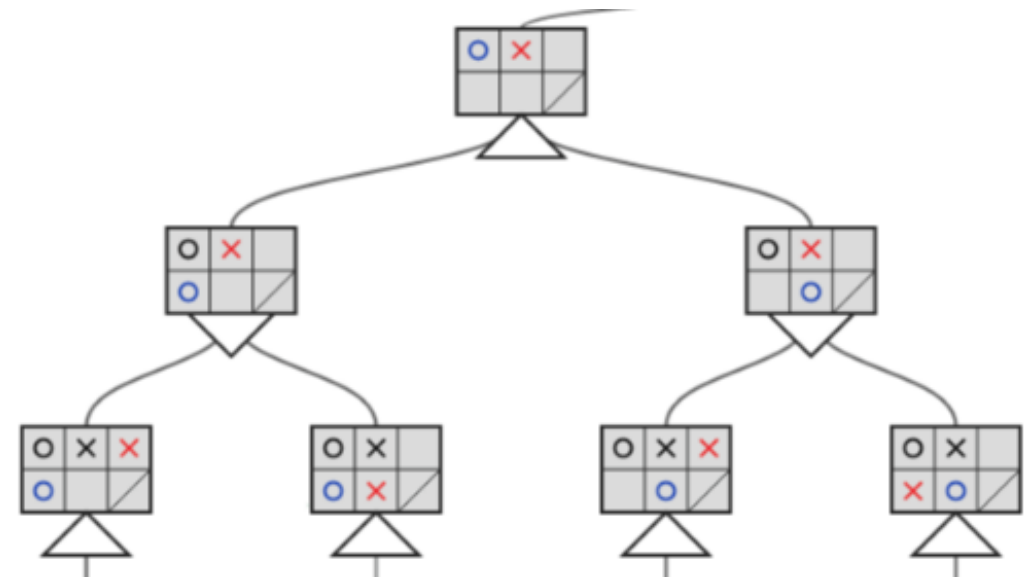




# Sistemas Inteligentes

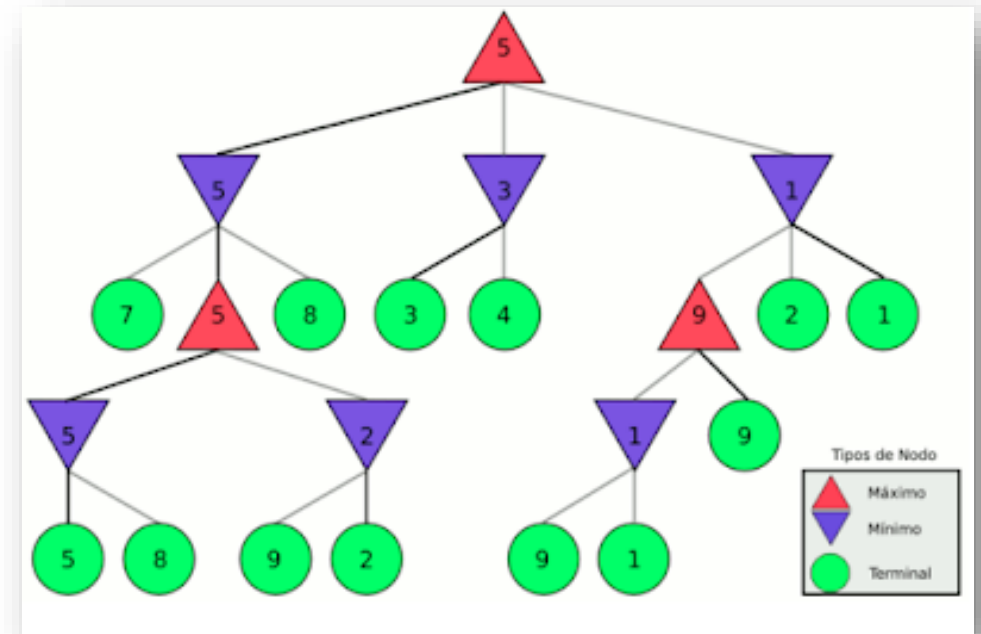
## Búsquedas con oponentes

- ✓ Minimax
- ✓ Poda  $\alpha$ - $\beta$



Dra. Yuridiana Alemán

# MINIMAX



# Buscar soluciones enfrentando adversarios

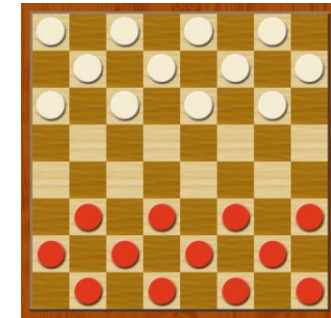
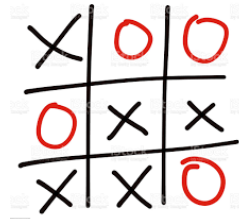
- ✓ En un ambiente de un videojuego, es muy común que un agente “conviva” con un conjunto de agentes
- ✓ ¿Qué hacer si el agente tiene que resolver un problema a partir de una búsqueda, pero la secuencia de acciones no depende exclusivamente del agente en sí?





# Juegos con Dos Agentes

- ✓ Considere un videojuego en el cual dos agentes A y B interactúan en un tablero dividido en celdas



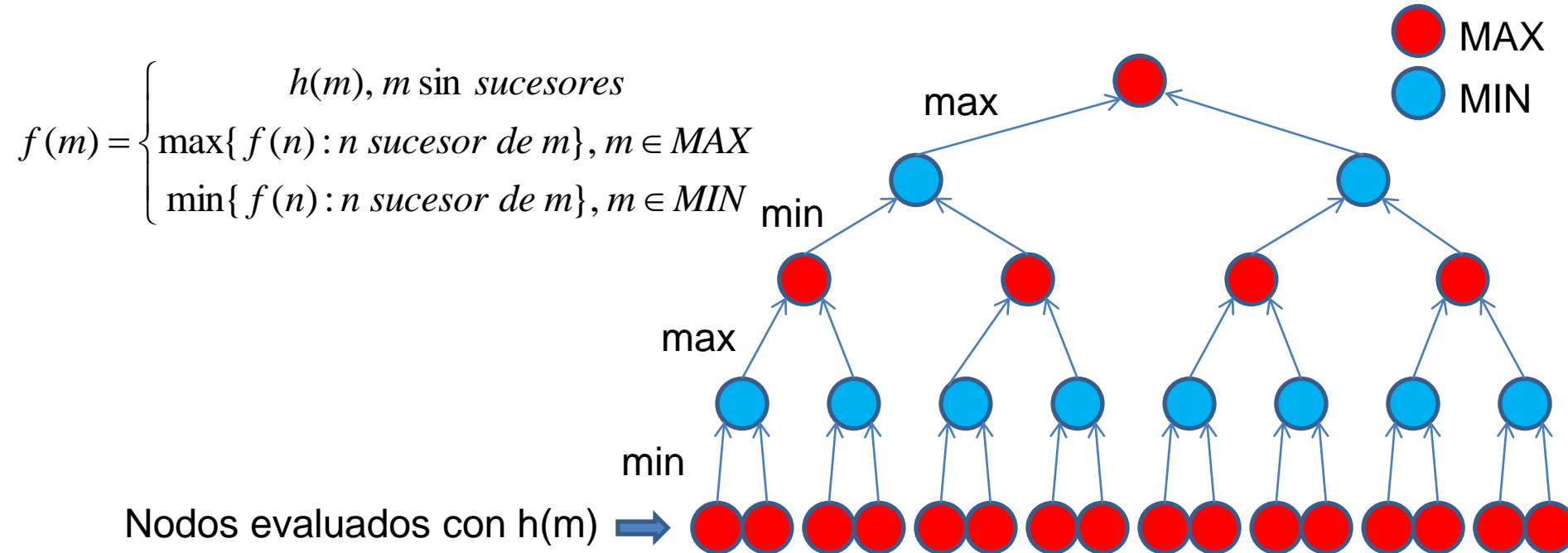
# Juegos con Dos Agentes

- ▶ Para resolver este problema, se puede actuar de la siguiente forma:
  - ▶ Para el agente A:
    - ▶ Determinar todos los posibles movimientos a partir de un estado dado y ver cual camino es el más ventajoso a través de una heurística
    - ▶ Dadas las limitantes de tiempo y cómputo, en ocasiones no es posible determinar con precisión el “mejor” movimiento

Análisis por horizonte limitado

- ▶ Se puede utilizar una estrategia de sensor / planear / actuar

# Procedimiento Minimax



# El procedimiento “minimax”

- ✓ Consideremos que en un juego tenemos a dos agentes, MAX y MIN
  - ✓ Asumamos que MAX tira primero seguido de un movimiento de MIN
  - ✓ Nodos en un nivel de profundidad impar corresponden a estados en donde MIN debe de tirar en seguida (nodos MIN)
  - ✓ Nodos a un nivel par corresponden a estados en donde MAX debe de ejecutar un movimiento (nodos MAX)
  - ✓ El nodo raíz es de profundidad cero
  - ✓ Cada jugador realizará una búsqueda de horizonte limitado para realizar su movimiento

# Etiquetado

- ▶ Como en cualquier búsqueda heurística, es necesario establecer una función “f” de evaluación para los nodos
- ▶ Consideremos como ejemplo de trabajo al juego del “gato” (tic-tac-toe)
  - ▶ Asumamos que el horizonte se establece a partir de una profundidad “2”
  - ▶ Como función de evaluación:
    - ▶  $f(\text{tablero}) = (\# \text{ de columnas, renglones o diagonales vacías disponibles para MAX}) - (\# \text{ de columnas, renglones o diagonales vacías disponibles para MIN})$
    - ▶  $f(\text{tablero}) = \infty$ , si tablero es una posición ganadora para MAX
    - ▶  $f(\text{tablero}) = -\infty$ , si tablero es una posición ganadora para MIN



# Etiquetado

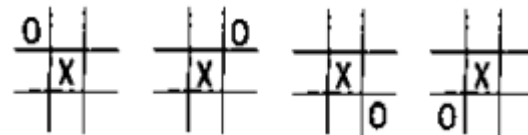
✓ Asumamos lo siguiente:

✓ 

0
X

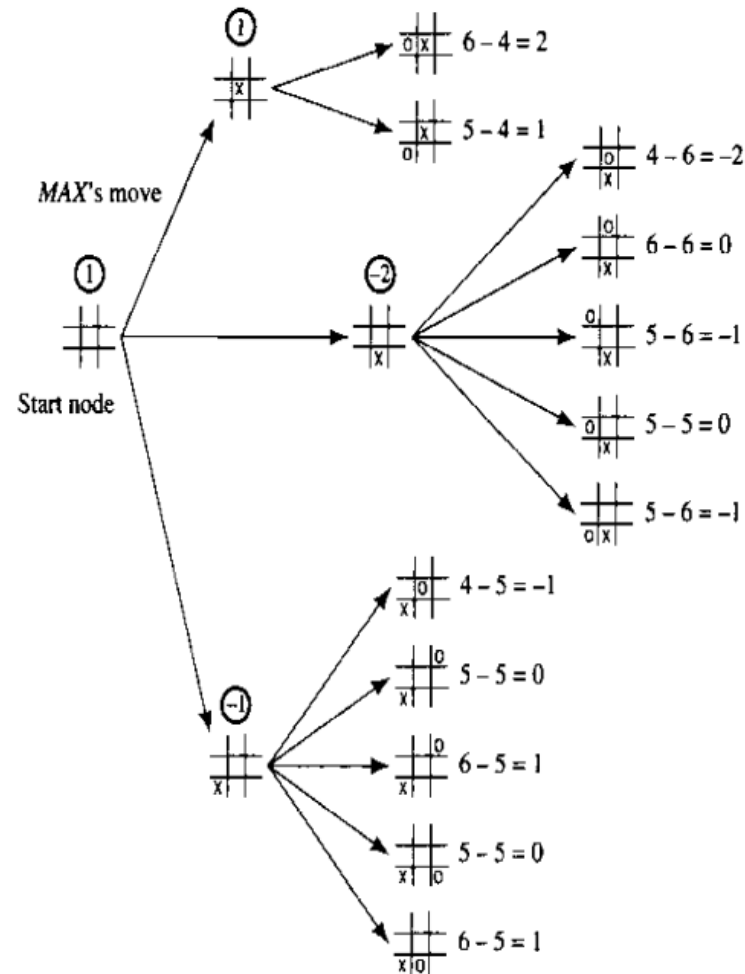
,  $f(\text{tablero}) = 6-4=2$

✓ Para simplificar el problema, se utilizará la simetría de los tableros (que den posiciones idénticas)



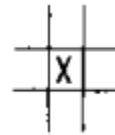
✓ Consideremos que MAX marca con una “x” al tablero y MIN con un “O” al tablero

# Procedimiento Minimax

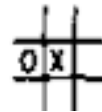


# Procedimiento Minimax

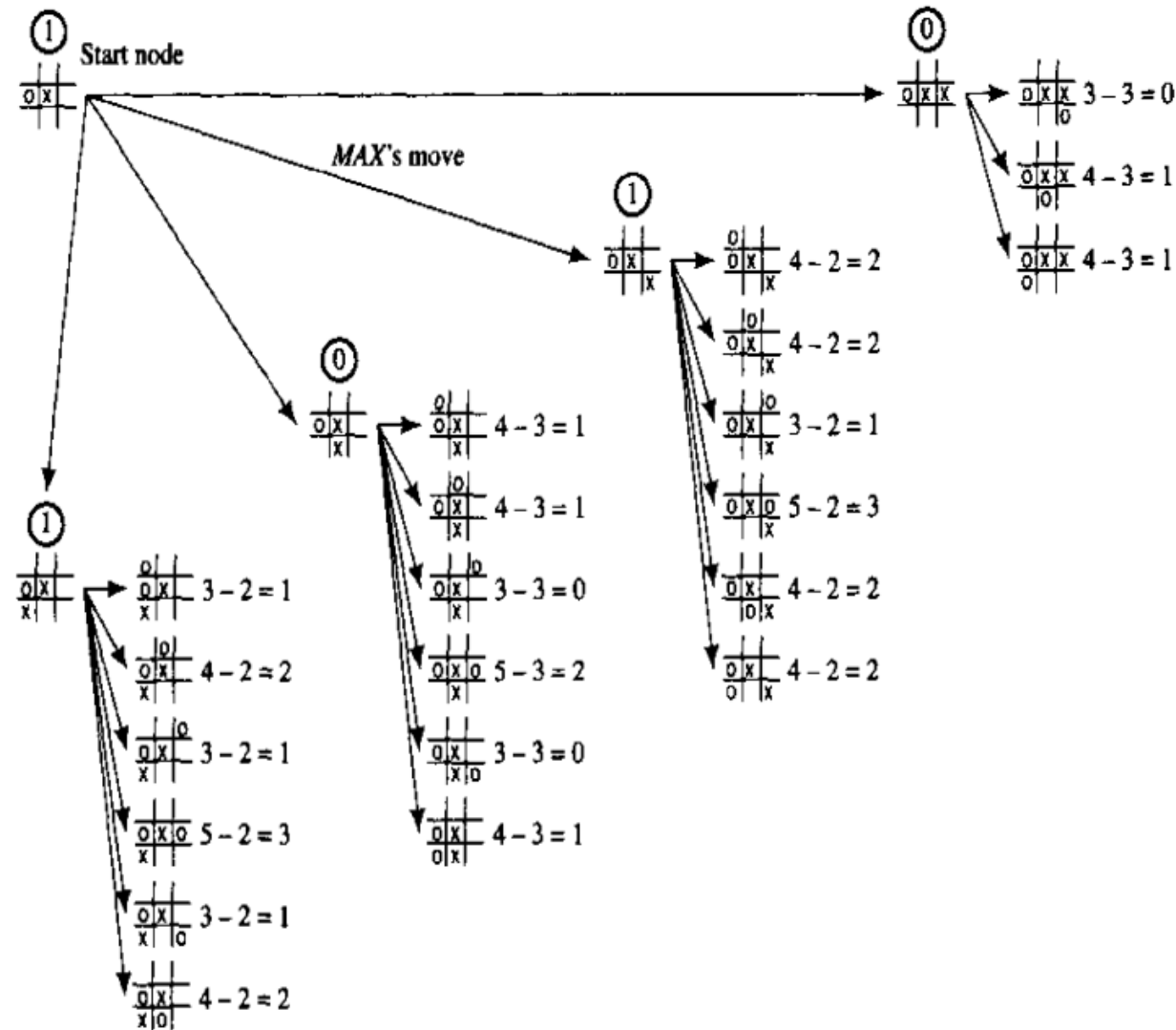
- ✓ A partir del árbol anterior, el estado ganador para MAX es:



- ✓ Considere que MIN responde a partir del siguiente movimiento



# Procedimiento Minimax



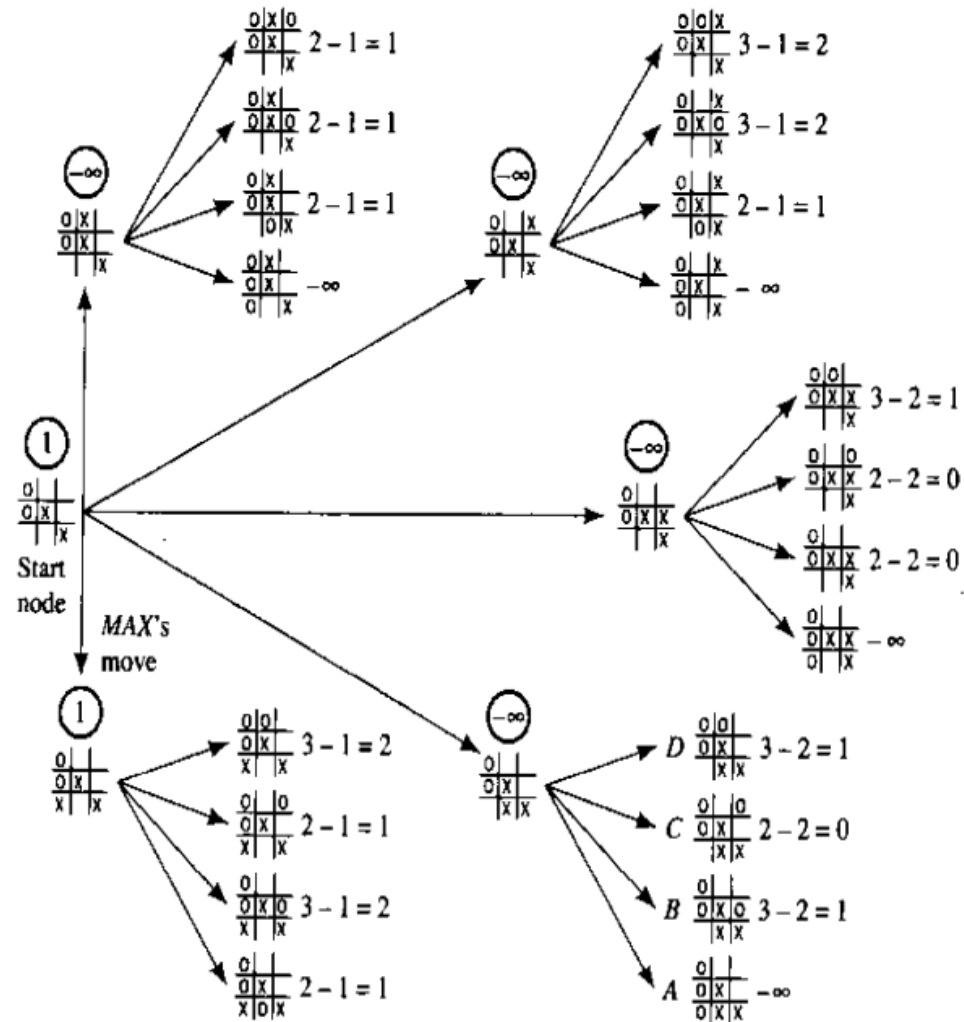


# Procedimiento Minimax

- ✓ A partir de este nuevo árbol de expansión
  - ✓ MAX tiene dos posibles movimientos (se elige uno aleatoriamente)
  - ✓ Considere que MAX elige la tirada marcada en el árbol
- ✓ Considere que MIN responde con el siguiente movimiento

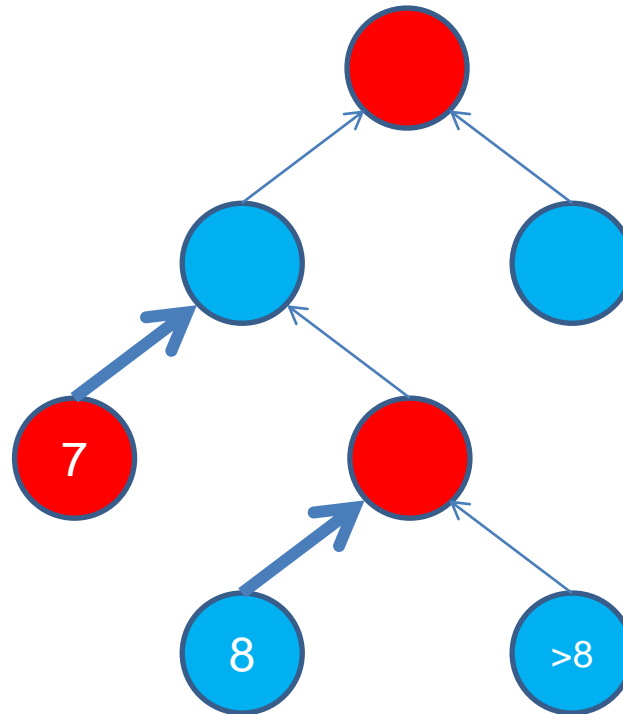
0		
0	X	
		X

# Procedimiento Minimax



# Procedimiento Minimax

- ✓ Minimax requiere realizar una búsqueda exhaustiva del árbol dentro del horizonte limitado
- ✓ Minimax no considera podas en el árbol



# **ALGORITMO PODA ALFA - BETA**



# Poda $\alpha - \beta$

- ✓ La estrategia “poda  $\alpha - \beta$ ” permite realizar podas a un árbol a partir de información que acarrea de los nodos inferiores
  - ✓ Mantiene en cada llamado recursivo dos valores ( $\alpha$ ,  $\beta$ ), donde  $\alpha$  es la cota inferior de los valores que se irán buscando en la parte superior del árbol y  $\beta$  la cota superior
  - ✓ Si  $\alpha$  llega a ser igual o superior a  $\beta$ , no se realiza la búsqueda en las demás ramas del nodo que se expande
    - ✓ Poda  $\alpha$  si el nodo que se expande es MAX
    - ✓ Poda  $\beta$  si el nodo que se expande es MIN

# Poda $\alpha$ - $\beta$ : Algoritmo

- ✓ J: nodo actual
- ✓  $J_k$ : k-ésimo hijo del nodo "J" ( $k = 1, \dots, b$ )
- ✓  $h(J)$ : valor de la función heurística evaluada en J

## Procedimiento $\alpha\beta(J, \alpha, \beta)$

1. Si J es terminal, devolver  $h(j)$
2.  $k \leftarrow 1$
3. Si J es MAX, hacer
  1.  $\alpha \leftarrow \max\{\alpha, \alpha\beta(J_k, \alpha, \beta)\}$
  2. Si  $\alpha \geq \beta$ , devolver  $\beta$ ; en caso contrario continuar
  3. Si  $k = b$  (#hijos de J), devolver  $\alpha$
  4.  $k \leftarrow k + 1$ , regresar a 3.1

# Poda $\alpha - \beta$ : Algoritmo

Procedimiento  $\alpha\beta(J, \alpha, \beta)$  (continuación...)

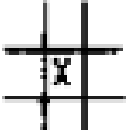
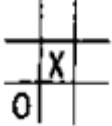
4. Si J es MIN, hacer
  1.  $\beta \leftarrow \min\{\beta, \alpha\beta(J_k, \alpha, \beta)\}$
  2. Si  $\alpha \geq \beta$ , devolver  $\alpha$ ; en caso contrario continuar
  3. Si  $k = b$  (#hijos de J), devolver  $\beta$
  4.  $k \leftarrow k + 1$ , regresar a 4.1

✓ NOTA: para el primer llamado del procedimiento,  $\alpha = -\infty$ ,  $\beta = \infty$

# EJERCICIOS: MINIMAX Y $\alpha - \beta$



# Ejercicios: Minimax y $\alpha - \beta$

De acuerdo con el ejercicio del Tic, Tac, Toe. Supón que el agente tira con el movimiento  y el usuario contesta con el movimiento 

Analiza cuál sería el mejor movimiento del agente, utiliza los algoritmos minimax y alfa-beta. Compara los resultados

# Otro ejemplo

De acuerdo con los valores calculados, determina cuál es el mejor de acuerdo al algoritmo minimax y alfa-beta. Compara los resultados

