

Sistemas Inteligentes I

Tema 3. Juegos

José A. Montenegro Montes

monte@lcc.uma.es

Resumen

- Juegos
- Algoritmo Minimax
- Poda Alfa-Beta
- Expectiminimax
- Funciones de Evaluación

Juegos

- Entornos multiagente, donde cada agente debe considerar las acciones de los otros agentes.
- **Juegos:** Entornos competitivos donde los objetivos del agente están en conflicto, dan lugar a problemas de búsqueda entre adversarios.
 - Ajedrez, Otelo, Backgammon, Go
 - Ajedrez: Árbol de búsqueda tiene 10^{154} nodos.
- Capacidad de tomar decisión cuando no es factible calcular la decisión óptima.
 - **Poda:** Nos permiten ignorar partes del árbol búsqueda.
 - **Funciones evaluación:** Heurísticas que permiten aproximar la utilidad sin hacer búsqueda completa.

Componentes Juegos

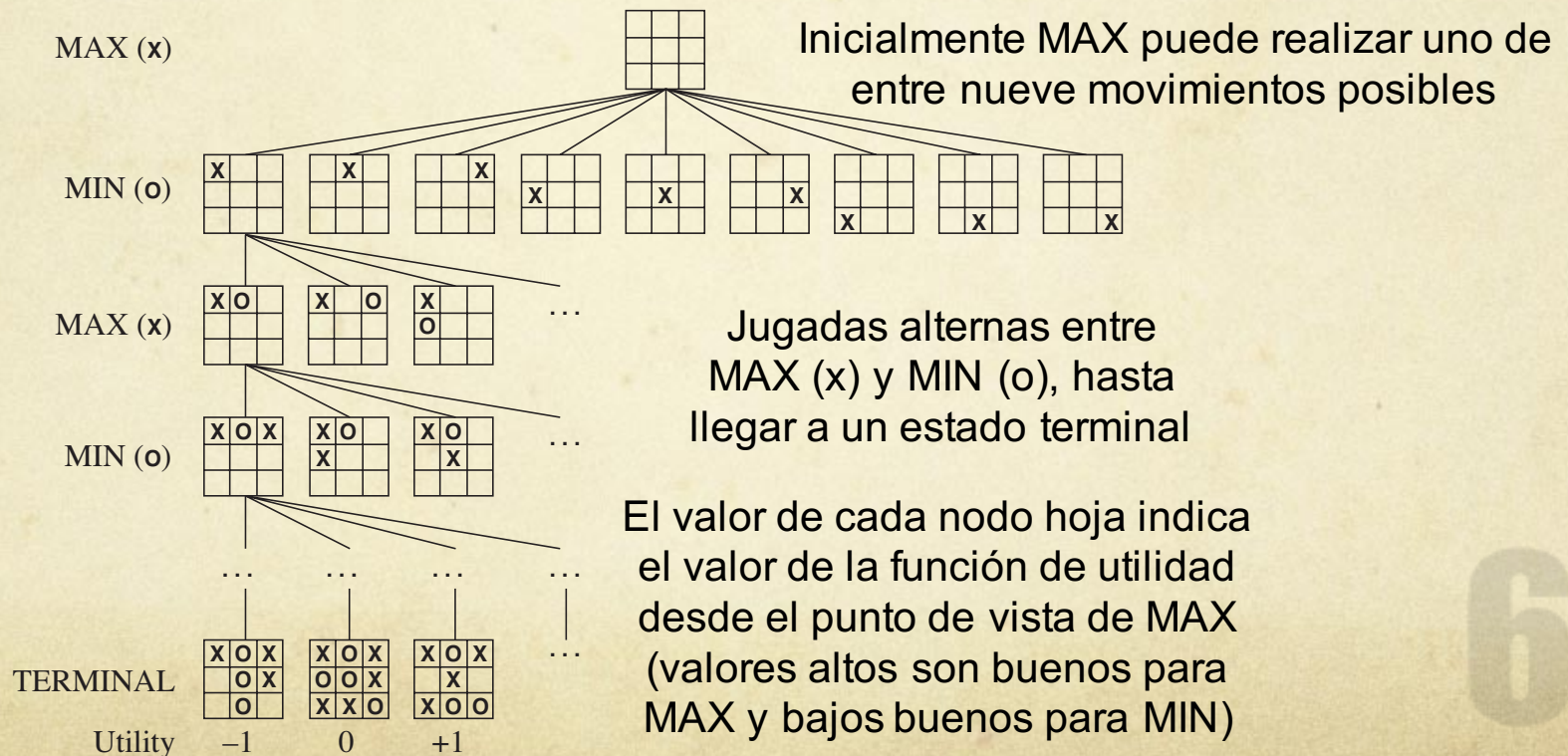
- Juegos, clase de problemas de búsqueda:
 - **Estado Inicial:** Posición tablero y jugador que mueve
 - **Función sucesor:** Lista pares (movimiento, estado) , estado resultante.
 - **Test terminal:** Cuando finaliza el juego. Estados terminales.
 - **Función Utilidad:** Valor numérico a los estados terminales.
 - Por ejemplo (suma nula): Triunfo +1, Pérdida -1, Empate 0
- **Árbol del juego:** Definido mediante estado inicial y los movimientos legales.

Algoritmo Minimax



Algoritmo Minimax

- Juegos 2 jugadores (MAX y MIN)
- Primero mueve MAX y luego MIN por turnos hasta que termina
- Árbol del juego del Tres en Raya

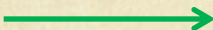


Algoritmo Minimax

- Tiene por objetivo decidir un movimiento para MAX.
- **HIPÓTESIS**
 - Jugador MAX trata de maximizar su beneficio (función de utilidad).
 - Jugador MIN trata de minimizar su pérdida.
 - Suponemos Jugadores juegan de forma óptima.
- Aplicación algoritmo:
 - 1) Generar árbol entero hasta nodos terminales
 - 2) Aplicar función *utilidad* a nodos terminales
 - 3) Propagar hacia arriba para generar nuevos valores de *utilidad* para todos los nodos
 - minimizando para MIN
 - Maximizando para MAX
 - 4) Elección jugada con máximo valor de *utilidad*

Algoritmo Minimax

function MINIMAX-DECISION(*state*) *returns an action*

 **return** $\arg \max_{a \in \text{ACTIONS}(s)} \text{MIN-VALUE}(\text{RESULT}(state, a))$

function MAX-VALUE(*state*) *returns a utility value*

if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)

$v \leftarrow -\infty$

for each *a* **in** ACTIONS(*state*) **do**

$v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$ 

return *v*

function MIN-VALUE(*state*) *returns a utility value*

if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)

$v \leftarrow \infty$

for each *a* **in** ACTIONS(*state*) **do**

$v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a)))$ 

return *v*

Algoritmo Minimax

MINIMAX(s)

UTILIDAD(s)

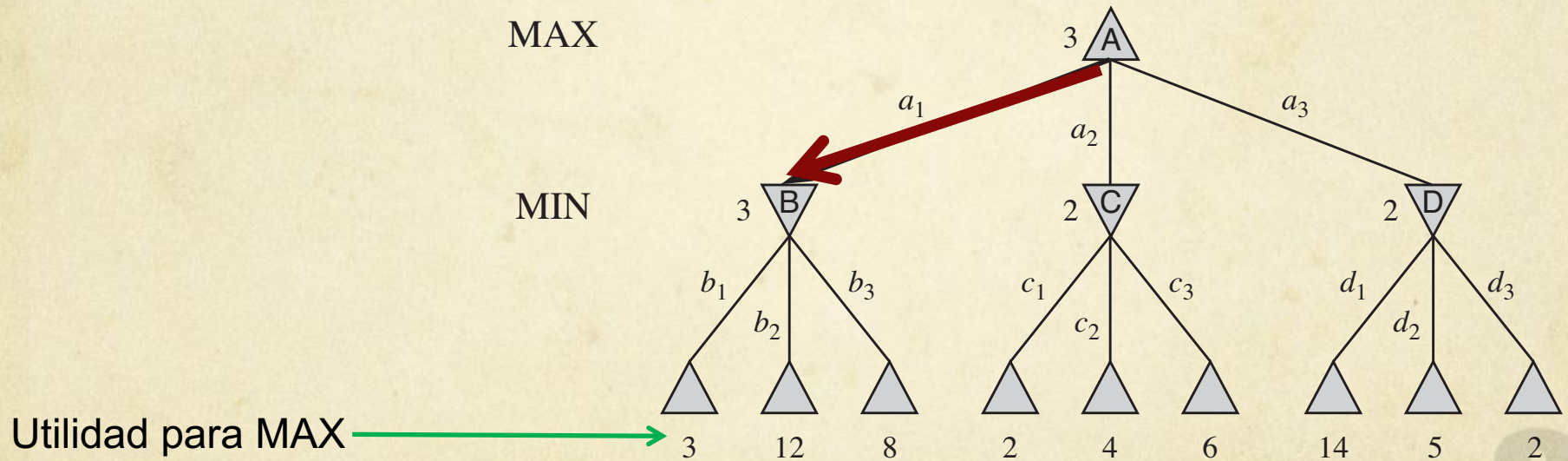
max_a MINIMAX (RESULT(s,a))

min_a MINIMAX (RESULT(s,a))

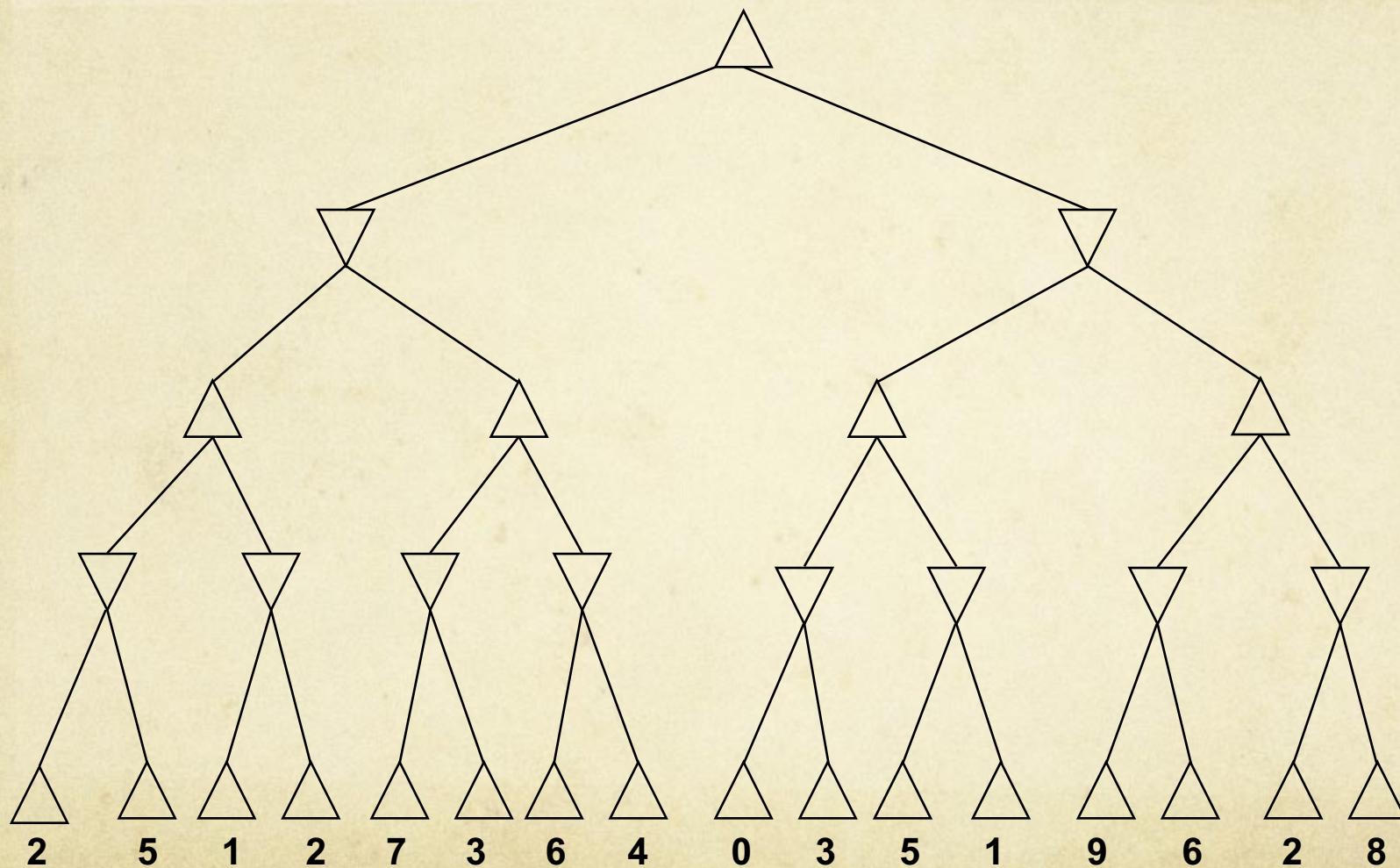
if TERMINAL-TEST(s)

if PLAYES(s) = MAX

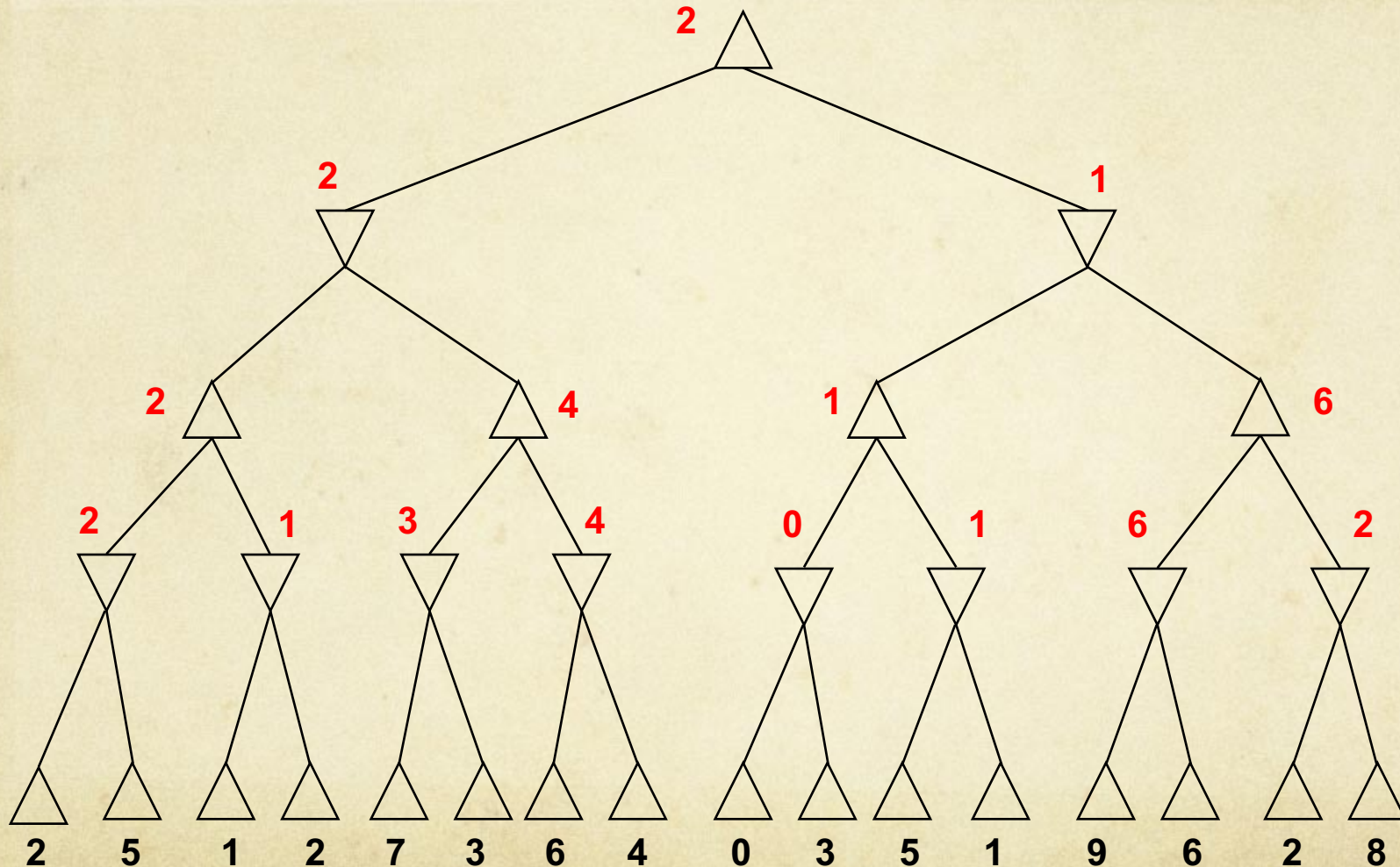
if PLAYES(s) = MIN



Ejemplo 1



Ejemplo 1 minimax



Poda Alfa-Beta



Poda Alfa-Beta

- No afecta al resultado final
- Su **efectividad** depende del **orden** en que se consideren los movimientos
- $\alpha \rightarrow$ valor de la mejor elección para MAX (valor más alto) que hemos encontrado en el camino hasta ahora
- $\beta \rightarrow$ valor de la mejor elección para MIN (valor más bajo) que hemos encontrado en el camino hasta ahora

Poda Alfa-Beta

Alfa

Beta

function ALPHA-BETA-SEARCH($state$) **returns** an action
 $v \leftarrow \text{MAX-VALUE}(state, \overset{\text{Alfa}}{\downarrow} -\infty, \overset{\text{Beta}}{\uparrow} +\infty)$
return the *action* in ACTIONS($state$) with value v

Poda Alfa-Beta

function MAX-VALUE($state, \alpha, \beta$) *returns a utility value*
if TERMINAL-TEST($state$) **then return** UTILITY($state$)

$v \leftarrow -\infty$

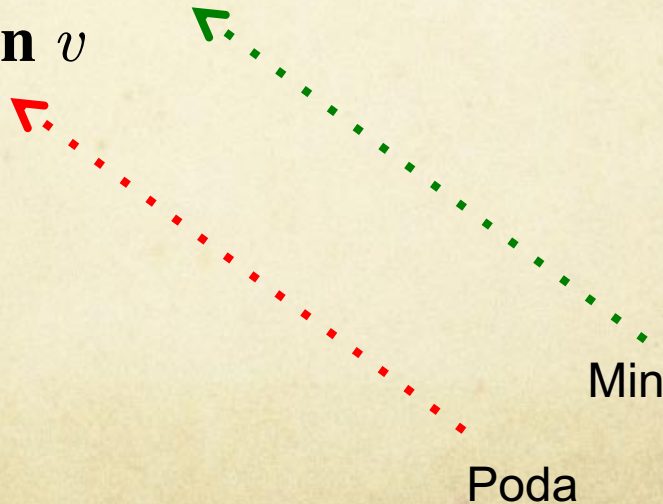
for each a **in** ACTIONS($state$) **do**

$v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$

if $v \geq \beta$ **then return** v

$\alpha \leftarrow \text{MAX}(\alpha, v)$

return v



Poda Alfa-Beta

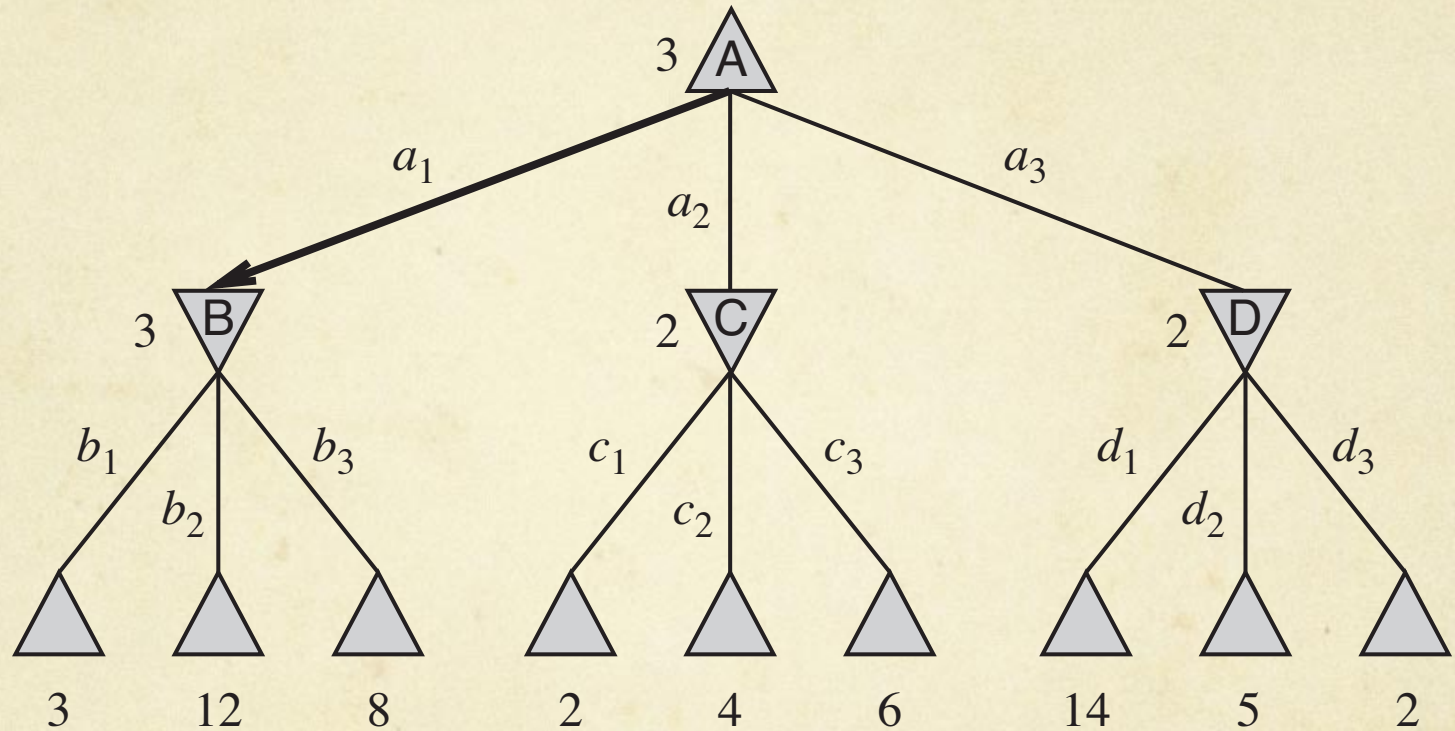
function MIN-VALUE($state, \alpha, \beta$) **returns** *a utility value*
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow +\infty$
for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
 if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v



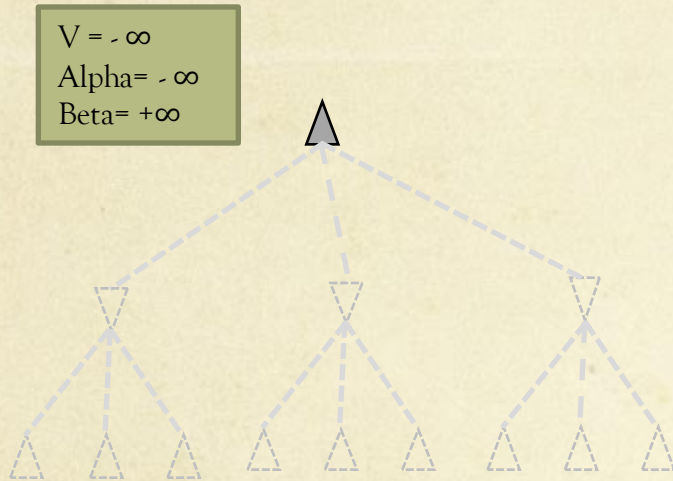
Ejemplo Alfa-Beta

MAX

MIN



Poda Alfa-Beta



function ALPHA-BETA-SEARCH(*state*) **returns** an action
 $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$
return the action in $\text{ACTIONS}(\text{state})$ with value v

1

function MAX-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$

$v \leftarrow -\infty$

for each a **in** $\text{ACTIONS}(\text{state})$ **do**

$v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$

if $v \geq \beta$ **then return** v

$\alpha \leftarrow \text{MAX}(\alpha, v)$

return v

2

function MIN-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$

$v \leftarrow +\infty$

for each a **in** $\text{ACTIONS}(\text{state})$ **do**

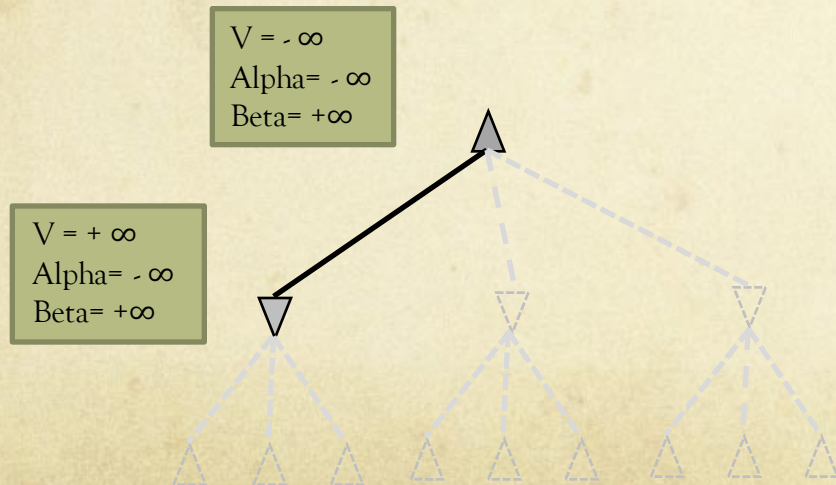
$v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$

if $v \leq \alpha$ **then return** v

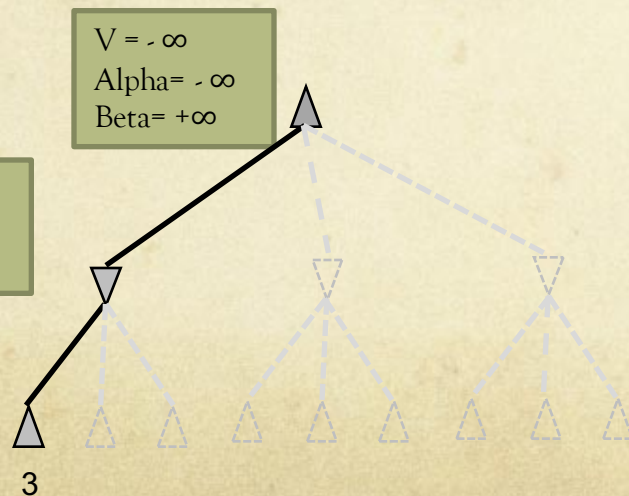
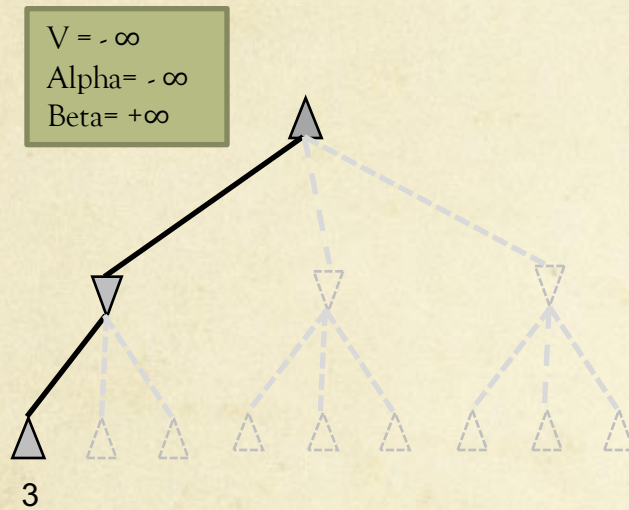
$\beta \leftarrow \text{MIN}(\beta, v)$

return v

3



Poda Alfa-Beta



function MAX-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)

$v \leftarrow -\infty$

for each a **in** ACTIONS($state$) **do**

$v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$

if $v \geq \beta$ **then return** v

$\alpha \leftarrow \text{MAX}(\alpha, v)$

return v

Return 3

function MIN-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)

$v \leftarrow +\infty$

for each a **in** ACTIONS($state$) **do**

$v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$

if $v \leq \alpha$ **then return** v

$\beta \leftarrow \text{MIN}(\beta, v)$

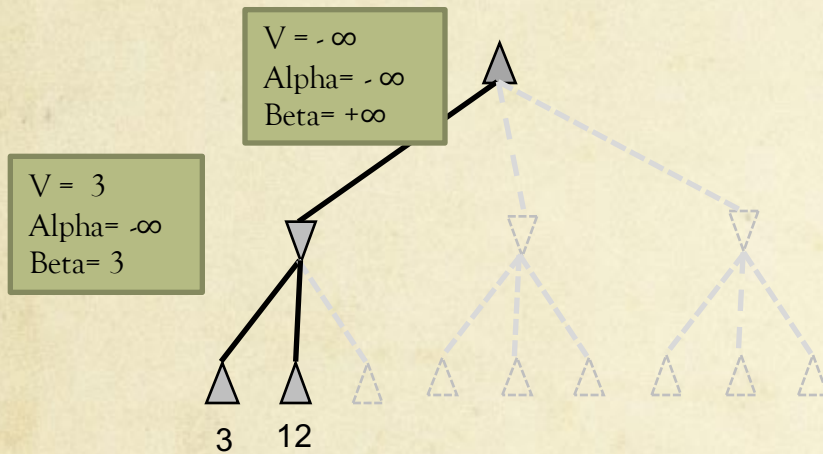
return v

$V = \text{Min}(+\infty, 3) = 3$

$3 \leq -\infty$ FALSE

$\text{Beta} = \text{Min}(+\infty, 3) = 3$

Poda Alfa-Beta



function MAX-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow -\infty$

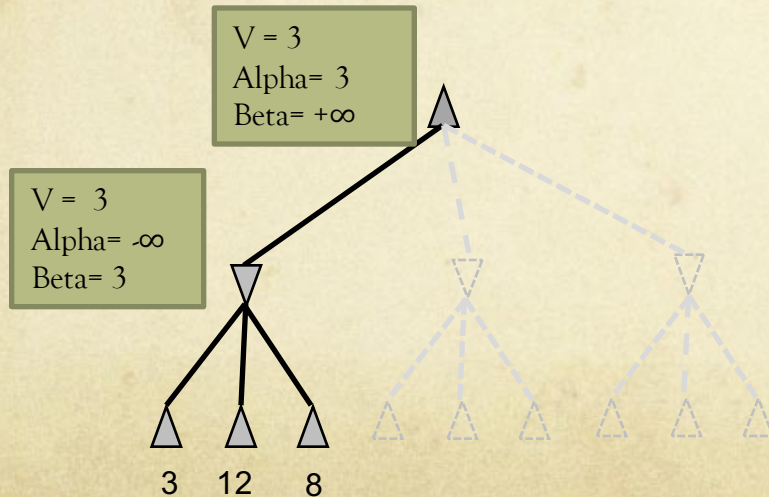
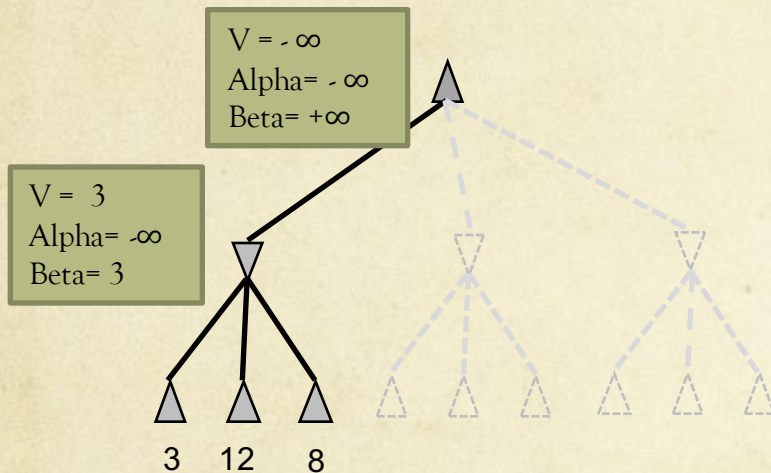
Return 12

for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

function MIN-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow +\infty$
for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

$V = \text{Min}(3, 12) = 3$
 $3 \leq -\infty$ FALSE
 $\text{Beta} = \text{Min}(3, 12) = 3$

Poda Alfa-Beta



function MAX-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow -\infty$

Return 8

for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

$V = \text{Max}(-\infty, 3) = 3$
 $3 \geq +\infty$ FALSE
 $\text{Alpha} = \text{Max}(-\infty, 3) = 3$

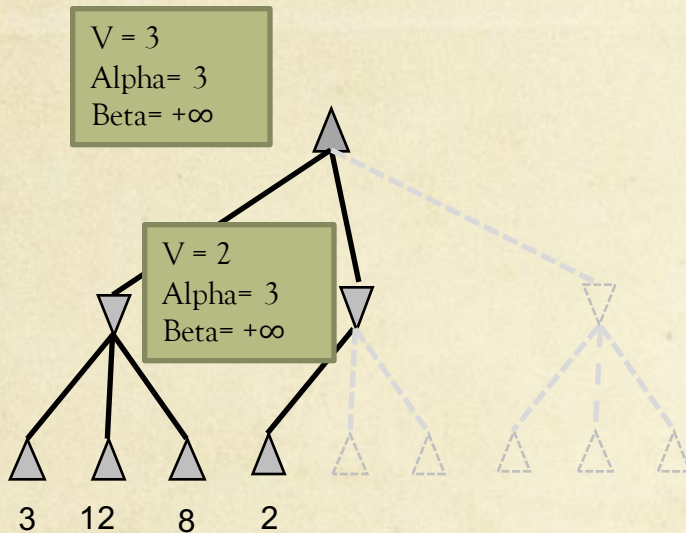
function MIN-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow +\infty$

for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

Return 3

$V = \text{Min}(3, 8) = 3$
 $3 \leq -\infty$ FALSE
 $\text{Beta} = \text{Min}(3, 8) = 3$

Poda Alfa-Beta



function MAX-VALUE($state, \alpha, \beta$) *returns a utility value*
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow -\infty$

Return 2

for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

$V = \text{Max}(3, 2) = 2$
 $3 \geq +\infty$ FALSE
 $\text{Alpha} = \text{Max}(3, 2) = 3$

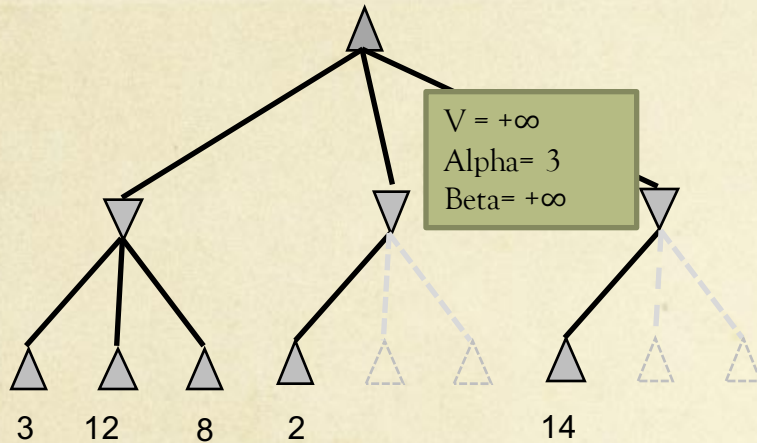
function MIN-VALUE($state, \alpha, \beta$) *returns a utility value*
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow +\infty$
for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

$V = \text{Min}(+\infty, 2) = 2$
 $2 \leq 3$ TRUE

Return 2

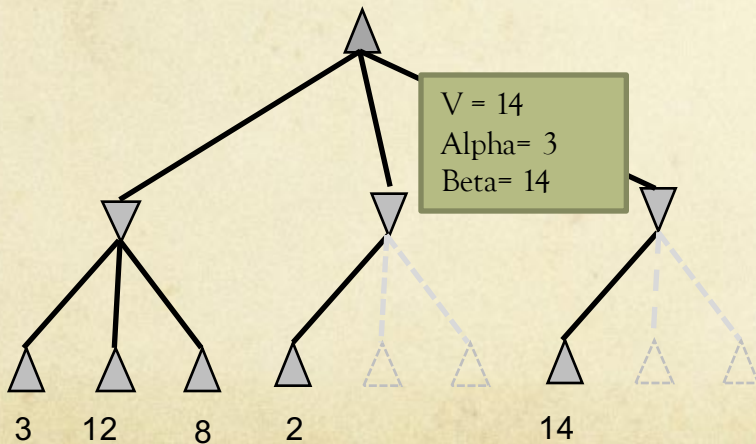
PODA!!!!!!

Poda Alfa-Beta



function MAX-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow -\infty$
for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

Return 14



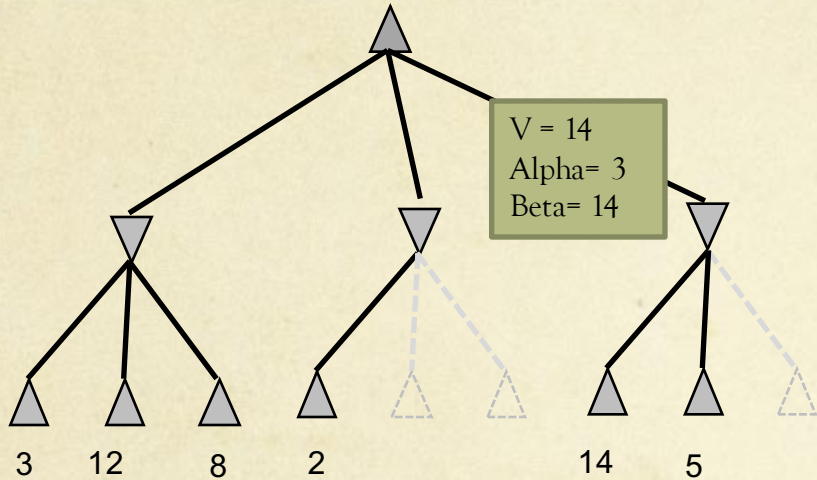
function MIN-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow +\infty$
for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

$V = \text{Min}(+\infty, 14) = 14$

$14 \leq 3$ FALSE

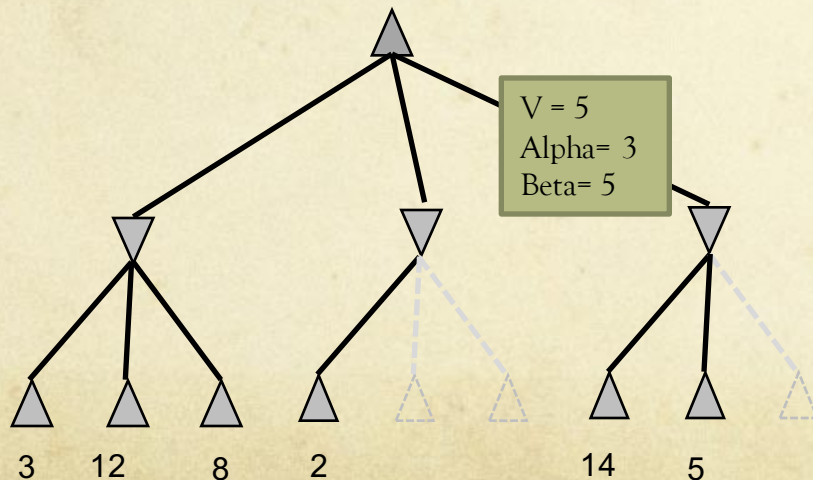
$\text{Beta} = \text{Min}(+\infty, 14) = 14$

Poda Alfa-Beta



function MAX-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow -\infty$
for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

Return 5



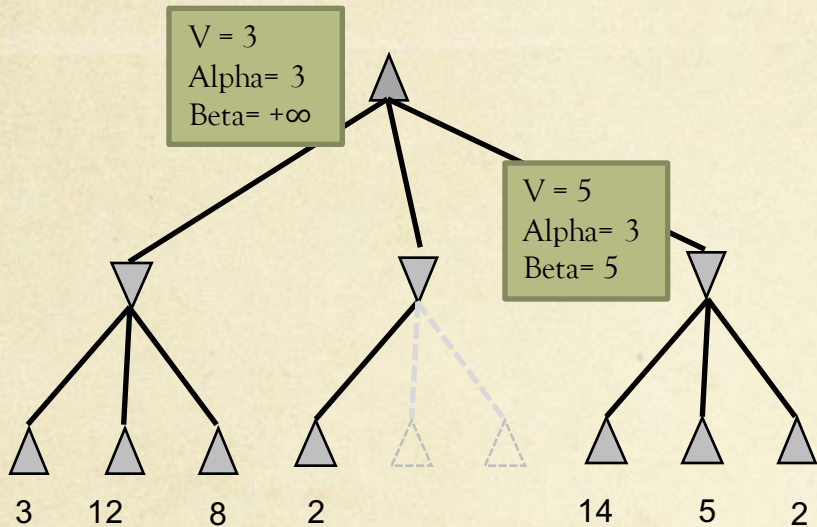
function MIN-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow +\infty$
for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

$V = \text{Min}(14, 5) = 5$

$5 \leq 3$ FALSE

$\text{Beta} = \text{Min}(14, 5) = 5$

Poda Alfa-Beta



function MAX-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)

$v \leftarrow -\infty$

for each a **in** ACTIONS($state$) **do**

$v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$

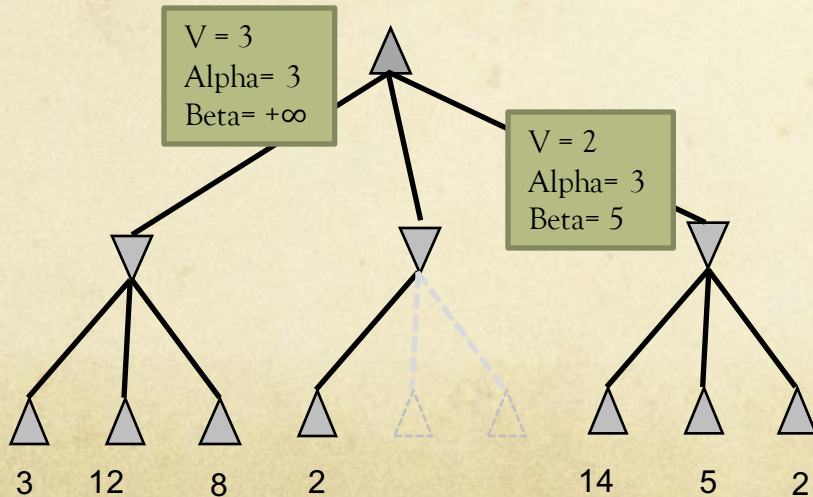
if $v \geq \beta$ **then return** v

$\alpha \leftarrow \text{MAX}(\alpha, v)$

return v

Return 2

$V = \text{Max}(3, 2) = 3$
 $3 \geq +\infty$ FALSE
 $\text{Alfa} = \text{Max}(3, 3) = 3$



function MIN-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)

$v \leftarrow +\infty$

for each a **in** ACTIONS($state$) **do**

$v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$

if $v \leq \alpha$ **then return** v

$\beta \leftarrow \text{MIN}(\beta, v)$

return v

$V = \text{Min}(5, 2) = 2$
 $2 \leq 3$ TRUE

Return 2

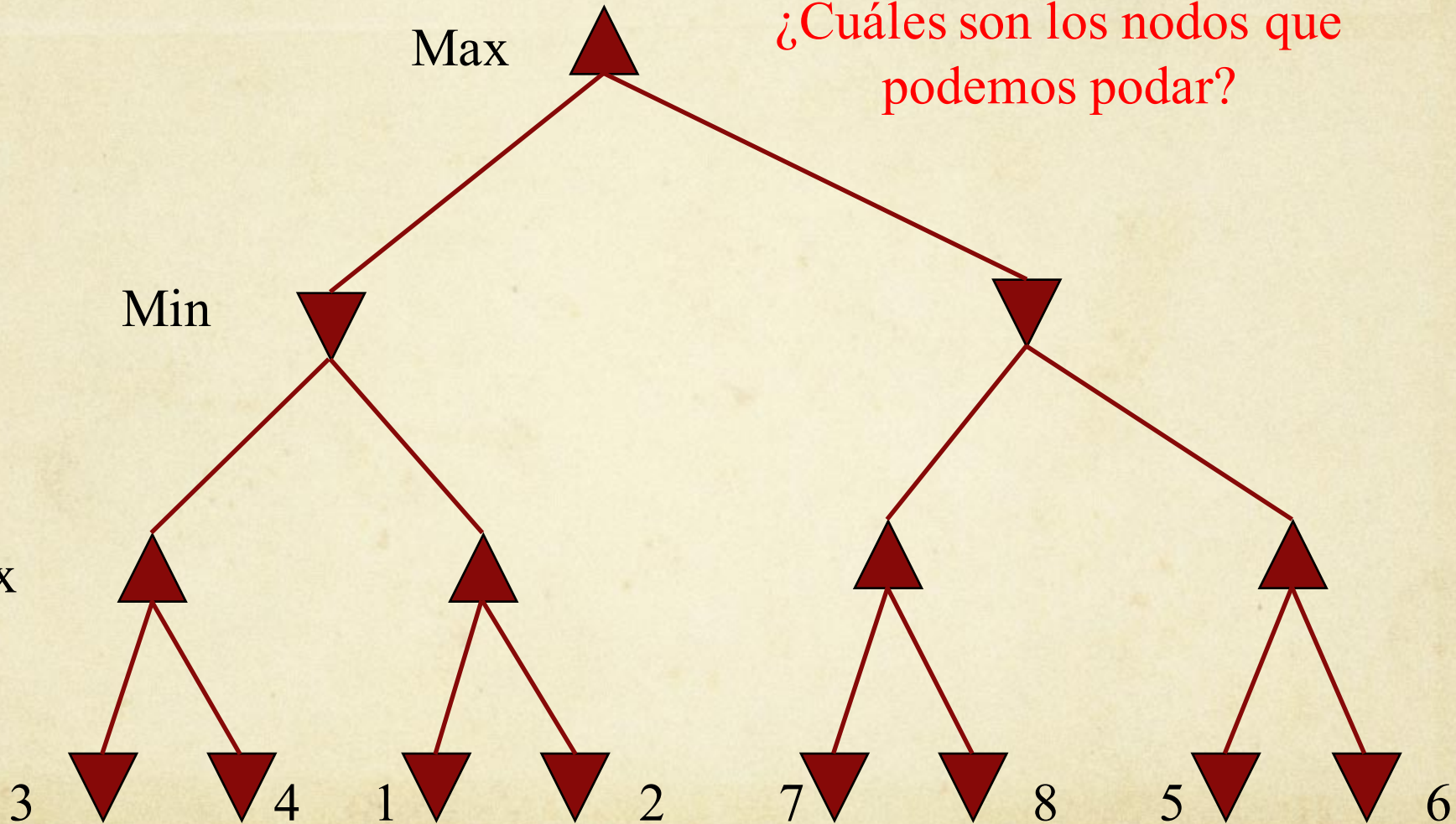
Ejemplo

Max

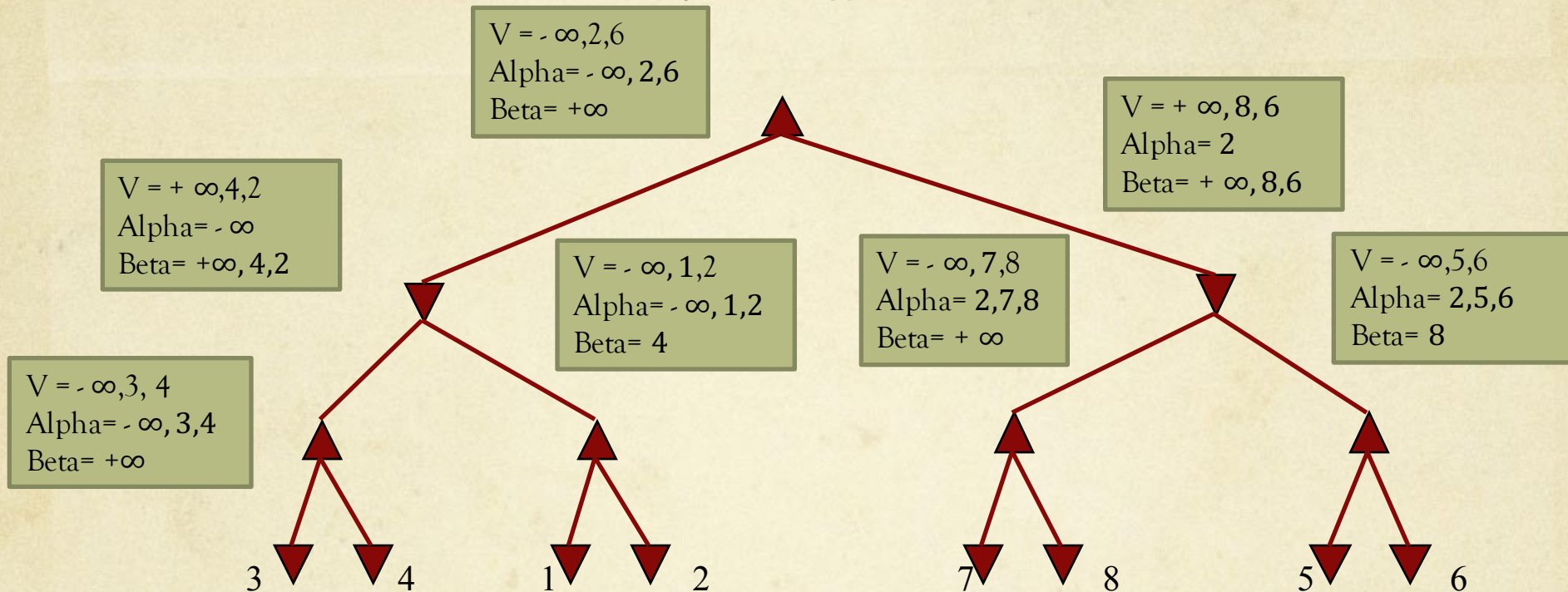
¿Cuáles son los nodos que podemos podar?

Min

Max



Ejemplo



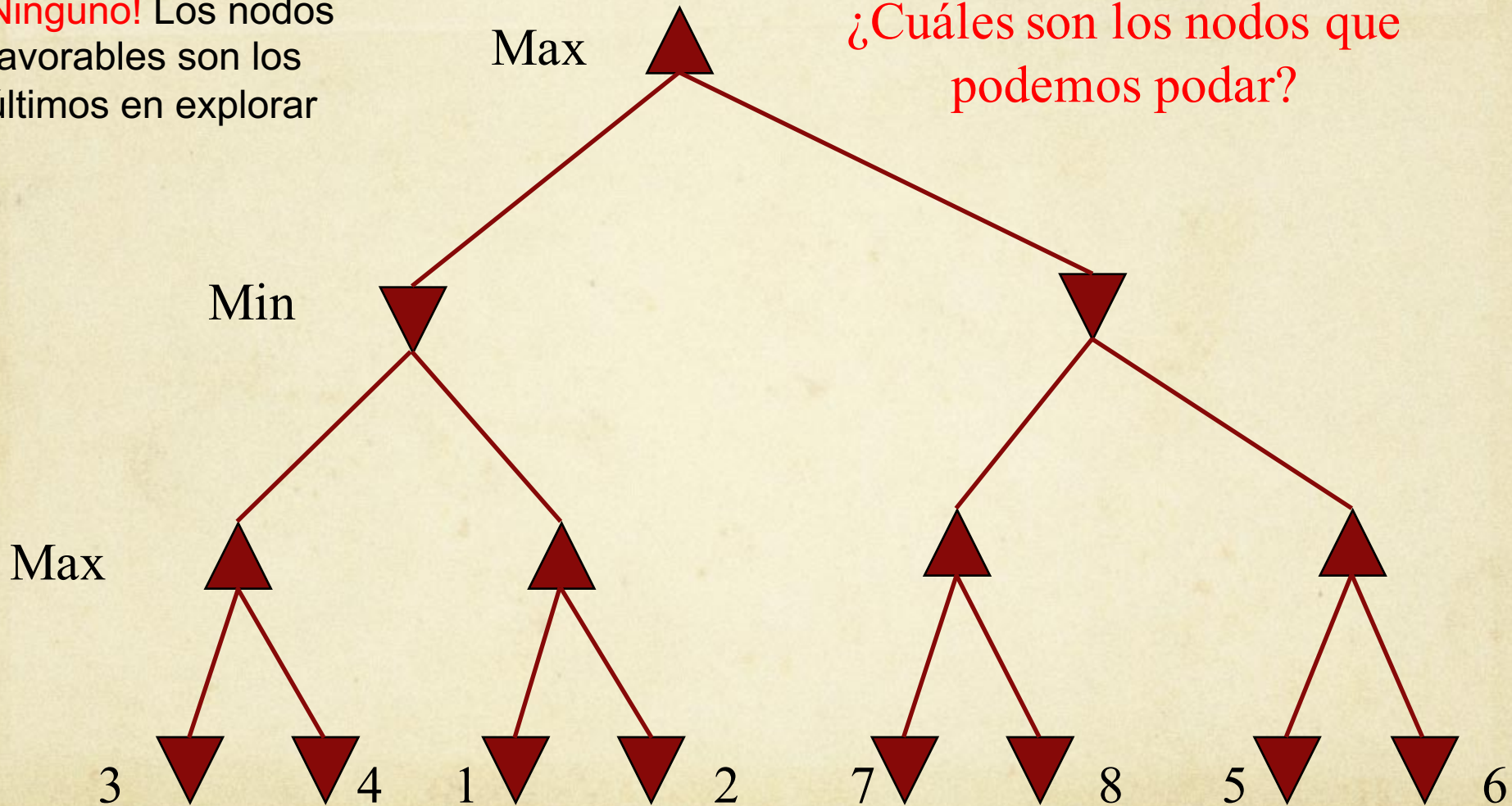
function MAX-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow -\infty$
for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
 if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

function MIN-VALUE($state, \alpha, \beta$) **returns** a utility value
if TERMINAL-TEST($state$) **then return** UTILITY($state$)
 $v \leftarrow +\infty$
for each a **in** ACTIONS($state$) **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
 if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

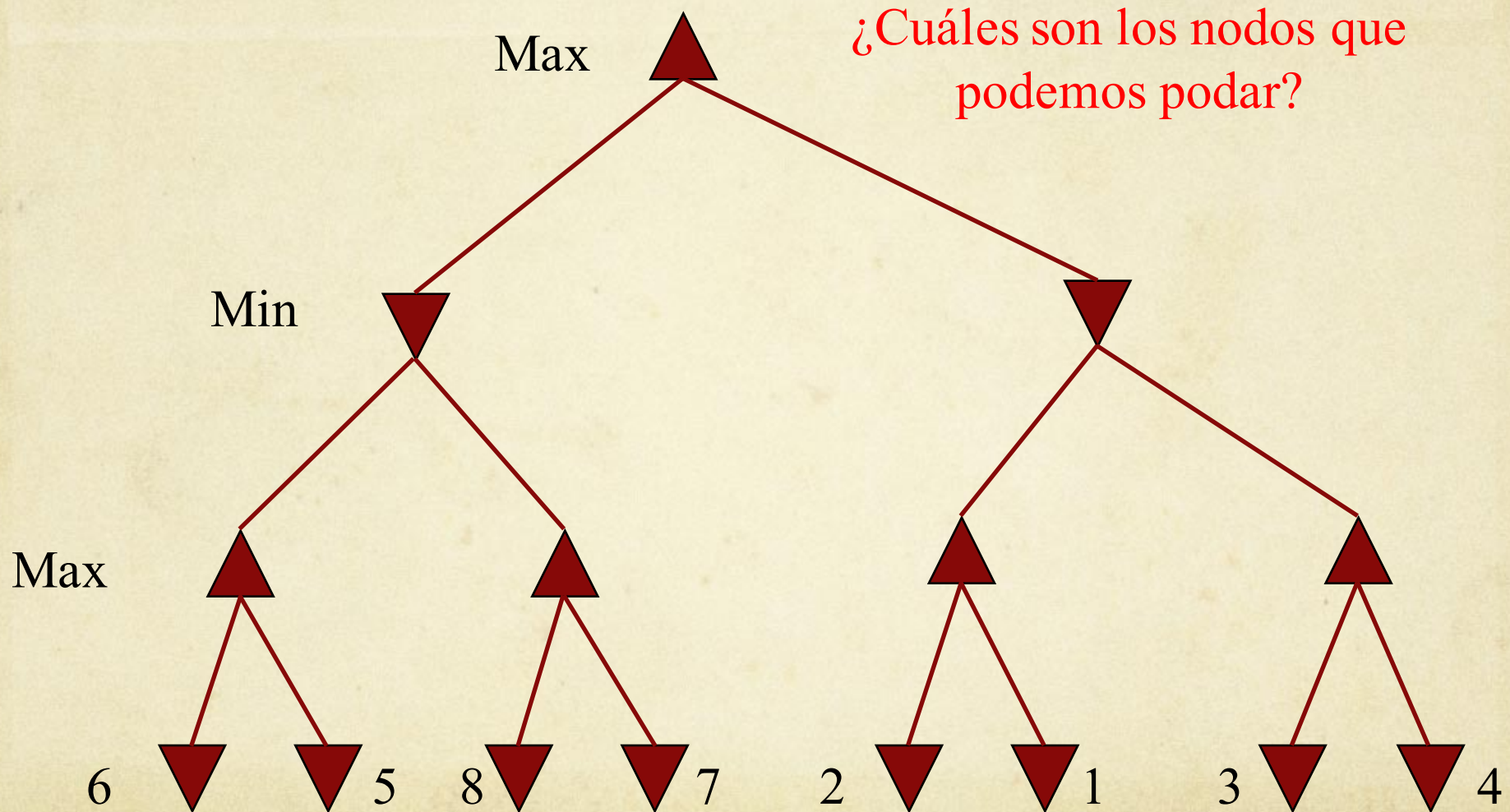
Ejemplo

Ninguno! Los nodos favorables son los últimos en explorar

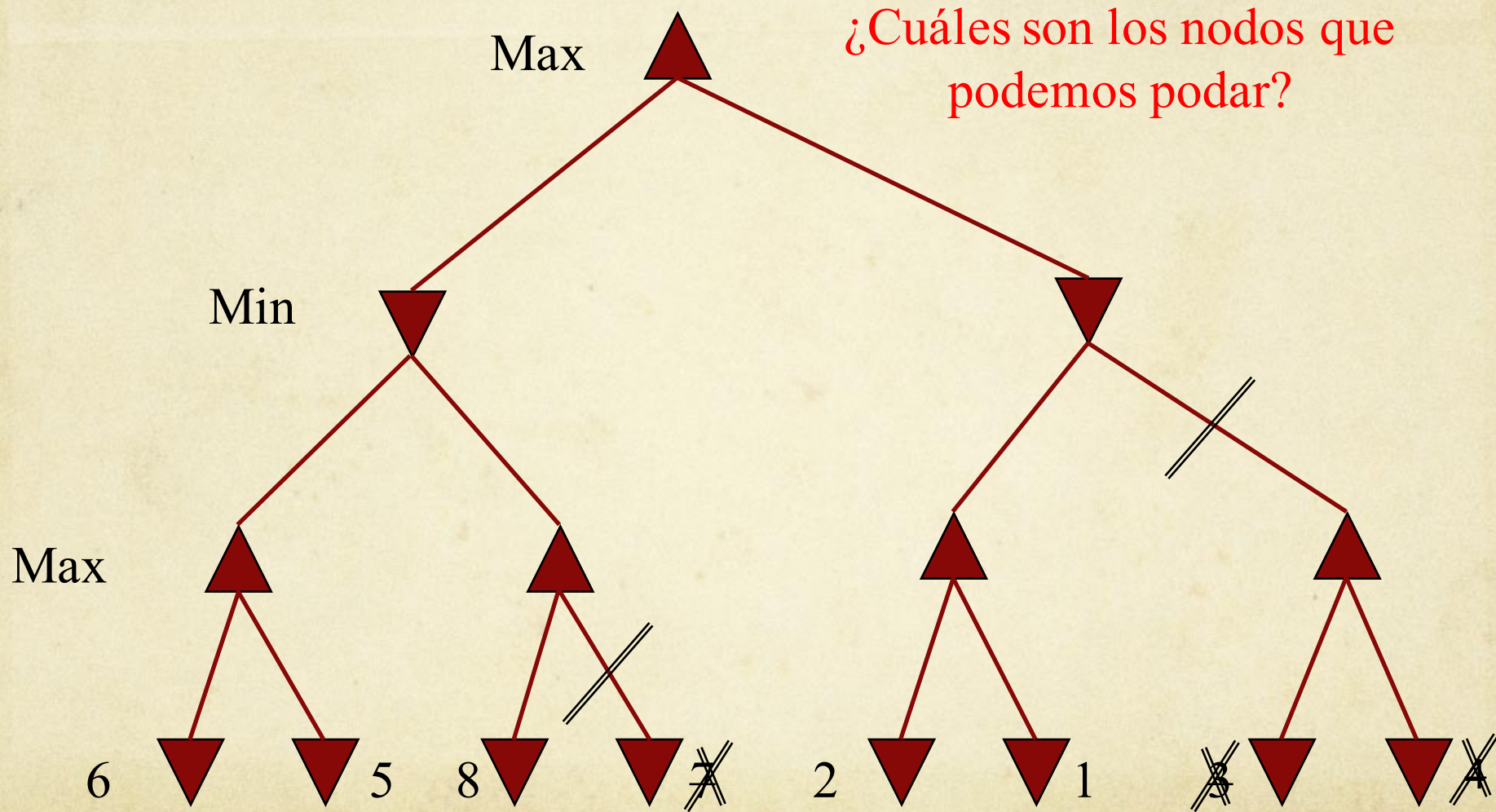
¿Cuáles son los nodos que podemos podar?



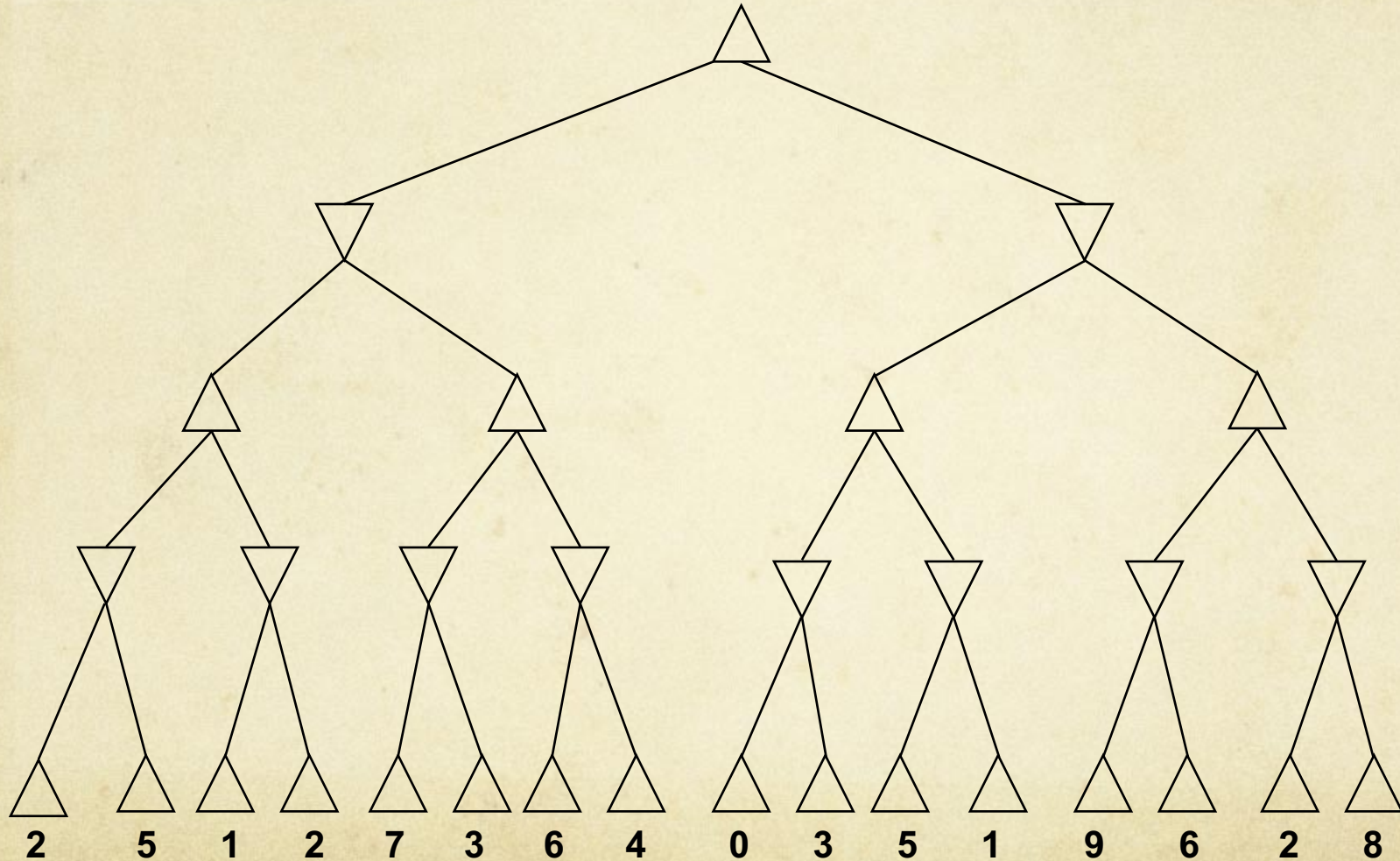
Ejemplo 2(Orden)



Ejemplo 2(Orden)

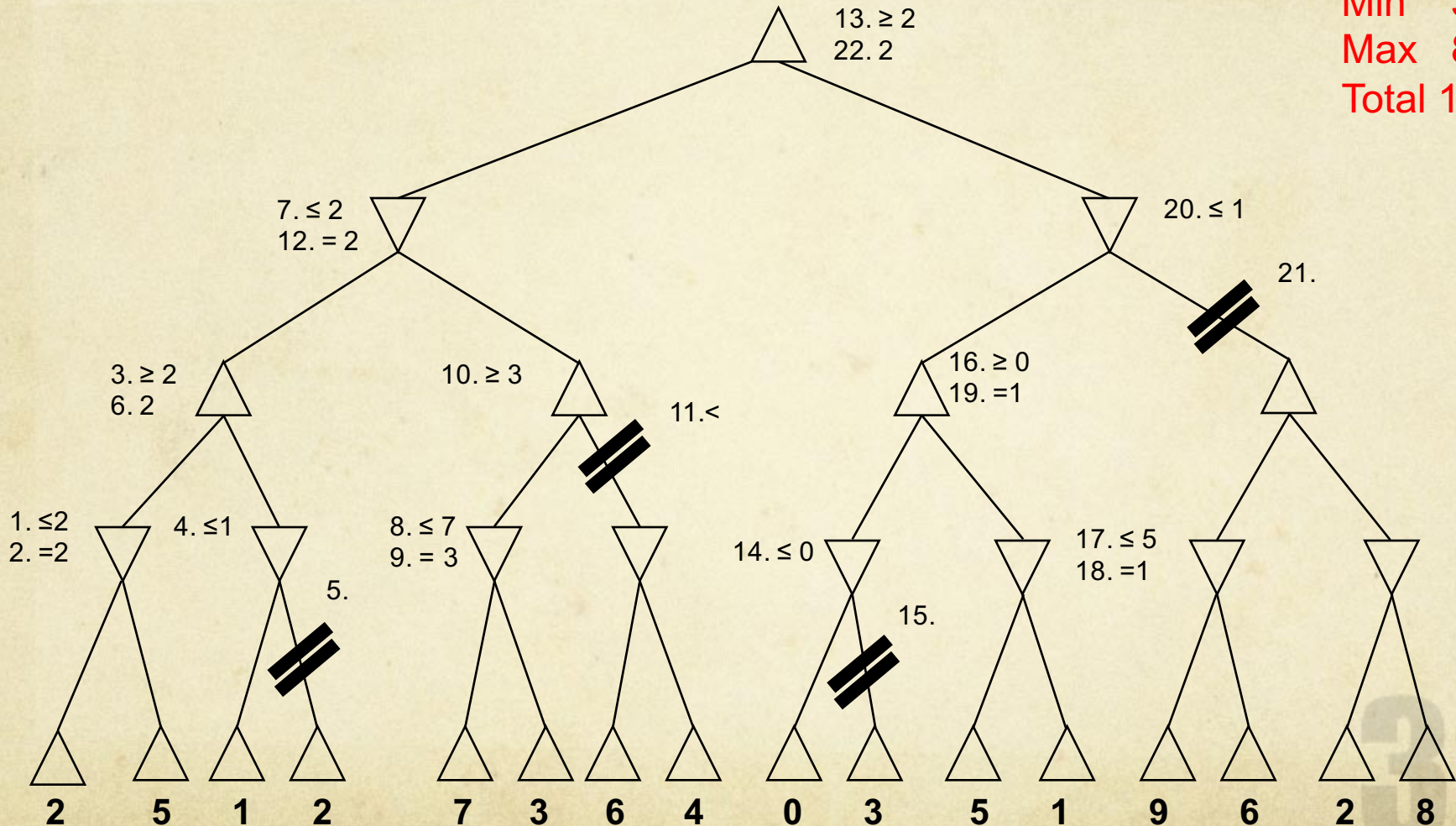


Ejemplo 1



Ejemplo 1 alfa beta

Min 3
Max 8
Total 11



Juegos No deterministas

Expectiminimax



Juegos estocásticos

- Introducimos un elemento aleatorio en el juego, p.e. Dado.
- Algoritmo similar al minimax pero incluimos un nodo CHANCE a los anteriores nodos MAX y MIN

EXPECTIMINIMAX(s)

UTILIDAD(s)

max_a EXPECTIMINIMAX (RESULT(s,a))

min_a EXPECTIMINIMAX (RESULT(s,a))

SUM P(r) max_a EXPECTIMINIMAX (RESULT(s,r))

if TERMINAL-TEST(s)

if PLAYES(s) = MAX

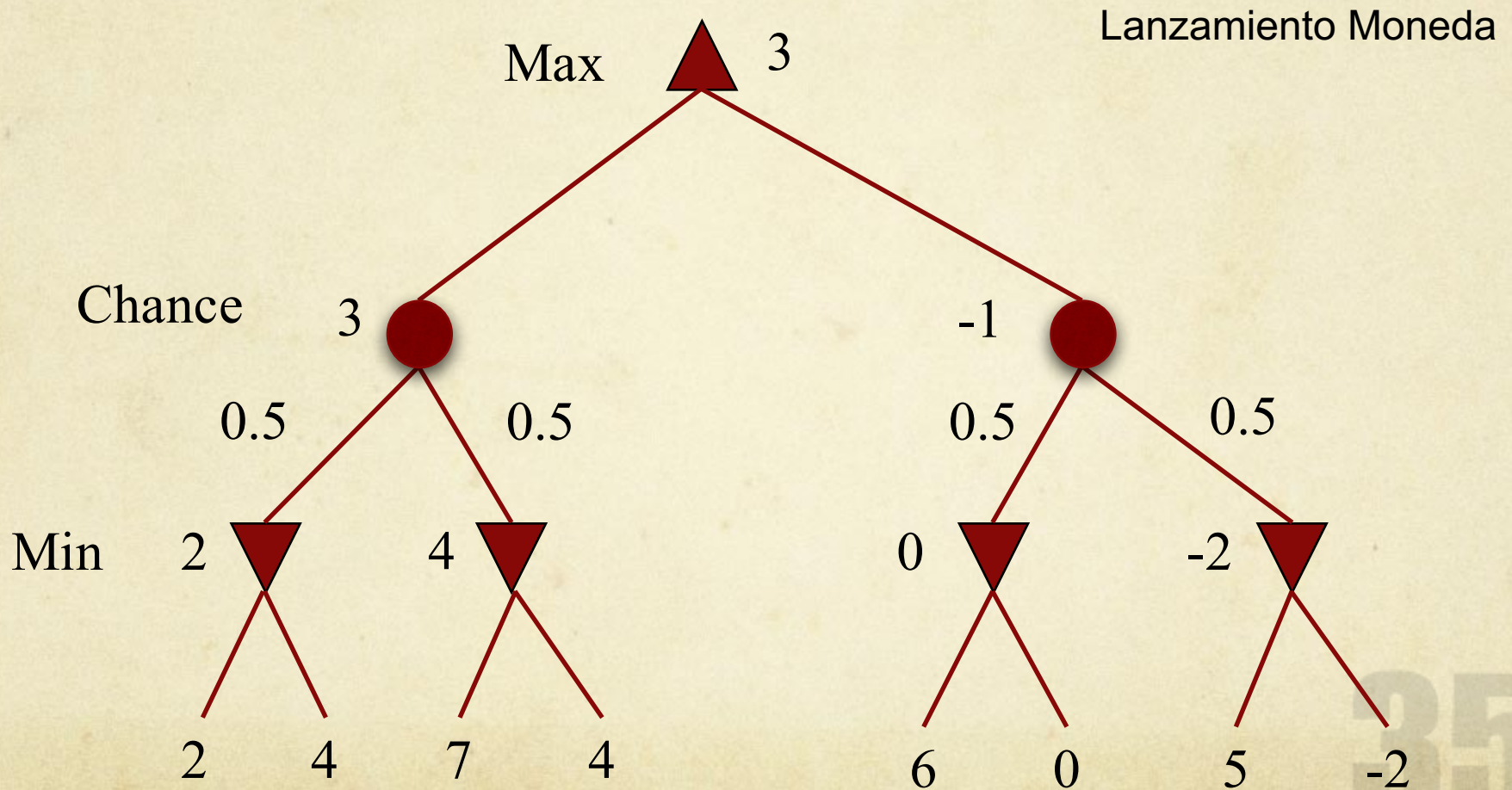
if PLAYES(s) = MIN

if PLAYES(s) = CHANCE

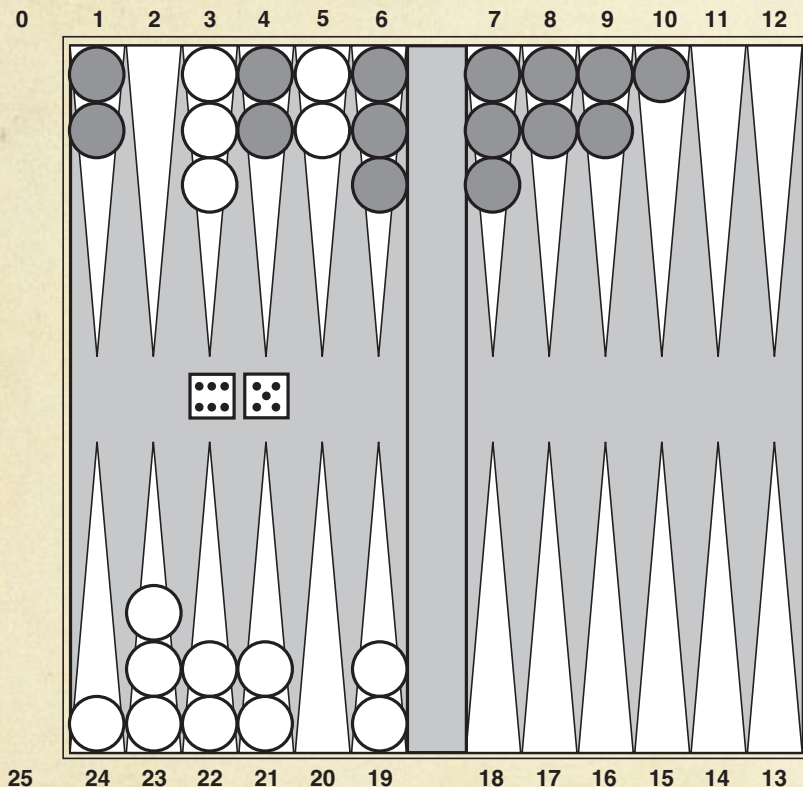
roll



EXPECTIMINIMAX



Backgammon



El objetivo del juego es mover todas la piezas fuera del tablero.

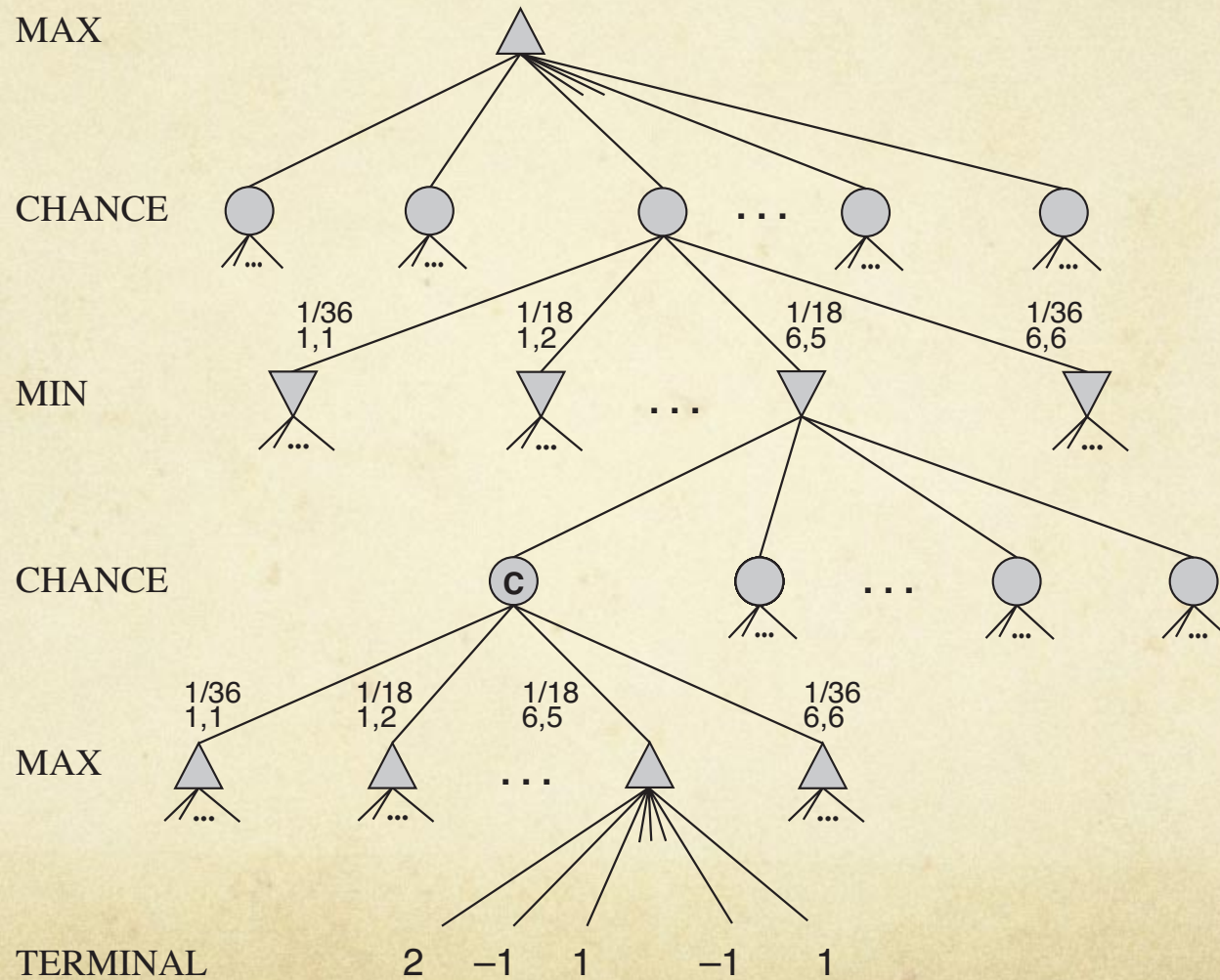
Las blancas salen por la posición 25 y las negras por la posición 0.

Una pieza puede mover a cualquiera posición a menos que varias piezas del oponente ocupen la posición.

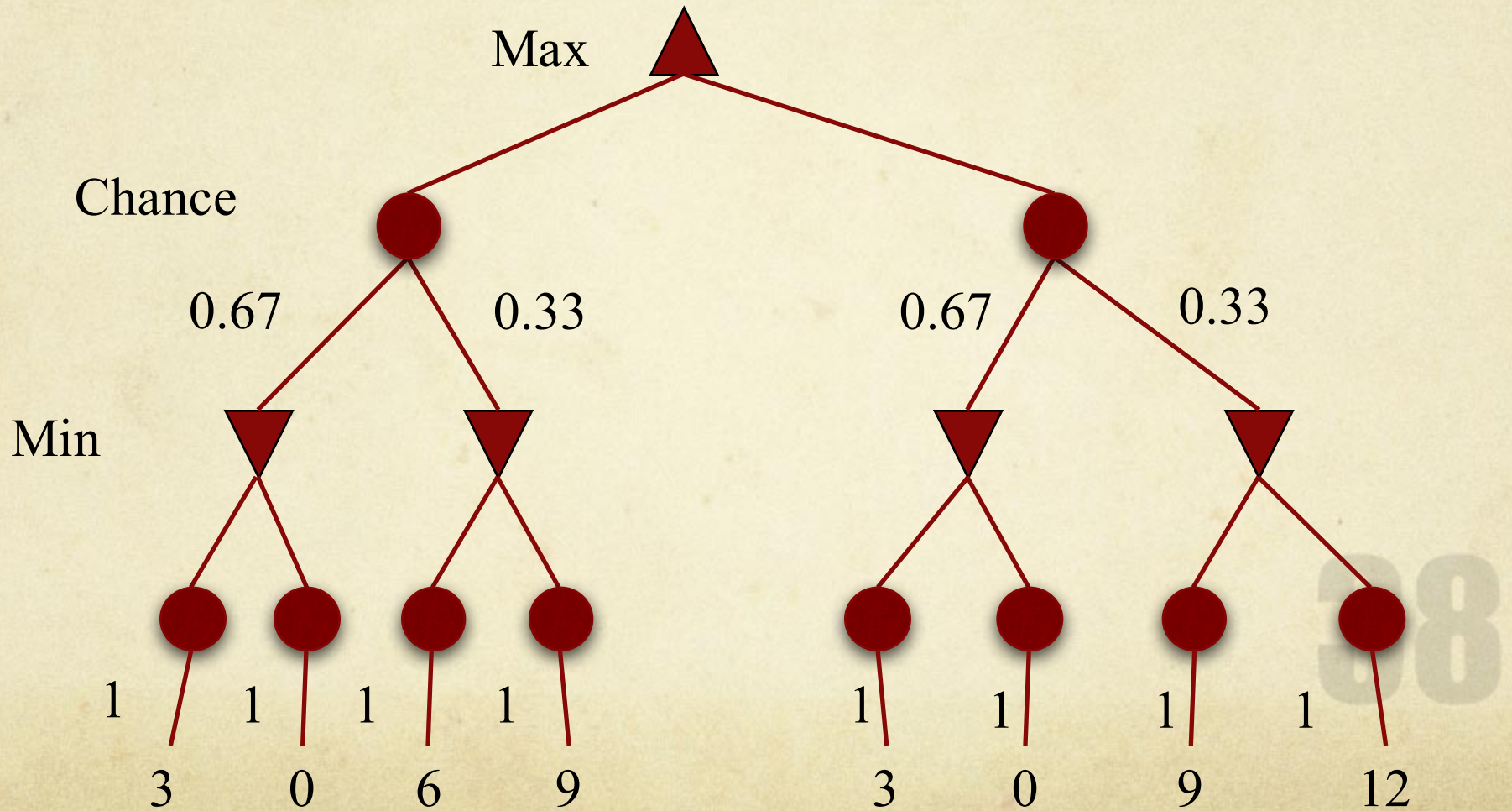
Blanca lanza y con los dados 6-5:
(5-10,5-11) (5-11,19-24)(5-10,10-16)
y (5-11,11-16)

Probabilidad dobles $1/36$ los demás $2/36=1/18$ doble posibilidad mismo resultado

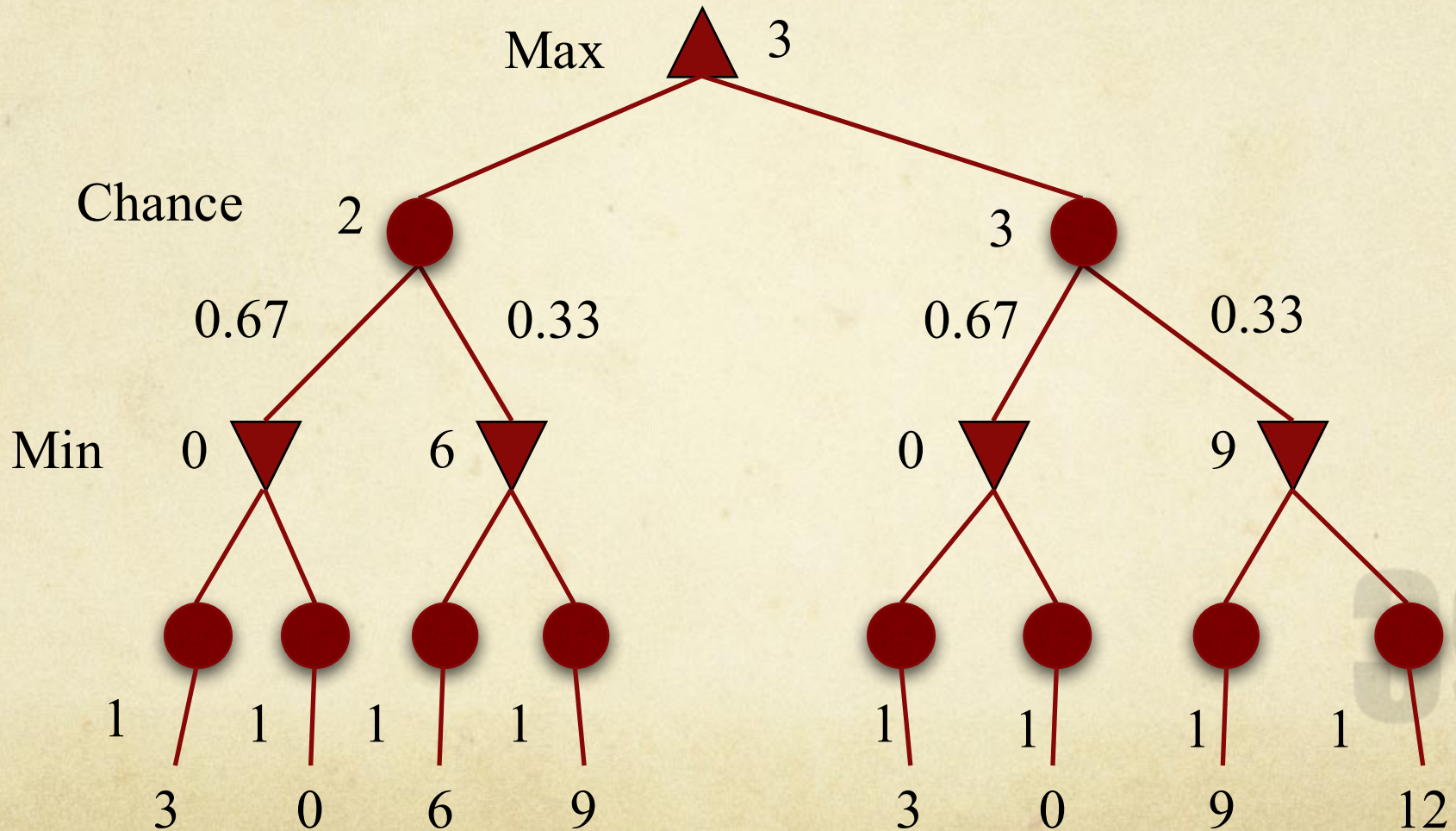
EXPECTIMINIMAX



EXPECTIMINIMAX



EXPECTIMINIMAX



Juegos

Funciones de Evaluación



Funciones de Evaluación

- **Minimax** genera el espacio de búsqueda **entero**, mientras que el algoritmo **alfa-beta** podemos **partes**.
 - Necesario llegar a estados terminales.
- Trabajo shannon (1950) *programming a computer for playing chess*, propone una función de evaluación heurística a los estados convirtiendo nodos no terminales a terminales.
 - Sustituye **función utilidad** por **función evaluación heurística**, que da una **estimación** de utilidad de la posición y establecemos un test **límite** que decide cuando terminar de aplicar función.
- Su cálculo ha de ser poco costoso

Decisiones imperfectas en juegos de dos adversarios

- **Decisión imperfecta:** Decisión tomada por el algoritmo sobre un horizonte que no alcanza el final del juego (se asume) y con función de evaluación estimada $f = \hat{u}$.
- Función de evaluación, ejemplos:
 - $f(n) =$
 - 1) si “n” no es terminal: (número de filas, columnas o diagonales libres para MAX) - (número de filas, columnas o diagonales libres para MIN)
 - 2) si gana MAX: ∞
 - 3) si gana MIN: $-\infty$

X		
O		

Nº pos max = 5

Nº pos min = 5

$$f(n) = 5 - 5 = 0$$

X		
	O	

Nº pos max = 4

Nº pos min = 5

$$f(n) = 4 - 5 = -1$$



Sistemas Inteligentes

José A. Montenegro Montes

monte@lcc.uma.es

