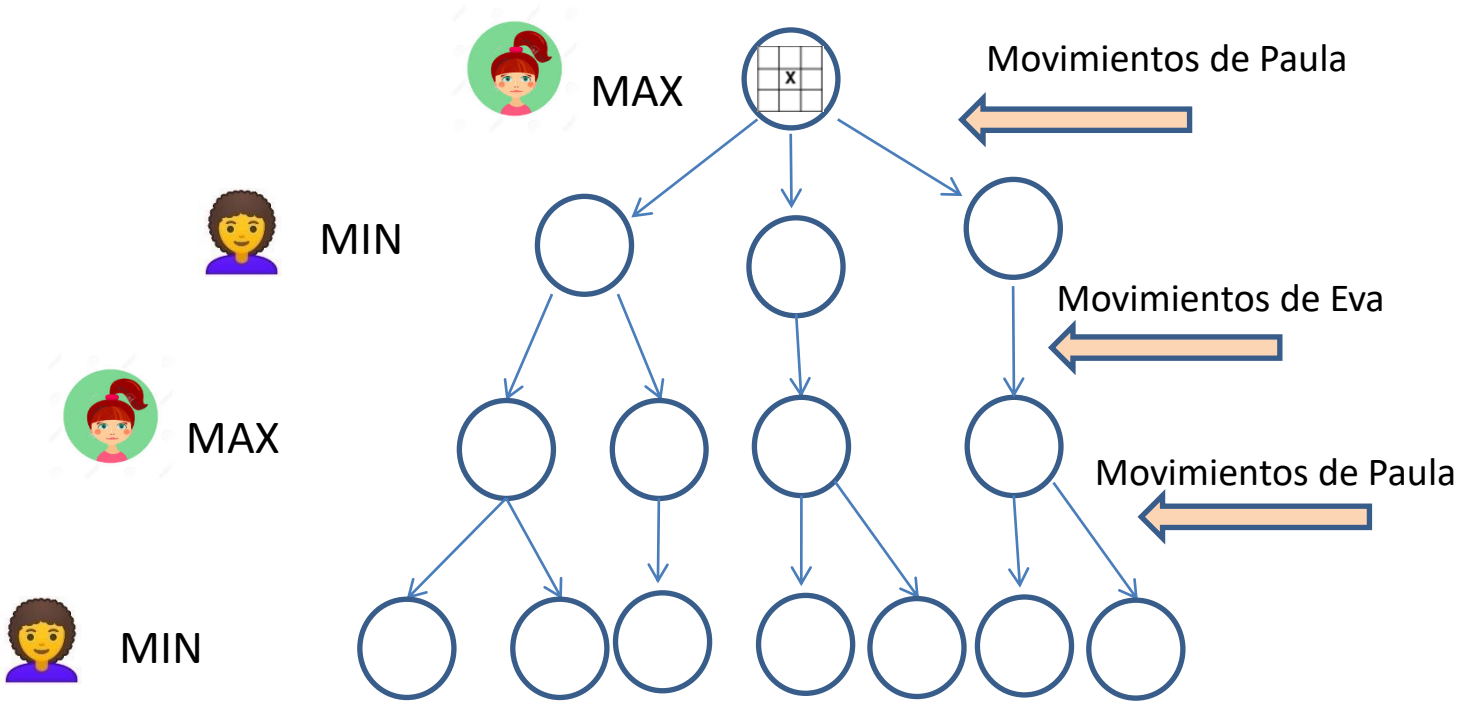
1. Definición del problema



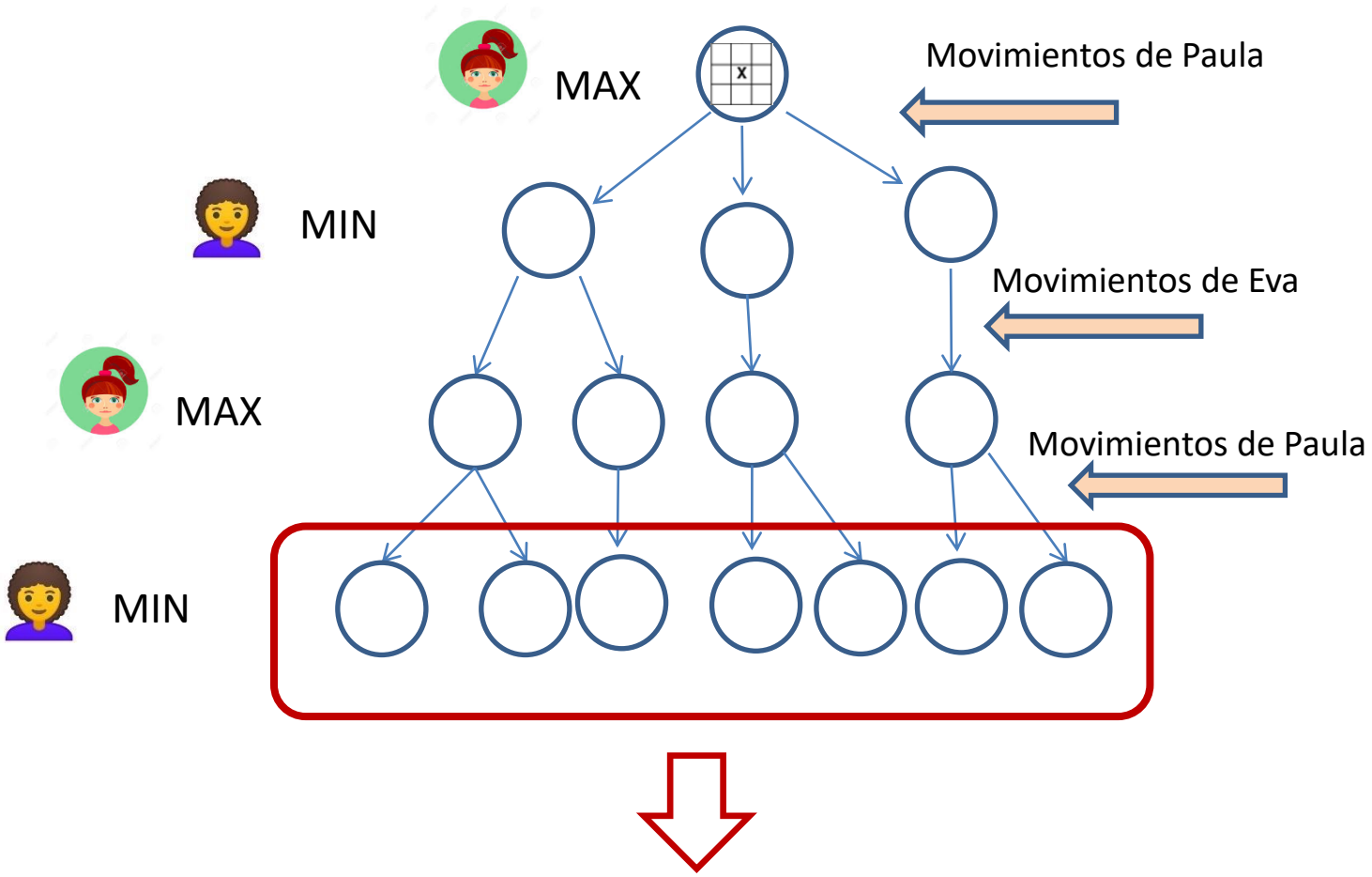
1. Definición del problema



1. Definición del problema

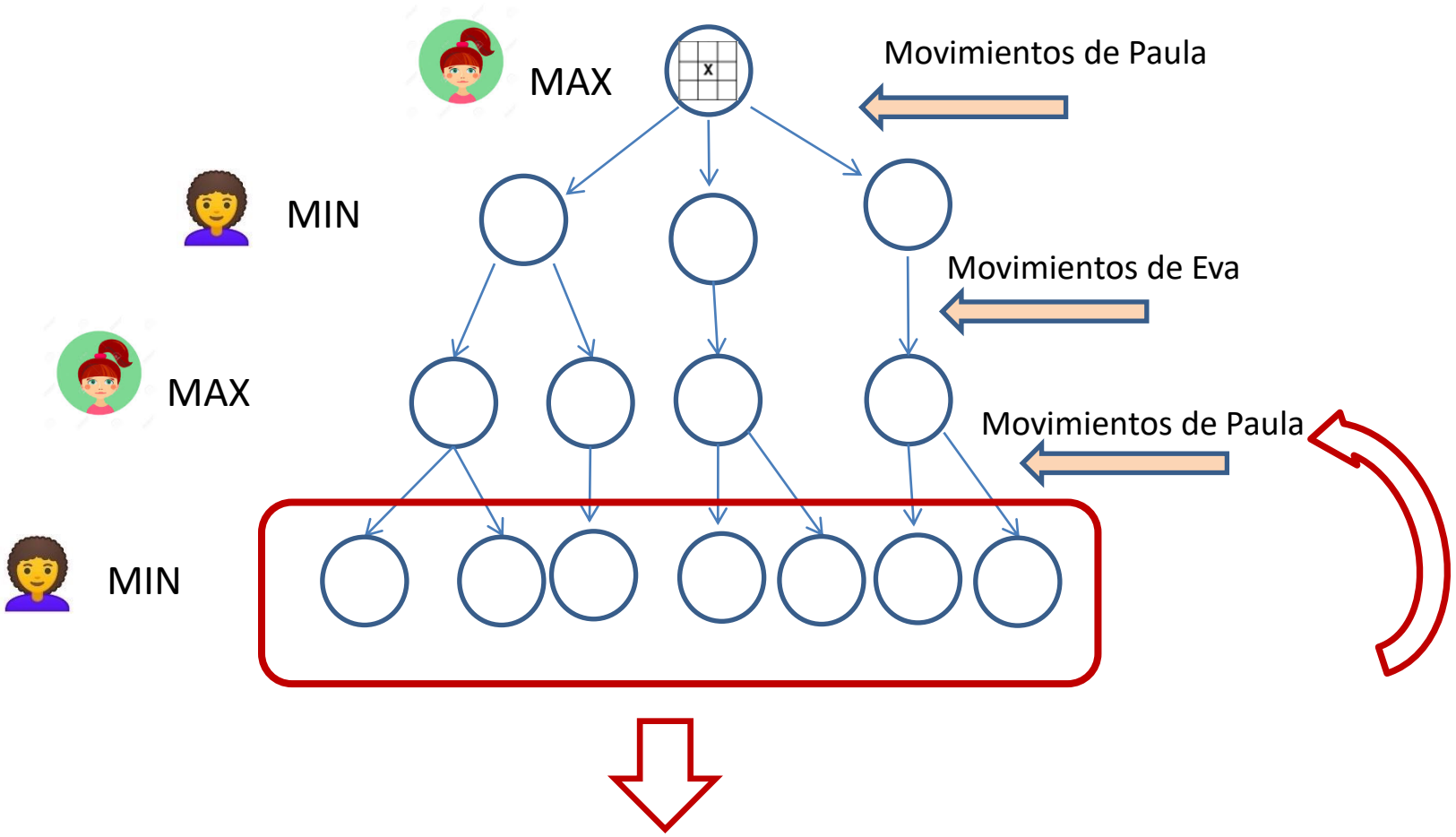


1. Definición del problema



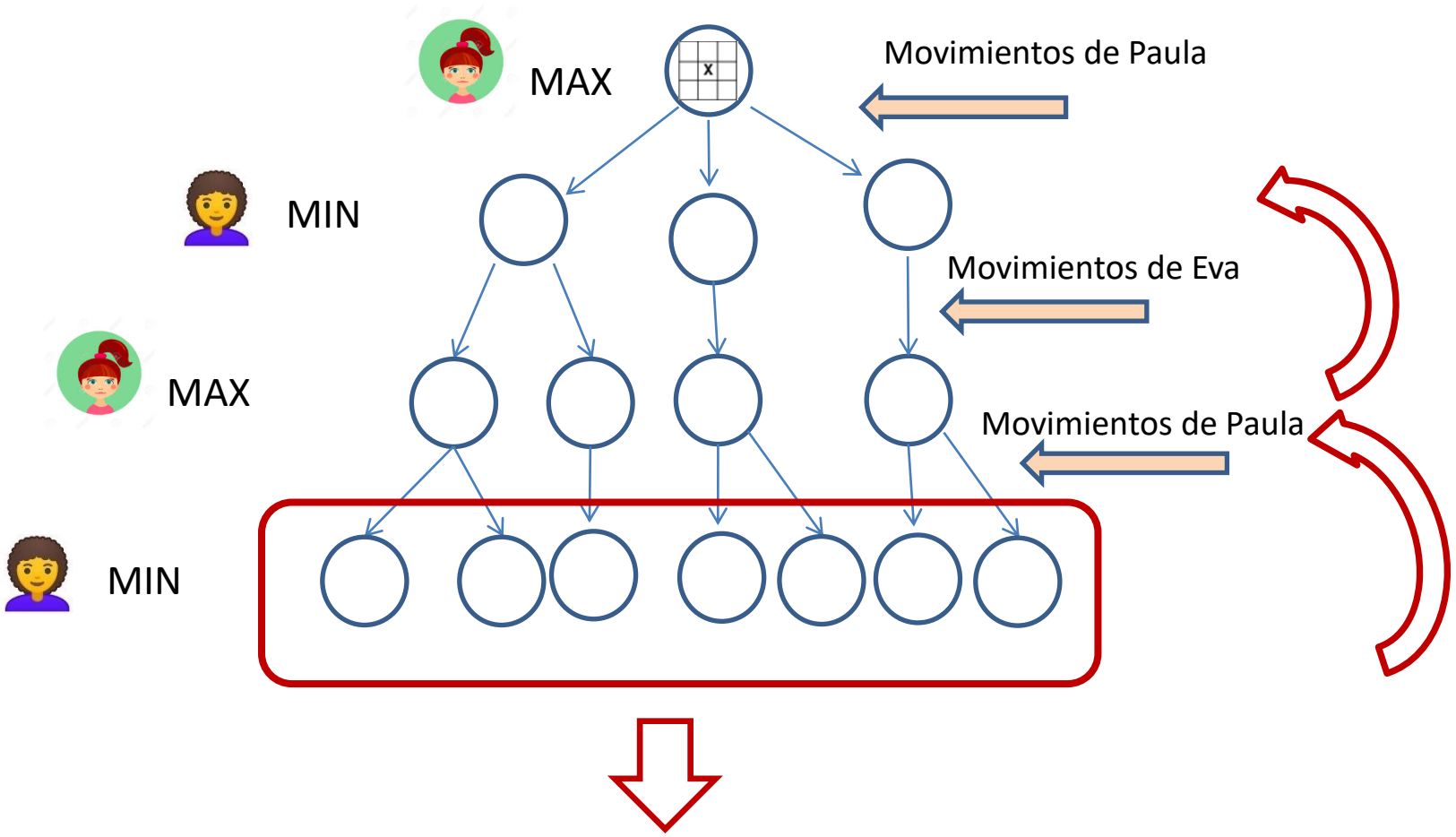
Aplicamos una función de evaluación o función de **utilidad** $f(n)$

1. Definición del problema



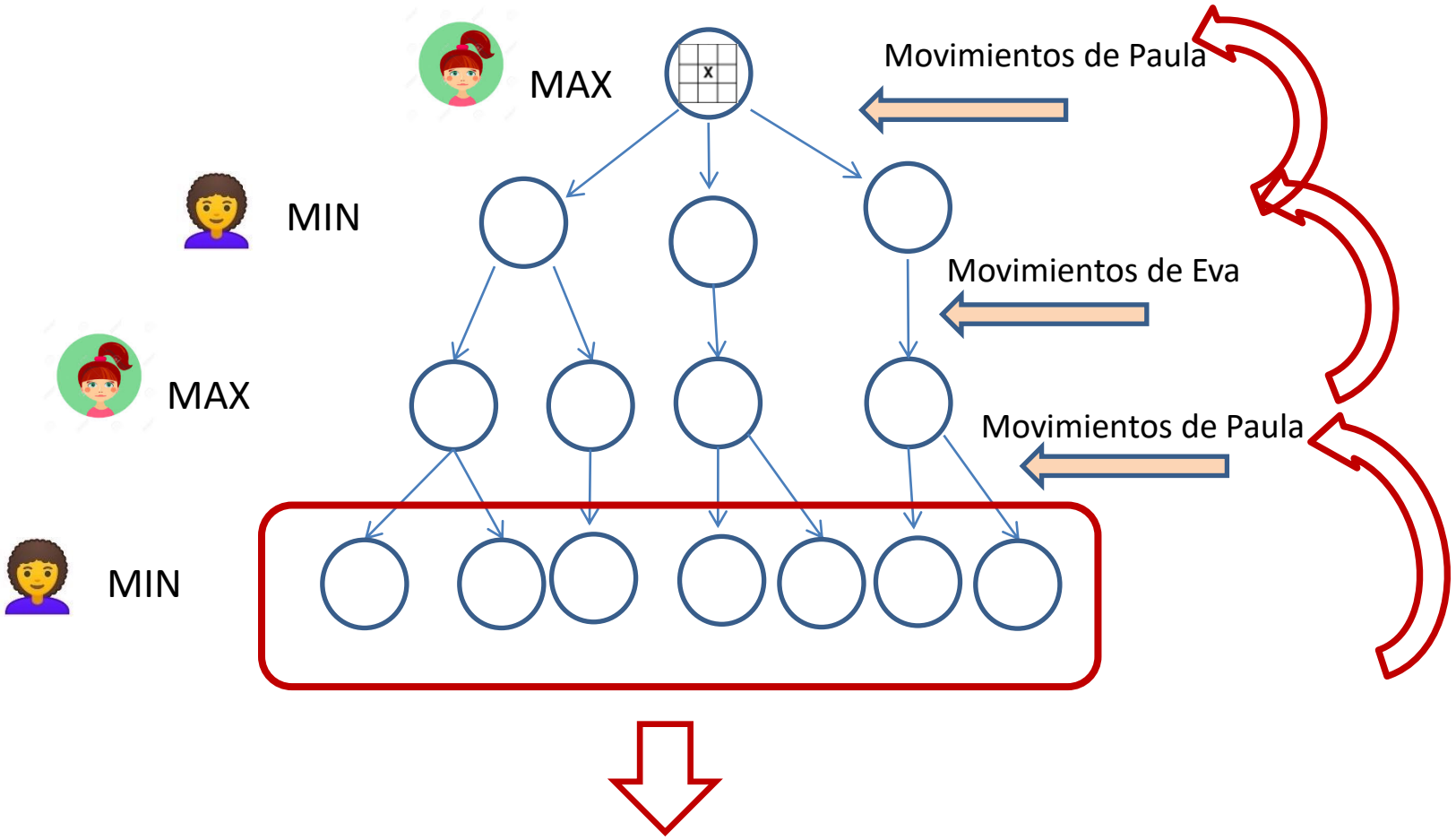
Aplicamos una función de evaluación o función de **utilidad** $f(n)$

1. Definición del problema



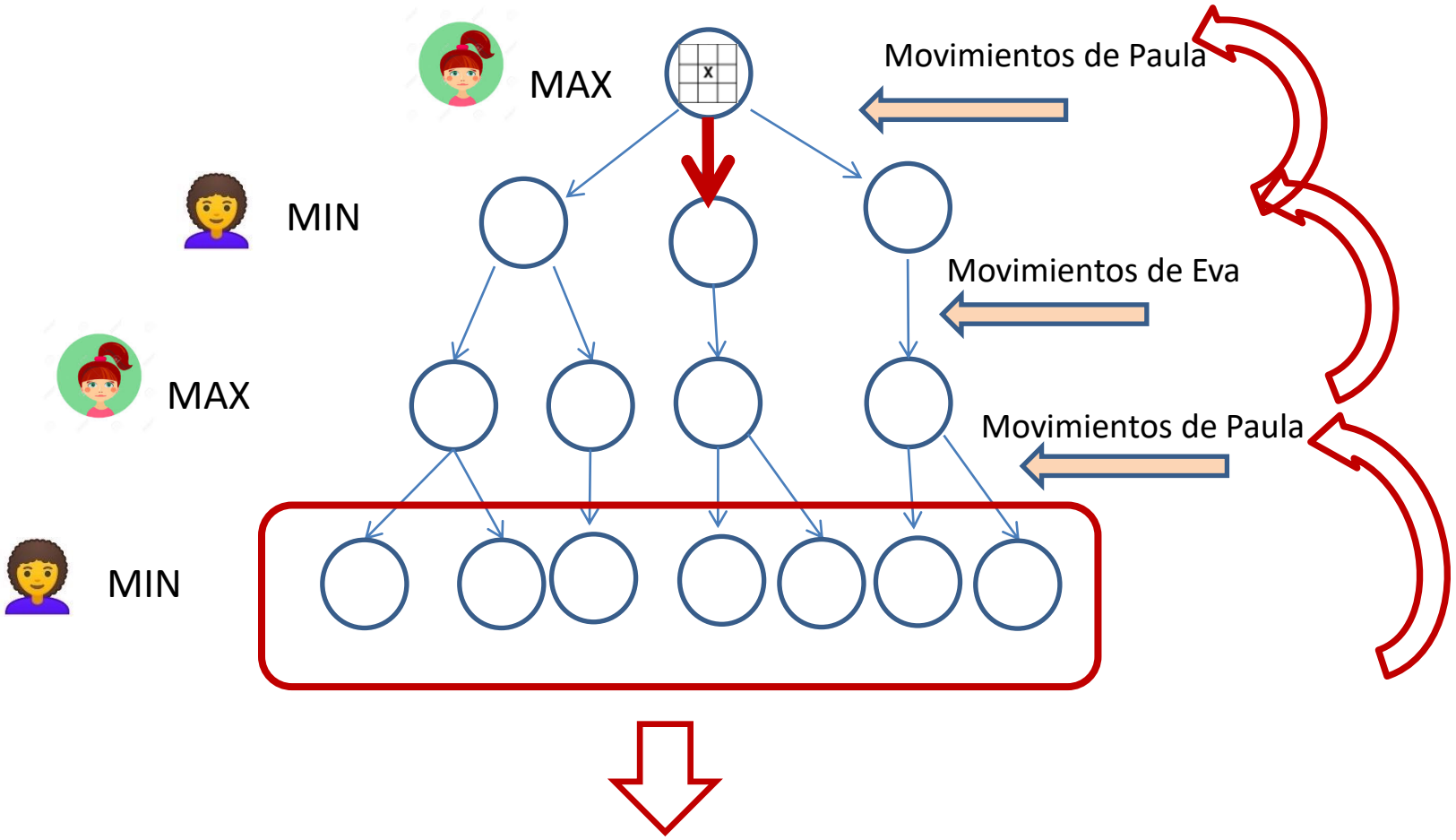
Aplicamos una función de evaluación o función de **utilidad** $f(n)$

1. Definición del problema



Aplicamos una función de evaluación o función de **utilidad** $f(n)$

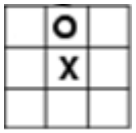
1. Definición del problema



Aplicamos una función de evaluación o función de **utilidad** $f(n)$

1. Definición del problema

Y Paula juega ….



1. Definición del problema: la función de utilidad

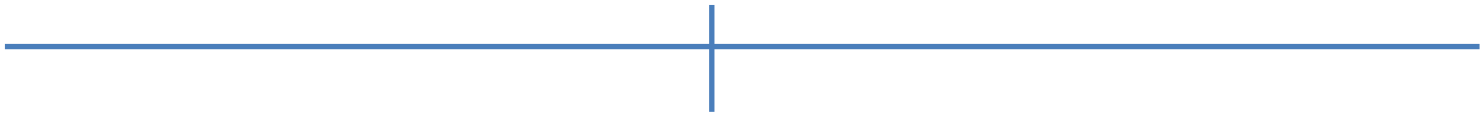
Como estamos en **juegos de dos jugadores**, la idea es definir una función que:

- valores positivos son buenos para MAX
- valores negativos son buenos para MIN

1. Definición del problema: la función de utilidad

Como estamos en **juegos de dos jugadores**, la idea es definir una función que:

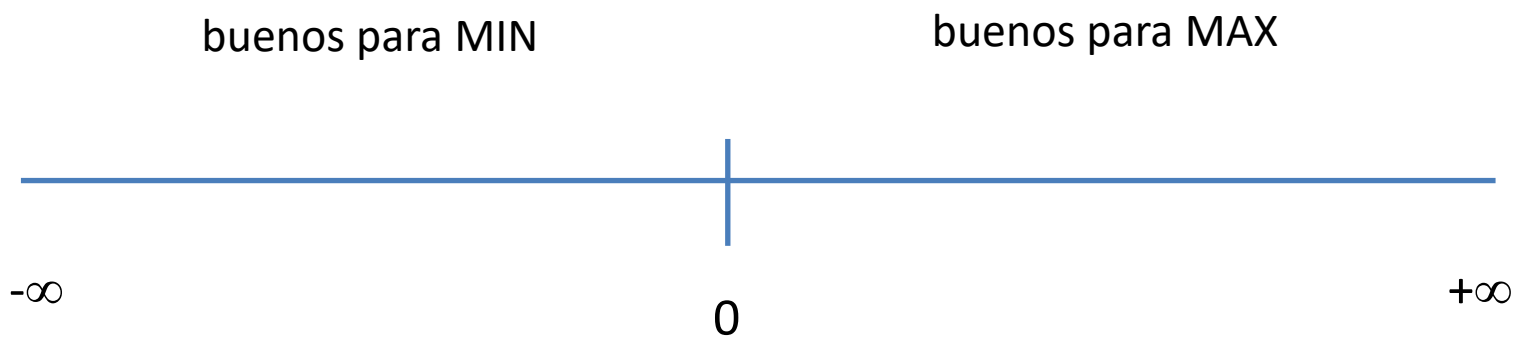
- valores positivos son buenos para MAX
- valores negativos son buenos para MIN



1. Definición del problema: la función de utilidad

Como estamos en **juegos de dos jugadores**, la idea es definir una función que:

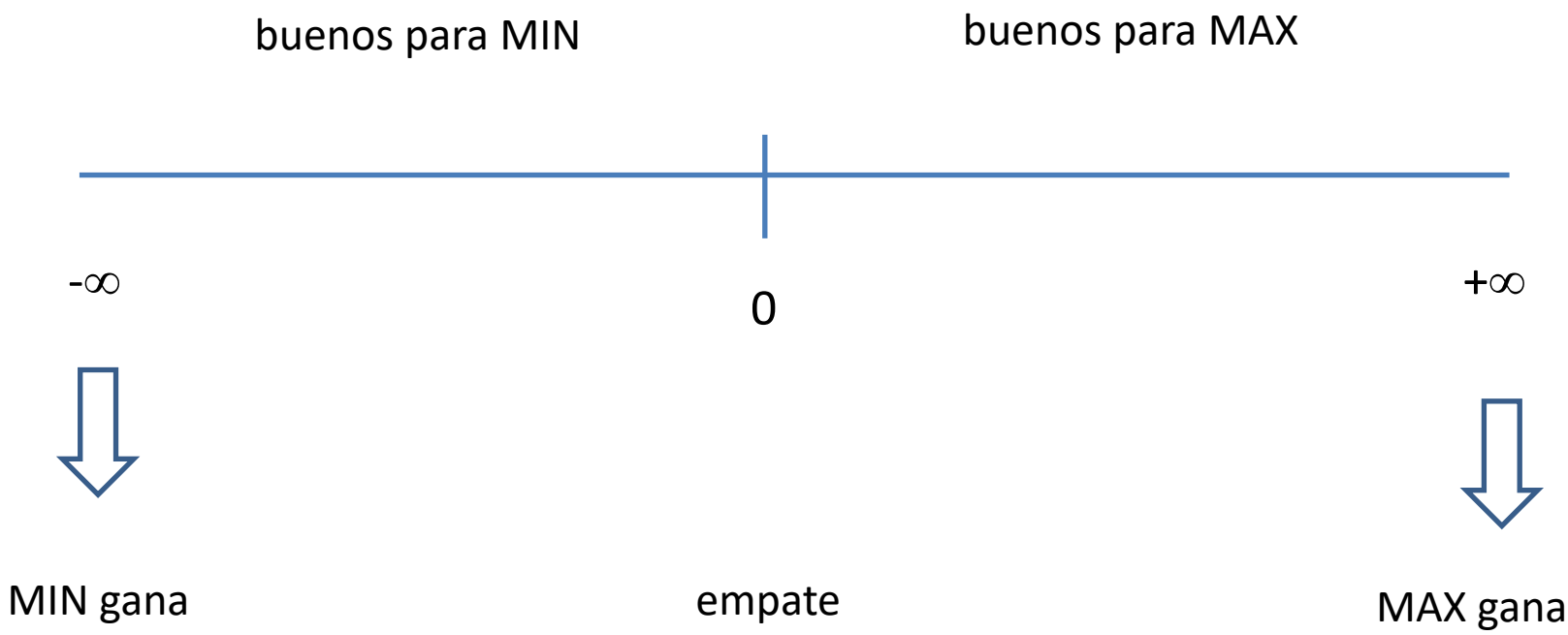
- valores positivos son buenos para MAX
- valores negativos son buenos para MIN



1. Definición del problema: la función de utilidad

Como estamos en **juegos de dos jugadores**, la idea es definir una función que:

- valores positivos son buenos para MAX
- valores negativos son buenos para MIN



1. Definición del problema: la función de utilidad

Max: X **MIN: 0**

f(n) =
(número de filas, columnas y diagonales abiertas para MAX) **–**
(número de filas, columnas y diagonales abiertas para MIN)

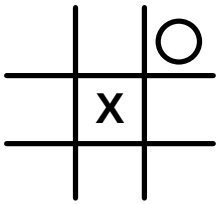
f(n)= +∞, si es una jugada ganadora para MAX
f(n)= - ∞, si es una jugada ganadora para MIN

1. Definición del problema: la función de utilidad

Max: X **MIN: 0**

f(n) =
(número de filas, columnas y diagonales abiertas para MAX) **–**
(número de filas, columnas y diagonales abiertas para MIN)

f(n)= +∞, si es una jugada ganadora para MAX
f(n)= - ∞, si es una jugada ganadora para MIN



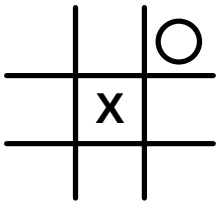
f(n)= 5 – 4 = 1

1. Definición del problema: la función de utilidad

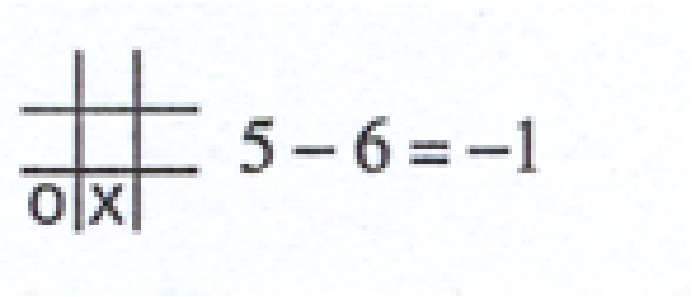
Max: X **MIN: 0**

f(n) =
(número de filas, columnas y diagonales abiertas para MAX) –
(número de filas, columnas y diagonales abiertas para MIN)

f(n)= +∞, si es una jugada ganadora para MAX
f(n)= - ∞, si es una jugada ganadora para MIN



f(n)= 5 – 4 = 1



2. Algoritmo Minimax

Cuando se busca la solución óptima para MAX (mejor jugada inicial para MAX), MIN tiene algo que decir al respecto. A grandes rasgos, una estrategia óptima conduce a resultados al menos tan buenos como cualquier otra estrategia cuando se juega contra un adversario infalible.

Es inviable, incluso para juegos sencillos, dibujar todo el árbol del juego.

2. Algoritmo Minimax

1. Generar en **ANCHURA el árbol del juego hasta un cierto nivel de profundidad** (hasta alcanzar los nodos terminales del nivel máximo de profundidad).
2. Aplicar la función de utilidad en cada nodo terminal para obtener su valor.
3. Usar los valores de utilidad de los nodos terminales para determinar la utilidad de los nodos del nivel superior. Volcar los valores de utilidad desde los nodos hoja hasta el raíz, nivel por nivel (en **PROFUNDIDAD**). En los nodos MIN se vuelca el menor valor de utilidad de sus sucesores; en los nodos MAX se vuelca el mayor valor de utilidad de sus sucesores.

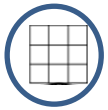
Valor_Minimax(n)=

$$\begin{cases} \text{Utilidad}(n) & \text{si } n \text{ es un nodo terminal} \\ \max_{s \in \text{Sucesor}(n)} \text{Valor\_Minimax}(s) & \text{si } n \text{ es un nodo MAX} \\ \min_{s \in \text{Succsor}(n)} \text{Valor\_Minimax}(s) & \text{si } n \text{ es un nodo MIN} \end{cases}$$

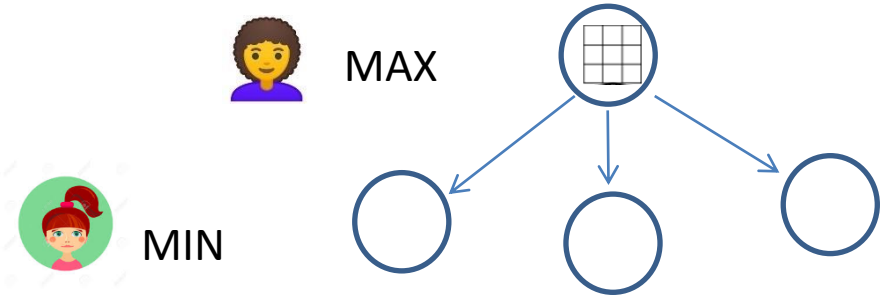
4. Eventualmente, el nodo raíz MAX adquiere un valor volcado. En este punto, MAX escoge el movimiento que le conduce al estado de máxima utilidad; es decir, escoge la acción correspondiente al mejor movimiento, el movimiento que reporta la mayor utilidad asumiendo que el oponente juega a minimizar la utilidad.

2. Algoritmo Minimax

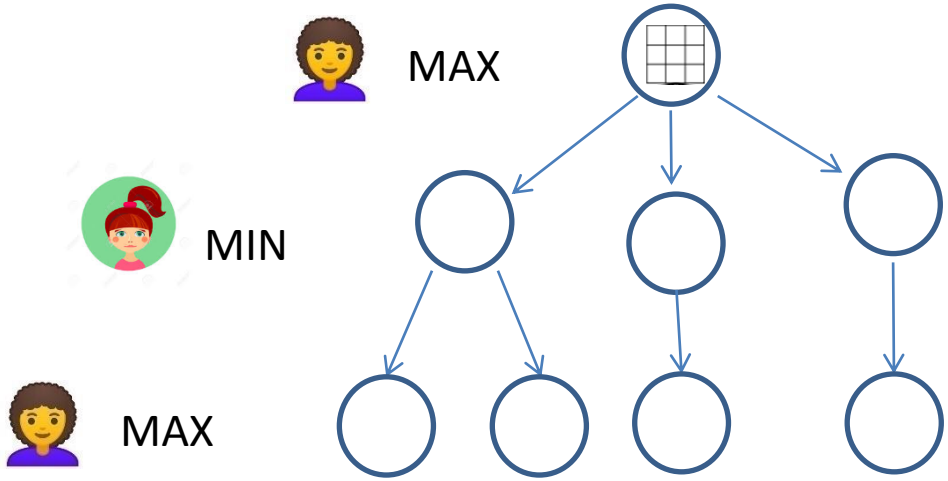
MAX



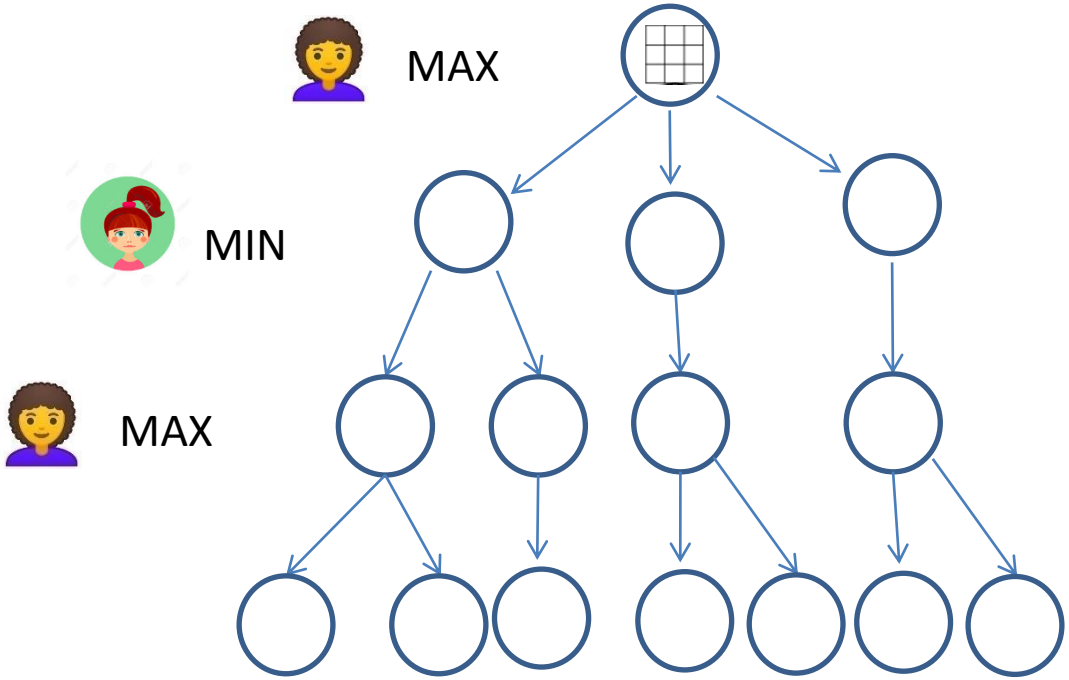
2. Algoritmo Minimax



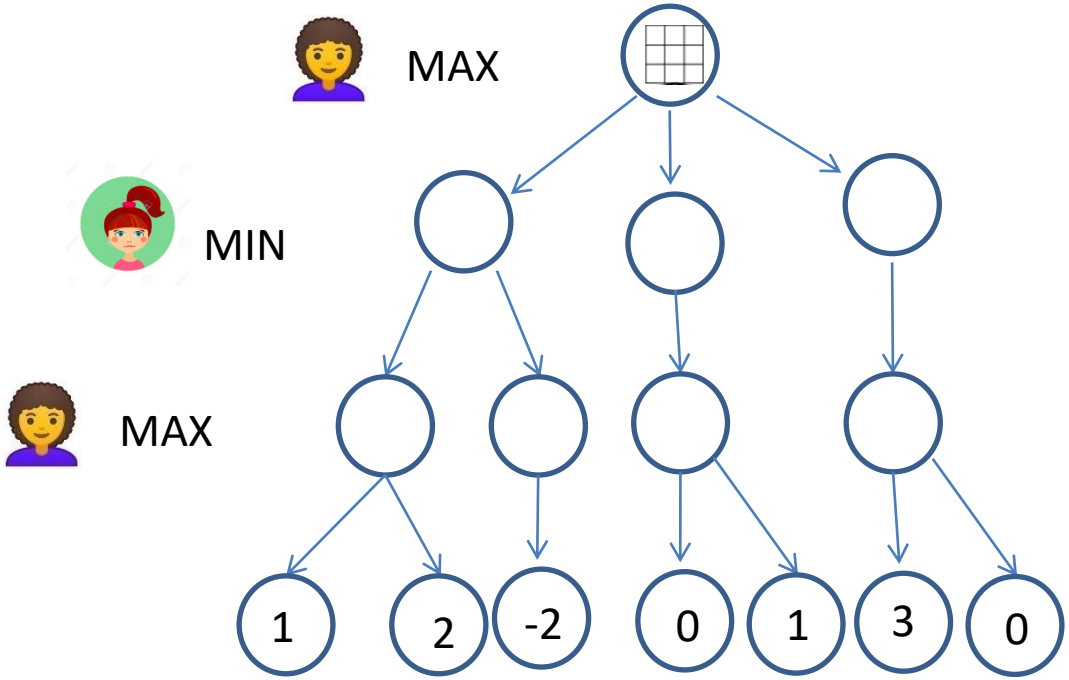
2. Algoritmo Minimax



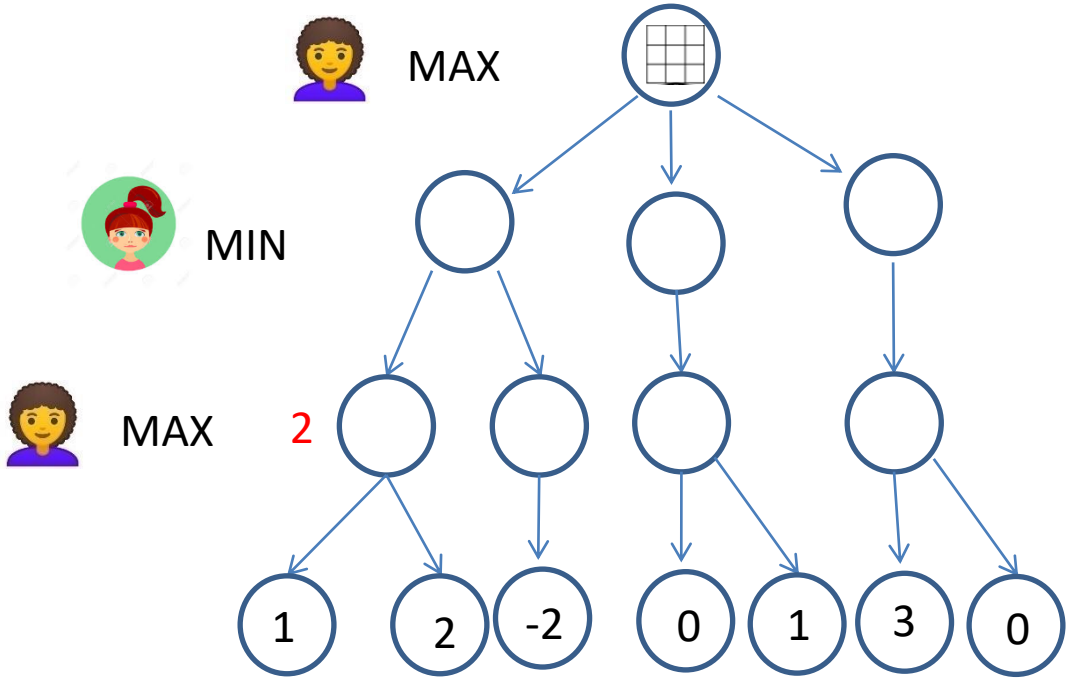
2. Algoritmo Minimax



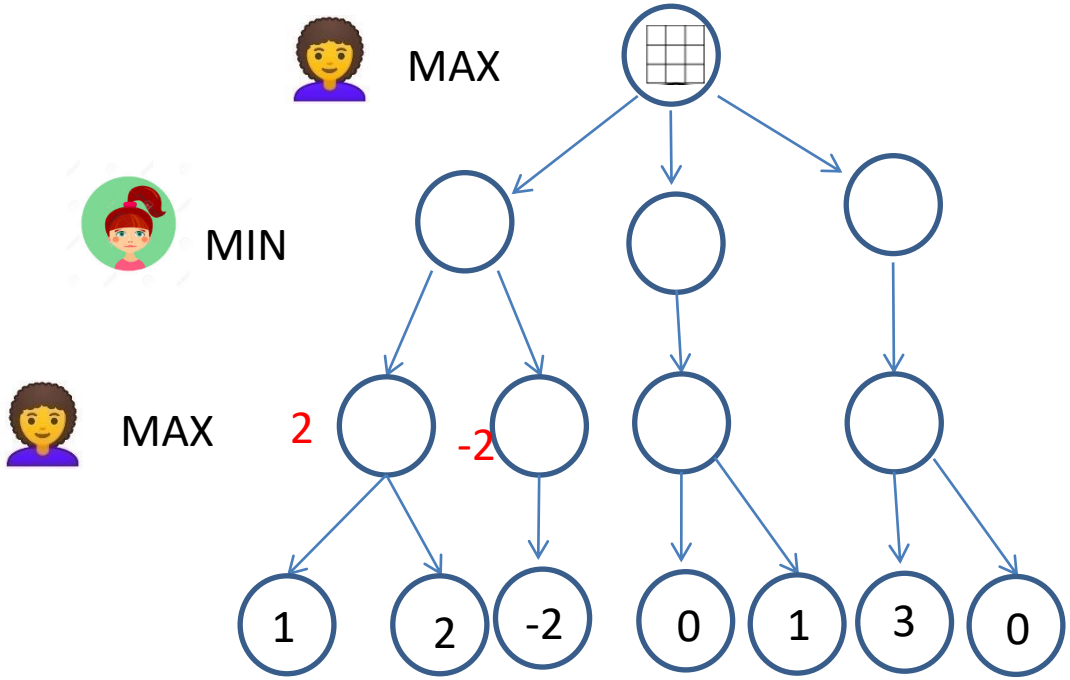
2. Algoritmo Minimax



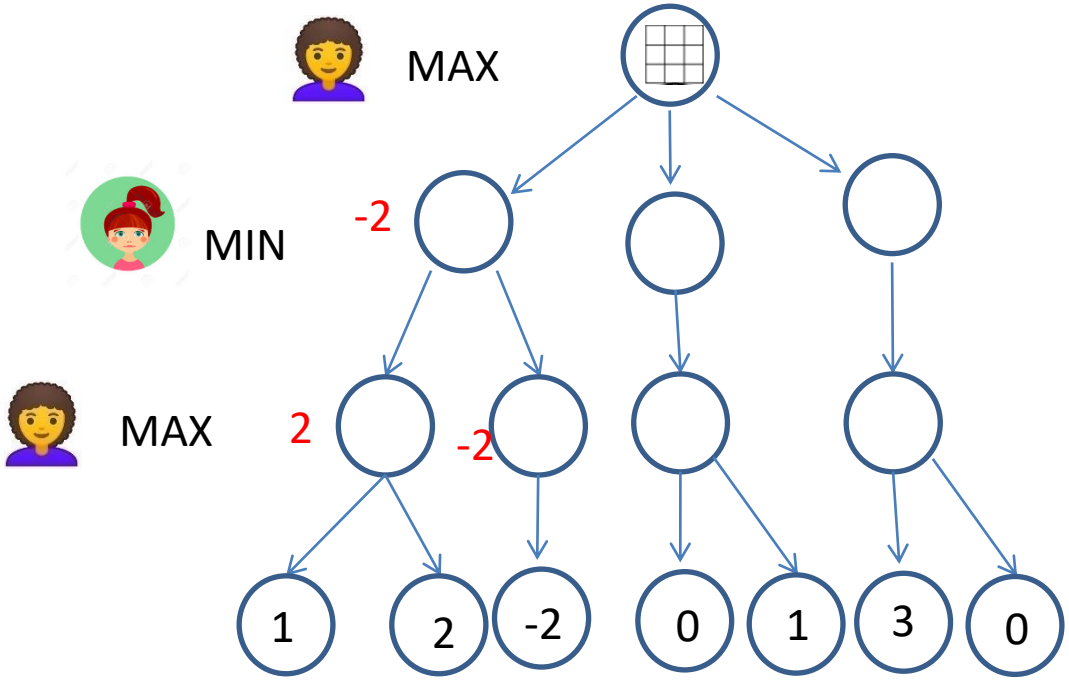
2. Algoritmo Minimax



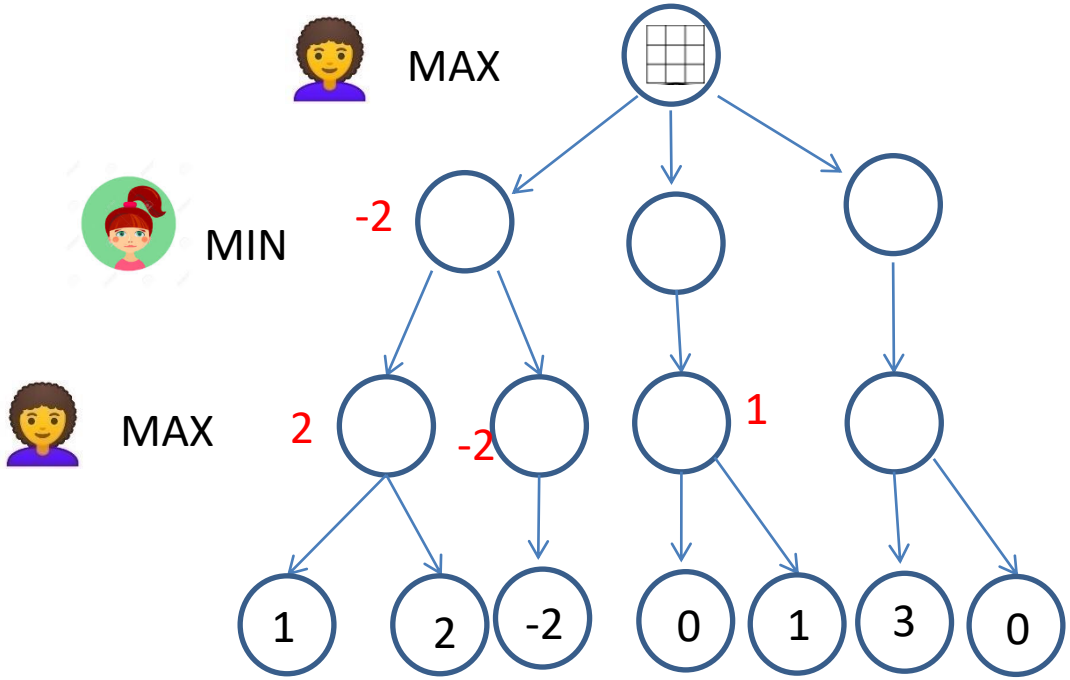
2. Algoritmo Minimax



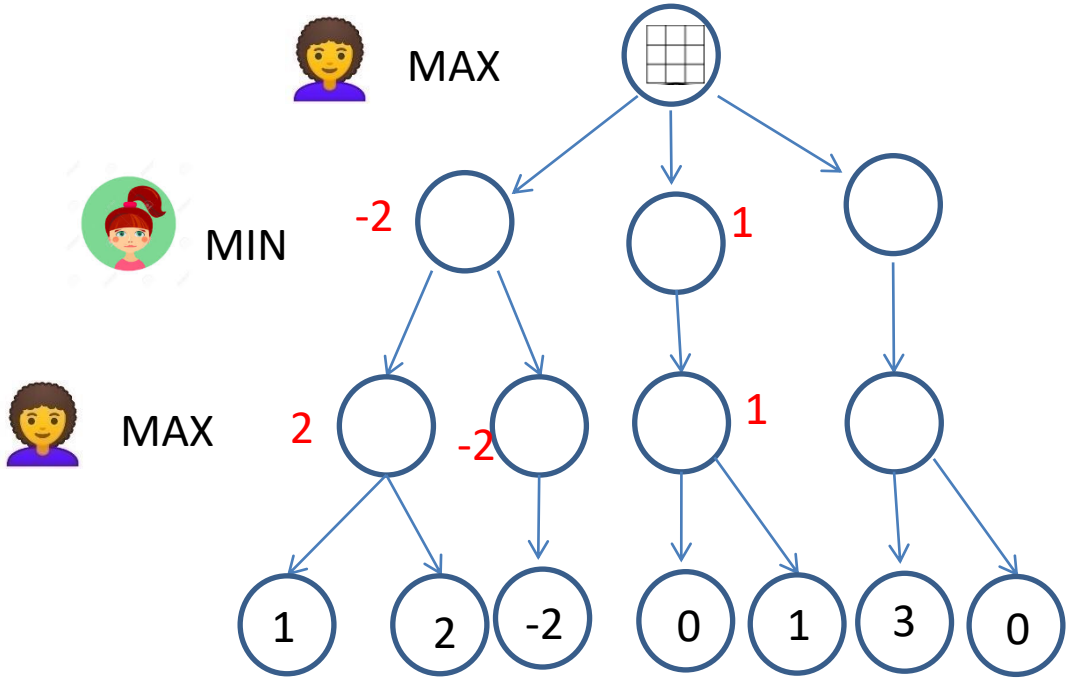
2. Algoritmo Minimax



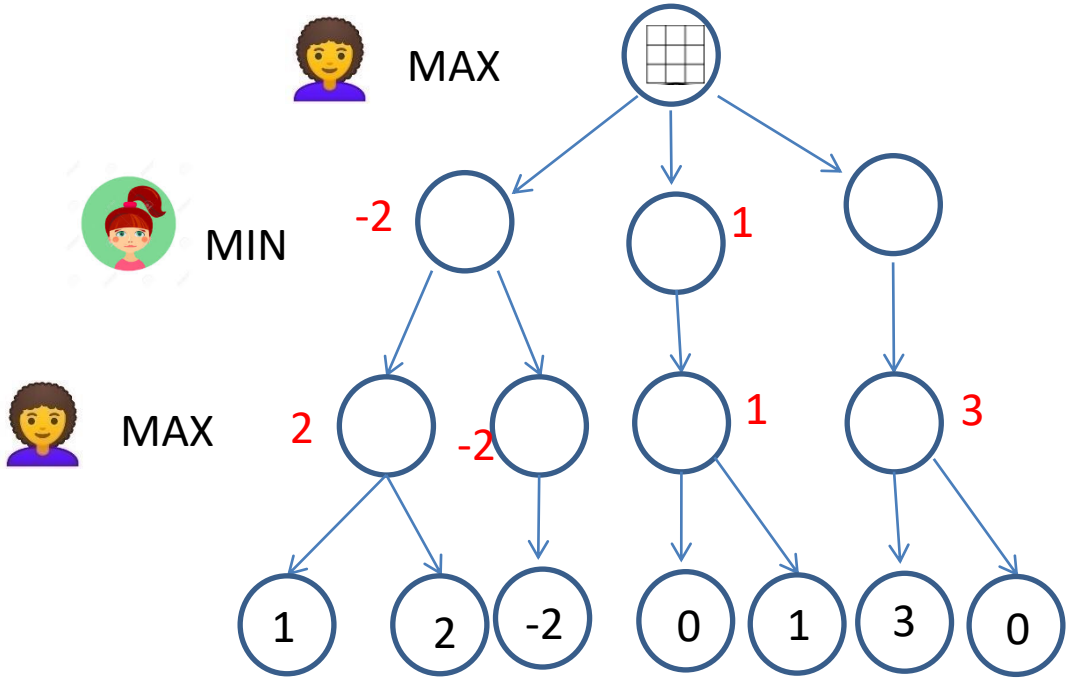
2. Algoritmo Minimax



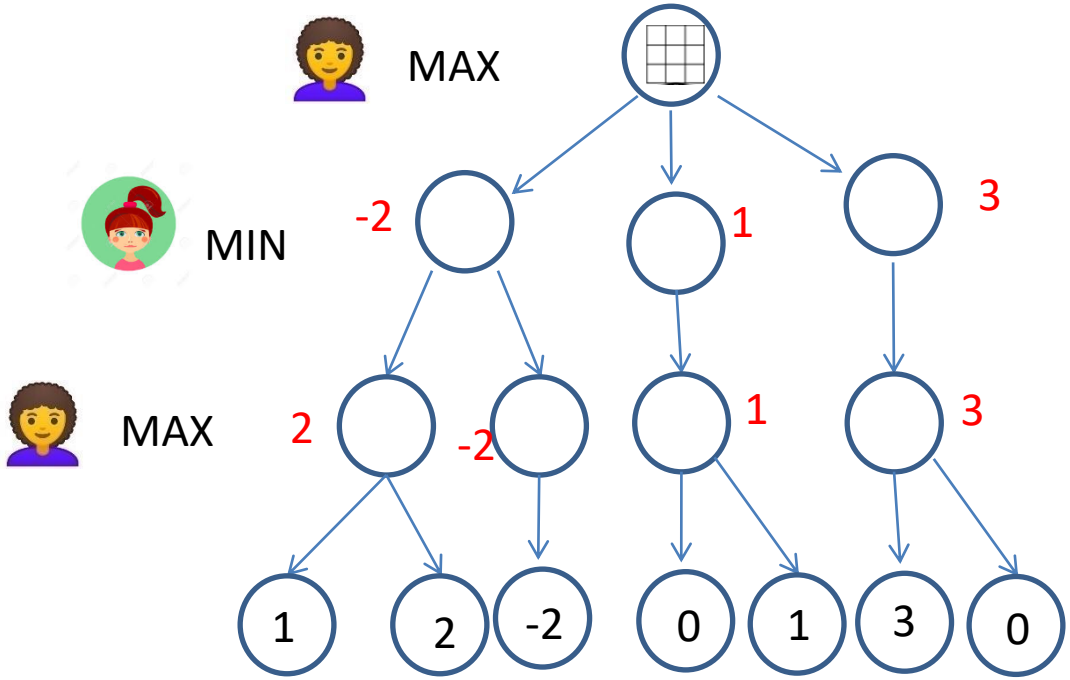
2. Algoritmo Minimax



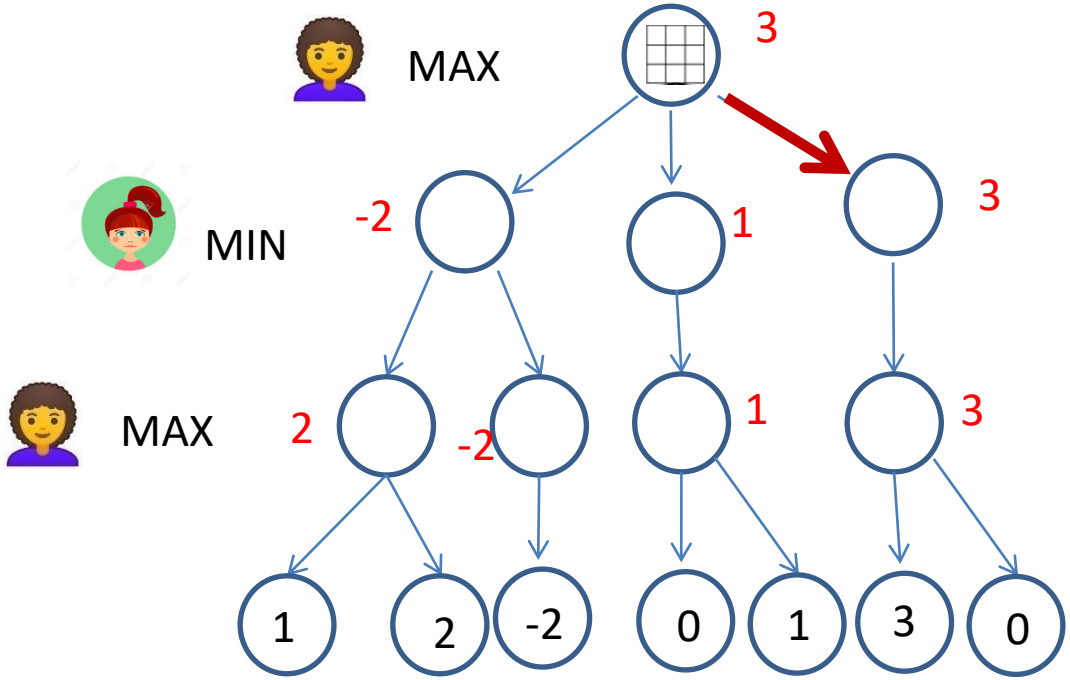
2. Algoritmo Minimax



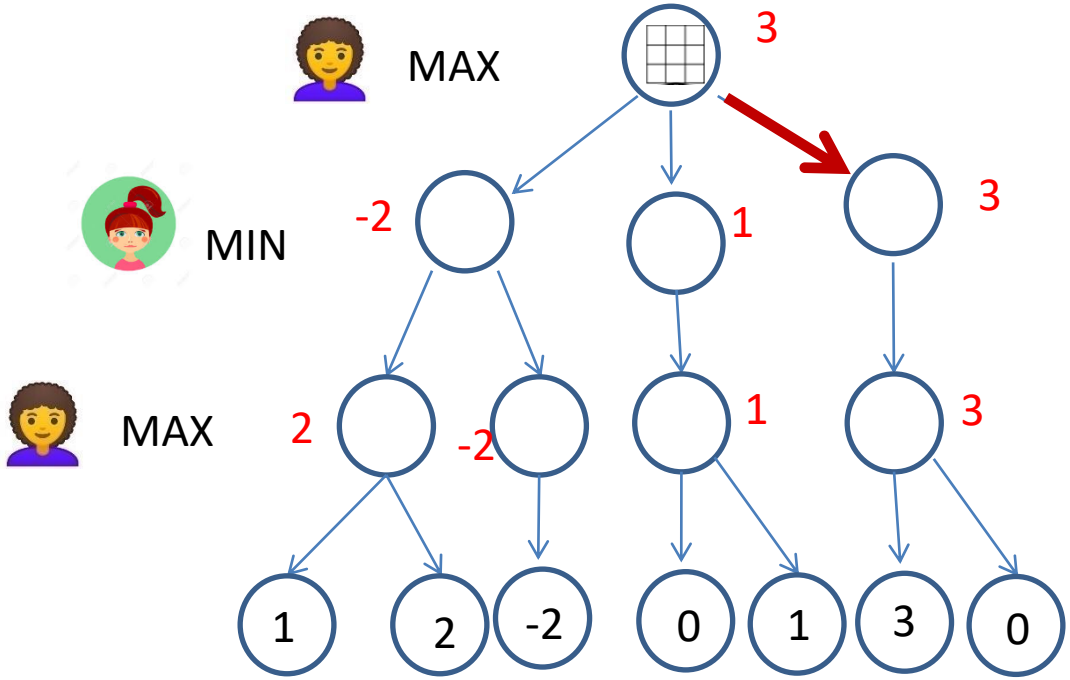
2. Algoritmo Minimax



2. Algoritmo Minimax



2. Algoritmo Minimax



En el caso de producirse empates entre los máximos valores de utilidad de algunos hijos del nodo raíz da igual cuál de ellos elijamos como mejor jugada. Cuando se produce uno de estos empates decimos que se produce una ***meseta***.

2. Algoritmo Minimax: tres en raya

Árbol de búsqueda:

- Profundidad: 2 niveles
- Las posiciones simétricas no se representan

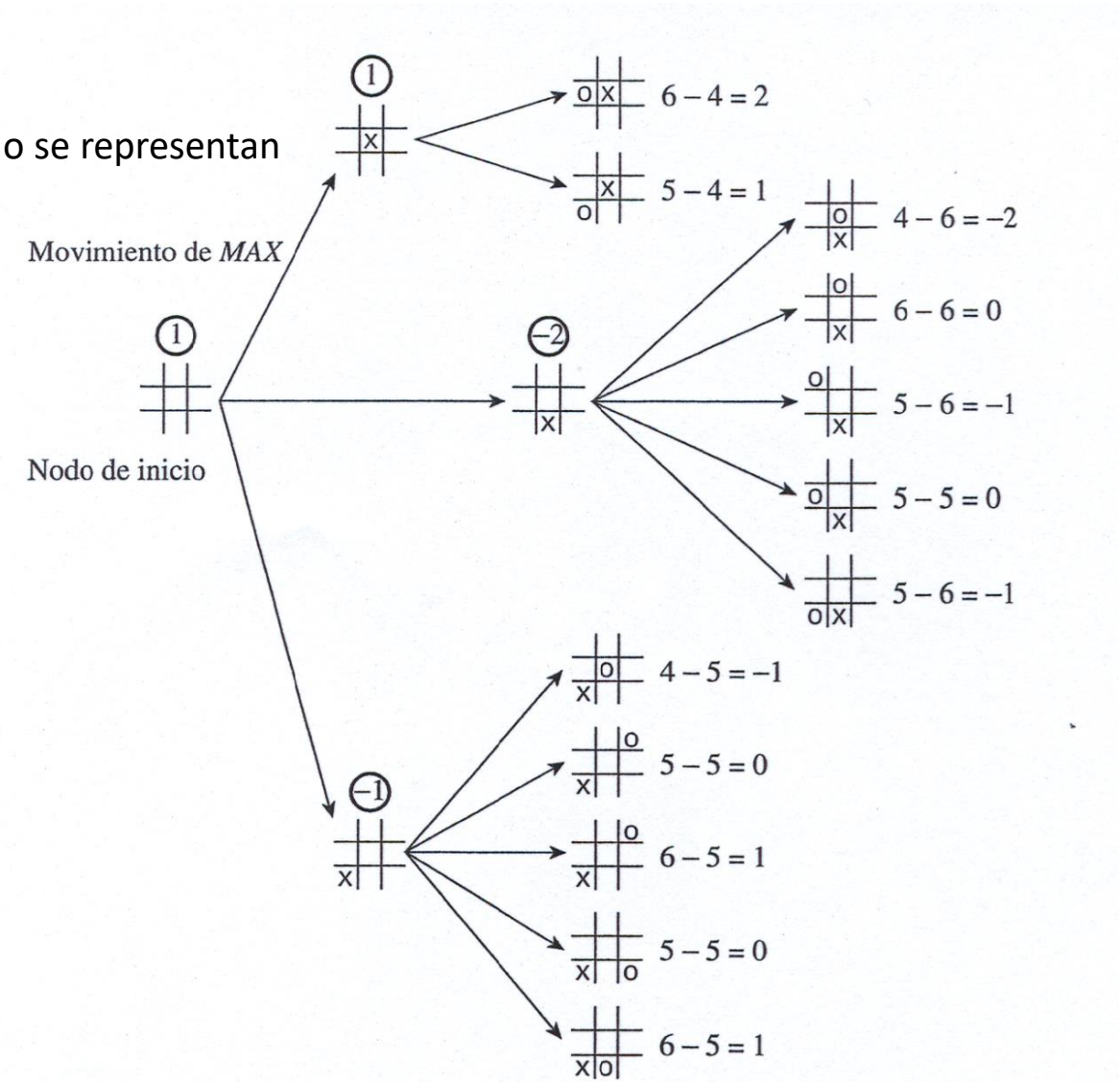


Figura 12.3.

2. Algoritmo Minimax: tres en raya

Árbol de búsqueda:

- Profundidad: 2 niveles
- Las posiciones simétricas no se representan

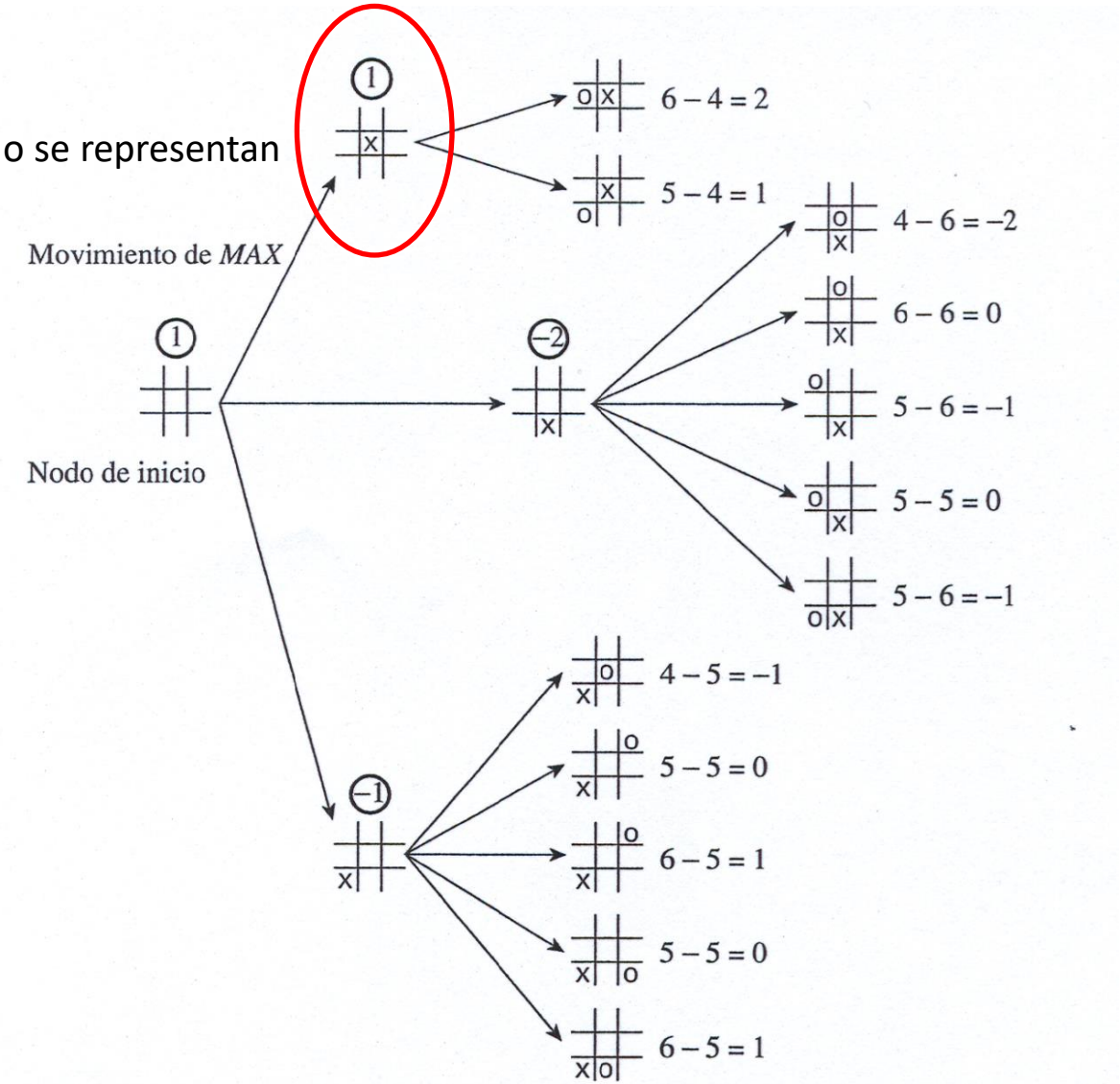
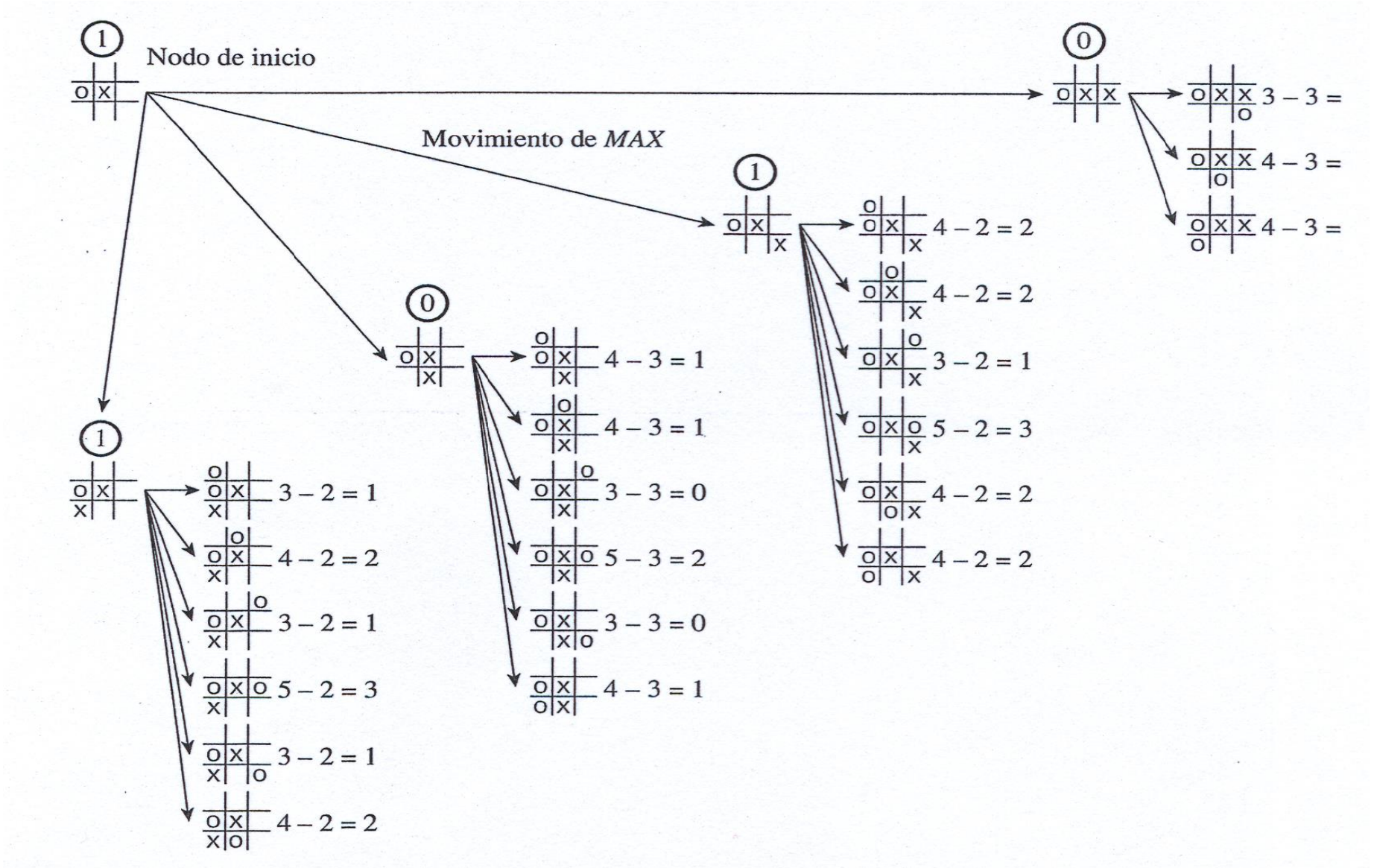
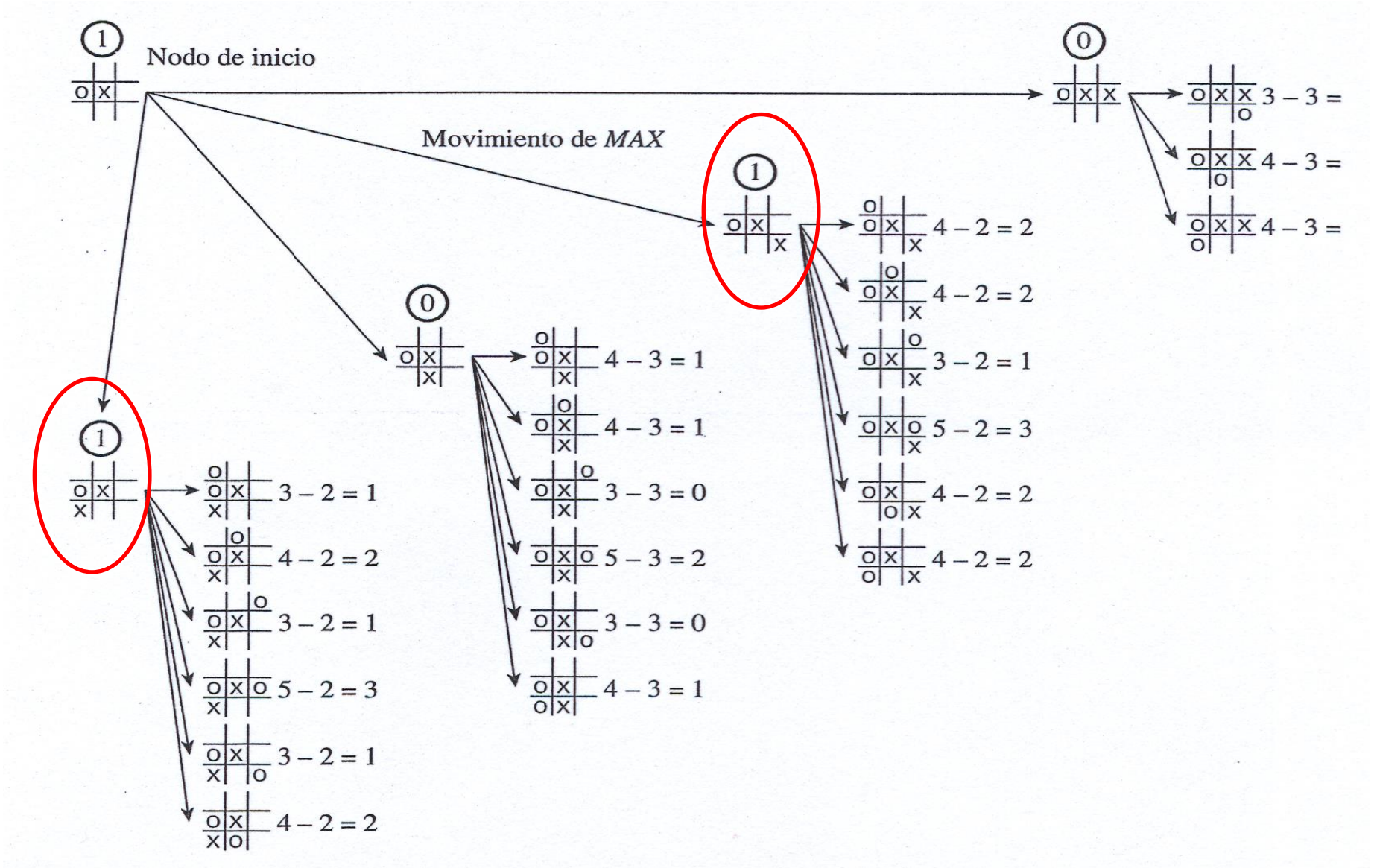


Figura 12.3.

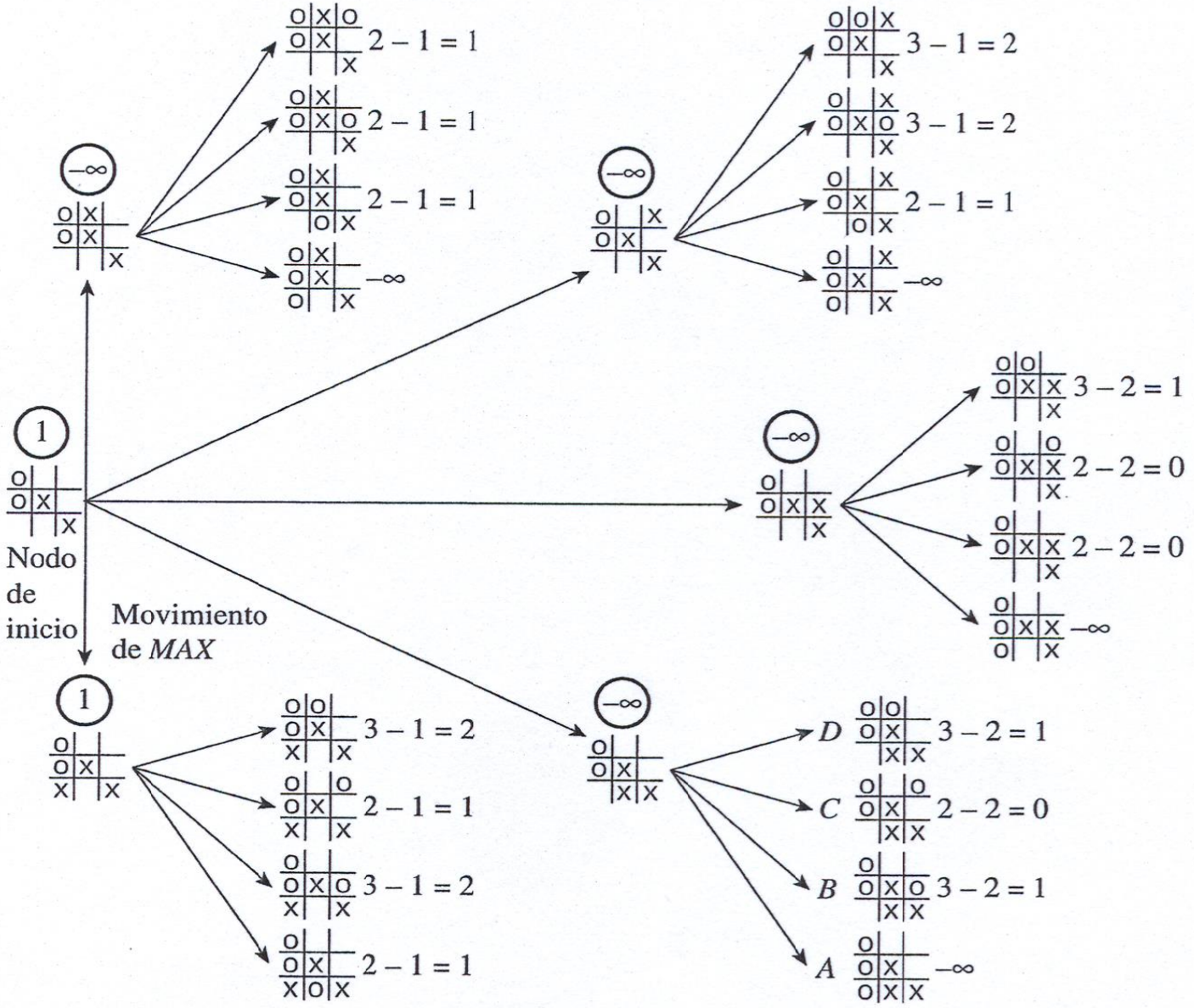
2. Algoritmo Minimax: tres en raya



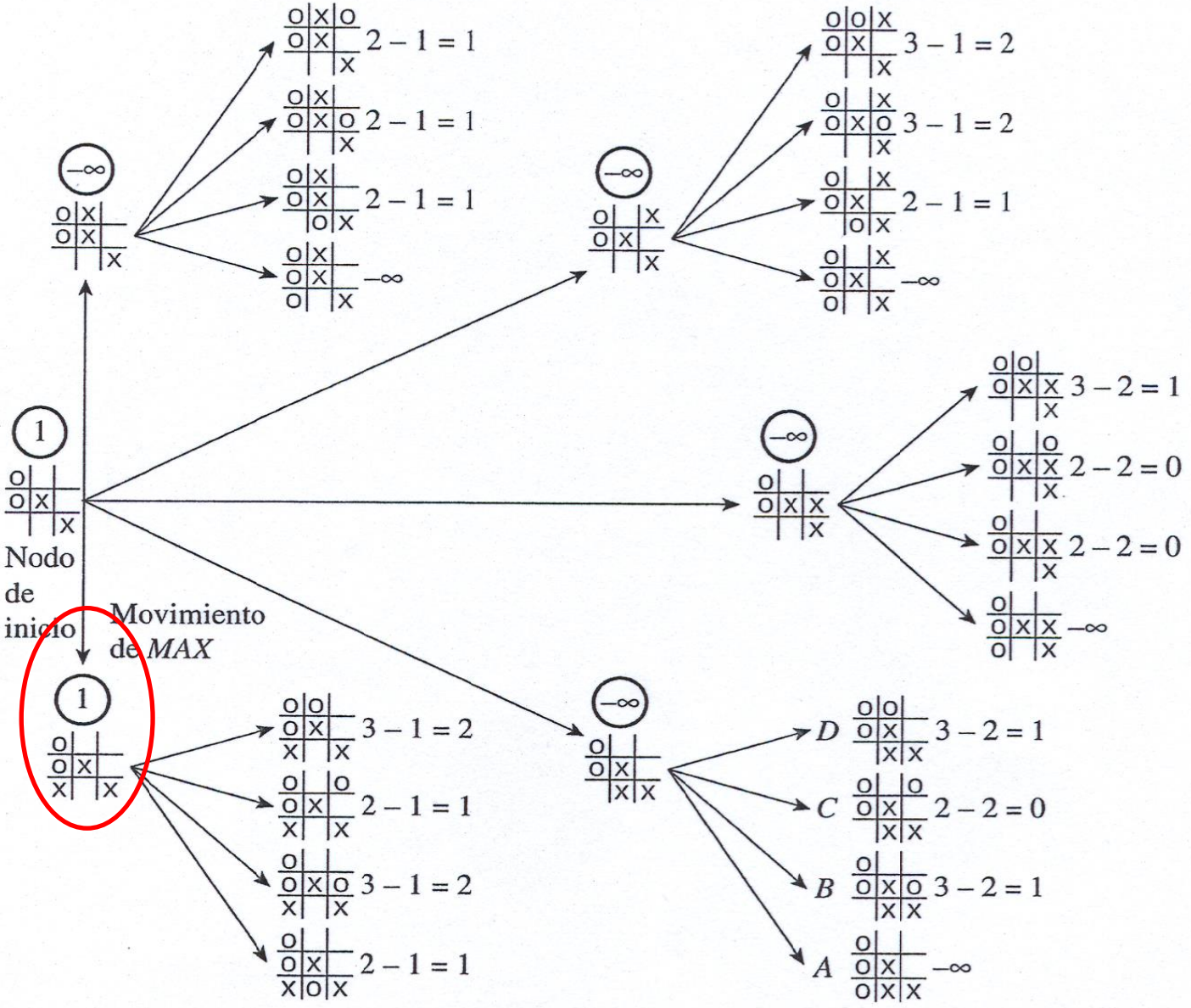
2. Algoritmo Minimax: tres en raya



2. Algoritmo Minimax: tres en raya



2. Algoritmo Minimax: tres en raya

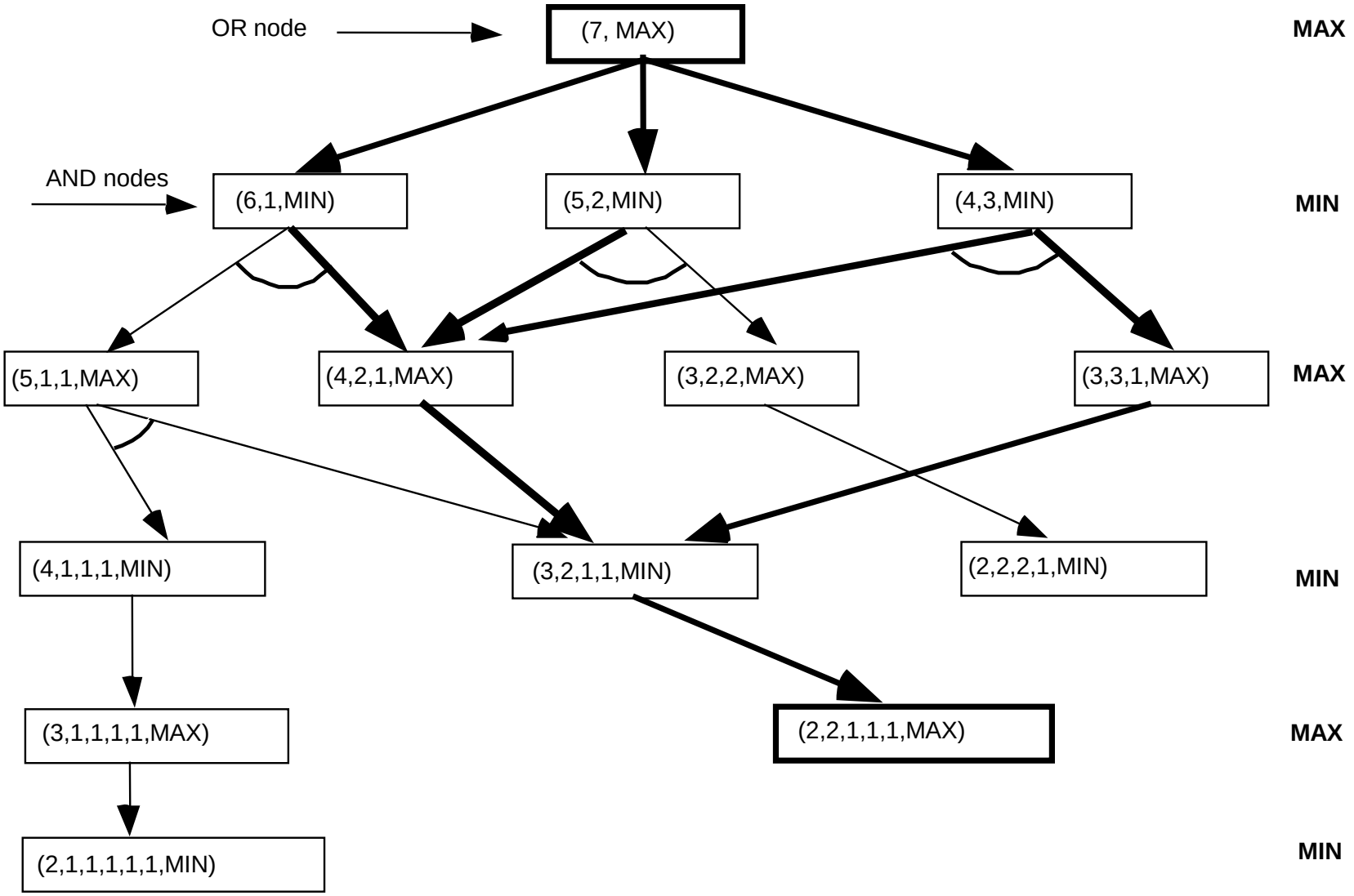


2. El juego del Grundy

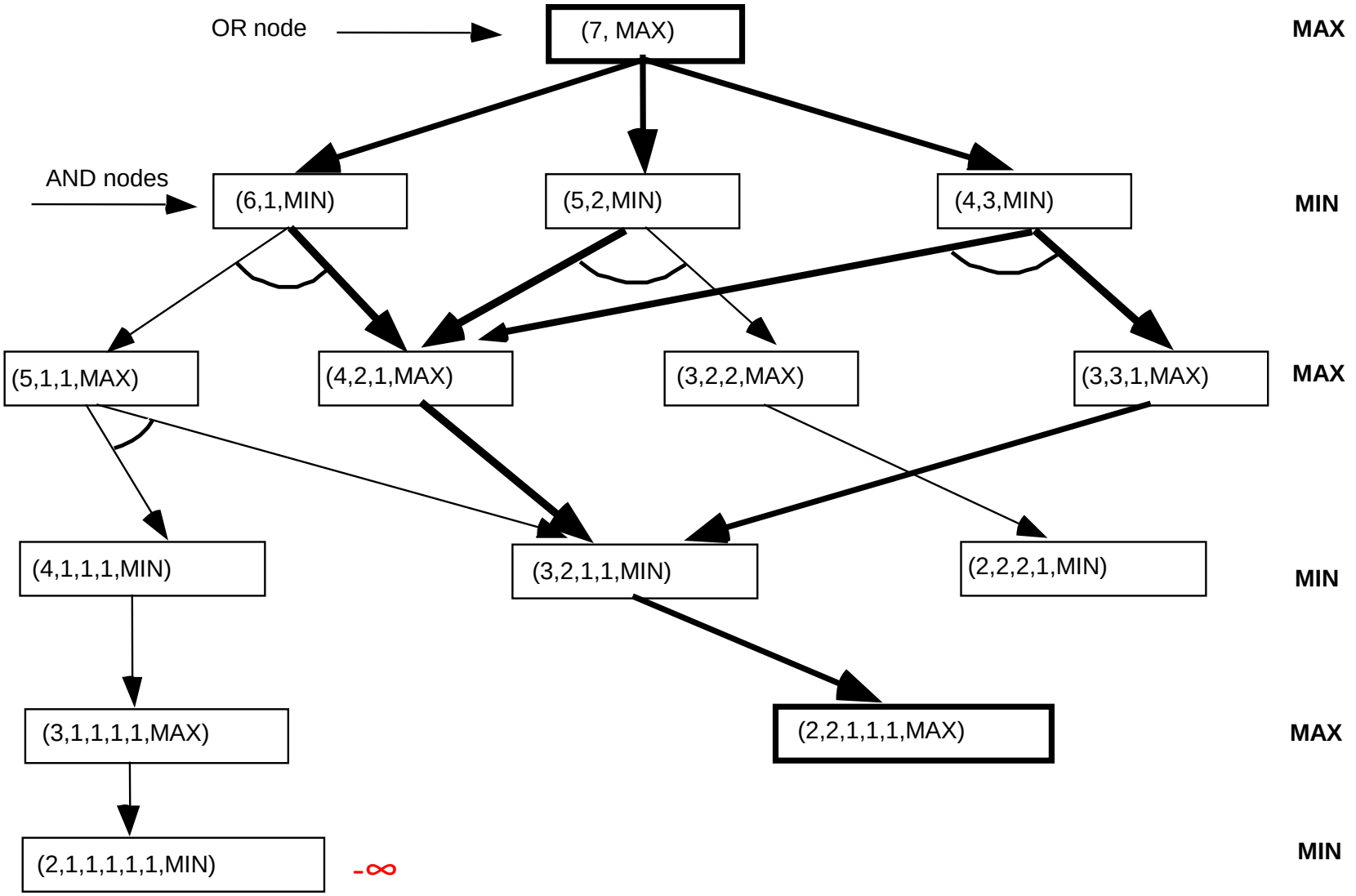
El juego de Grundy es un juego matemático de estrategia para dos jugadores.

La configuración inicial parte de una pila de objetos, y los dos jugadores se turnan para dividir una única pila en dos pilas de diferentes tamaños. El juego termina cuando solo quedan montones de tamaño dos o más pequeños, ninguno de los cuales se puede dividir de manera desigual. El juego generalmente se juega como un juego normal, lo que significa que la última persona que puede hacer un movimiento permitido gana.

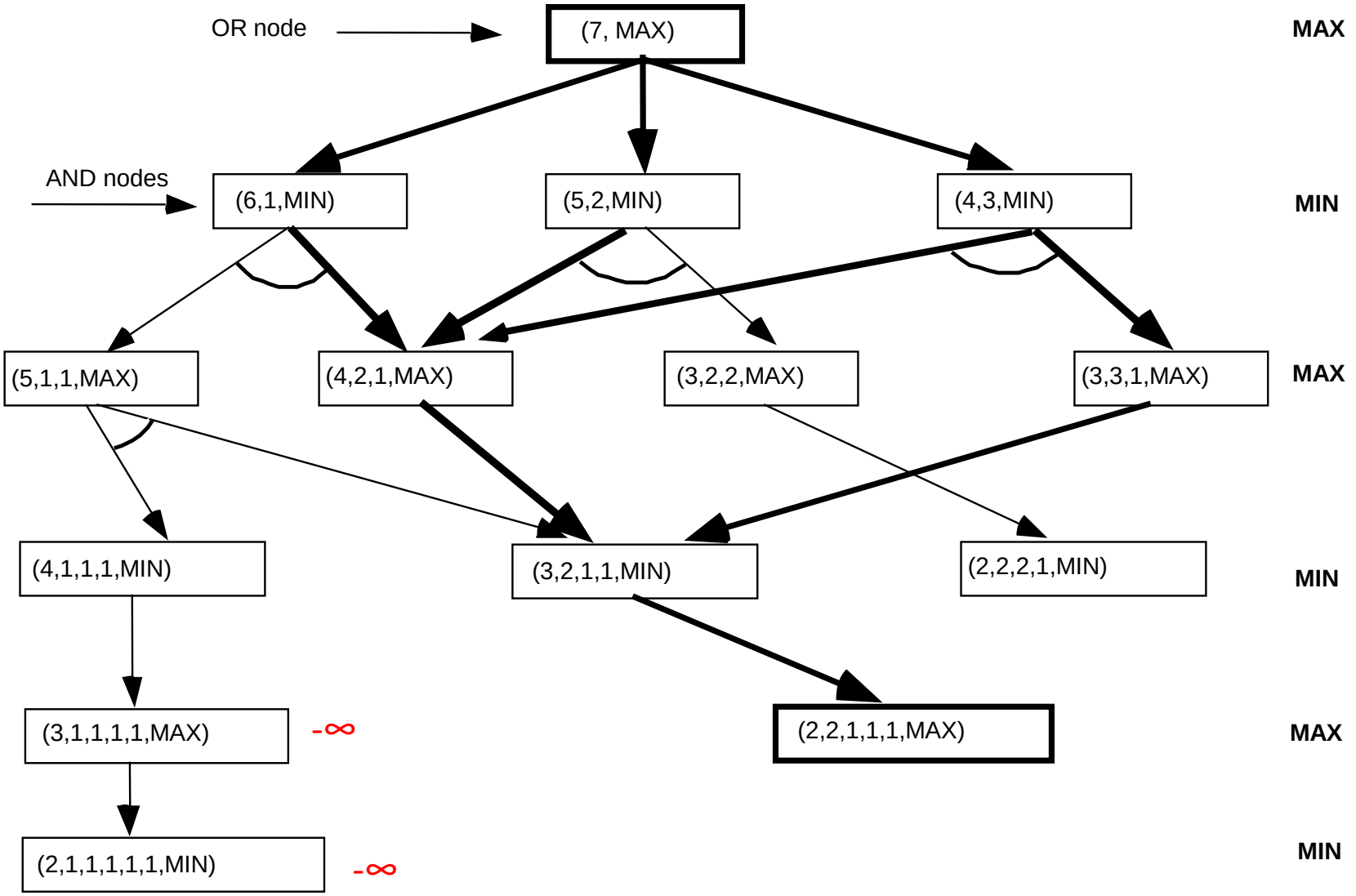
2. Algoritmo Minimax: juego del Grundy



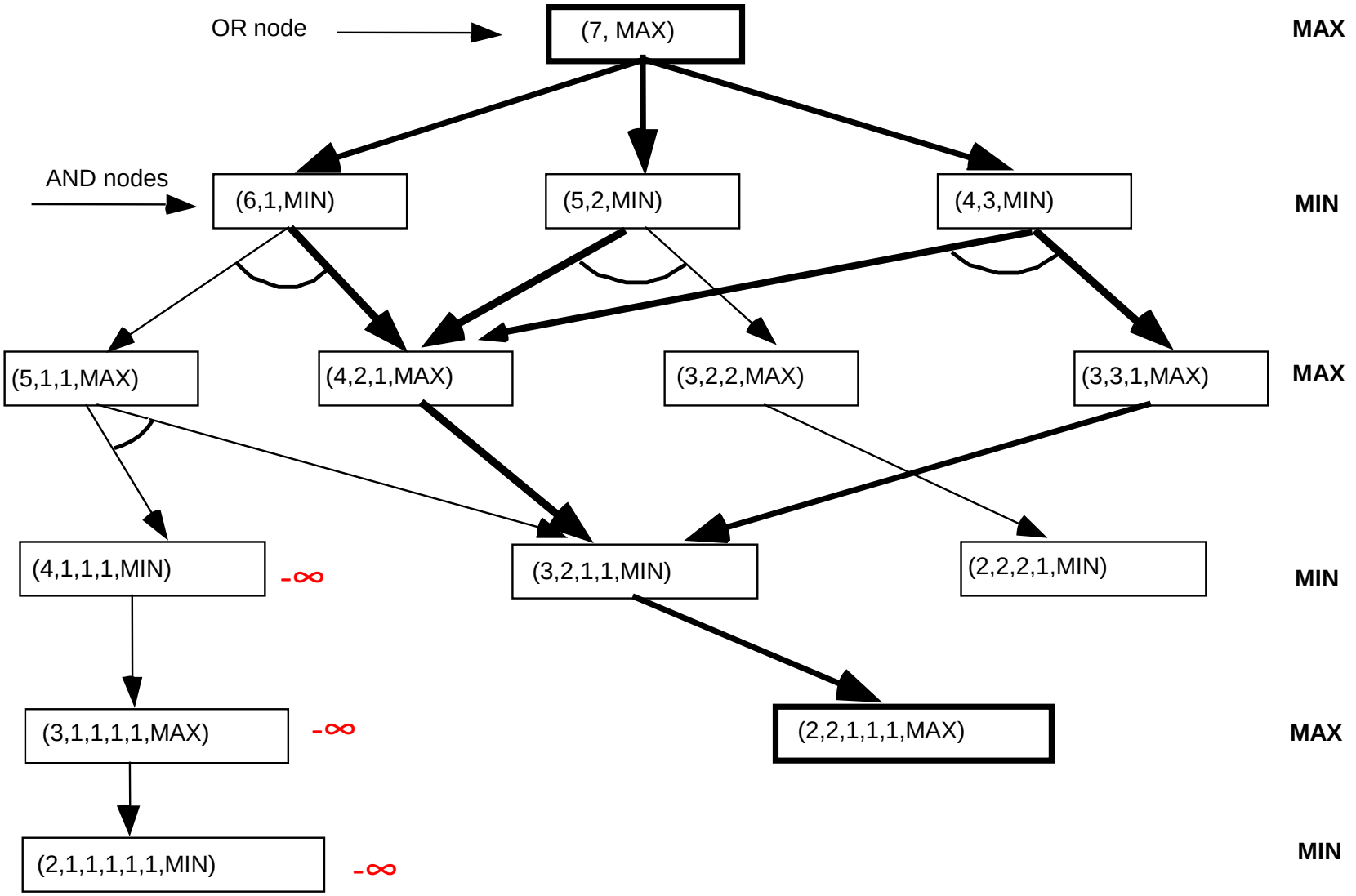
2. Algoritmo Minimax: juego del Grundy



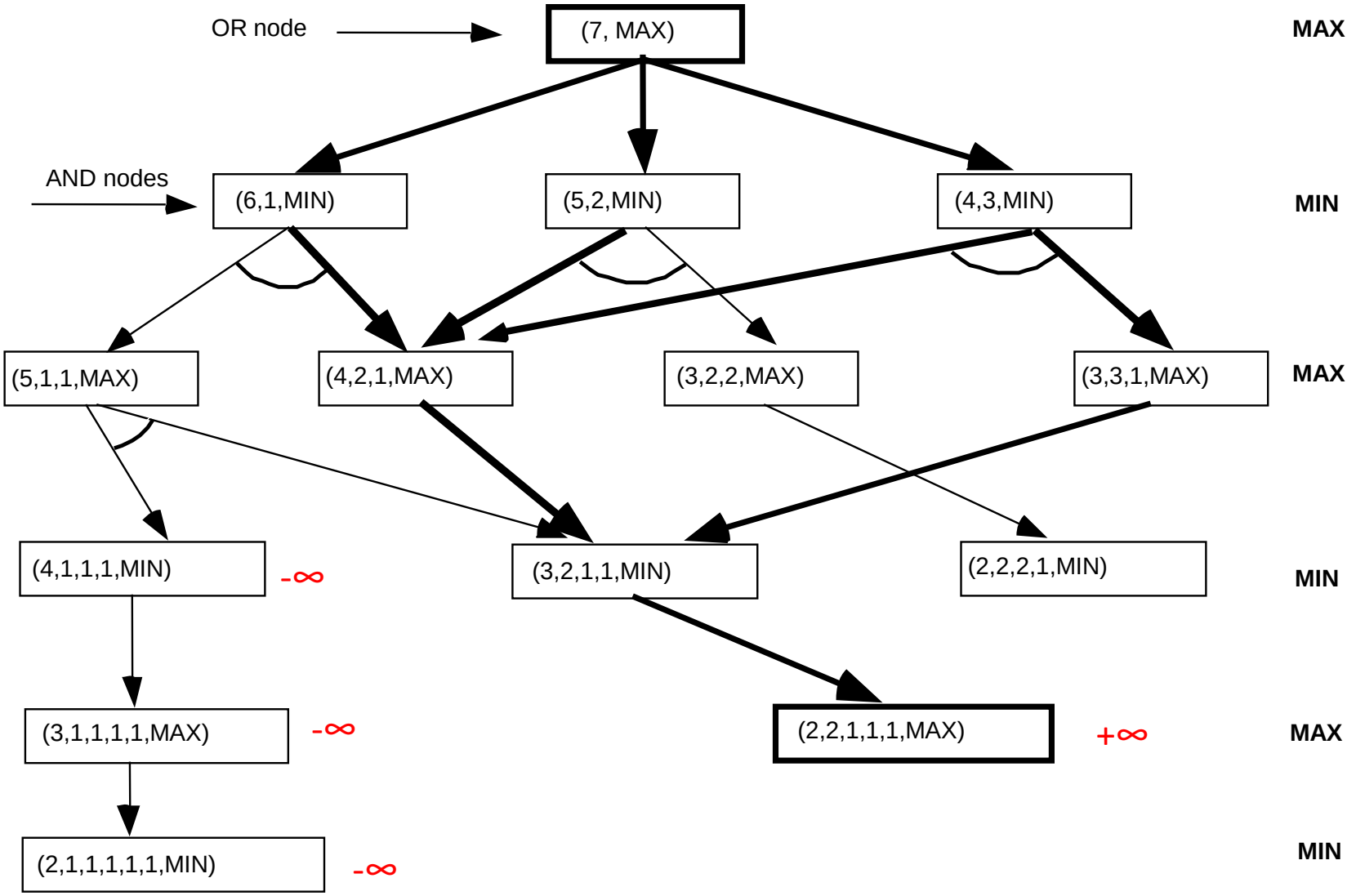
2. Algoritmo Minimax: juego del Grundy



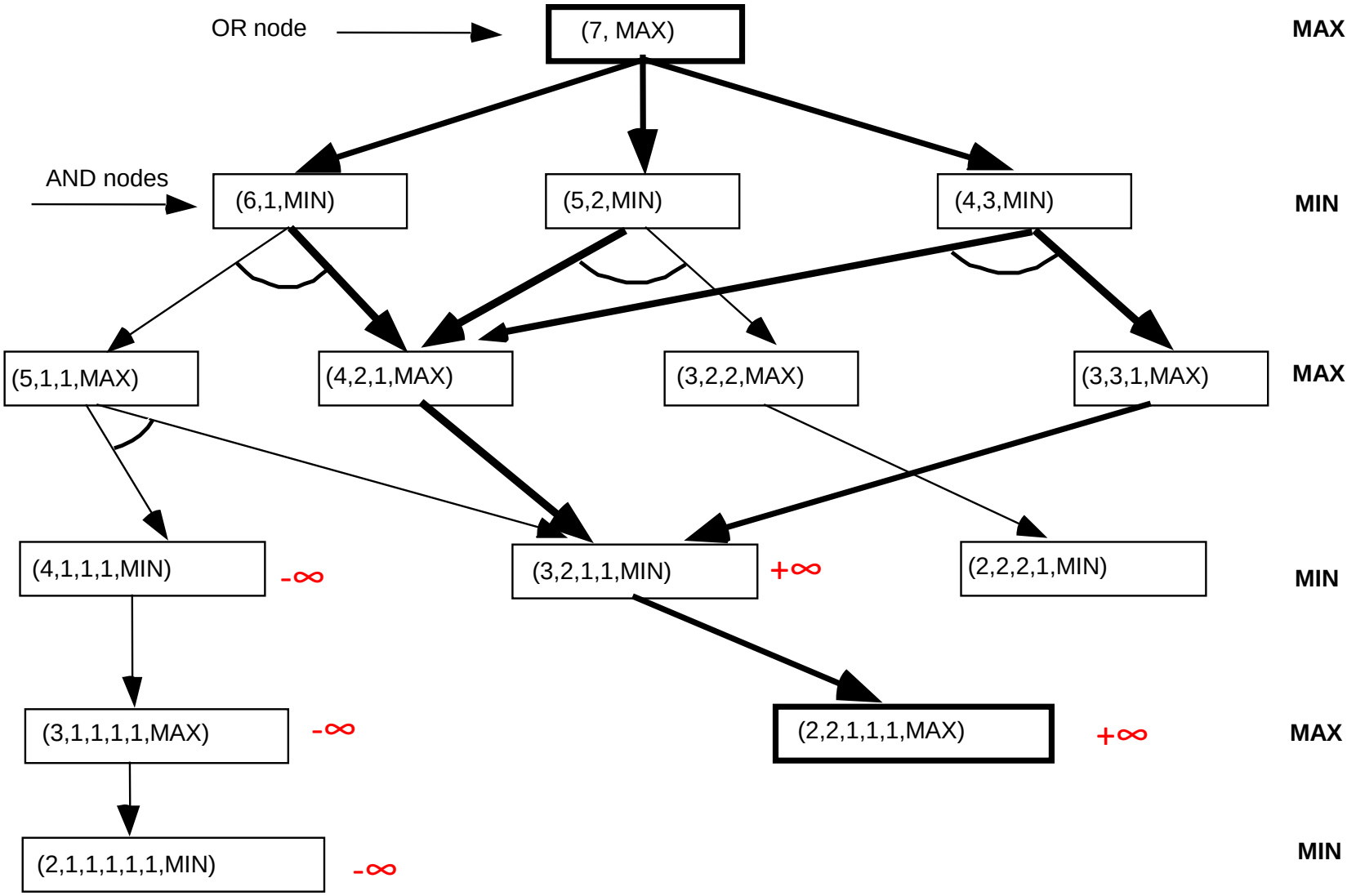
2. Algoritmo Minimax: juego del Grundy



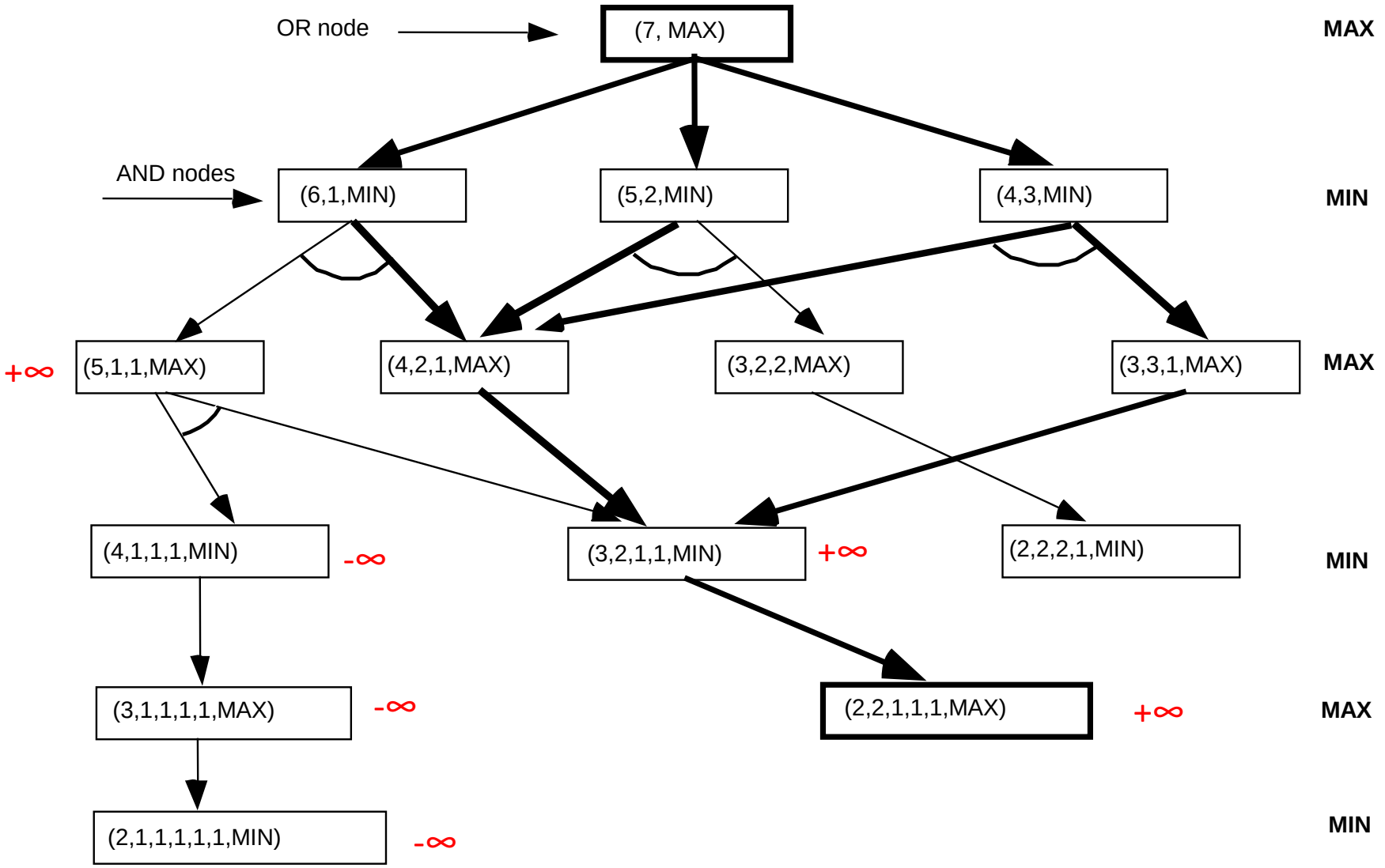
2. Algoritmo Minimax: juego del Grundy



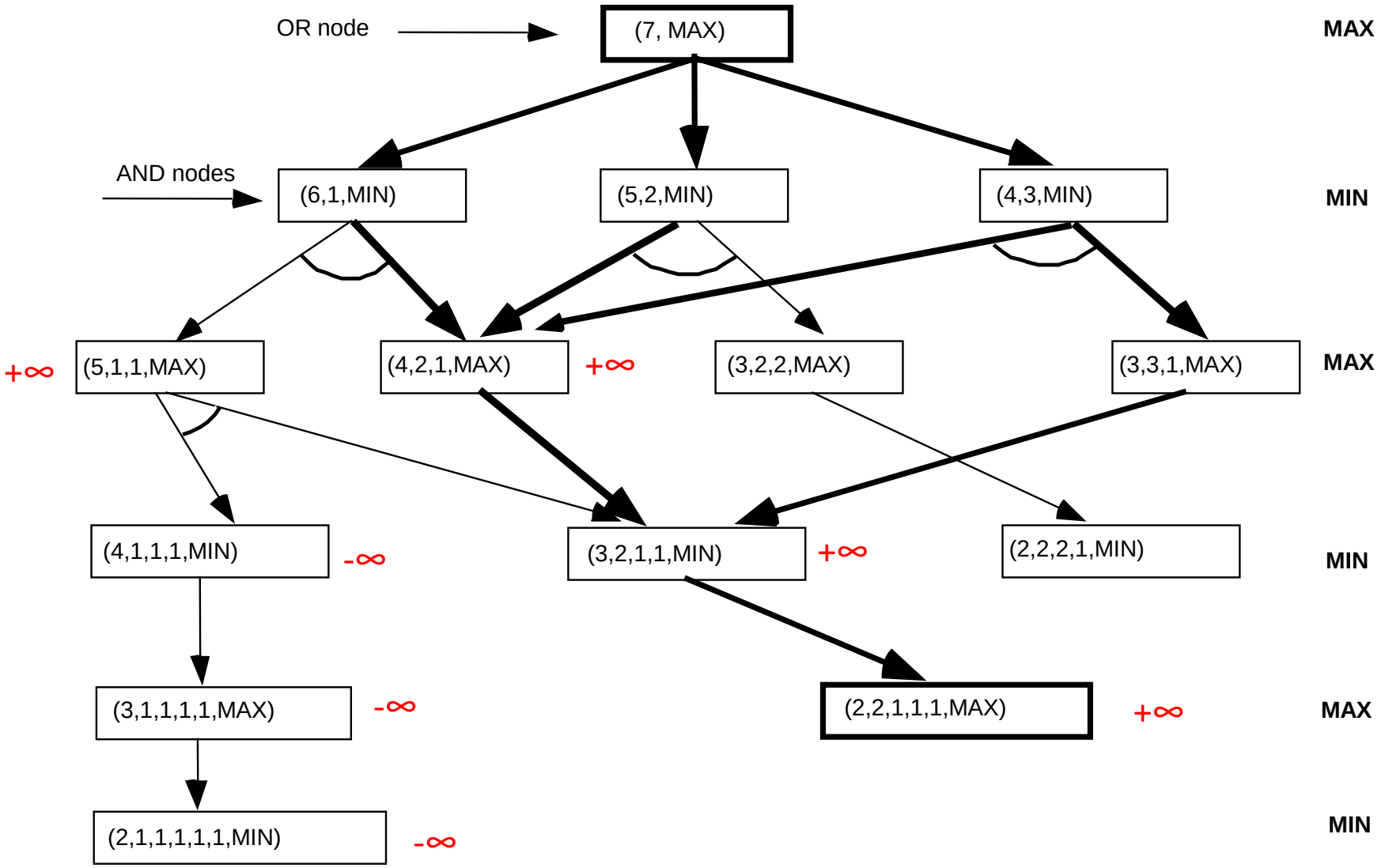
2. Algoritmo Minimax: juego del Grundy



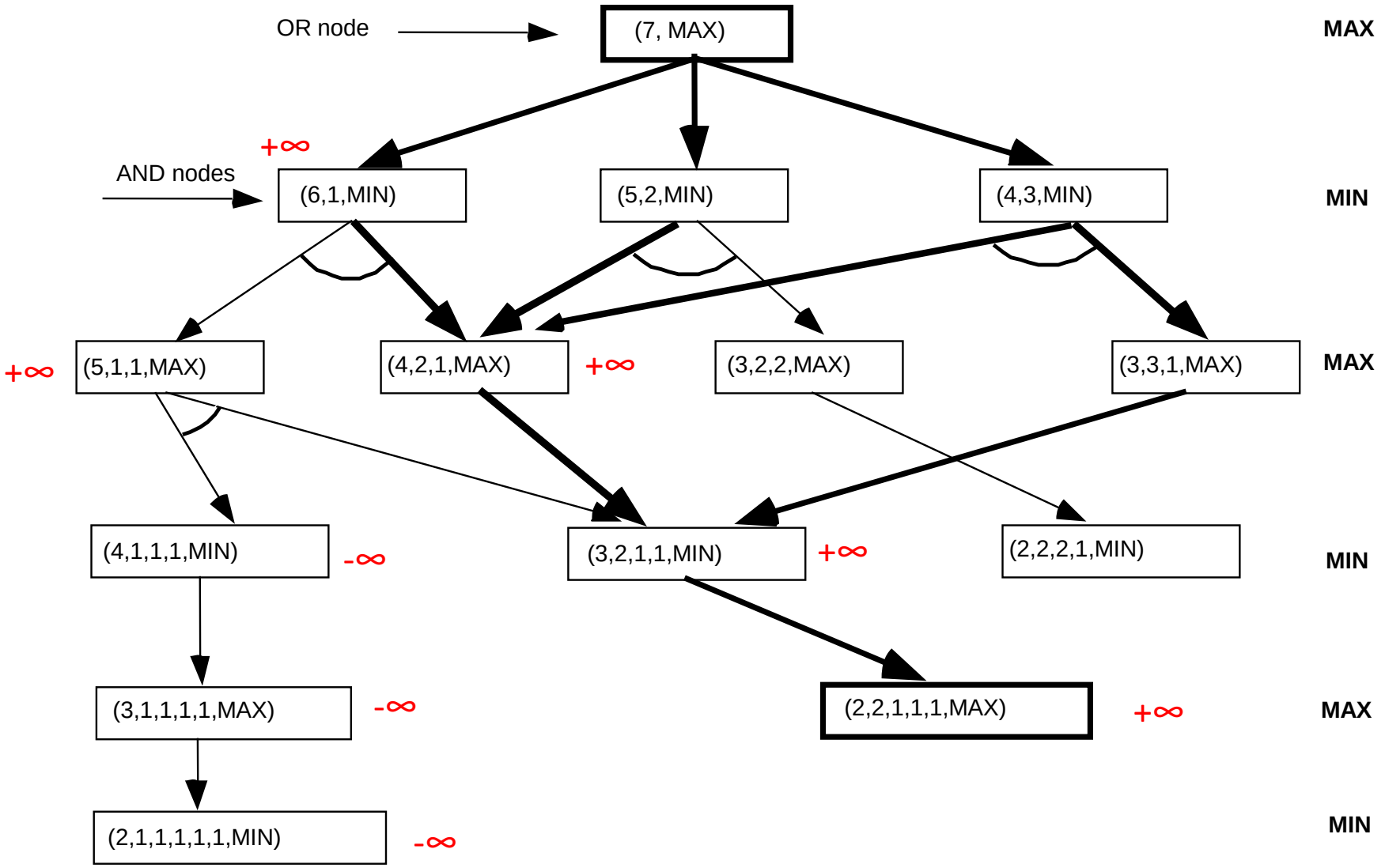
2. Algoritmo Minimax: juego del Grundy



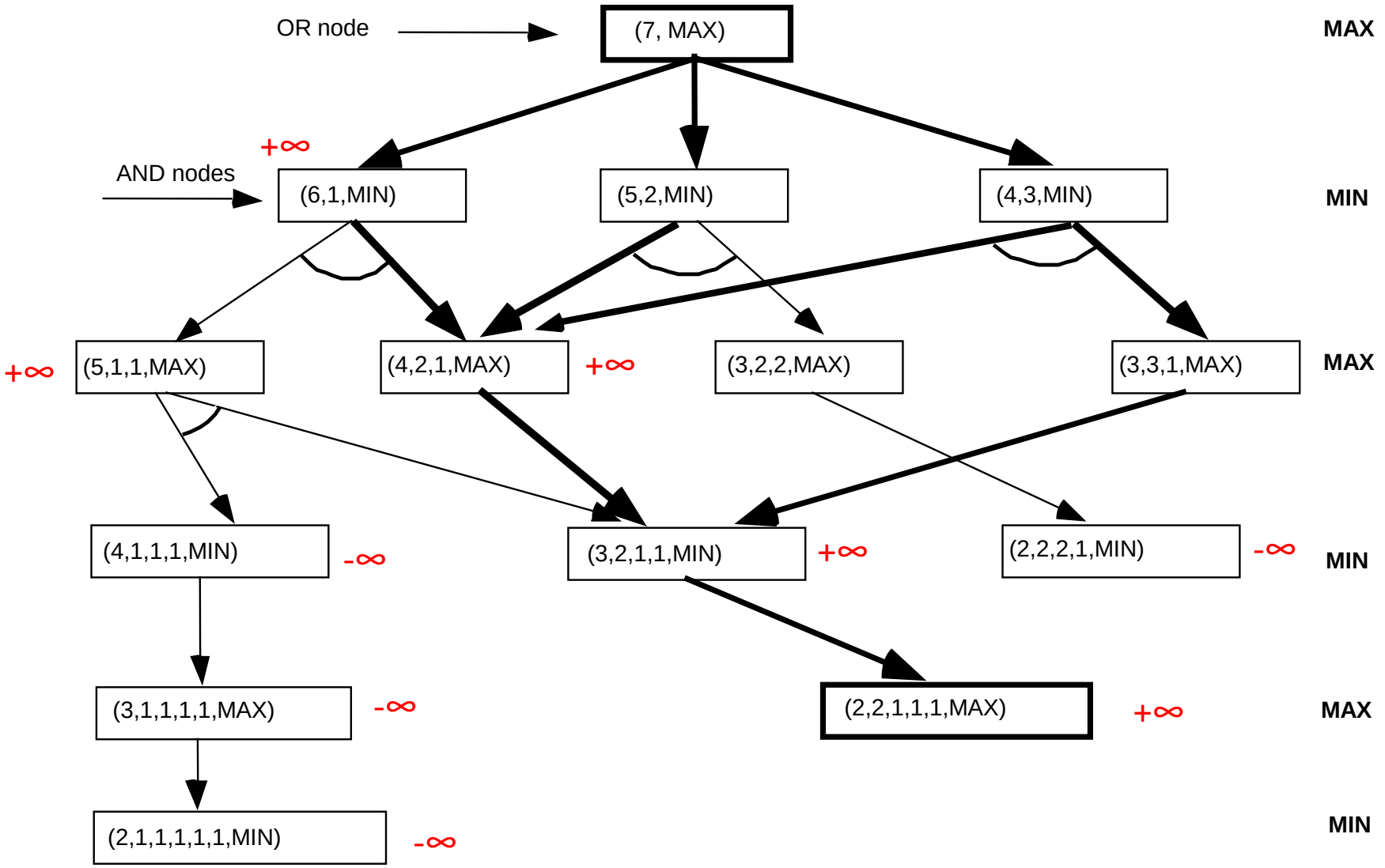
2. Algoritmo Minimax: juego del Grundy



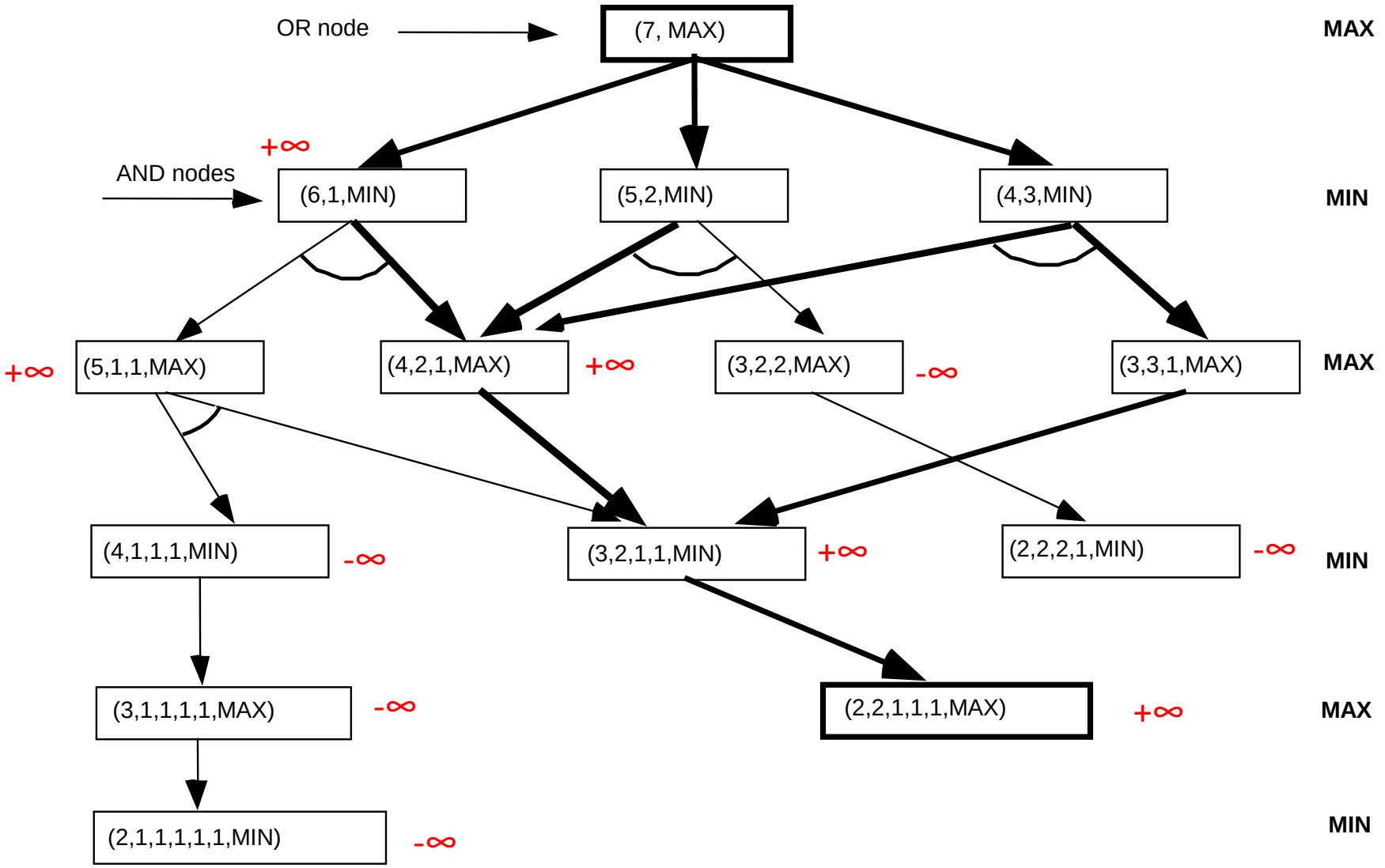
2. Algoritmo Minimax: juego del Grundy



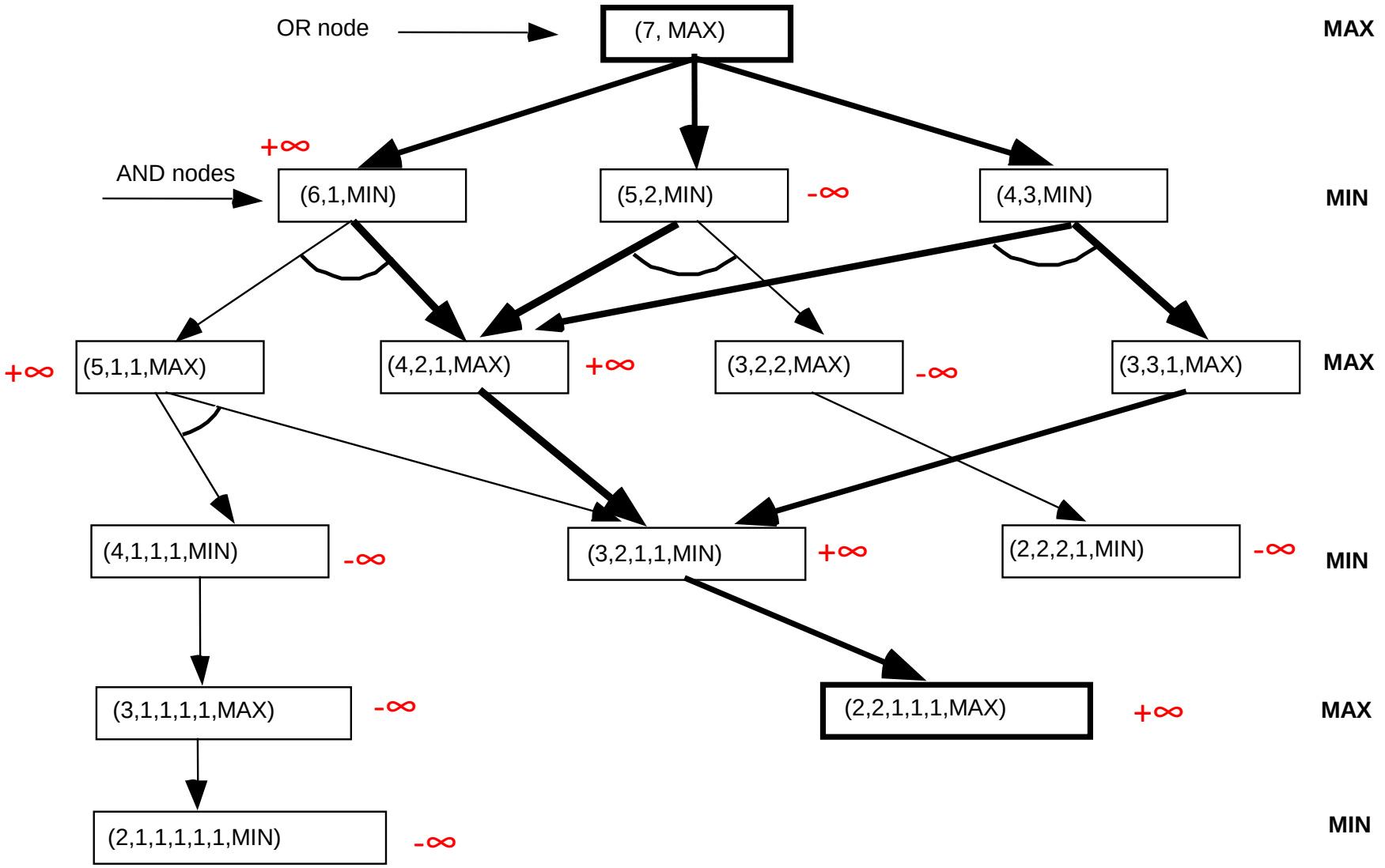
2. Algoritmo Minimax: juego del Grundy



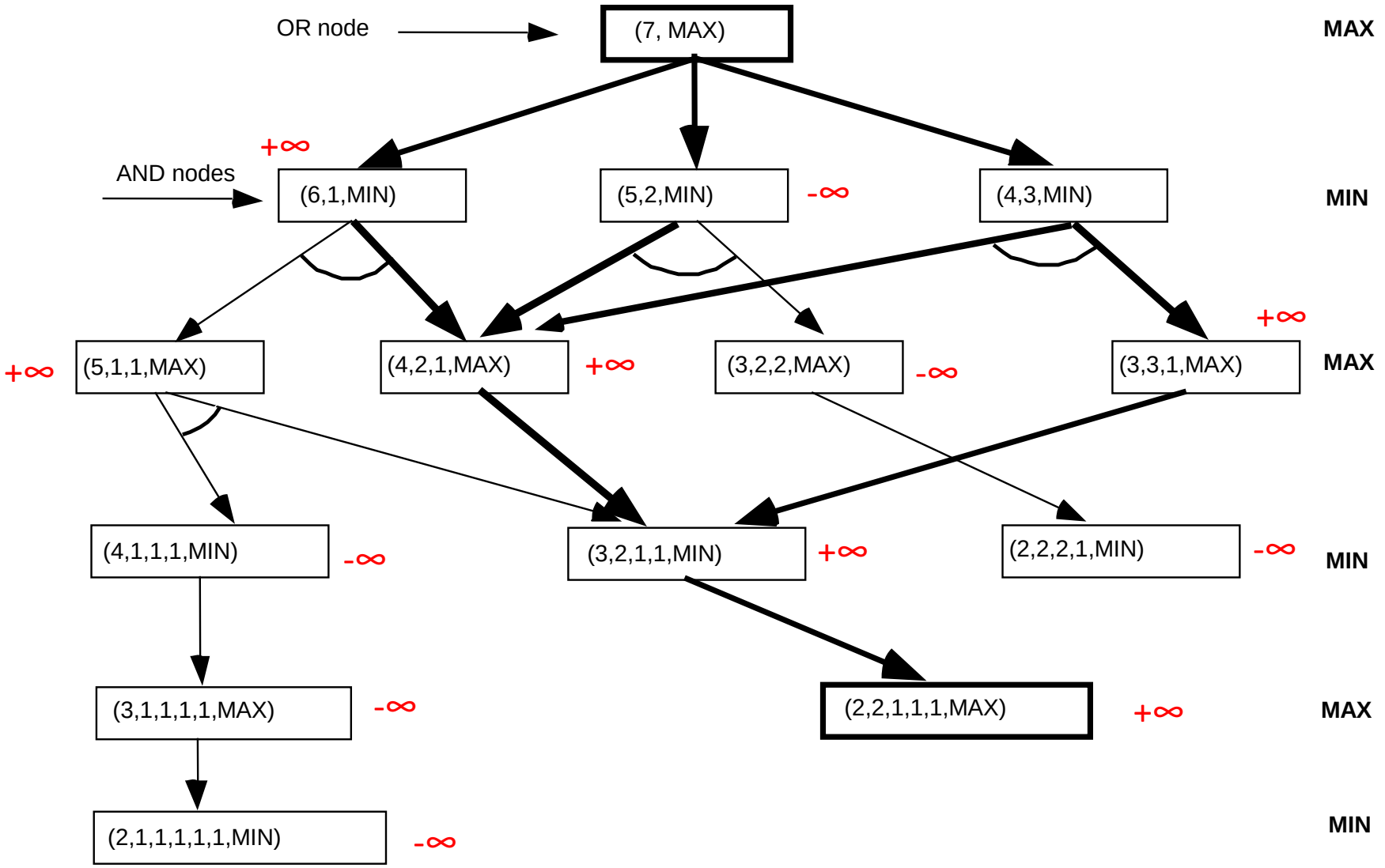
2. Algoritmo Minimax: juego del Grundy



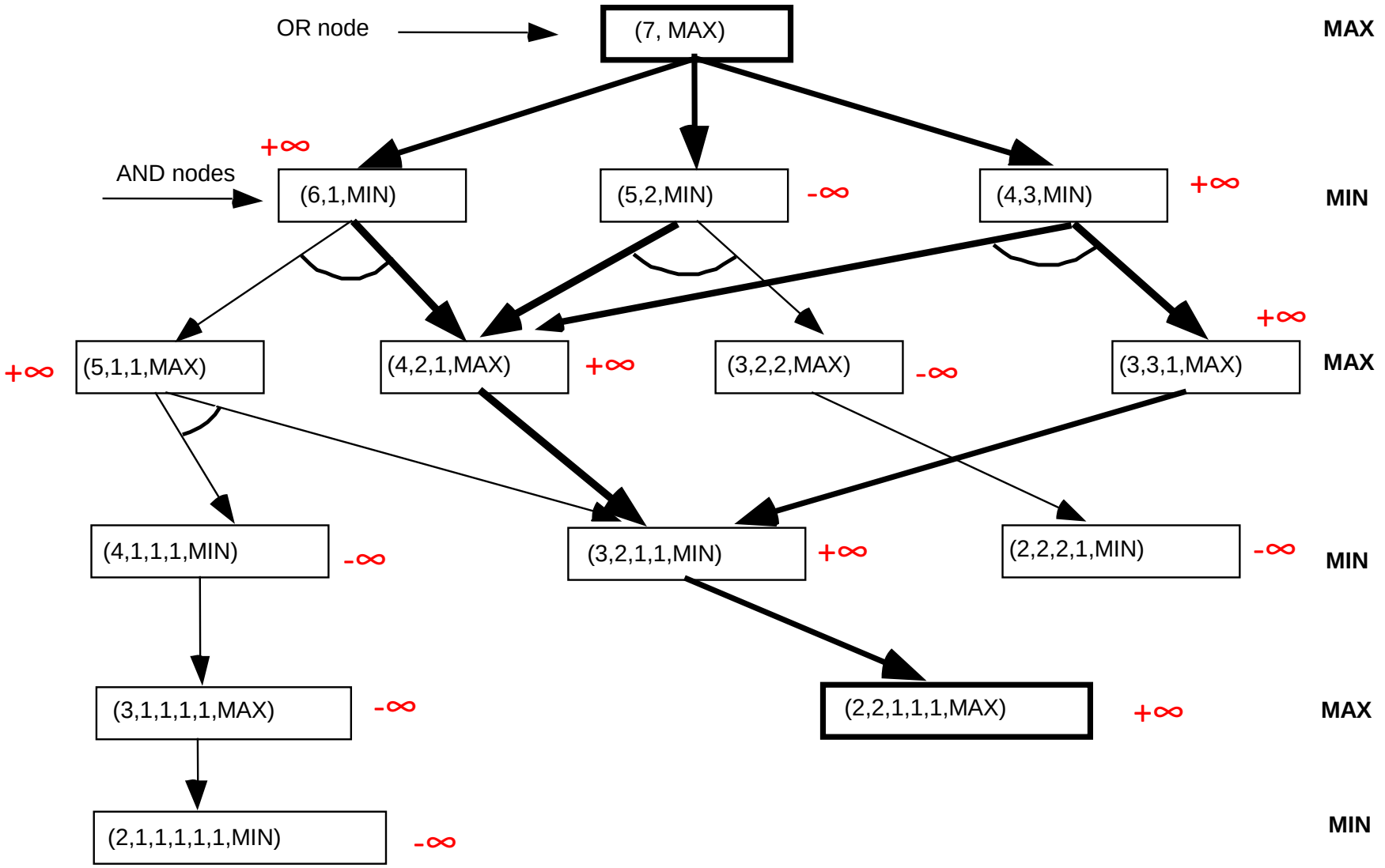
2. Algoritmo Minimax: juego del Grundy



2. Algoritmo Minimax: juego del Grundy



2. Algoritmo Minimax: juego del Grundy



2. Algoritmo Minimax: propiedades

Utilidad del Minimax:

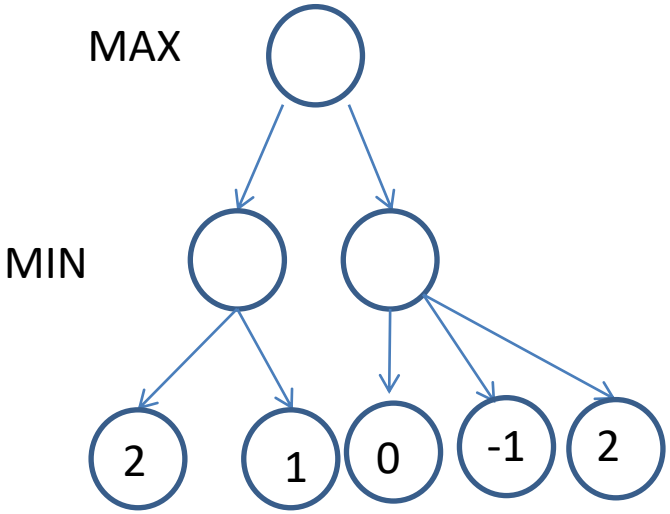
La selección del mejor movimiento se realiza teniendo en cuenta la evolución del juego en niveles más profundos: el valor volcado de un nodo es un valor mucho más informado que el resultado de aplicar directamente la función de utilidad sobre dicho nodo.

Eficiencia del Minimax:

- Nivel de profundidad en el árbol de juego
- Función de utilidad

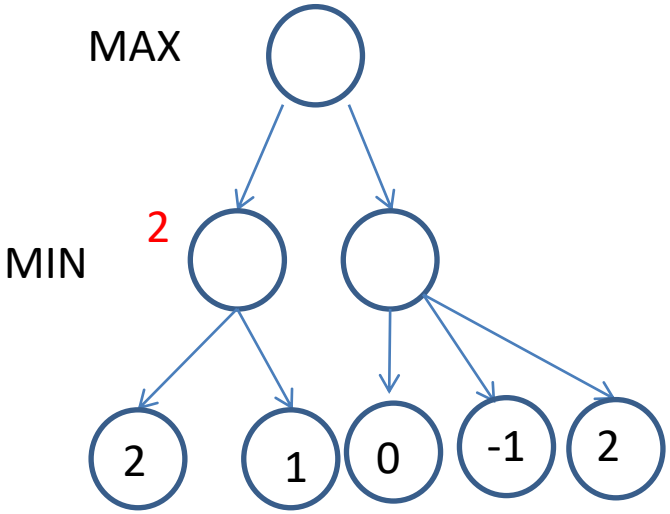
3. Algoritmo Minimax: poda alfa-beta (α-β)

Motivación:



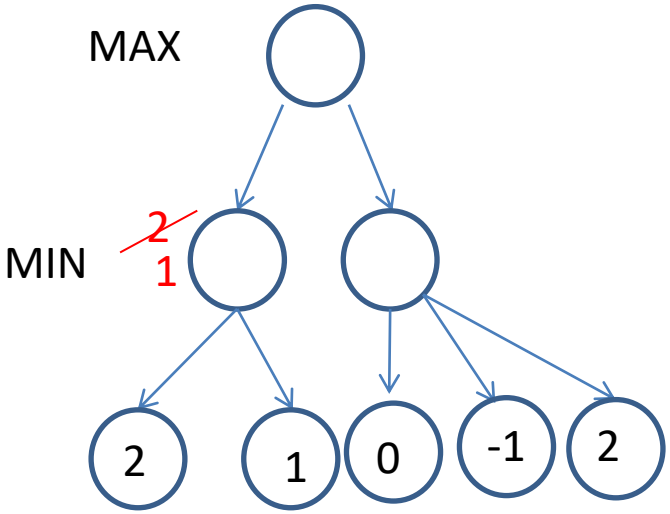
3. Algoritmo Minimax: poda alfa-beta (α-β)

Motivación:



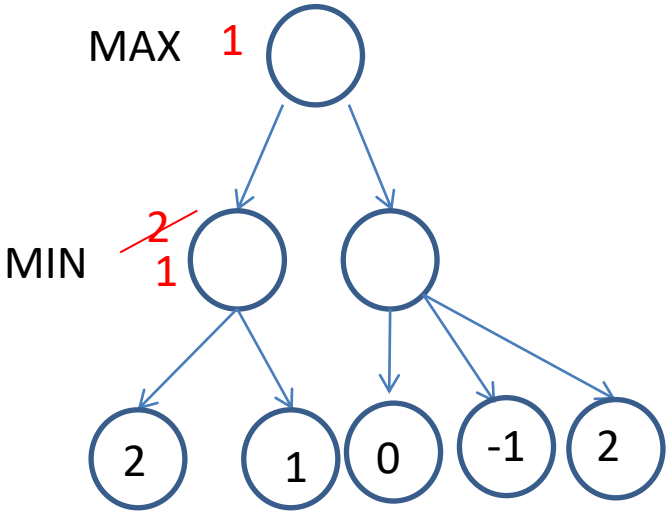
3. Algoritmo Minimax: poda alfa-beta (α-β)

Motivación:



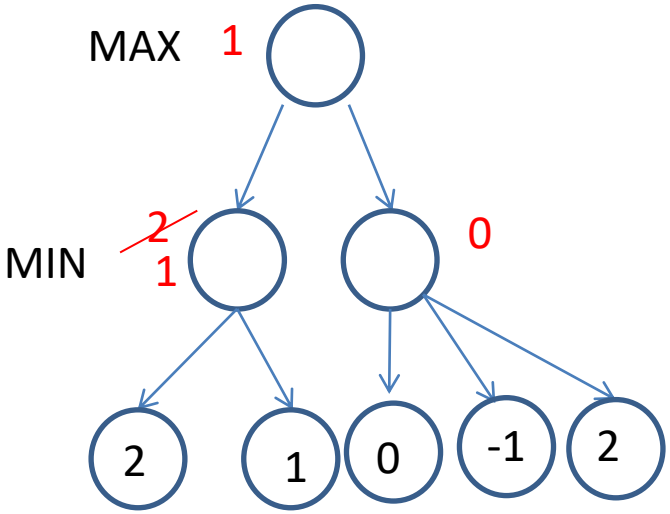
3. Algoritmo Minimax: poda alfa-beta (α-β)

Motivación:



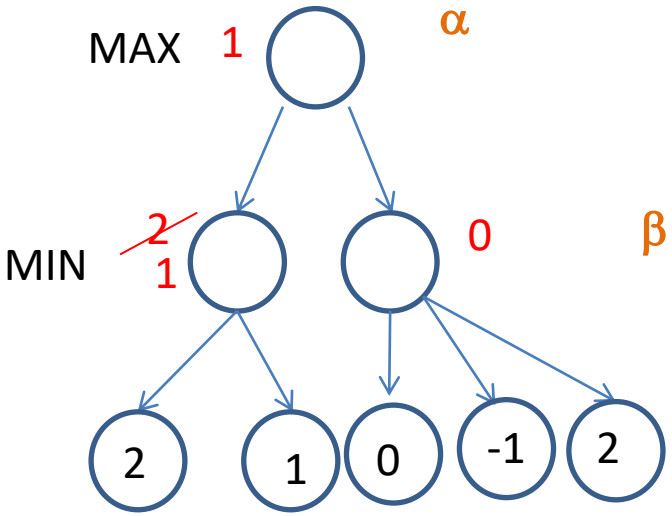
3. Algoritmo Minimax: poda alfa-beta (α-β)

Motivación:



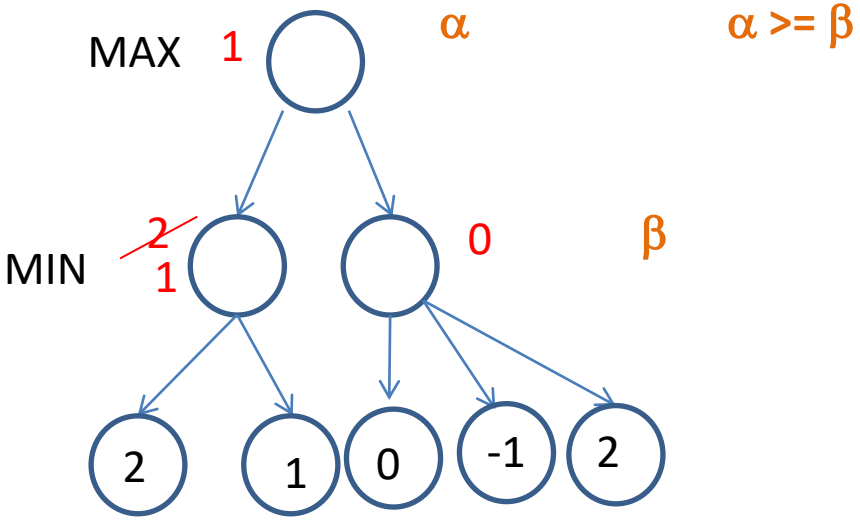
3. Algoritmo Minimax: poda alfa-beta (α-β)

Motivación:



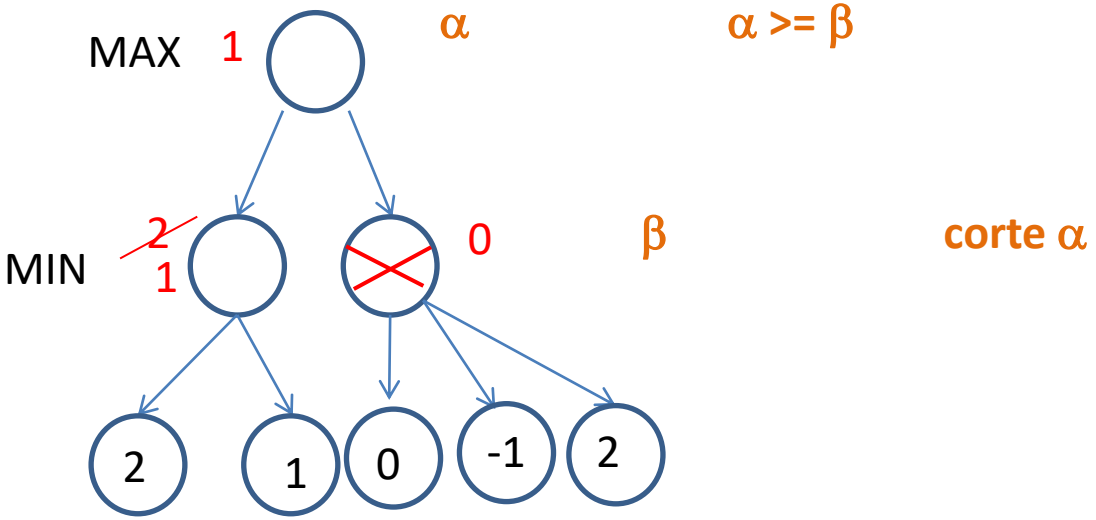
3. Algoritmo Minimax: poda alfa-beta (α-β)

Motivación:



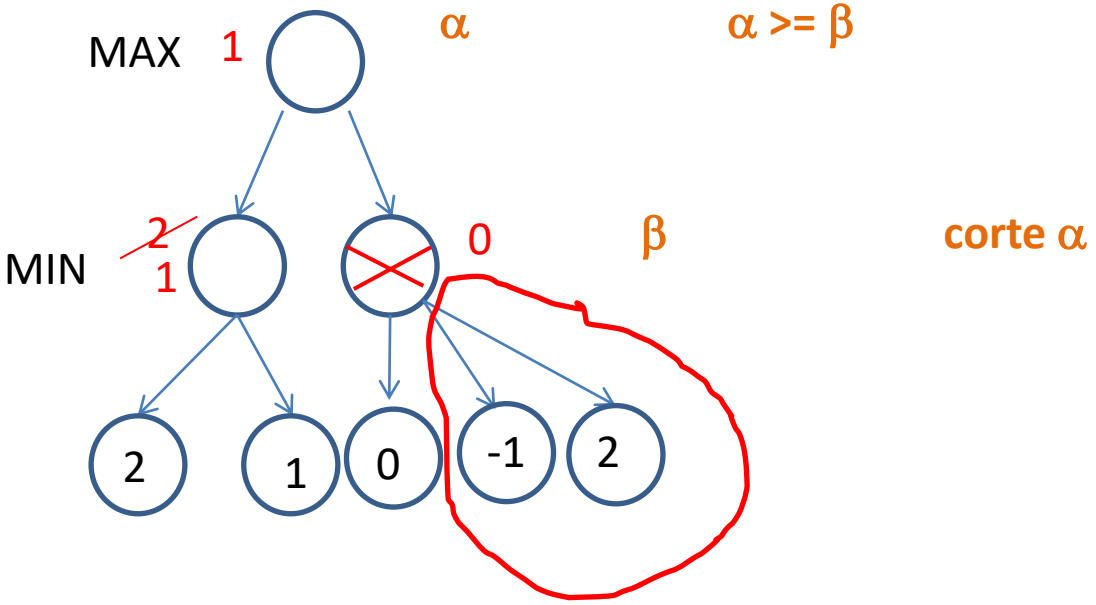
3. Algoritmo Minimax: poda alfa-beta (α-β)

Motivación:



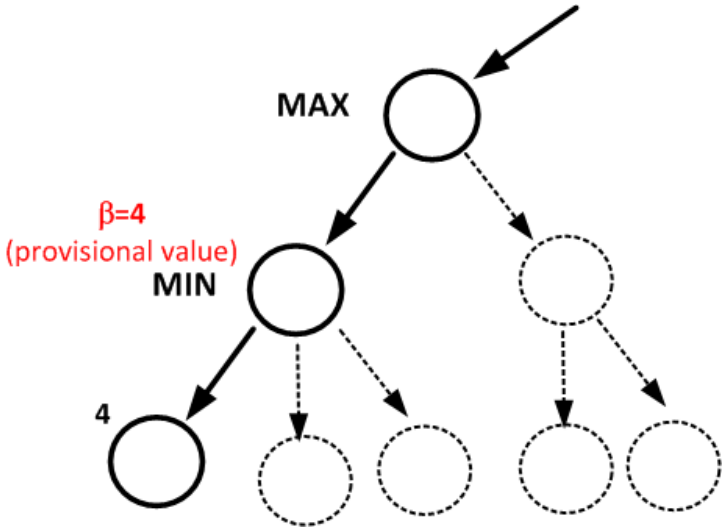
3. Algoritmo Minimax: poda alfa-beta (α-β)

Motivación:

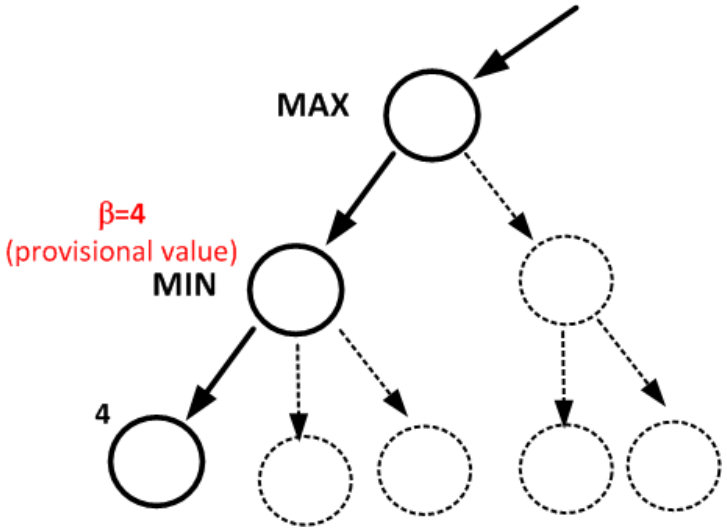


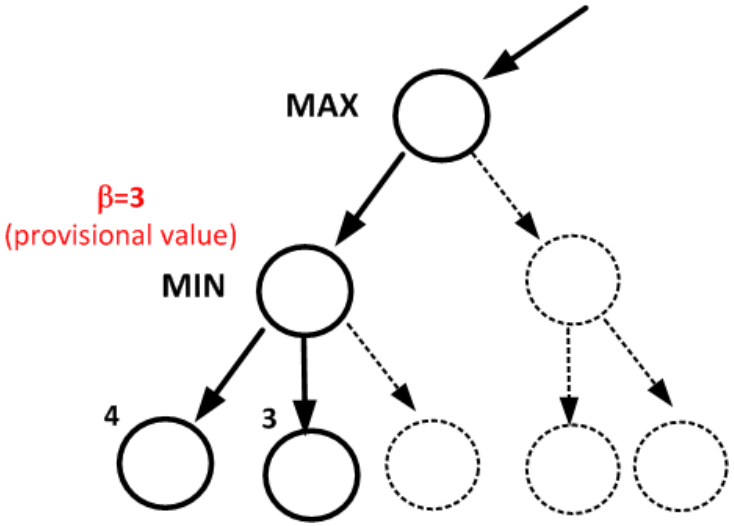
3. Poda alfa-beta (α-β)

3. Poda alfa-beta (α-β)

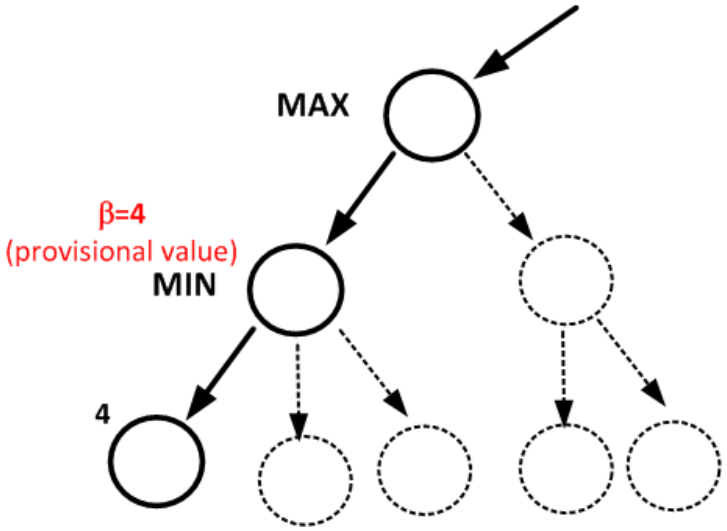


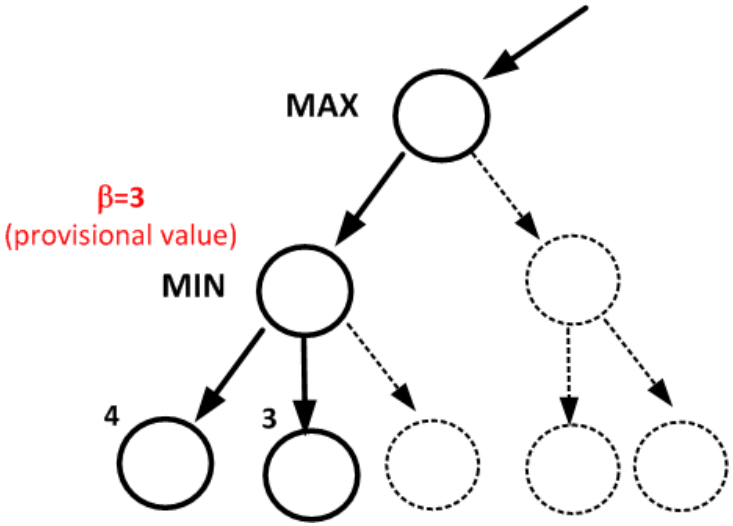
3. Poda alfa-beta (α-β)

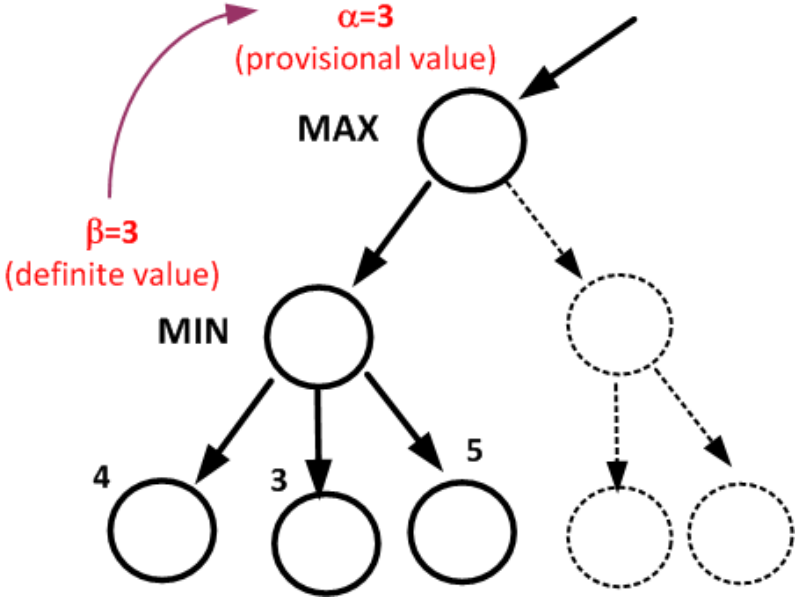




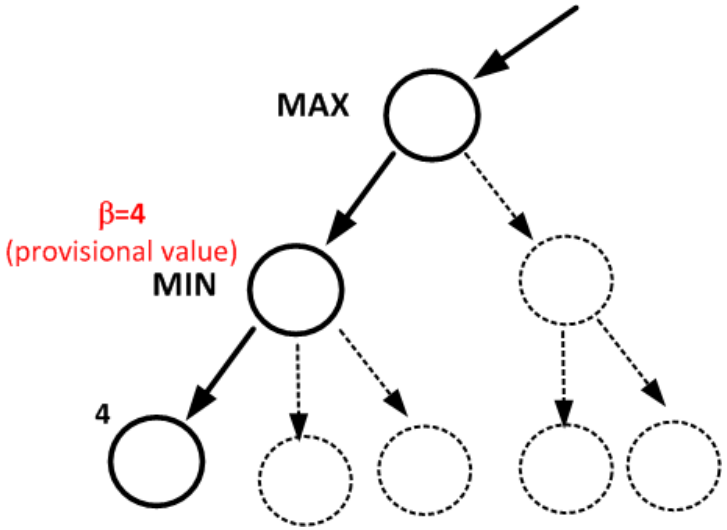
3. Poda alfa-beta (α-β)

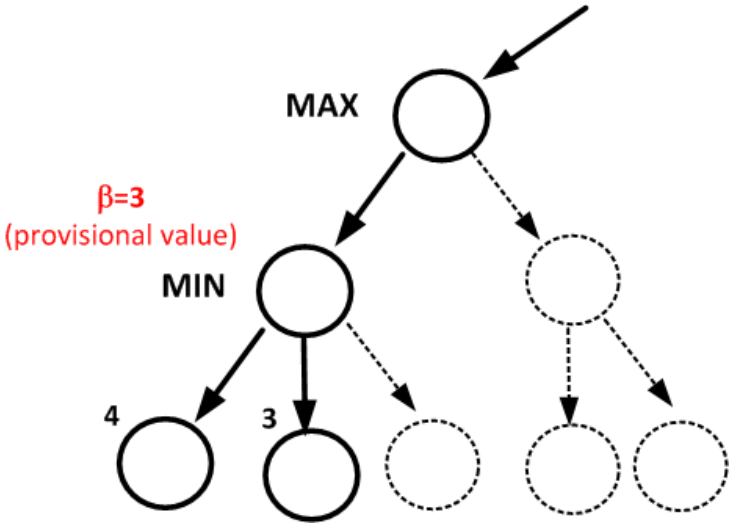


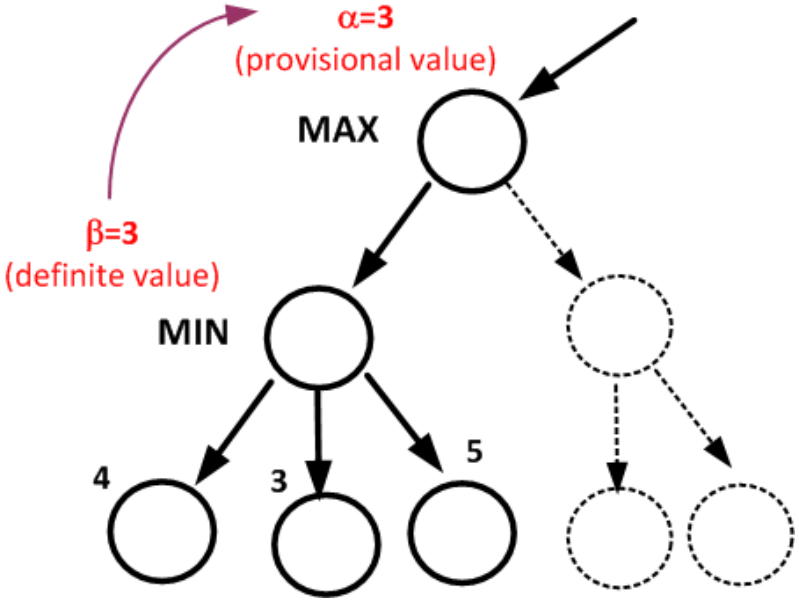


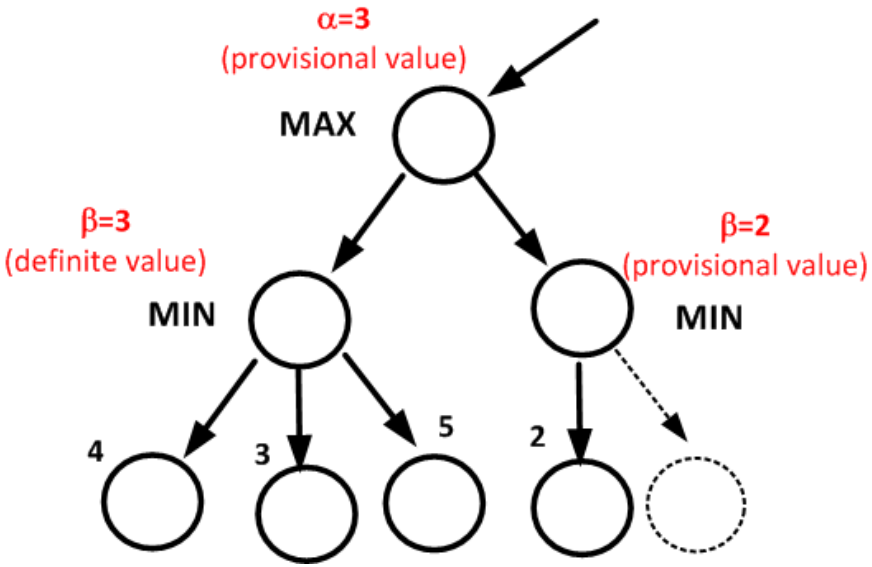


3. Poda alfa-beta (α-β)

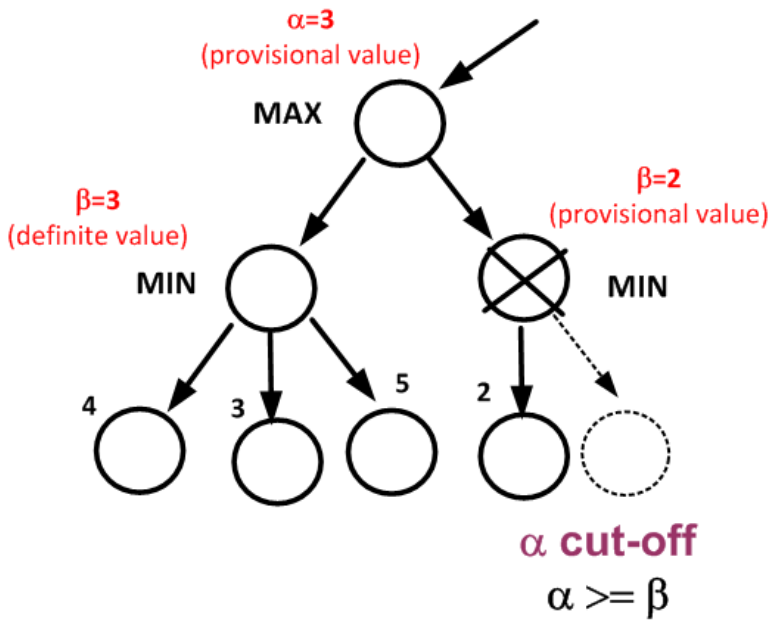




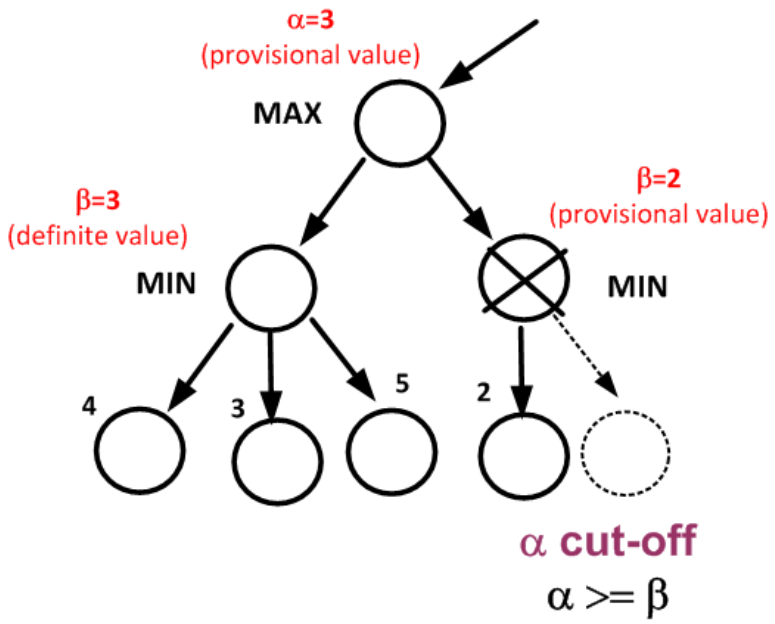


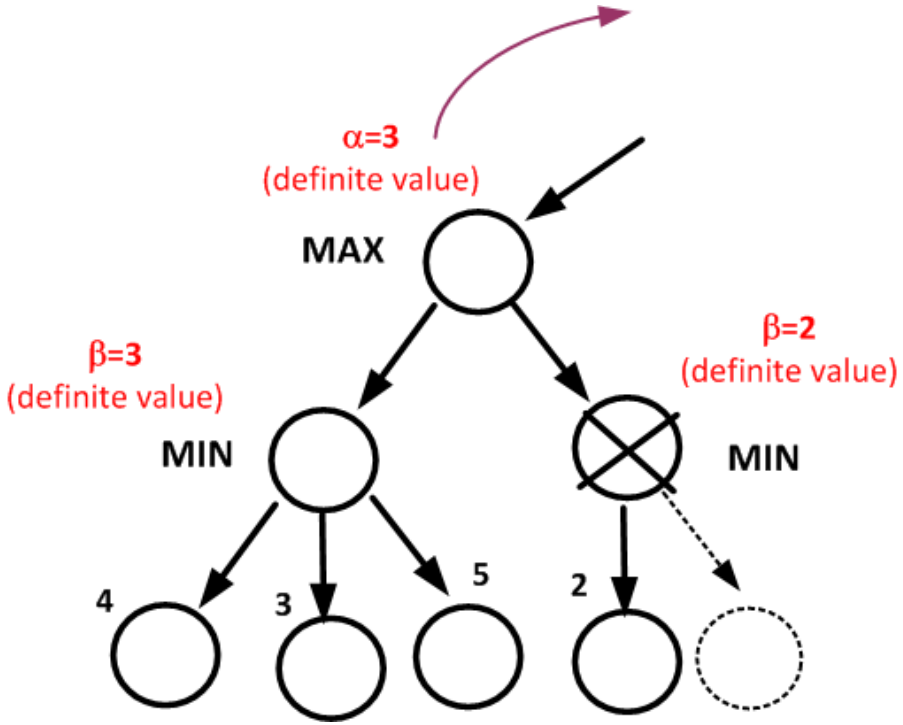


3. Poda alfa-beta (α-β)

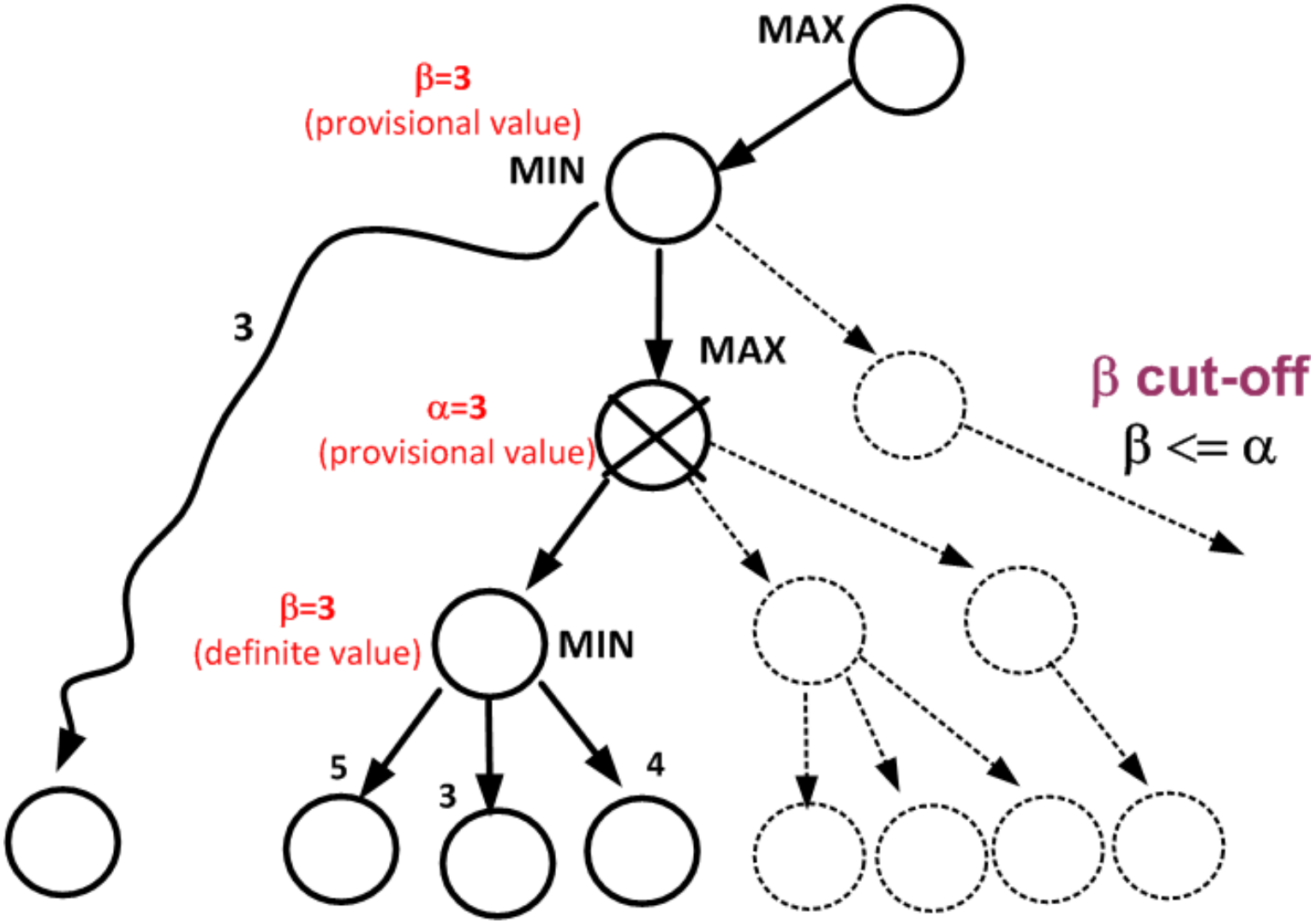


3. Poda alfa-beta (α-β)

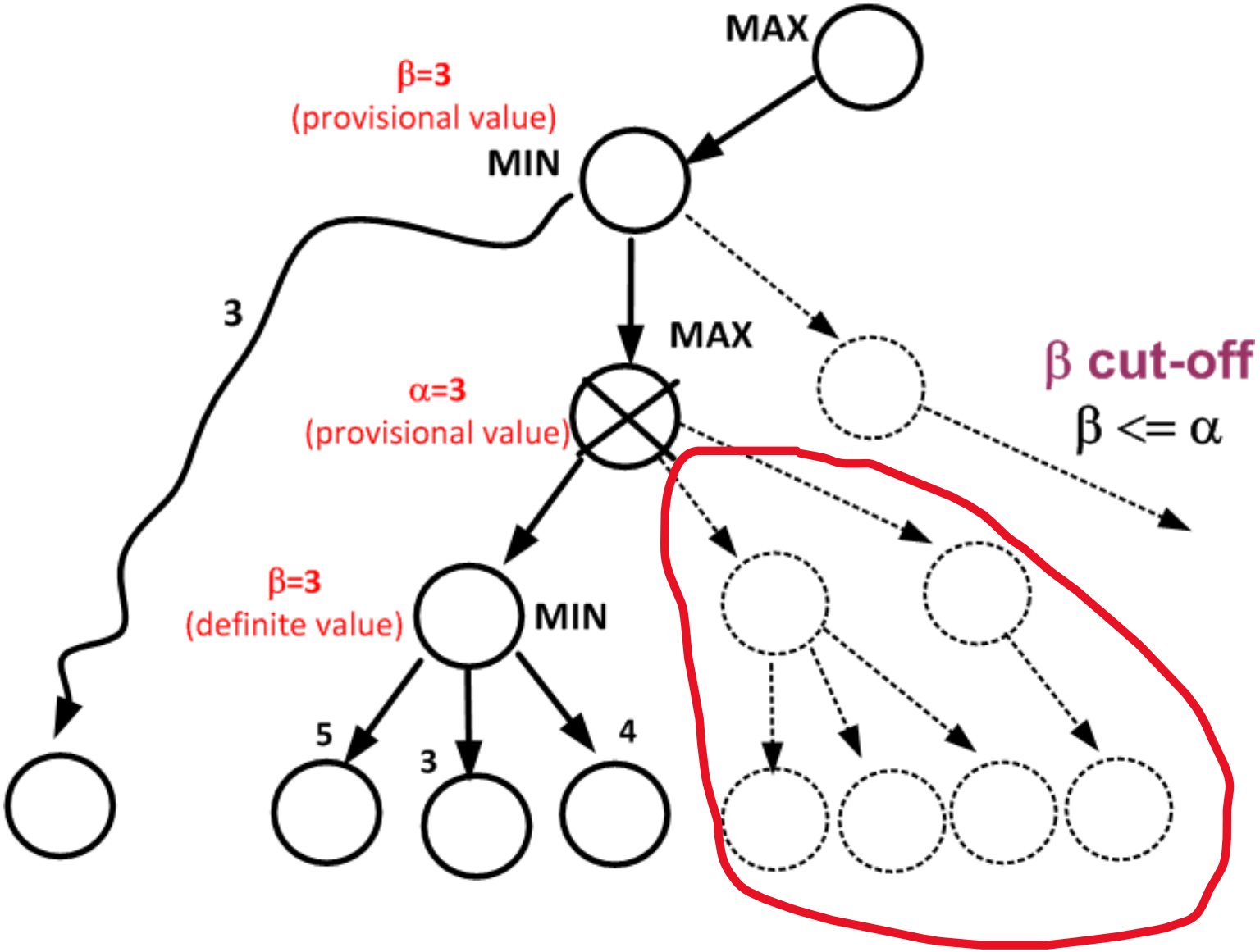




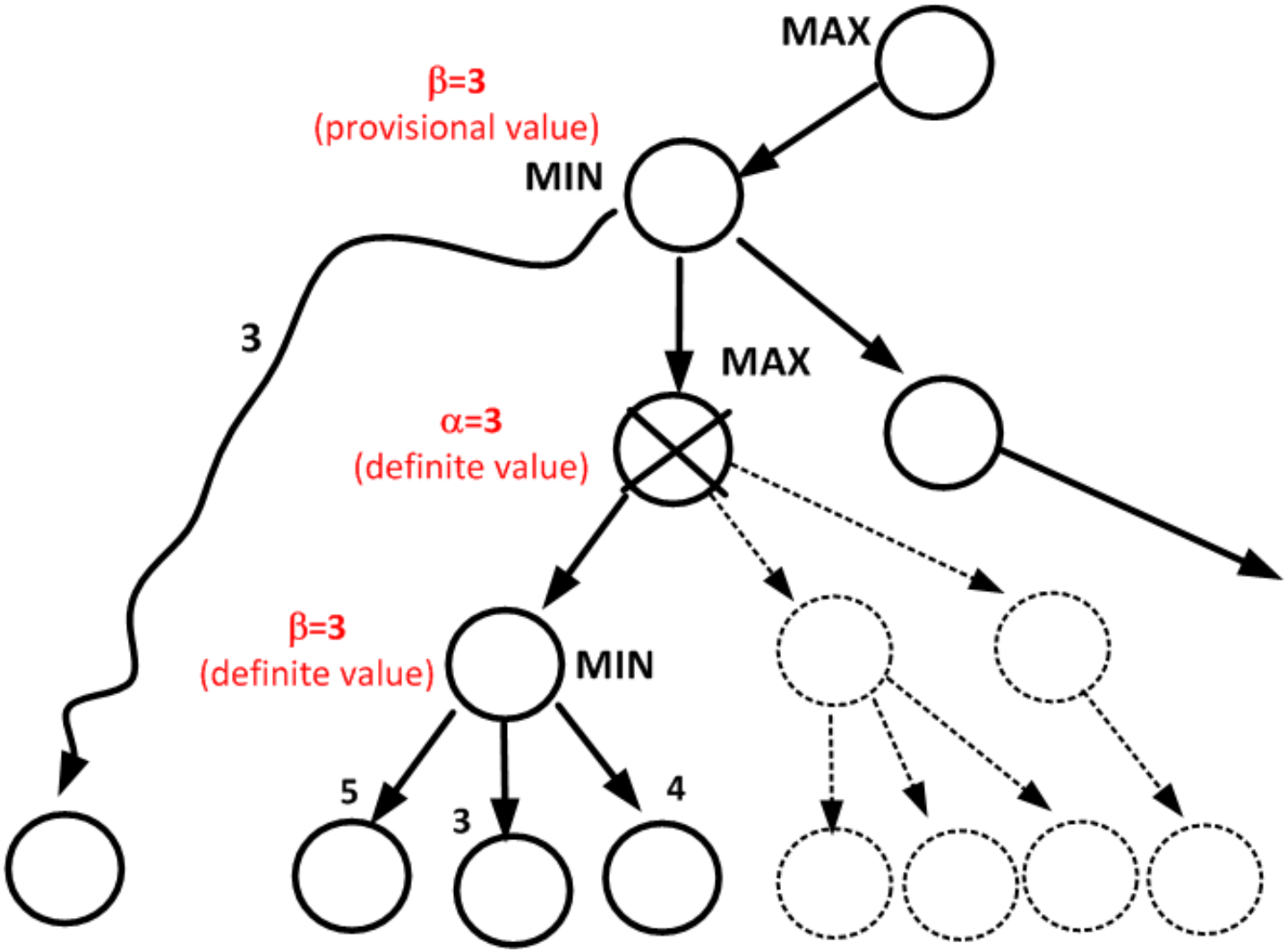
3. Poda alfa-beta (α-β)



3. Poda alfa-beta (α-β)



3. Poda alfa-beta (α-β)



3. Poda alfa-beta (α-β)

1) Inicialmente:

Nodos MAX: valores α (inicialmente $\alpha = -\infty$)
Nodos MIN: valores β (inicialmente $\beta = +\infty$)

2) Generar un nodo terminal en profundidad. Calcular su valor de utilidad.

3) Volcar el valor al nodo padre y mantener el mejor valor para el padre.

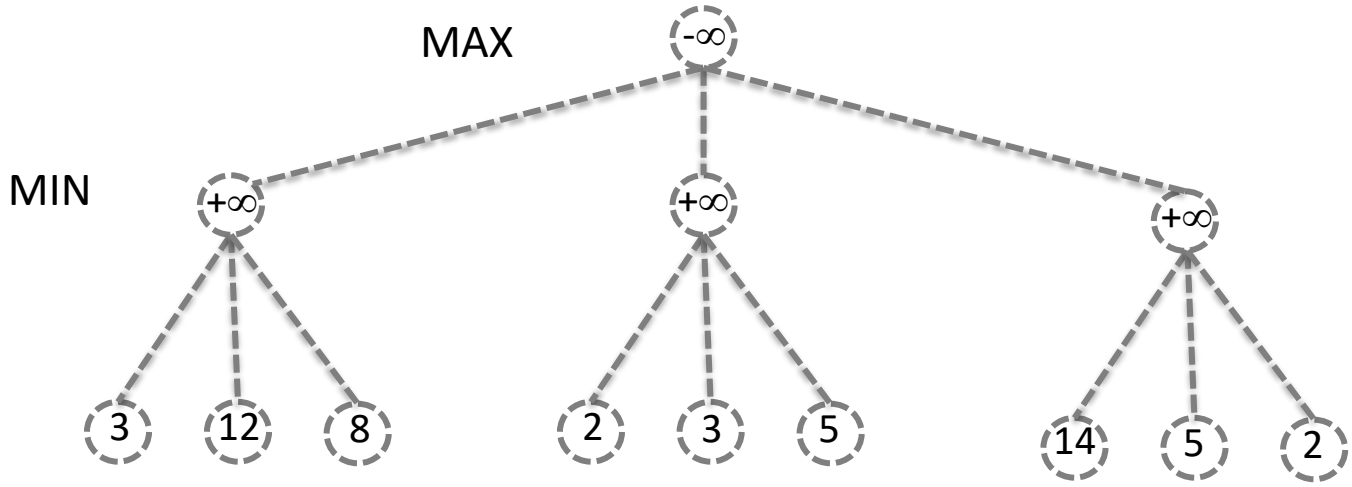
4) Pregunta de corte. Cuando se vuelca un valor a un nodo (MAX o MIN), este valor se compara con los valores de todos los nodos predecesores contrarios con objeto de ver si se produce un corte α o β.

- Si se produce un corte, ir a 3

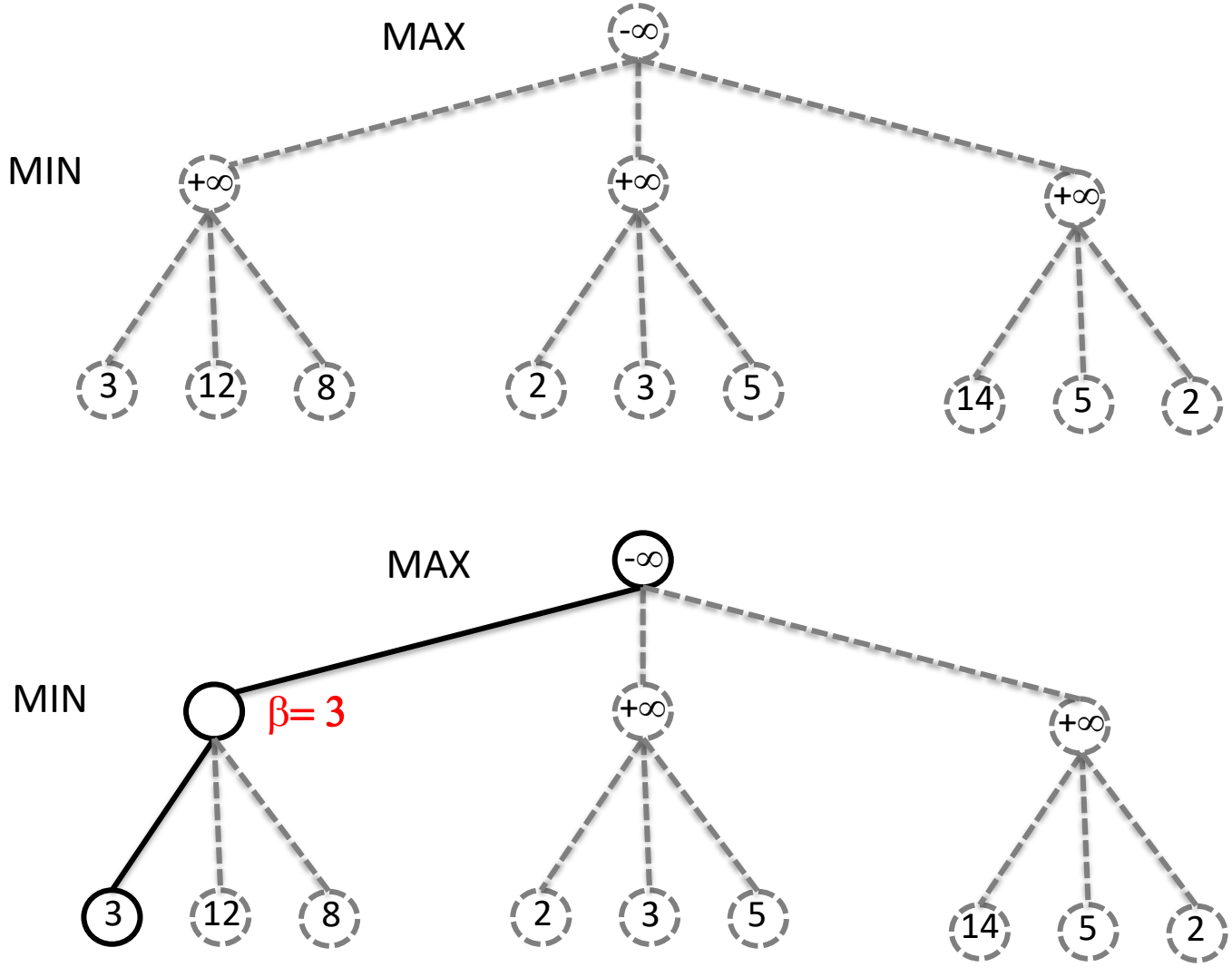
5) Valor provisional/definitivo:

- Si es un valor provisional, ir a 2
- Si es un valor definitivo, ir a 3

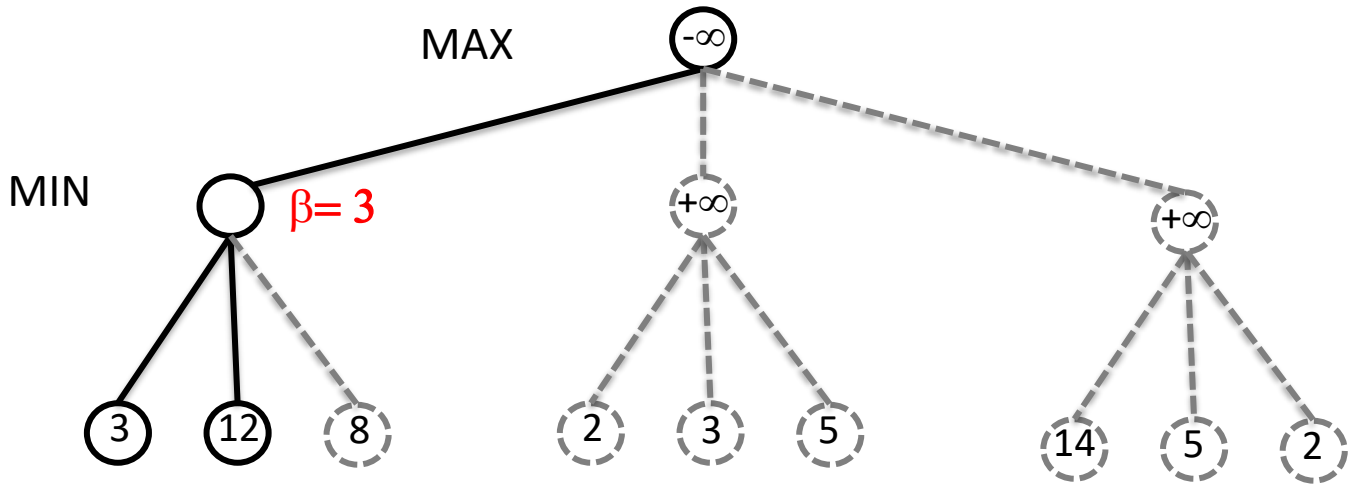
3. Poda alfa-beta (α-β): ejemplo 1



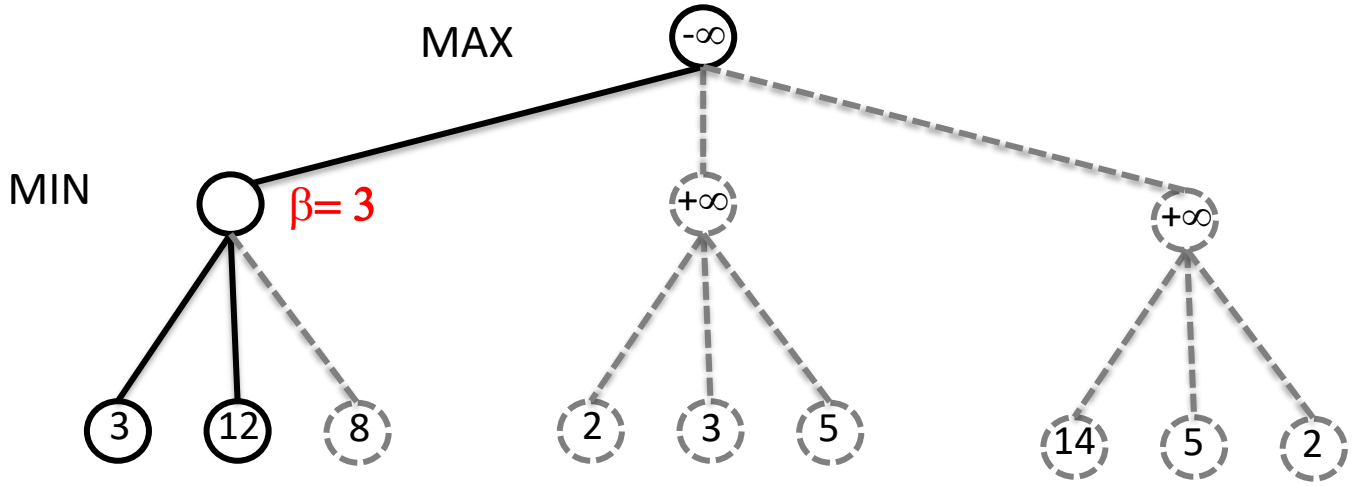
3. Poda alfa-beta (α-β): ejemplo 1

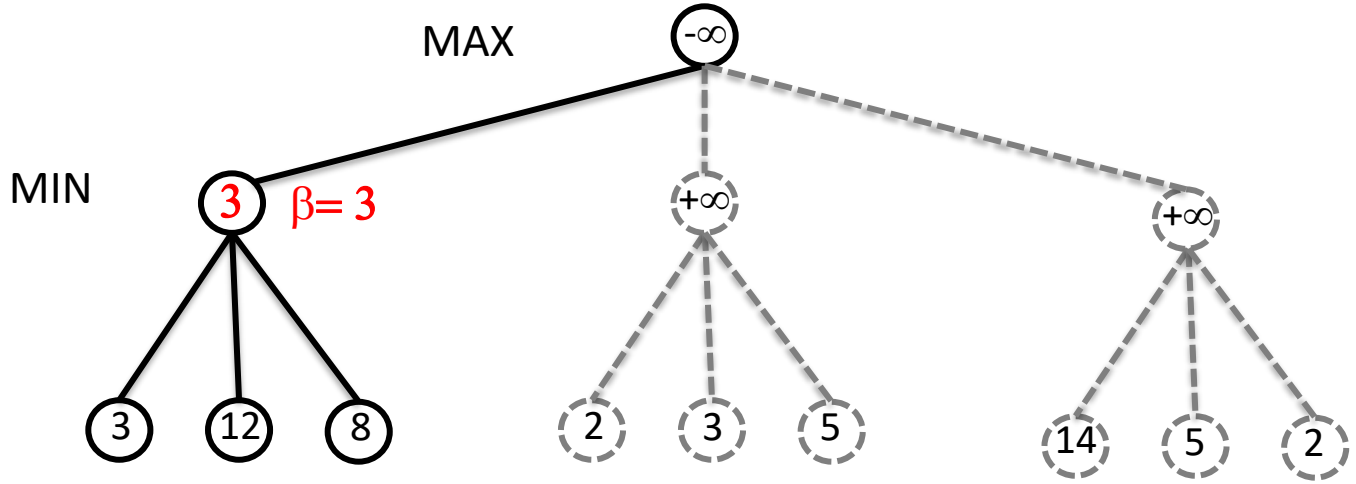


3. Poda alfa-beta (α-β): ejemplo 1

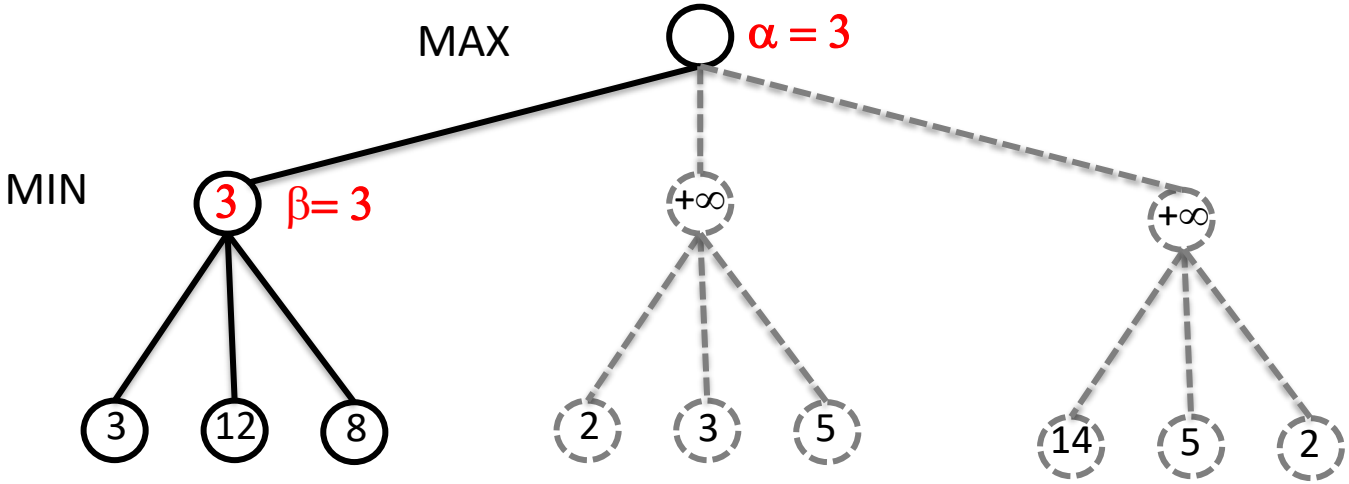


3. Poda alfa-beta (α-β): ejemplo 1

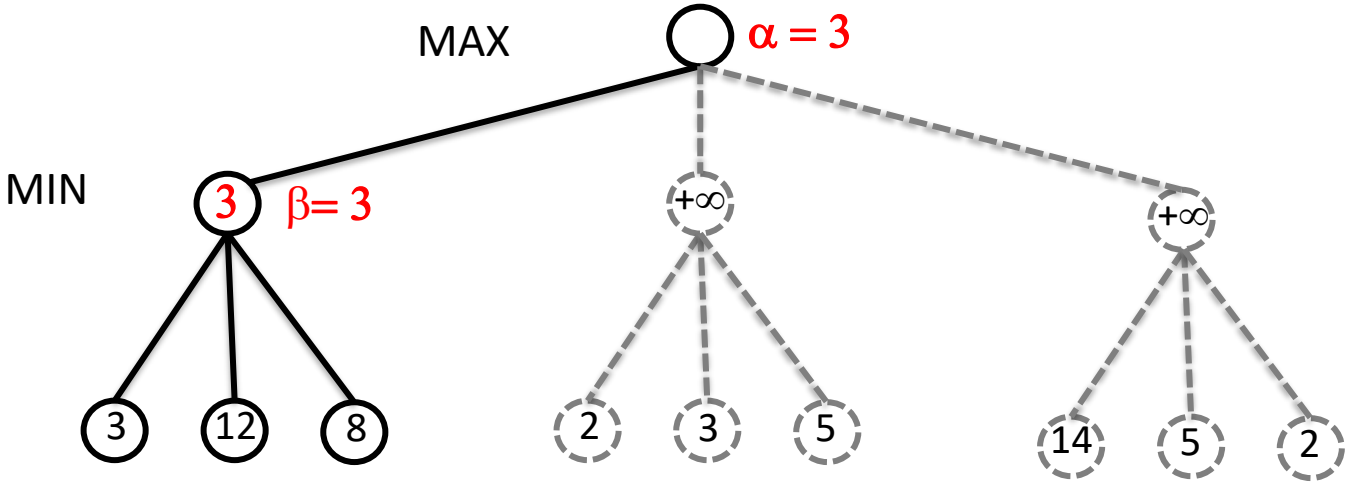


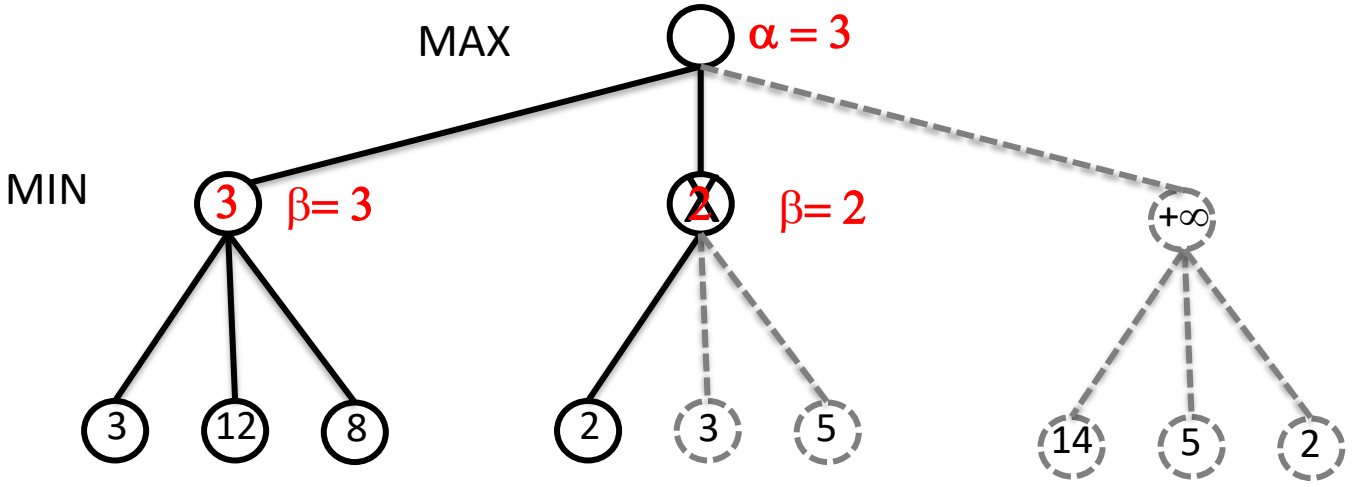


3. Poda alfa-beta (α-β): ejemplo 1

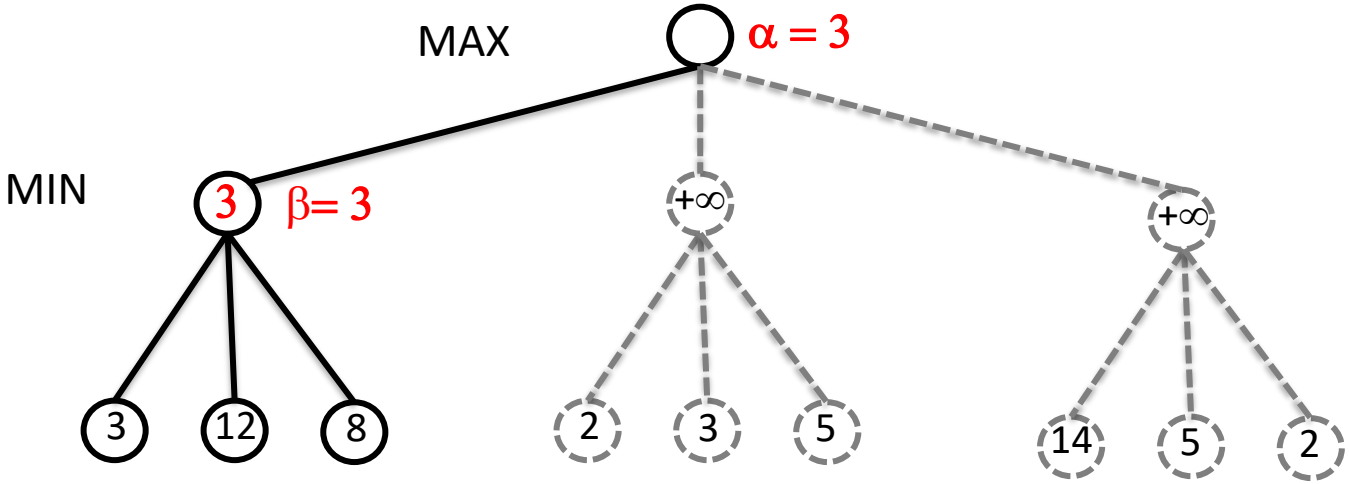


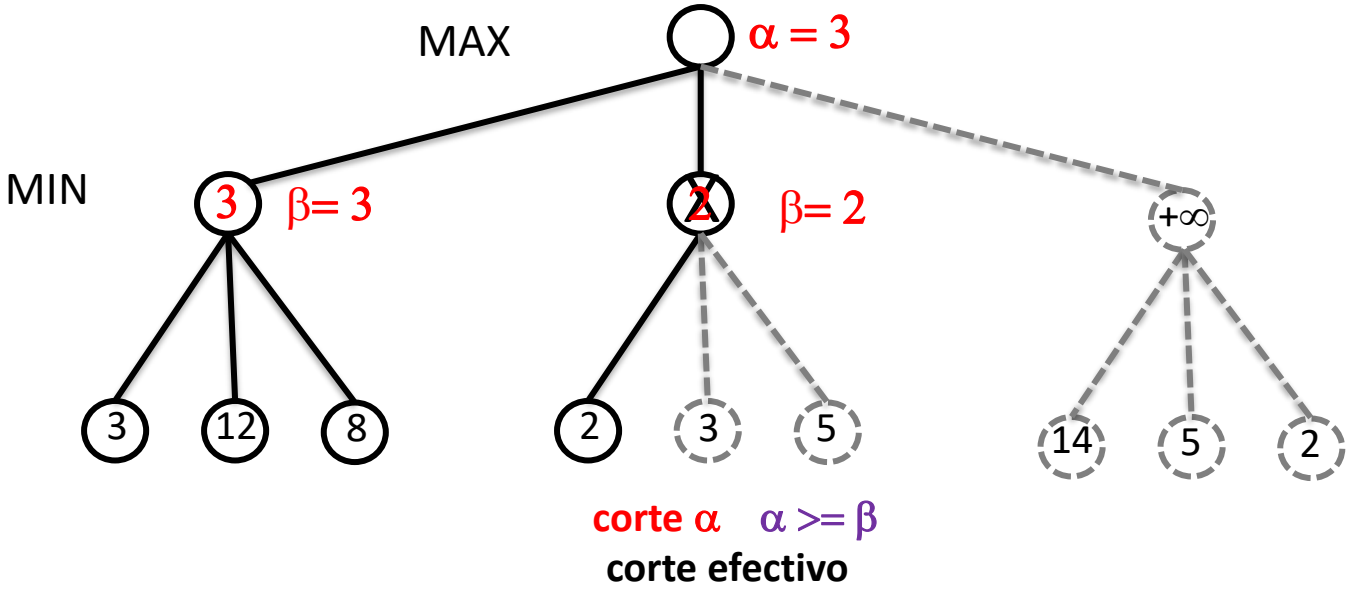
3. Poda alfa-beta (α-β): ejemplo 1



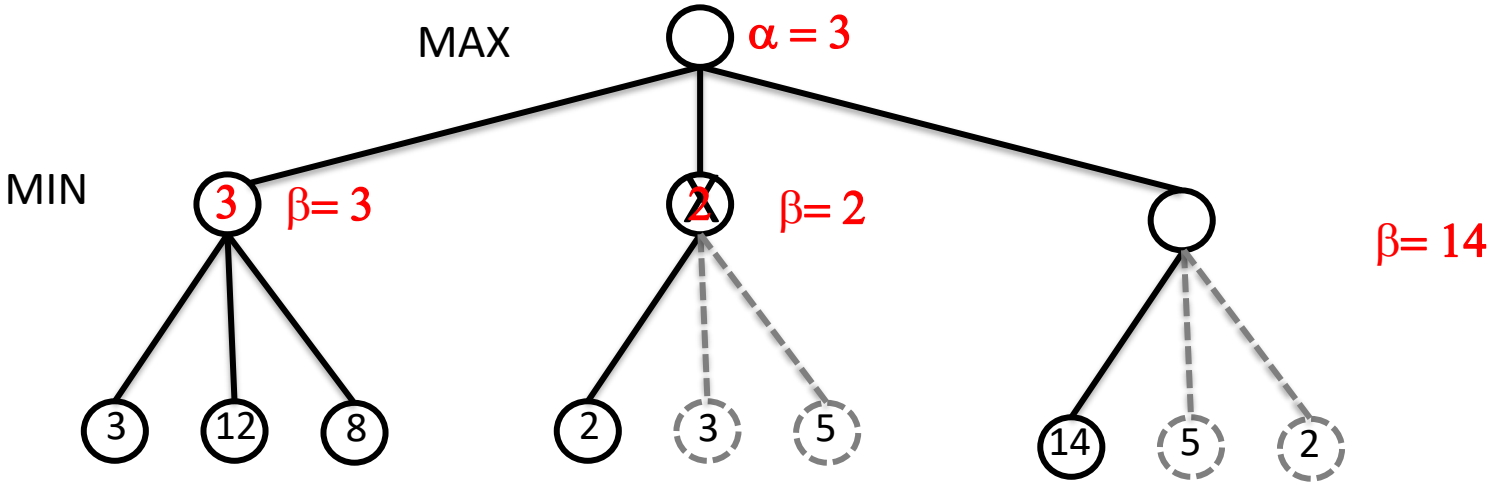


3. Poda alfa-beta (α-β): ejemplo 1

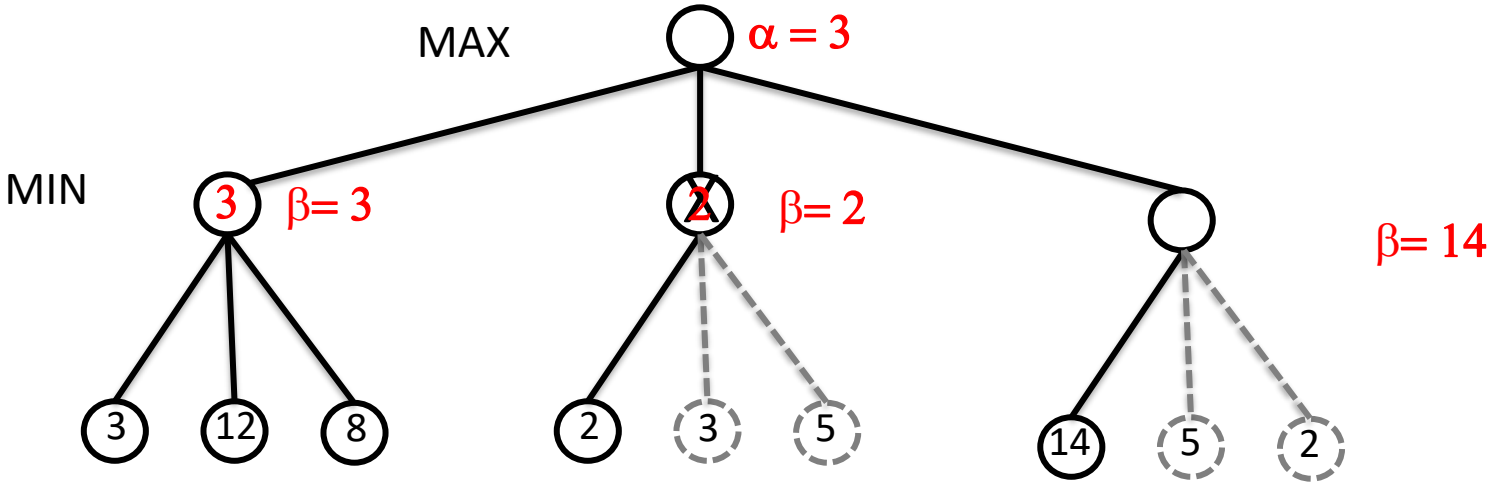


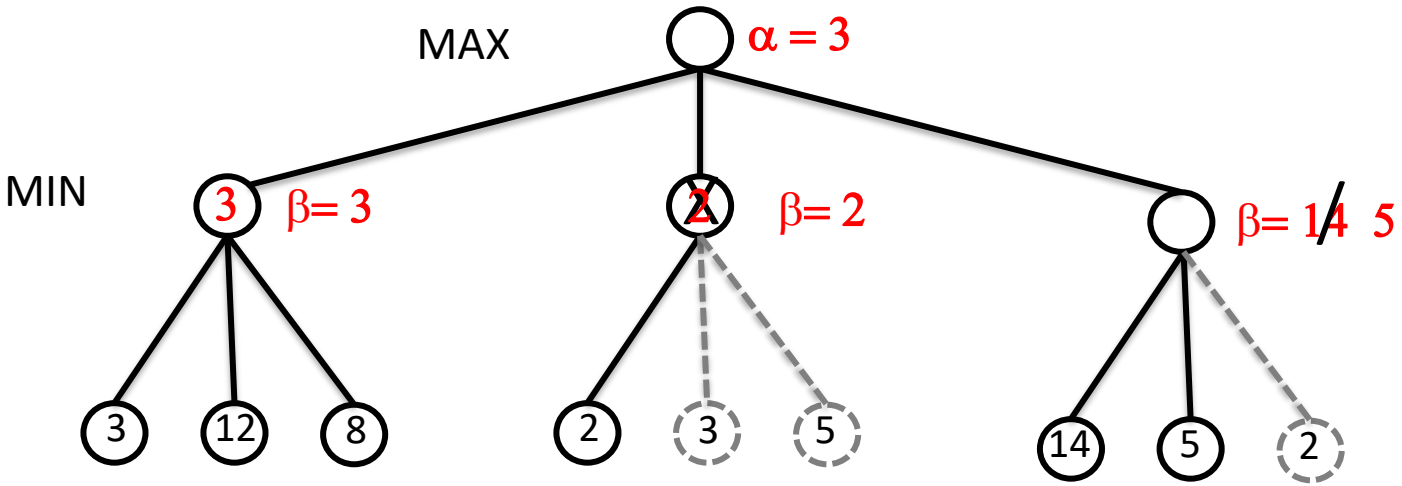


3. Poda alfa-beta (α-β): ejemplo 1

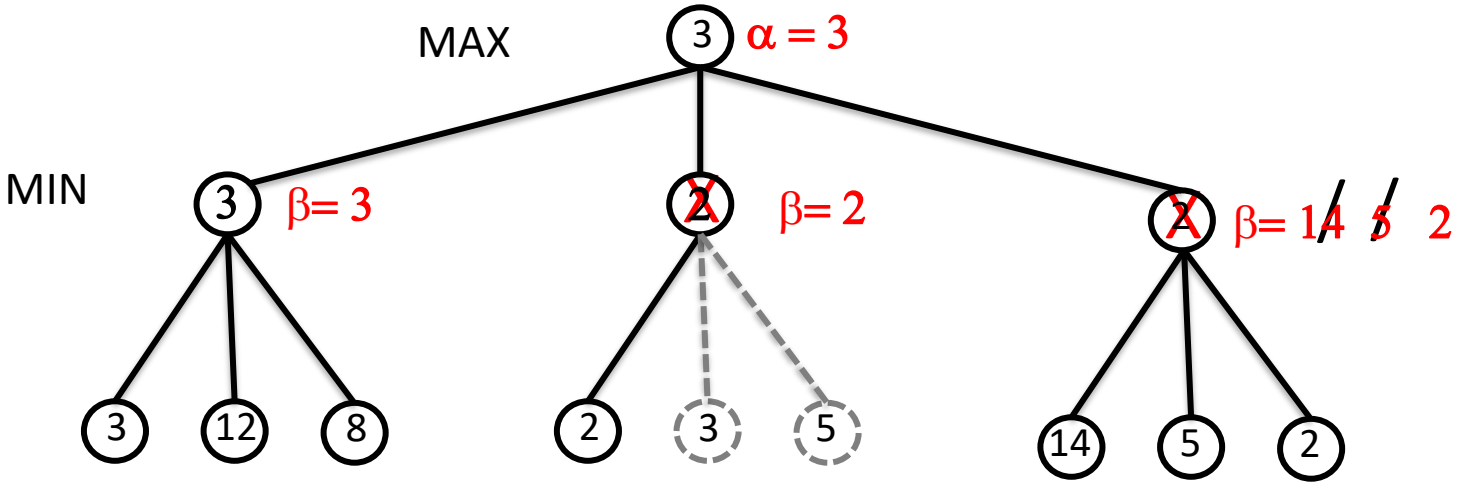


3. Poda alfa-beta (α-β): ejemplo 1

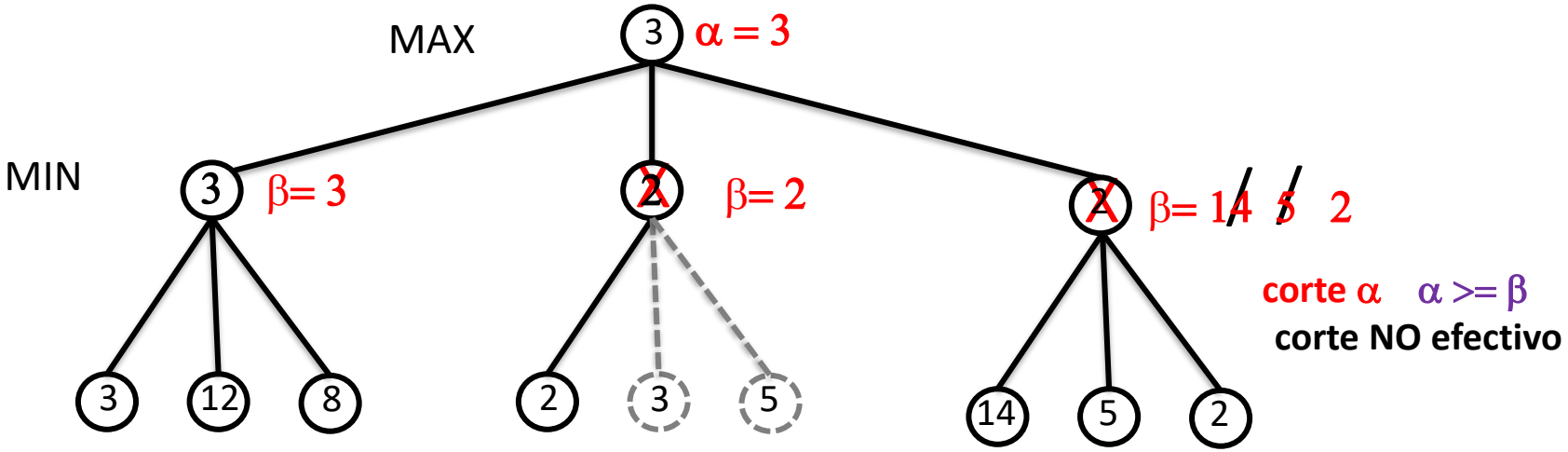




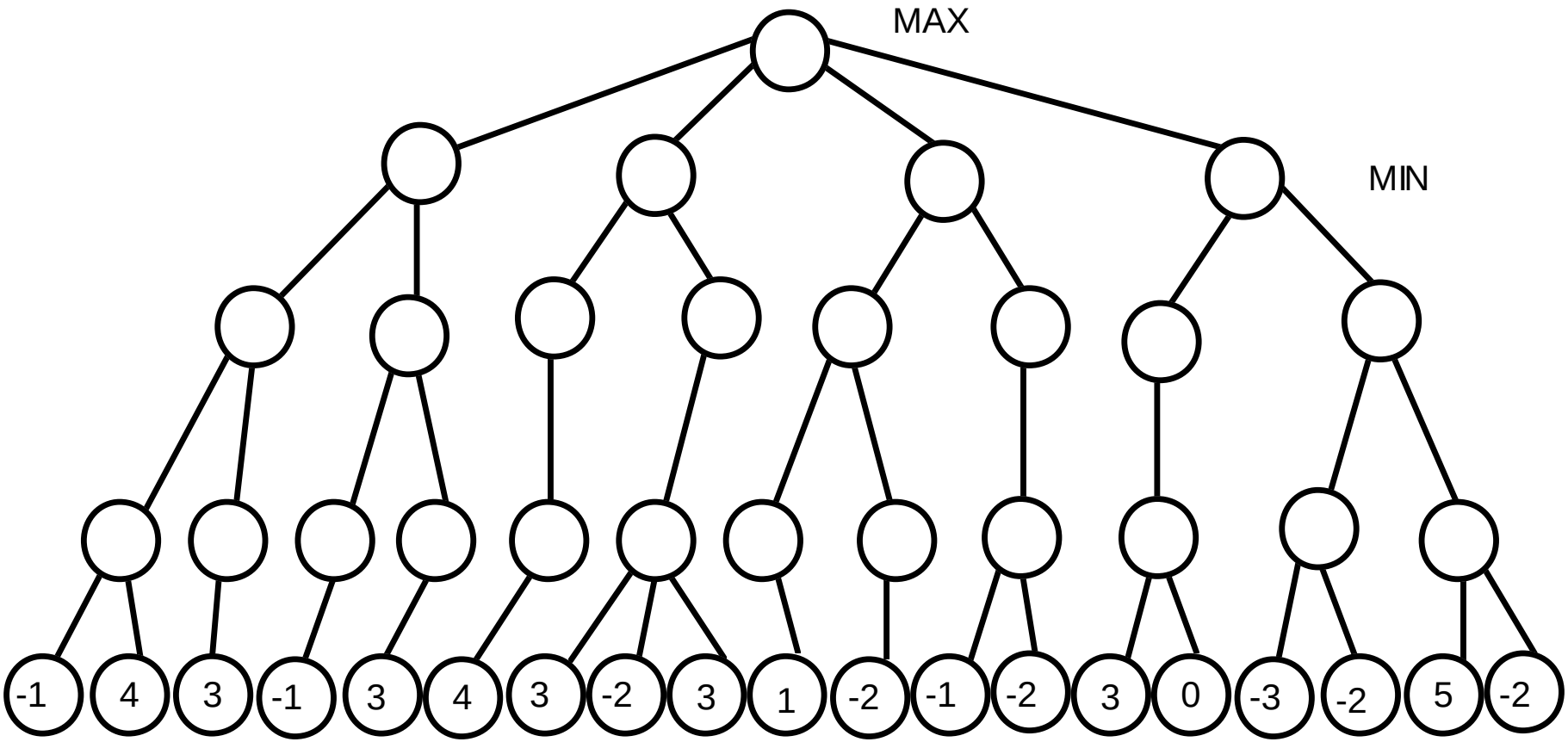
3. Poda alfa-beta (α-β): ejemplo 1



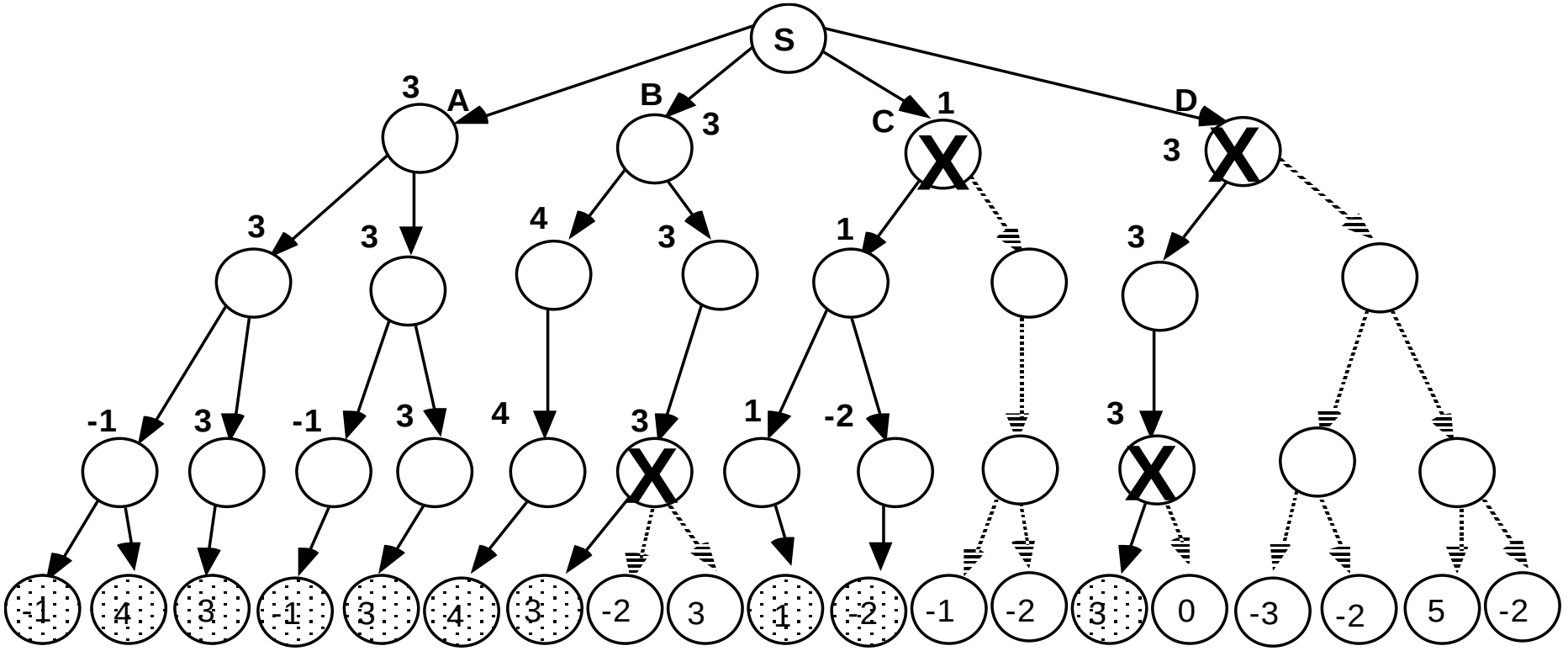
3. Poda alfa-beta (α-β): ejemplo 1



3. Poda alfa-beta (α-β): ejemplo 2



3. Poda alfa-beta (α-β): ejemplo 2



3. Poda alfa-beta (α-β): características

Los nodos MAX tienen valores α:

- Tienen garantizado conseguir, a partir de ese nodo, un estado con un valor ≥ α
- Los valores α provisionales de un nodo MAX son cotas inferiores del nodo y nunca pueden disminuir al desarrollar la búsqueda.
- El valor α de un nodo no terminal es el mayor valor entre los valores volcados de sus nodos sucesores.

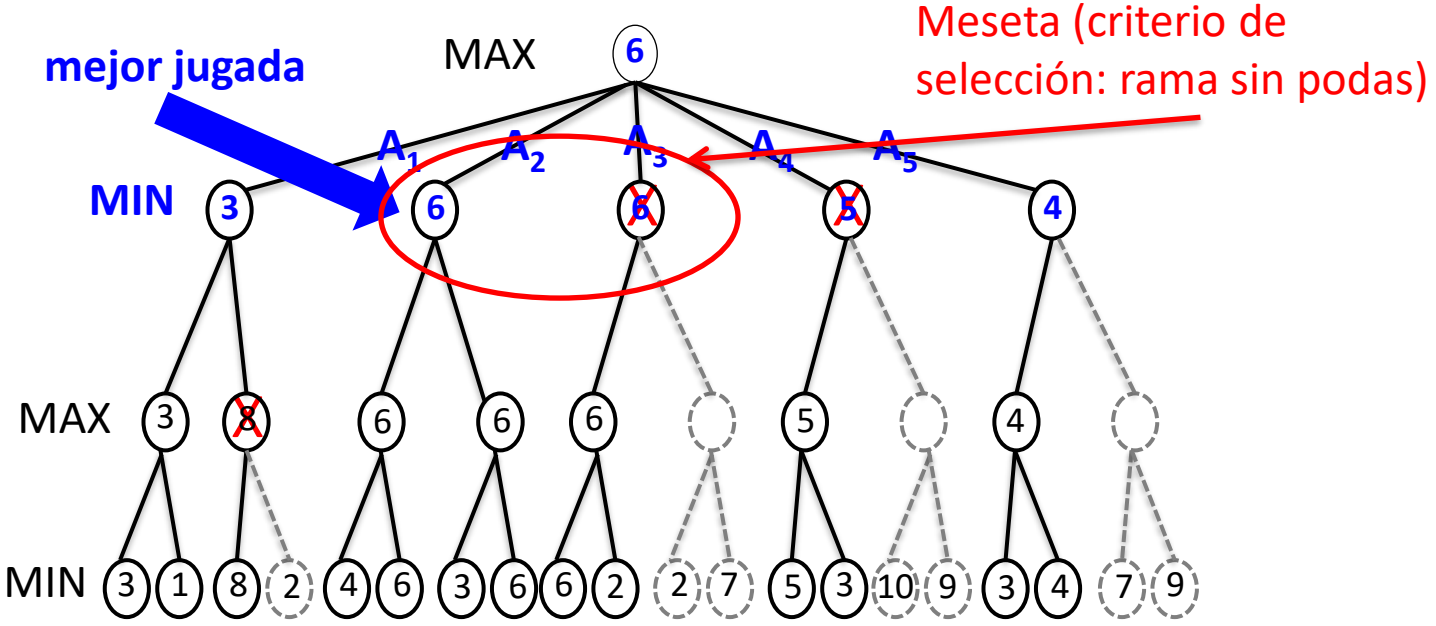
Los nodos MIN tienen valores β :

- Tienen garantizado conseguir, a partir de ese nodo, un estado con un valor ≤ β.
- Los valores β provisionales de un nodo MIN son cotas superiores del nodo, y nunca pueden aumentar al desarrollar la búsqueda.
- El valor β de un nodo no terminal es el menor valor entre los valores volcados de sus nodos sucesores.

3. Poda alfa-beta (α-β): características

Los valores finales de los nodos en el procedimiento $\alpha-\beta$ son iguales que en el procedimiento MINIMAX, excepto aquellos nodos cuyos valores provengan de cortes α (podría ser menor) o β (podría ser mayor).

Si se produce un corte en un nodo **n** seleccionable por el nodo MAX inicial, el nodo **n** no se escogerá porque su valor es mejorado por otro nodo seleccionable que no ha sido cortado.



3. Poda alfa-beta (α-β): características

La eficiencia del procedimiento alfa-beta depende del nº de cortes que se realizan para un orden de expansión aleatorio.

El número de cortes que se pueden hacer depende del orden en que aparezcan los valores alfa-beta que permiten un corte. El algoritmo alfa-beta será más eficiente si examinamos en primer lugar los sucesores que probablemente sean los mejores.

El valor final asumido por el nodo inicial es igual al valor de la evaluación de un nodo terminal (nodo óptimo en el nivel obtenido): si este nodo aparece pronto, en la búsqueda en profundidad, el nº de cortes será máximo, generando el mínimo árbol de búsqueda.

En un alfa-beta óptimo se generaran $O(b^{d/2})$ mientras que en Mini-Max se generan $O(b^d)$, luego el factor de ramificación eficaz es $\sqrt{b}$ en lugar de b, por lo tanto, el número de nodos terminales que se generan en un árbol con profundidad n, es igual al número de nodos terminales que se generarían en un procedimiento Minimax de profundidad n/2. Es decir, dada una limitación espacio/tiempo, el procedimiento alfa-beta permite una profundidad doble que el procedimiento Minimax.

4. Algoritmo α-β: refinamientos

Diferentes variantes del $\alpha-\beta$ para encontrar el mejor nodo terminal lo antes posible y maximizar el número de cortes:

A) Expansión ordenada a cada nivel
B) Orden preliminar

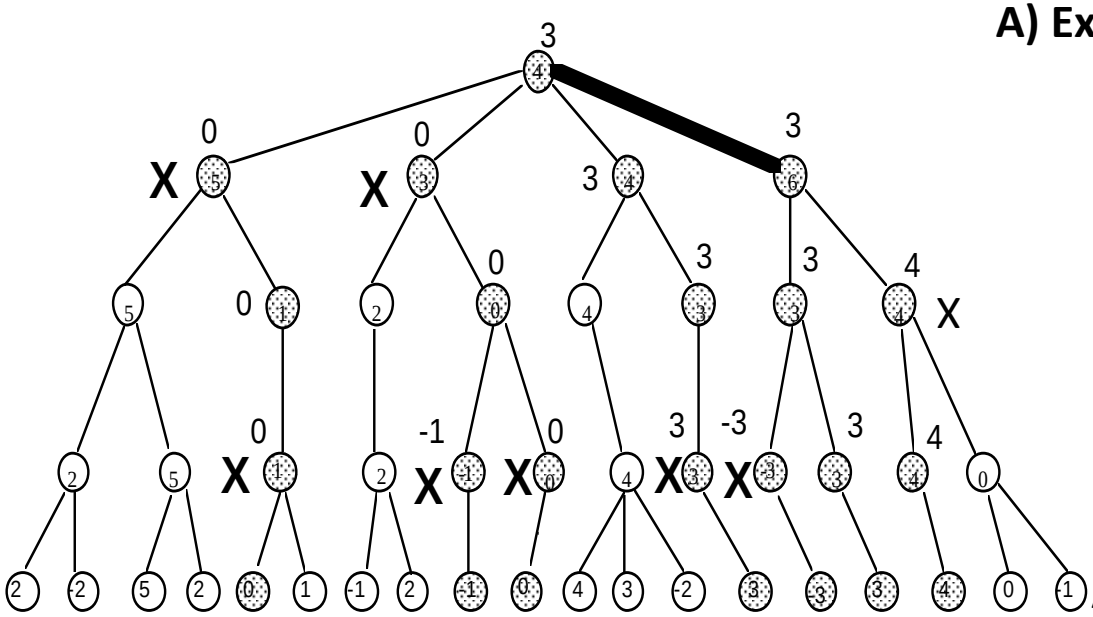
En Juegos es necesario:

- Utilizar funciones heurísticas para podar el árbol
- Utilizar procedimientos más inteligentes para conseguir un comportamiento dinámico

Ejemplos:

- Tener una serie de movimientos clásicos (biblioteca de movimientos, salidas, preferencia de piezas, zonas de tablero, …)
- Tener patrones de tablero y mejores movimientos para esas posiciones
- Tácticas y estrategias de juego, hábitos del oponente, fallos (movimientos erróneos), etc.
- Aprendizaje de la función de evaluación.

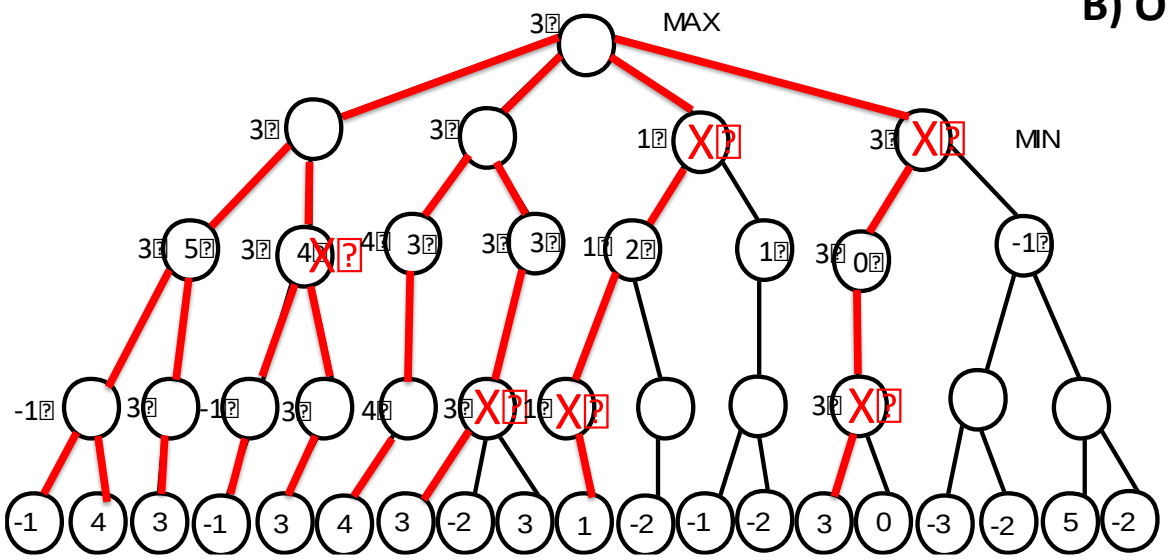
4. Algoritmo α-β: refinamientos



A) Expansión ordenada a cada nivel:

- Se expanden todos los sucesores del nodo expandido (a partir del nodo inicial).
- Se evalúan los nodos sucesores expandidos.
- Se expande aquel nodo que:
  - tenga un valor máximo, si es sucesor de un nodo max
  - tenga un valor mínimo, si es sucesor de un nodo min
- Este método es una combinación anchura / profundidad, y supone no elegir los sucesores a expandir de cada nodo de forma arbitraria.

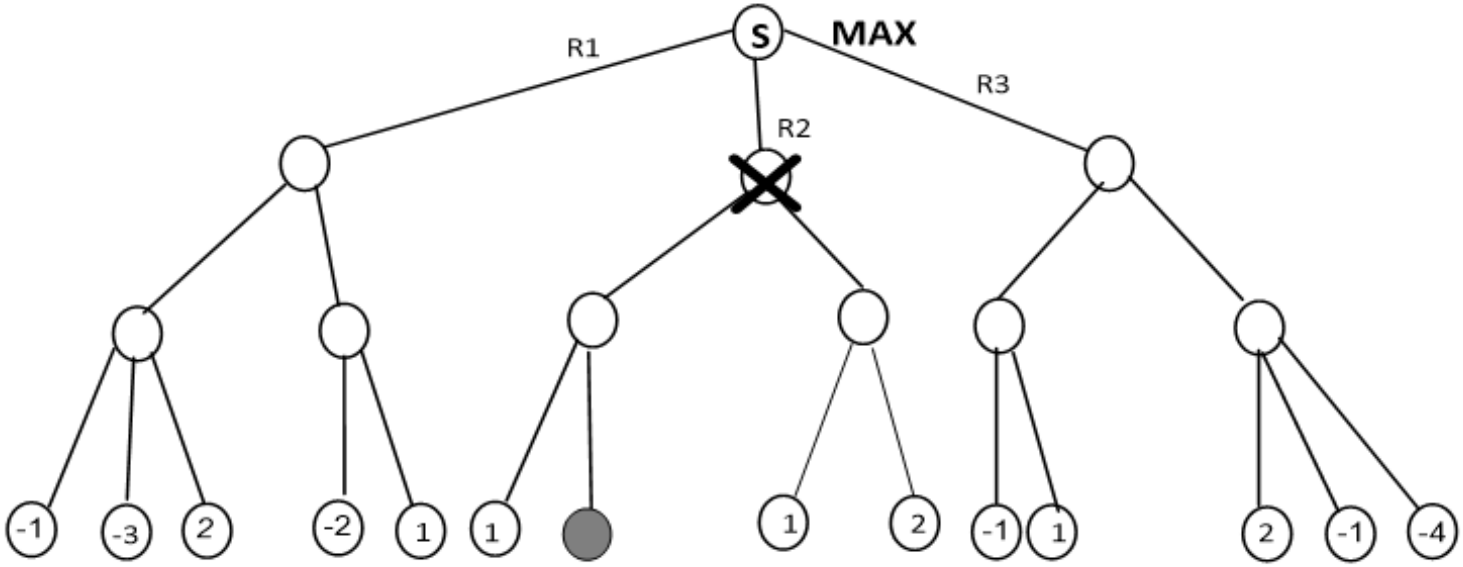
4. Algoritmo α-β: refinamientos



B) Orden preliminar:

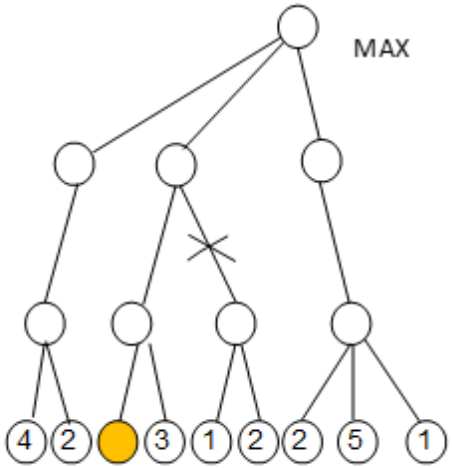
- Expandir el árbol en anchura hasta un nivel m.
- Evaluamos los nodos terminales del nivel m.
- Ordenar los nodos del nivel m en función de su mejor posición para el nodo inicial.
- Empezamos un procedimiento $\alpha-\beta$ a partir de ese nivel hasta el deseado, pero expandiendo los nodos en orden a su mejor promesa.

17) Dado el espacio de búsqueda de un juego representado en la figura siguiente, asumiendo que se aplica un procedimiento alfa-beta, indica el valor que debería tomar el nodo sombreado para que se produzca el corte señalado en la rama R2:



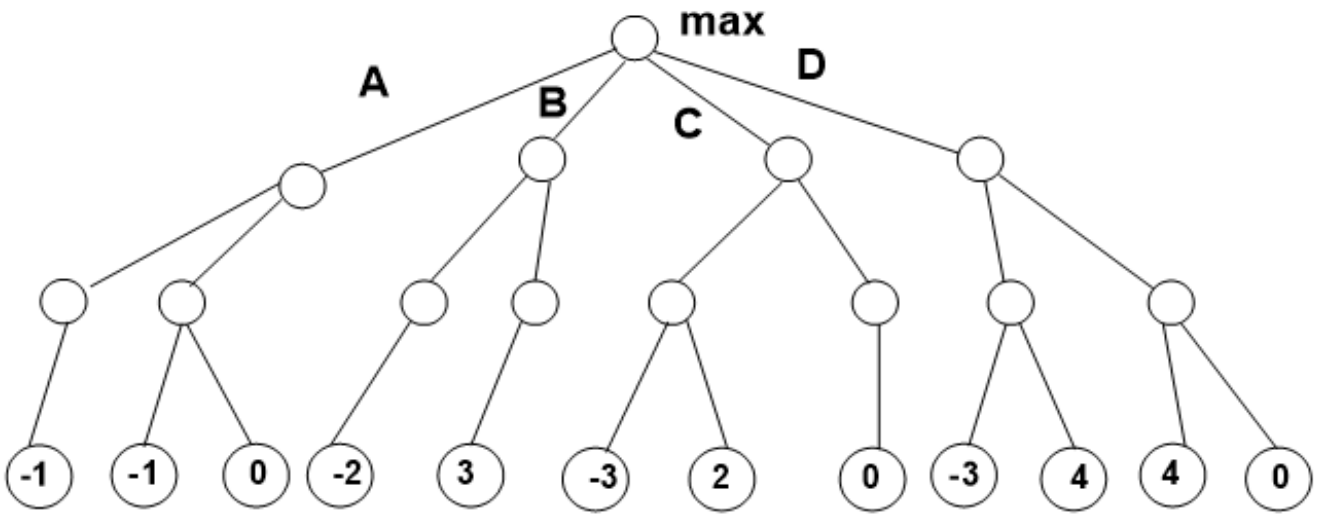
A. Un valor en [- ∝, 1]
B. Un valor en [1, +∝]
C. El nodo sombreado solo puede tomar el valor 1
D. No se puede producir el corte de la figura.

14) ¿Qué valores debería tener el nodo sombreado para que se produzca siempre el corte mostrado en la figura?



A. Cualquier valor comprendido en [-∞ 4]
B. Cualquier valor.
C. Cualquier valor comprendido en [4 +∞]
D. Nunca se producirá

25) Dado el árbol de juego de la figura, ¿Cuántos nodos evitamos generar respecto a un algoritmo MINIMAX si realizamos una exploración alfa-beta?

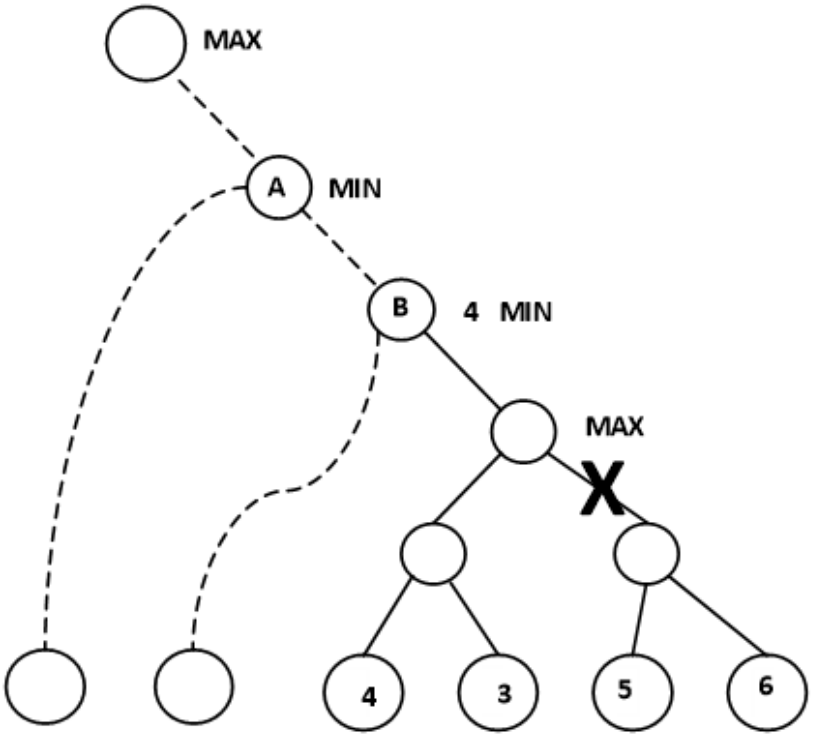


A. 3
B. 4
C. 5
D. 6

22) Sea n1 y n2 los dos únicos nodos hijo de un nodo n el cual es un nodo MAX en un árbol de juego. Asumimos que se explora primero el nodo n1 y luego n2. Indica la respuesta CORRECTA:

A. El valor definitivo del nodo n será el máximo entre el valor definitivo de n1 y n2 solo cuando n1 y n2 son nodos terminales.
B. Cuando se vuelca el valor de n1 al nodo padre n, este puede tener asociado un valor volcado anteriormente.
C. Cuando se vuelva el valor de n1 al nodo padre n, se puede producir un corte beta en n.
D. Ninguna de las respuestas anteriores es correcta.

21) Dado el desarrollo parcial de una búsqueda alfa-beta indicado en la figura, donde el nodo B tiene un valor volcado provisional de 4 ¿Qué valor volcado provisional debe tener el nodo A para que se produzca el corte efectivo indicado en la figura?



A. Nunca se producirá el corte
B. Menor o igual que 3
C. Mayor o igual que 3
D. Menor que 3