

## 4ta. Práctica

***Búsqueda en árbol con contrincante:  
MiniMax con poda Alfa-Beta***

# Decisiones **Perfectas** en Juegos de **DOS** Participantes

## *Definición de Juego*

- ☞ **Estado Inicial**: a) posición en el tablero; b) indicación de a quién le toca jugar.
- ☞ **Conjunto de Operadores**: definen que jugadas están permitidas.
- ☞ **Prueba terminal**: define el término del juego.
- ☞ **Función Utilidad**: asigna un valor numérico al resultado obtenido en un juego (ejem. +1, -1 ó 0)

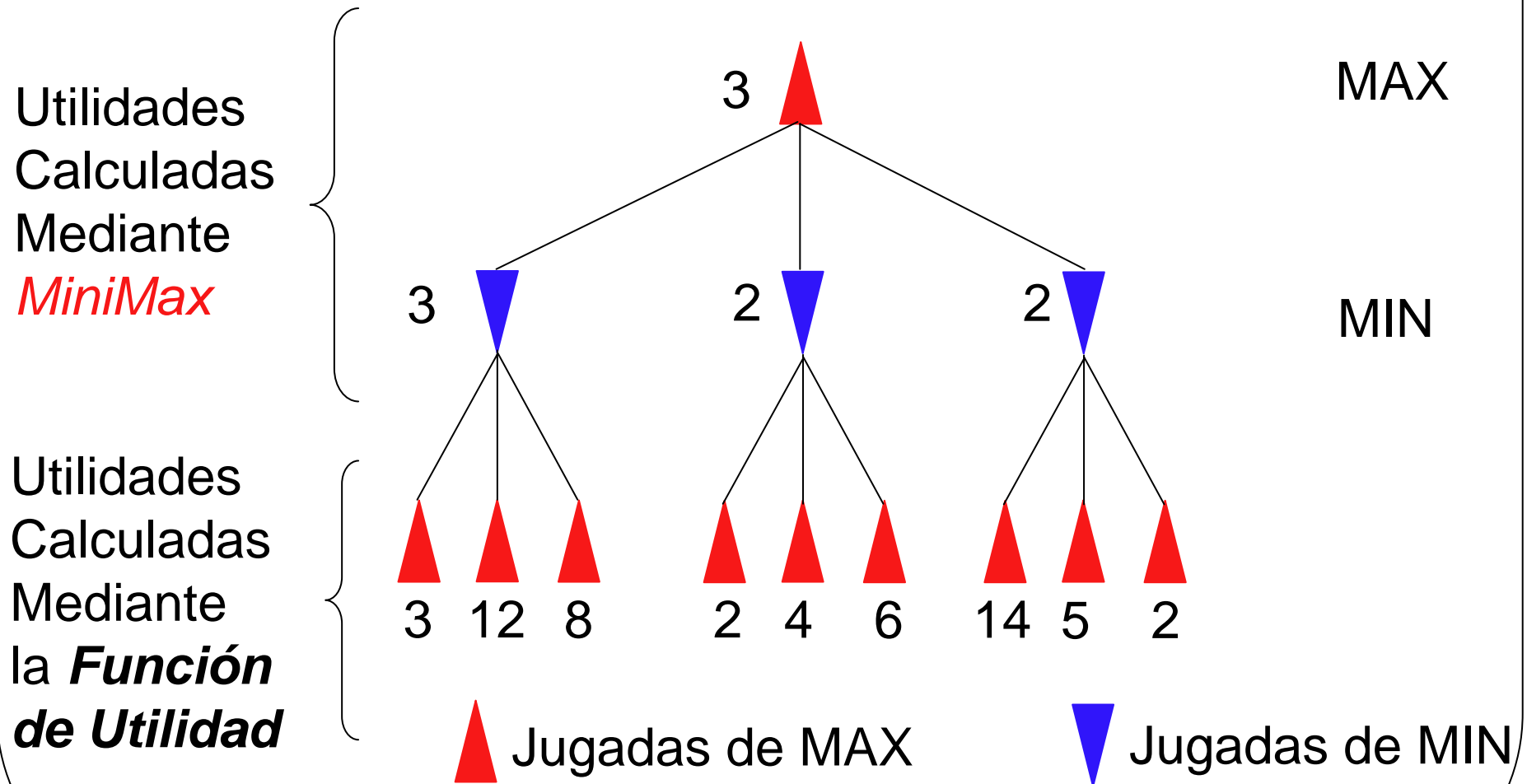
# Decisiones **Perfectas** en Juegos de **DOS** Participantes

## *Algoritmo MiniMax*

- ☞ Dos Jugadores: MAX, MIN.
- ☞ *MiniMax*: determina la estrategia óptima para MAX, independientemente de lo que haga MIN.



# Decisiones Perfectas en Juegos de DOS Participantes

## Algoritmo MiniMax (ejemplo)



## Decisiones **Perfectas** en Juegos de **DOS** Participantes

### *Algoritmo MiniMax (pasos)*

1. *Generación de TODO el árbol de juego (hasta alcanzar los estados terminales).*
2. Aplicar la Función de Utilidad a cada estado terminal.
3. Utilizando la utilidad de los nodos terminales, calcular la utilidad de los nodos del nivel superior en el árbol de búsqueda según corresponda (  MIN ,  MAX).
4. Aplicar (3.), en dirección a la raíz, una capa a la vez.
5. Al llegar a la raíz, MAX elegirá como jugada aquella que le permita obtener el valor más alto, a esto se lo llama decisión MiniMax (en la ilustración anterior {3, 2, 2} MAX elegirá 3)

## Decisiones **Perfectas** en Juegos de **DOS** Participantes

### *Algoritmo MiniMax*

- ☞ Si la profundidad del árbol es  $m$  y  $b$  representa las jugadas permitidas en cada punto, la complejidad en tiempo del algoritmo *MiniMax* es  $O(b^m)$
- ☞ Por tanto, en general, es impráctico utilizar *MiniMax*, si embargo es UTIL como PUNTO DE PARTIDA

→ ***Decisiones Imperfectas !!!***

## Decisiones **Imperfectas** en Juegos de **DOS** Participantes

### *Variantes sobre el Algoritmo MiniMax*

- ☞ En vez de llegar a los estados terminales, suspender la búsqueda antes, y aplicar a esas hojas una **Función de Evaluación** (**heurística**).
- ☞ La **Función de Evaluación** produce una *Estimación* de la utilidad esperada. Por ejemplo en ajedrez se asigna un valor a cada una de las piezas: peones = 1; caballo & alfil = 3; torre = 5; reina = 9. Otras heurísticas ponderan “*una buena estructura de peones*”, ó, “*la seguridad del rey*” .....

## Decisiones **Imperfectas** en Juegos de **DOS** Participantes

### *Definición de una Función de Evaluación*

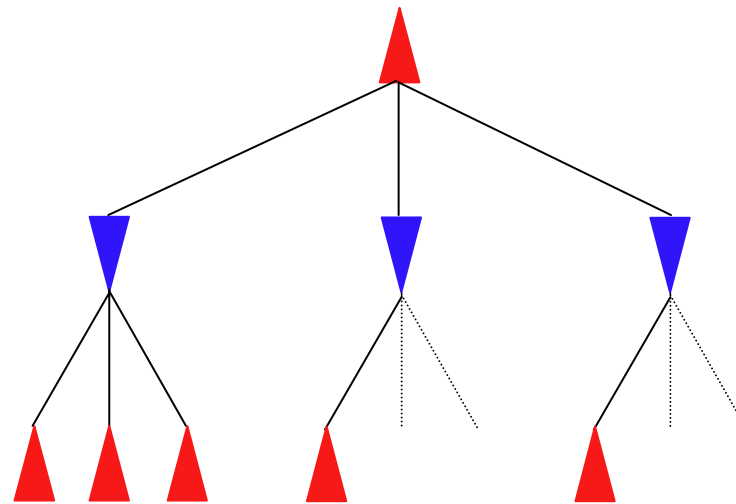
- ☞ La **Función de Evaluación** deberá coincidir con la **Función de Utilidad** de los estados terminales.
- ☞ Complejidad inferior a **MINIMAX** (compromiso entre precisión de la función de evaluación y su respectivo coste en el tiempo).

*Sin embargo, para el caso del Ajedrez, **MiniMax** con una función de evaluación razonable y un ordenador normal podrá buscar 1000 posiciones por segundo (a 150 segundos por jugada se podrán explorar 150000 posiciones, considerando un factor de ramificación de 35, esto representa **4 capas**, Nivel de NOVATO, en promedio **un humano** puede anticipar **8 capas**)*

## Decisiones **Imperfectas** en Juegos de **DOS** Participantes

### *Definición de una Función de Evaluación*

- ➡ Otra modificación sobre **MINIMAX** es descartar aquellos caminos del árbol que no afectarán el resultado final:



➔ Poda **Alfa-Beta**

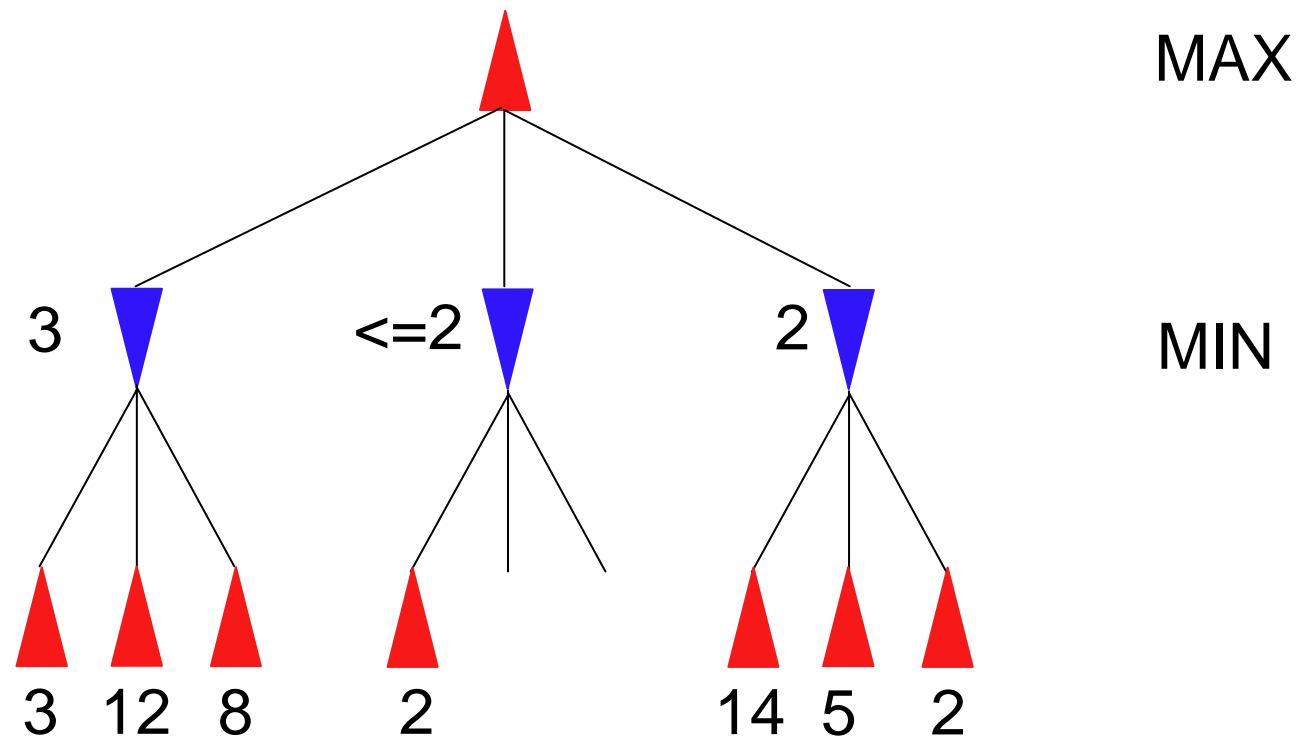
## Decisiones **Imperfectas** en Juegos de **DOS** Participantes

### *Poda Alfa-Beta*

- ☞ Produce la misma jugada que se obtendría con *MiniMax*, pero elimina todas las ramas que posiblemente no influirán en la decisión final.
- ☞ **Alfa**: valor de la mejor opción encontrada hasta entonces en un punto de elección a través de la ruta de **MAX**;
- ☞ **Beta**: el valor de la mejor (valor más bajo) opción encontrada hasta entonces en un punto de opción a lo largo de la ruta **MIN**.
- ☞ Conforme se efectúa la búsqueda Alfa-Beta se van actualizando los valores de alfa y beta y se poda

# Decisiones **Imperfectas** en Juegos de **DOS** Participantes

## *Poda Alfa-Beta*



▲ Jugadas de MAX

▼ Jugadas de MIN

## Cuarta Práctica (entrega antes del 10 de enero)

- 1) Implementar una función heurística que evalúe el estado del tablero de Othello (\*)
- 2) Implementar la función de búsqueda en árbol con contrincante **MiniMax** con poda **Alfa-Beta** (\*)

(\*) Ver enunciado de la práctica en la página Web

