Algorítmica

Capítulo 5: Algoritmos para la Exploración de Grafos.

Tema 13: Grafos implícitos

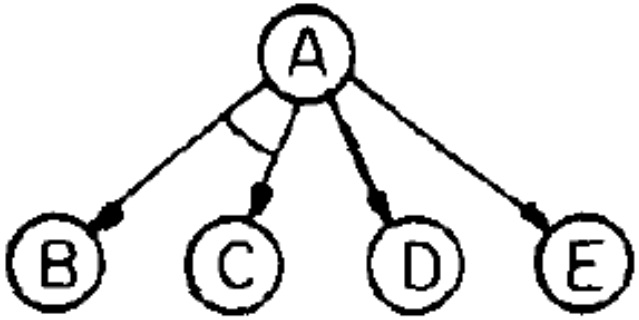
- Grafos Implícitos.
- Árboles para Juegos.
- Algoritmos de solución para juegos elementales

Grafos implícitos. Grafos Y/O

- Muchos problemas complejos pueden descomponerse en una serie de subproblemas tales que la solución de todos, o algunos de ellos, constituya la solución del problema original.
- A su vez, estos subproblemas pueden descomponerse en sub-subproblemas, y asi sucesivamente, hasta conseguir problemas lo suficientemente triviales como para poderlos resolver sin dificultad.
- Esta descomposición de un problema en subproblemas puede representarse mediante un grafo dirigido, en el que los nodos serian problemas, y los descendientes de un nodo serian los subproblemas asociados al mismo.
- Es un tema absolutamente actual cuyas aplicaciones son importantisimas: www, “data mining”, recuperación de información, ...

Grafos Y/O

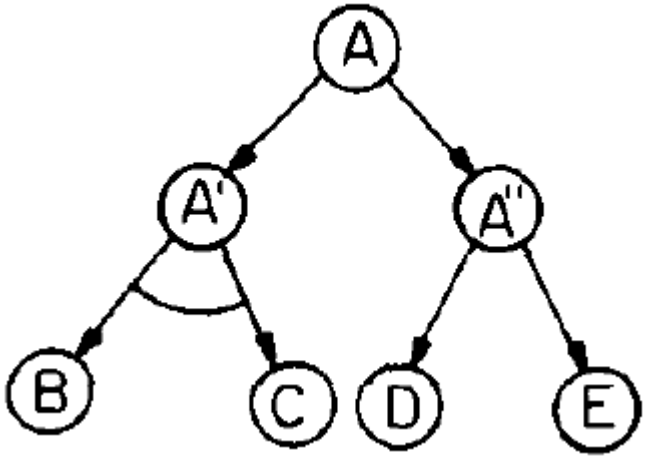
- Por ejemplo, el grafo de la figura



- representa un problema A que puede resolverse bien mediante la resolución de los dos subproblemas B y C, o bien mediante la de D o E aisladamente.
- Los grupos de subproblemas que deben resolverse para implicar una solución del nodo padre, se conectan mediante un arco que una las respectivas aristas (caso de B y C aquí).
- Podemos introducir nodos fantasma para uniformar el grafo

Grafos Y/O

- Todos los nodos serán tales que sus soluciones requieran que haya que resolver todos sus descendientes (**nodos Y**: A'), o bien que solo haya que resolver un descendiente (**nodos O**: A y A'').

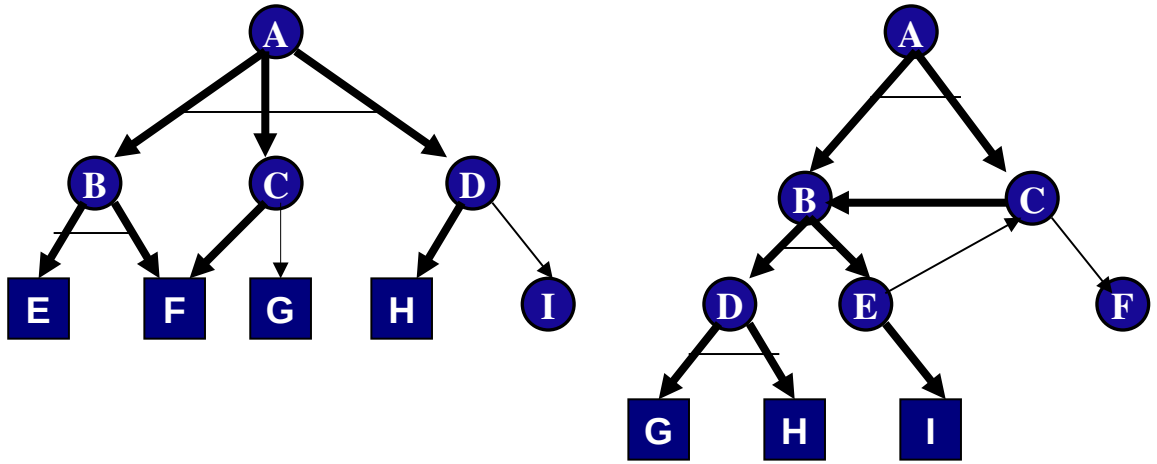


- Los nodos sin descendientes se llamarán terminales.
- Los nodos terminales representan problemas primitivos, y se marcan como resolubles o no resolubles. Los nodos terminales resolubles se representaran por rectángulos.

Grafos Y/O

- La descomposición de un problema en diversos subproblemas se conoce con el nombre de Reducción del Problema.
- La Reducción de Problemas se usa frecuentemente en ámbitos como la demostración de teoremas, la integración simbólica o el análisis de calendarios industriales.
- Cuando se usa la Reducción de Problemas, dos problemas diferentes pueden generar un subproblema común.
- En ese caso puede ser deseable tener solo un nodo representando a este subproblema, ya que esto podría significar tener que resolver ese subproblema solo una vez
- Lo vemos en la siguiente transparencia

Grafos Y/O



Nótese que el grafo no es un árbol.

Puede ser además que tales grafos tengan ciclos dirigidos, como en la figura derecha. La presencia de un ciclo dirigido no implica en si misma la irresolubilidad del problema.

Llamaremos grafo solución a un subgrafo de nodos resolubles que muestra que el problema esta resuelto.

Los grafos solución de los grafos de la anterior figura se representan por aristas gruesas.

Determinando la resolubilidad

- Veamos primero como determinar si un árbol Y/0 dado representa o no un problema resoluble (la extensión al caso de grafos se deja como ejercicio).

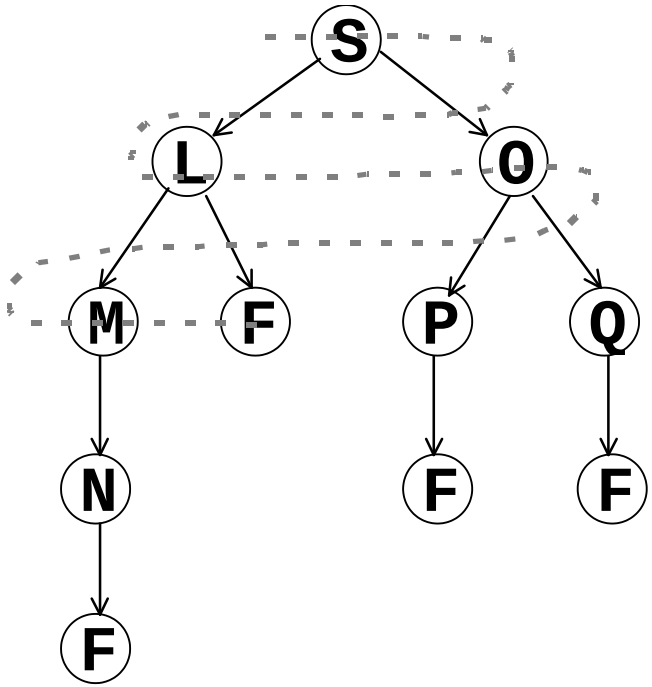
- Está claro que, realizando un recorrido en post-orden del árbol Y/O, podemos determinar si un problema es resoluble o no.
- En lugar de evaluar todos los hijos de un nodo, el algoritmo finalizaría tan pronto como descubriera que un nodo es irresoluble o resoluble.
- Esto reduce la cantidad de trabajo que el algoritmo tiene que hacer, sin que por ello la salida se afecte

Determinando la resolubilidad

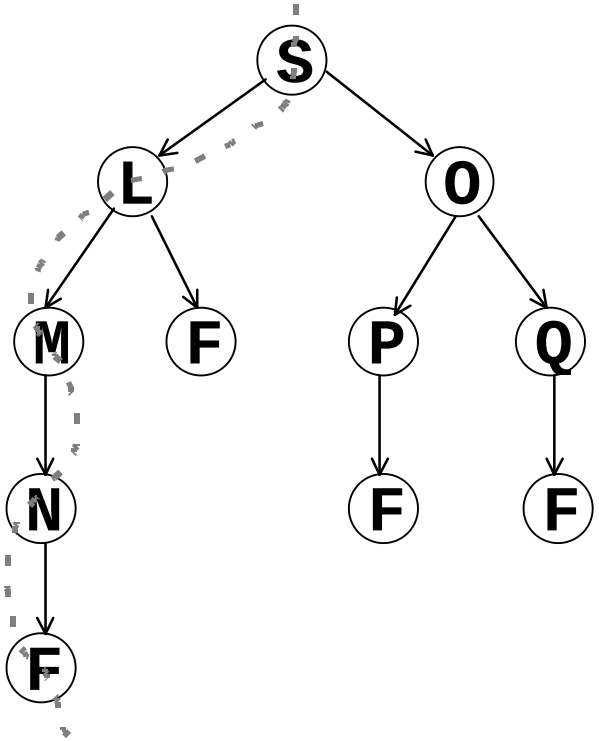
```
Procedimiento RESUELVE (T)
    {T es un árbol Y/O con raíz T. T ≠ 0. El algoritmo devuelve 1
   si  el problema es resoluble, y 0 en otro caso}
Begin
   Caso          :T es un nodo terminal:       If T es resoluble
                                               Then Return (1)
                                                    Else Return (0)
          : T es un nodo Y:       For cada hijo S de T do
                                  If RESUELVE (S) = 0
                                  Then Return (0)
                                       Return (1)
          : En otro caso:                 For cada hijo S de T do
                                  If RESUELVE (S) = 1
                                  Then Return (1)
                                               Return (0)
End
```

Generación de grafos implícitos

Primero en anchura



Primero en profundidad



Generación de grafos implícitos

- A menudo, solo de dispone de forma implícita del árbol Y/O correspondiente a un problema.
- Por tanto, vamos a dar una función F que genere todos los hijos de un nodo ya generado.
- Los nodos del árbol pueden generarse primero en profundidad o primero en anchura.
- Como es posible que un árbol Y/O tenga profundidad infinita, si comenzáramos una generación del árbol primero en profundidad, generando todos los nodos en un camino infinito desde la raíz, podríamos no llegar a determinar un subarbol solución (incluso aunque existiera).
- Esto se evita restringiendo la búsqueda primero en profundidad a la generación del árbol Y/O dentro solo de una profundidad d.

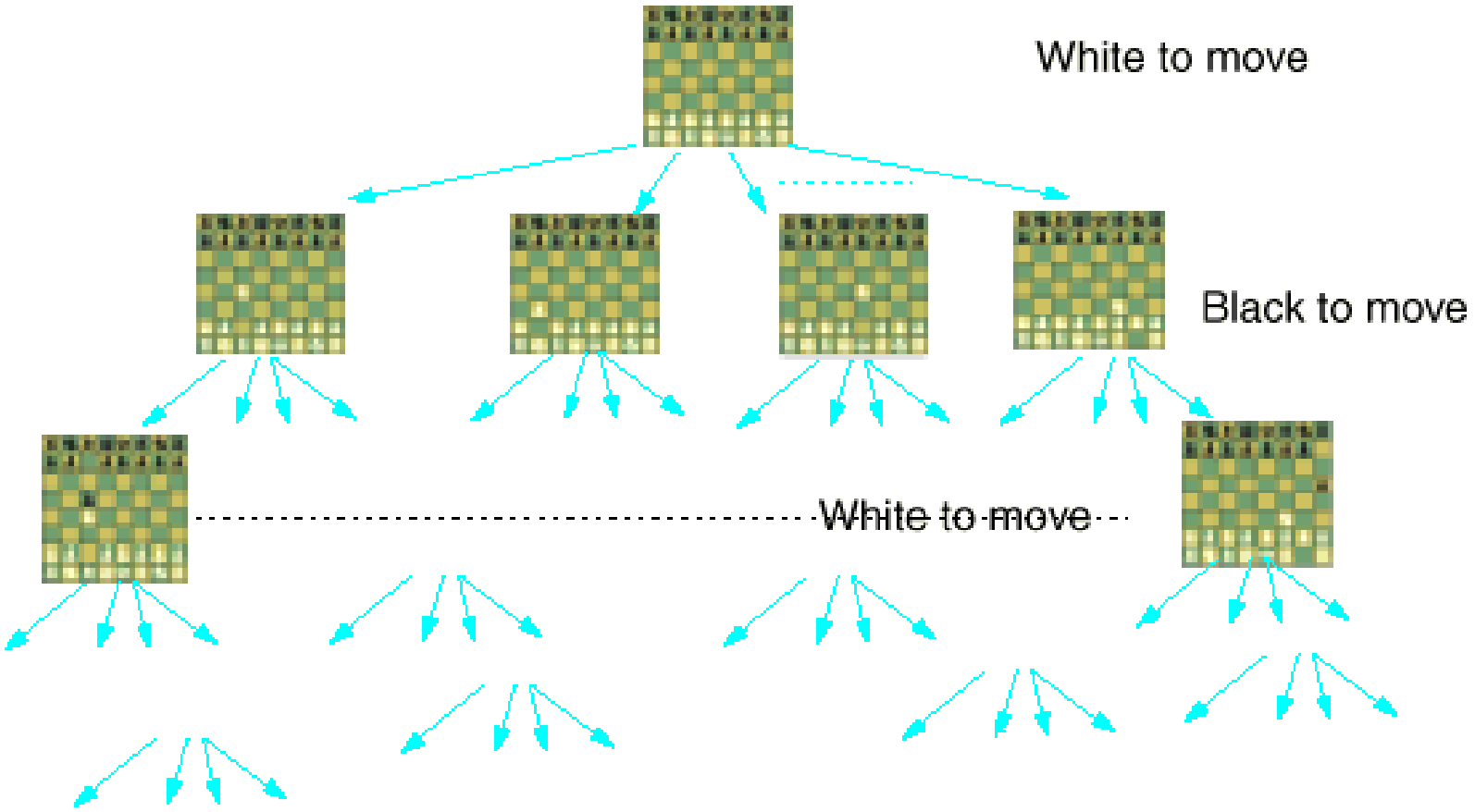
Generación de grafos implícitos

- Así, los nodos que a profundidad d sean no terminales, se etiquetarán irresolubles. En este caso, queda garantizado que la búsqueda primero en profundidad encontrará un subárbol solución, dado que hay una profundidad no mayor que d.
- La búsqueda (o generación) primero en anchura no presenta este inconveniente. Ya que cada nodo puede tener solo un número finito de hijos, ningún nivel en el árbol Y/O puede tener un número infinito de nodos.
- La generación primero en anchura del árbol Y/O garantiza encontrar un subárbol solución, si es que este existe.
- El algoritmo se deja como ejercicio

Árboles para juegos.

- La exploración de arboles se aplica continuamente en diferentes campos
- Una aplicación interesante de los árboles se encuentra en el desarrollo de juegos de estrategia (cartas, ajedrez, damas, etc.).
- Aunque solo sea desde el punto de vista de sus múltiples aplicaciones, el concepto de juego es uno de los mas interesantes que podemos encontrar.

Un típico árbol de juego



Un ejemplo clásico

- ¿Que movimiento debemos hacer?

Mueven blancas

Un ejemplo popular

- Deep Blue
  - El considerado mejor jugador de ajedrez del mundo Garry Kasparov en mayo de 1997 se enfrentó en Nueva York contra Deep Blue, en juego no sólo estaba el millón de dolares para el ganador sino la supremacía del hombre frente a las máquinas.
  - Desarrollado por un equipo de científicos de IBM en un proyecto que se inició 10 años atrás, era capaz de calcular más de 200 millones de posiciones del ajedrez por segundo, además de esta impresionante capacidad de cómputo y a la inteligencia artificial con la que se le había dotado (con una gran biblioteca de jugadas precalculadas) el ordenador contaba con su incapacidad de cansancio o de tener un sólo descuido.
  - 32 P2SC Procesadores
  - Era capaz de analizar entre 50 y 100.000 millones de posiciones en menos de TRES minutos
  - 1000 veces mas rápido que su predecesor, Deep Thought
- Kasparov ganó en 1996

En 1997 perdió

Otro ejemplo popular (mas reciente)

- **AlphaGo** de Google
  - El Go se originó en China hace más de 2.500 años. Confucio escribió sobre el juego, que se considera una de las cuatro artes esenciales que debe conocer cualquier erudito chino. Jugado por más de 40 millones de personas en todo el mundo, las reglas del juego son simples:
    - Los jugadores se turnan para colocar piedras negras o blancas en un tablero, tratando de capturar las piedras del oponente o rodear los espacios vacíos para hacer puntos por territorio. El juego se juega principalmente a través de la intuición y la sensación, y debido a su belleza, sutileza y profundidad intelectual, ha capturado la imaginación humana por siglos.
- El número de movimientos legales es (exactamente):
  - 208,168,199,381,979,984,699,478,633,344,862,770,286,522,453,884,530,548,425,639,456,820,927,419,612,738,015,378,525,648,451,698,519,643,907,259,916,015,628,128,546,089,888,314,427,129,715,319,317,557,736,620,**397,247,064,840,935**.
  - El 19 de marzo de 2016, el mejor jugador de Go del mundo, Lee Sedol, jugó contra el programa de I.A. de Google **AlphaGo**.

Y perdió

Definición de juego.

- **Concepto previo**
- Un árbol topológico, o árbol del juego, es una colección finita de vértices, conectados por arcos que constituyen una figura conexa que no incluye curvas cerradas simples.
- Dados dos vértices cualesquiera A y B, solamente hay una secuencia de arcos y vértices que unen A y B.
- Si Γ es un árbol topológico con un vértice distinguido (raíz) A, decimos que el vértice C sigue al vértice B si la sucesión de arcos que une A con C pasa a través de B.
- Se dice que C sigue inmediatamente a B, si C sigue a B y, además, hay un arco que une B con C. Un vértice X se dice terminal si no hay ningún vértice que lo siga.

Definición de juego.

- **Un juego n-personal en forma extensiva se define como,**
- Un árbol topológico Γ, con un vértice distinguido A que suele llamarse punto de partida de Γ.
- Una función, llamada función de pagos, que asigna un n-vector a cada vértice terminal de Γ.
- Una partición de los vértices no terminales de Γ en n+1 conjuntos, llamados conjuntos de jugadores, S_0, S_1, ..., S_n .
- Una distribución de probabilidad definida en cada vértice de S_0 .

Definición de juego.

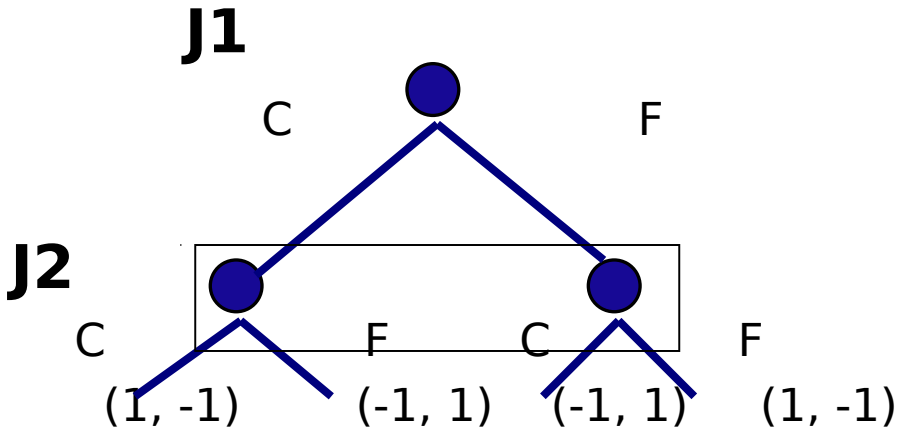
- Para cada i = 1, ..., n, una subpartición de S_i en subconjuntos S_i^j, llamados conjuntos de información, tales que dos vértices en el mismo conjunto de información tiene el mismo número de vértices seguidores inmediatos, y ningún vértice puede seguir a ningún otro en un mismo conjunto de información.
- Para cada conjunto de información S_i^j existe un conjunto de índices I_i^j, y una aplicación uno a uno de I_i^j en el conjunto de los vértices inmediatos seguidores de cada vértice de S_i^j .

Definición de juego: Interpretación

- Los elementos del juego se interpretan de la siguiente forma.
  - La condición a) establece el punto de comienzo.
  - La b) da la forma en que se van a obtener recompensas.
  - La c) divide los movimientos en movimientos de azar (i = 0) y movimientos personales (i = 1, ..., n) correspondientes a los n jugadores.
  - La d) define un esquema de aleatorización en cada movimiento de azar.
  - La e) divide los movimientos de los jugadores en conjuntos de información: cada uno de ellos puede conocer en que conjunto de información se encuentra, pero desconoce en que vértice esta.

Ejemplo

- Supongamos que el jugador 1 escoge cara (C) o cruz (F). Entonces el jugador 2, sin saber lo que el 1 ha escogido, también escoge entre cara o cruz. Si ambos jugadores eligen de forma desigual, entonces el jugador 2 le gana una unidad al 1. En cualquier otro caso, el primer jugador le ganara una unidad al 2.



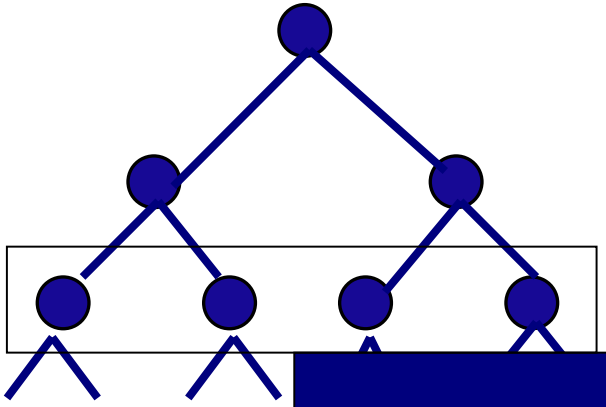
Ejemplo

- En la jugada I el primer jugador elige un número x entre 1 y 2; en la jugada II, J2 informado del número elegido por J1, elige a su vez un número y entre 1 y 2; en la jugada III, J1 sin conocer la elección hecha por J2 en la jugada II, y habiendo olvidado el número x elegido en la I, elige un numero z entre 1 y 2. Una vez elegidos los números x, y, z, J2 paga a J1 la cantidad M(x,y,z):

J1

J2

J1



	[1,1]	**[1,2]**	**[2,1]**	**[2,2]**
(1,1)	-2	-2	3	3
(1,2)	-1	-1	-4	-4
(2,1)	5	2	5	2
(2,2)	2	6	2	6

M(1,1,1) = -2 M(1,1,2) = -1
M(2,1,1) = 5 M(1,2,2) = -4
M(1,2,1) = 3 M(2,1,2) = 2
M(2,2,1) = 2 M(2,2,2) = 6

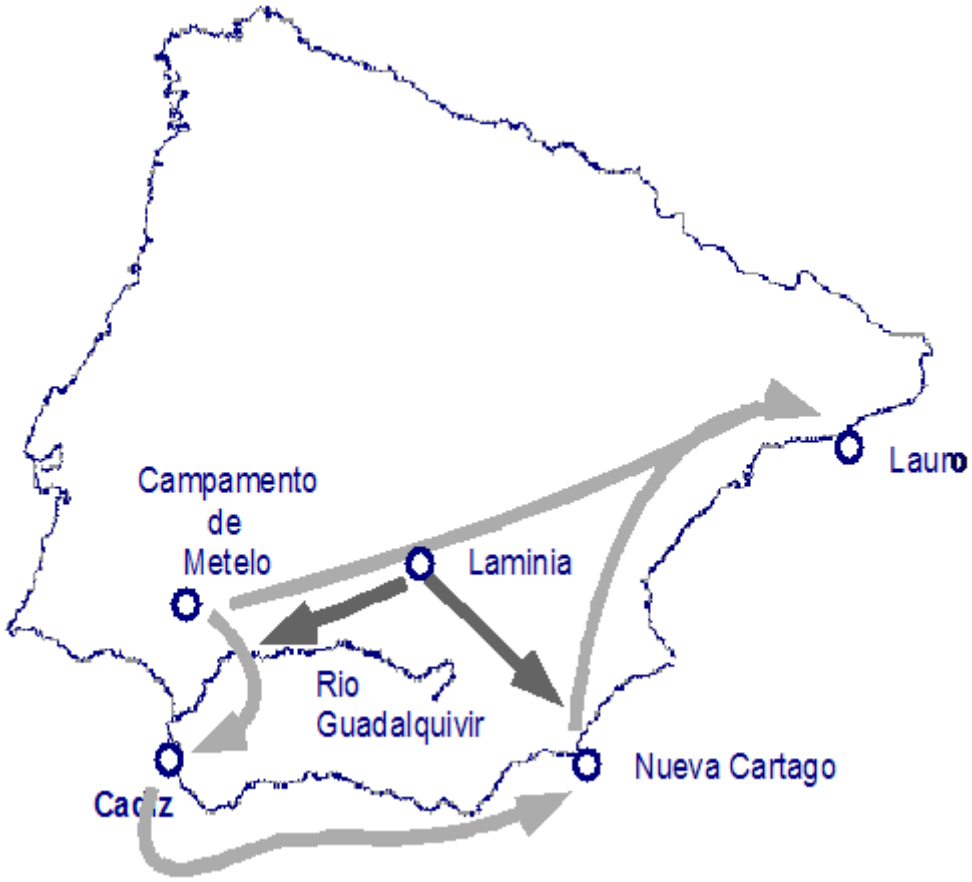
Ejemplo: Una de romanos

- Ha de Cristo co
- Lo Pompe ro la ibéri
- Po berar la e Nu o pudo i (L e Serto ca
- Er s, ambos “f
- Asi las cosas, los protagonistas de este “juego” pasaron a ser los respectivos segundos jefes, a saber:
    - Metelo, por Roma, cuyo objetivo era desplazar

Las estrategias de Metelo

Metelo tenía dos estrategias posibles.

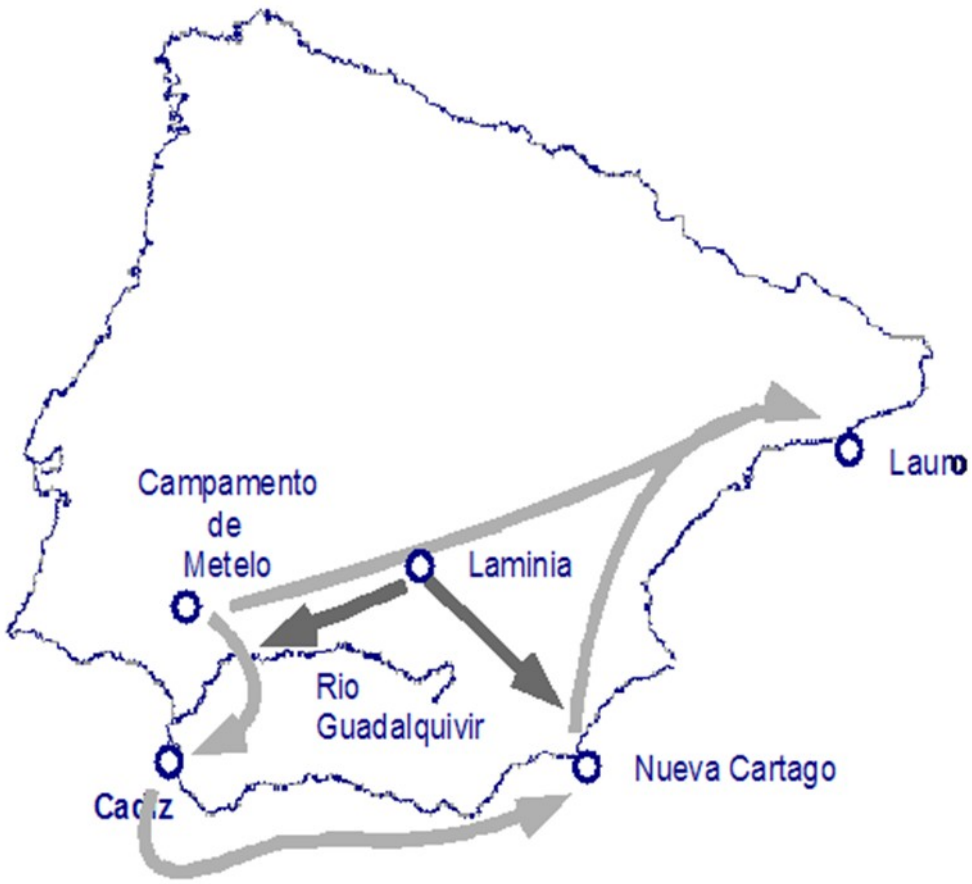
1. Podía atacar a Hirtuleyo, tomar Laminia (junto a Ciudad Real), y entonces dirigirse a Lauro. Pero sus opciones de exito eran pocas.
2. Dirigirse a Cádiz y entonces embarcar hacia Nueva Cartago, liberar la fortaleza que Pompeyo no había podido liberar, y entonces dirigirse hacia Lauro.



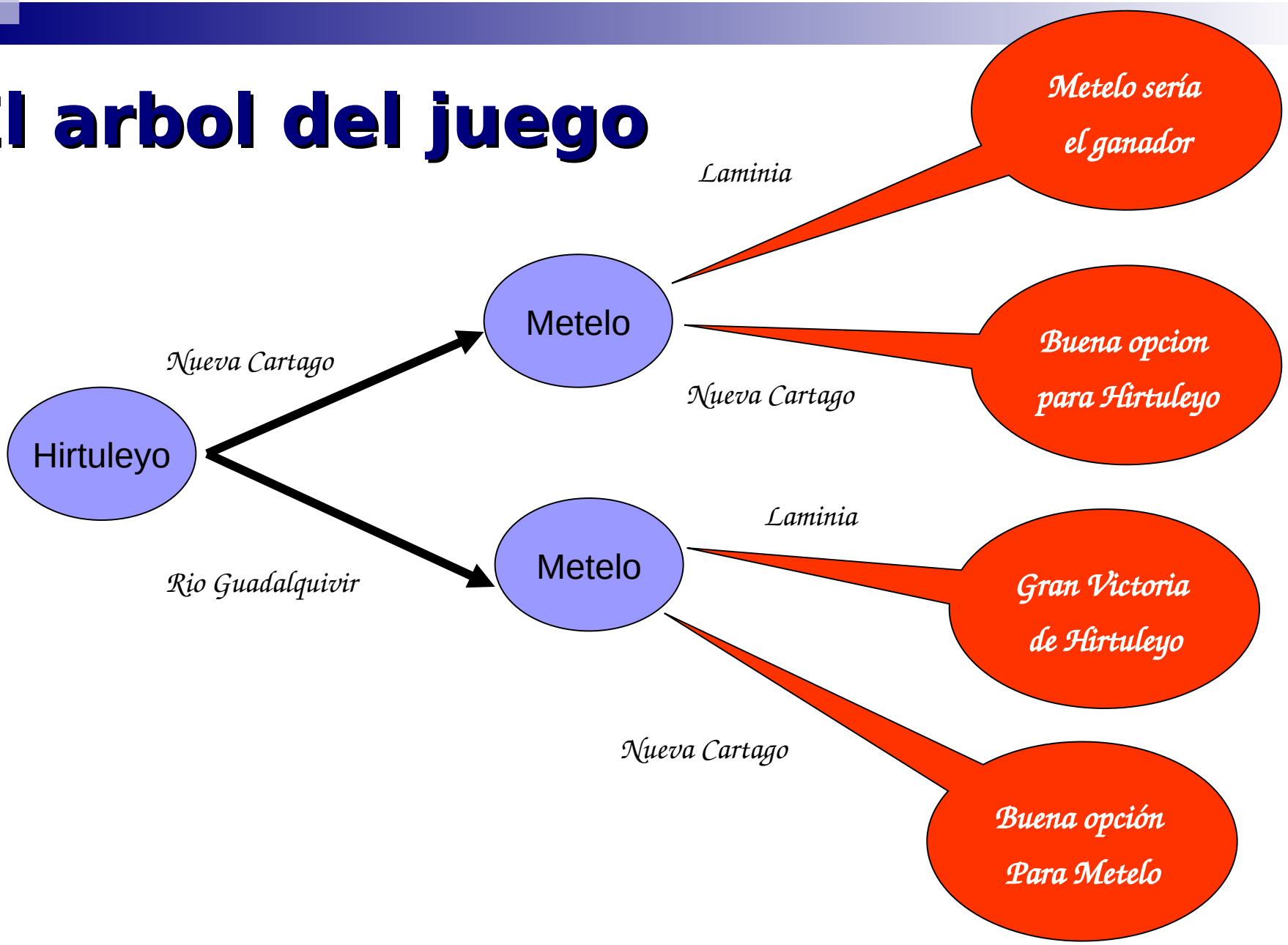
Estrategias de Hirtuleyo

1. Hirtuleyo podía dirigirse directamente hacia Nueva Cartago, sus posibilidades de derrotar a Metelo, en este caso, eran muy altas, pero Metelo sabría que Hirtuleyo estaba camino de Nueva Cartago y entonces podria tomar Laminia sin batalla, y dirigirse hacia Lauro.

2. Alternativamente, Hirtuleyo podría quedarse en Laminia hasta que Metelo saliera de su campamento, y



El arbol del juego



¿Cómo jugar estos juegos?

Supongamos un juego definido por una matriz,

-5	3	1	20
5	5	4	6
-4	-2	0	-5

Lo que interesa es saber que estrategia (fila) debe escoger J1, y que estrategia (columna) debe escoger J2 para no perder

Solución de estos juegos

- Si quisiéramos encontrar una forma de jugar en la que ambos jugadores se encontraran cómodos, podríamos emplear la conocida estrategia maximin
- El primer jugador escogería aquella estrategia que le proporcionara el valor

$$\mathrm{Max}_i \, \mathrm{Min}_j \, a_{ij}$$

- J2 tendría que jugar aquella estrategia que le proporcionara el valor,

$$\mathrm{Min}_j \, \mathrm{Max}_i \, a_{ij}$$

- El problema que aparece es que generalmente ambos valores no coinciden, por lo que hay que pasar a esquemas de resolución de juegos mas complejos y que no interesan para los objetivos de este tema.

Ejemplo de estrategia maximin

-5	3	1	20
5	5	4	6
-4	-2	0	-5

$$\text{Max}_i \, \text{Min}_j \, a_{ij} = \text{Max}\{-5,4,-5\} = 4$$

$$\text{Min}_j \, \text{Max}_i \, a_{ij} = \text{Min}\{5,5,4,20\} = 4$$

Ejemplo: juego de los palillos (Nim)

- Juegan dos jugadores A y B. El juego queda descrito por un panel que inicialmente contiene n palillos.
- Los jugadores A y B hacen alternativamente sus movimientos, empezando a jugar A. Un movimiento legal consiste en eliminar 1, 2 o 3 palillos del panel. Sin embargo, un jugador no puede quitar mas palillos de los que hay en el panel.
- El jugador que quita el último palillo pierde el juego y el otro gana.
- El panel esta completamente especificado en cualquier instante por el número de palillos que quedan, y en cualquier momento el status del juego se determina por la configuración del panel, junto con el jugador que le toca jugar a continuación.
- El juego de Nim no puede terminar en empate.

Ejemplo: juego de los palillos (Nim)

- Una configuración terminal es aquella que representa una situación de ganar, de perder o de empate. Todas las demás configuraciones son no terminales.
- En el juego de Nim solo hay una configuración terminal: Cuando no quedan palillos en el panel.
- Esta configuración es de ganancia para A, si B hizo el último movimiento o viceversa.
- Este es un juego para el que existe una forma óptima de jugar: Nunca se pierde
- El juego de Nim es importante porque es un juego de **Información Perfecta**, porque es un juego equitativo y porque cualquier otro juego equitativo es equivalente a uno de Nim.

Formalización: juegos mas generales

- Una sucesión C(1),...,C(m) de configuraciones se llama valida si
  - C(1) es la configuración inicial del juego
  - C(i), 0<i<m, son configuraciones no terminales, y
  - C(i+1) se obtiene de C(i) mediante un movimiento legal hecho por el jugador A si i es impar, y por el jugador B si i es par.
- Se supone que hay un número finito de movimientos legales.
- Una sucesión válida C(1),...,C(m) de configuraciones, en la que C(m) es una configuración terminal, se llama un caso del juego. La longitud de la sucesión C(1),...,C(m) es m.
- Un **Juego Finito** es un juego para el que no existen sucesiones válidas de longitud infinita.

Desarrollo de un juego general

- Todos los posibles casos de un juego finito pueden representarse por el árbol del juego.
- En el árbol del juego, cada nodo representaría una configuración, y el nodo raíz sería la configuración inicial C(1).
- Las transiciones desde un nivel al siguiente se hacen según mueva A o B.
  - En el juego de Nim las transiciones desde un nivel impar representarían los movimientos de A, mientras que todas las demás transiciones serian los movimientos de B.
- Las configuraciones terminales se representan por nodos hoja y suelen etiquetarse con el nombre del jugador que gana cuando se alcanza esa configuración.

Desarrollo de un juego general

- El grado de cualquier nodo en el árbol del juego es igual, a lo mas, al número de movimientos legales distintos que haya.
- Por definición, el número de movimientos legales en cualquier configuración es finito.
- La profundidad de un árbol de juego es la longitud del mas largo caso del juego.
- Los árboles de juego son útiles para determinar el siguiente movimiento que un jugador debe hacer:
- Partiendo de la configuración inicial representada por la raíz, el jugador A se enfrenta con tener que elegir entre varios posibles movimientos.
- Supuesto que A quiera ganar el juego, A debería elegir el movimiento que maximice su posibilidad de ganar.

Desarrollo de un juego general

- Si la recompensa no está claramente definida, se puede usar una función de evaluación E(X) que asigne un valor numérico a cada configuración X.
- Esta función dará la recompensa para el jugador A asociada a la configuración X.
  - Para un juego como el Nim con pocos nodos, basta definir E(X) sobre las configuraciones terminales.
- E(x) = 1 si X es una configuración ganadora para A
  = -1 si X es una configuración perdedora para A
- Usando esta función de evaluación, ahora lo que queremos es determinar cual de las posibles configuraciones que el jugador A tiene como posibles (supongamos que sean b, c y d) debe elegir.

Desarrollo de un juego general

- Si las configuraciones a las que pede moverse son b, c y d entonces:
- Claramente, la elección será aquella cuyo valor sea el del Max{V(b), V(c), V(d)}, donde V(x) es el valor de la configuración x.
- Para nodos hoja x, V(x) se toma como E(x).
- Para todos los demás nodos x, sea d $\geq$ 1 el grado de x, y sea C(1),...,C(d) la configuración representada por los hijos de x.
- Entonces V(x) se define como

V(x) = Max {V[C(i)]} $1 \leq i \leq d$ si x es un nodo de cuadrado (A)

= Min {V[C(i)]} $1 \leq i \leq d$ si x es un nodo circular (B)

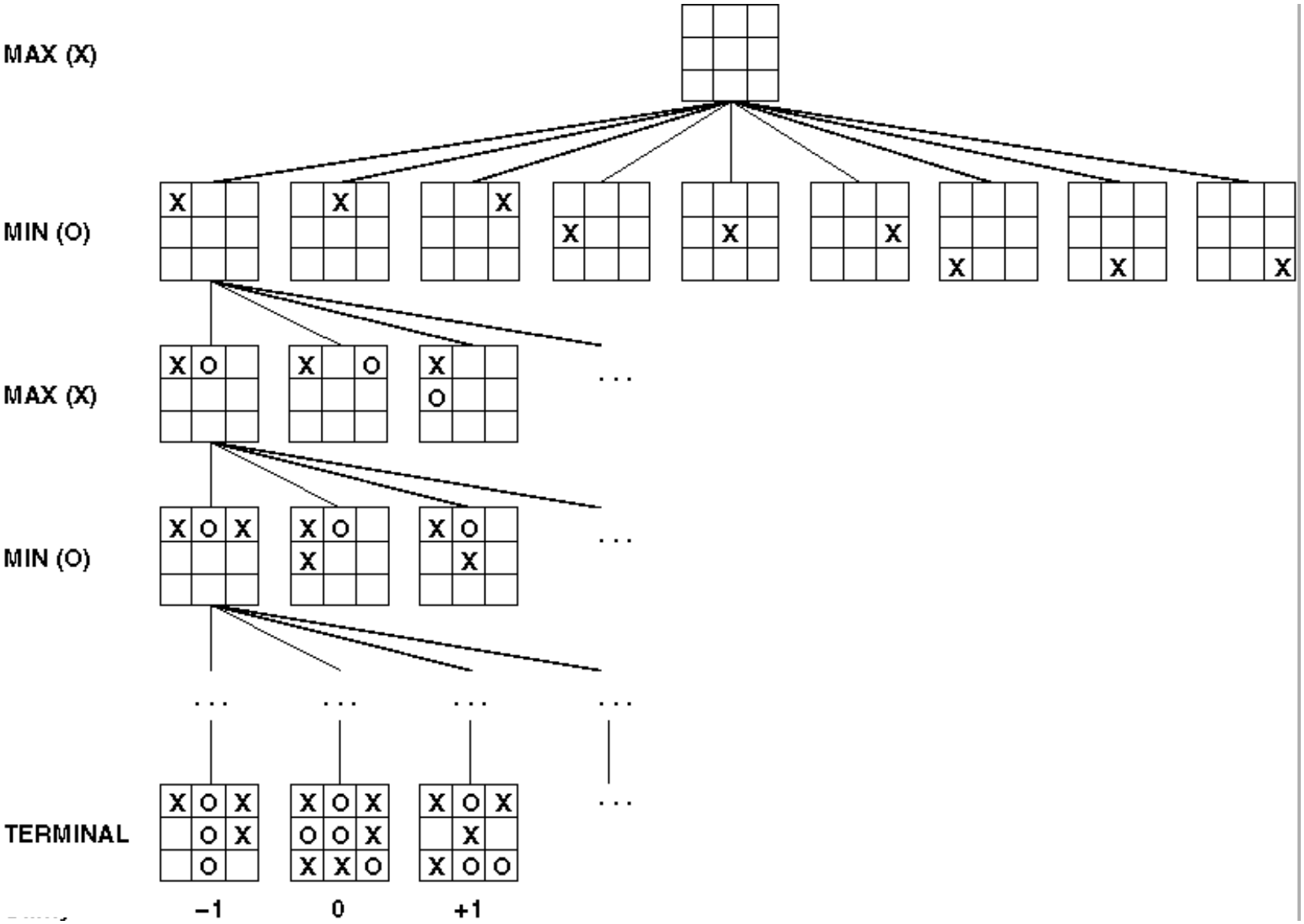
Generalización de la estrategia maximin

- La justificación de la formula anterior es simple.
- Si x es un nodo cuadrado, entonces estará en un nivel impar, y sería el turno para elegir A desde aquí, si alguna vez el juego alcanzara este nodo.
- Como A quiere ganar, A se moverá a un nodo hijo con valor máximo.
- En el caso de que x sea un nodo circular, debe estar en un nivel par, y si el nodo alguna vez alcanza ese nodo entonces será el turno para que mueva B.
- Como B no juega a ganar, hará un movimiento que minimice la posibilidad de que gane A. En este caso, la siguiente configuración será la que de el valor

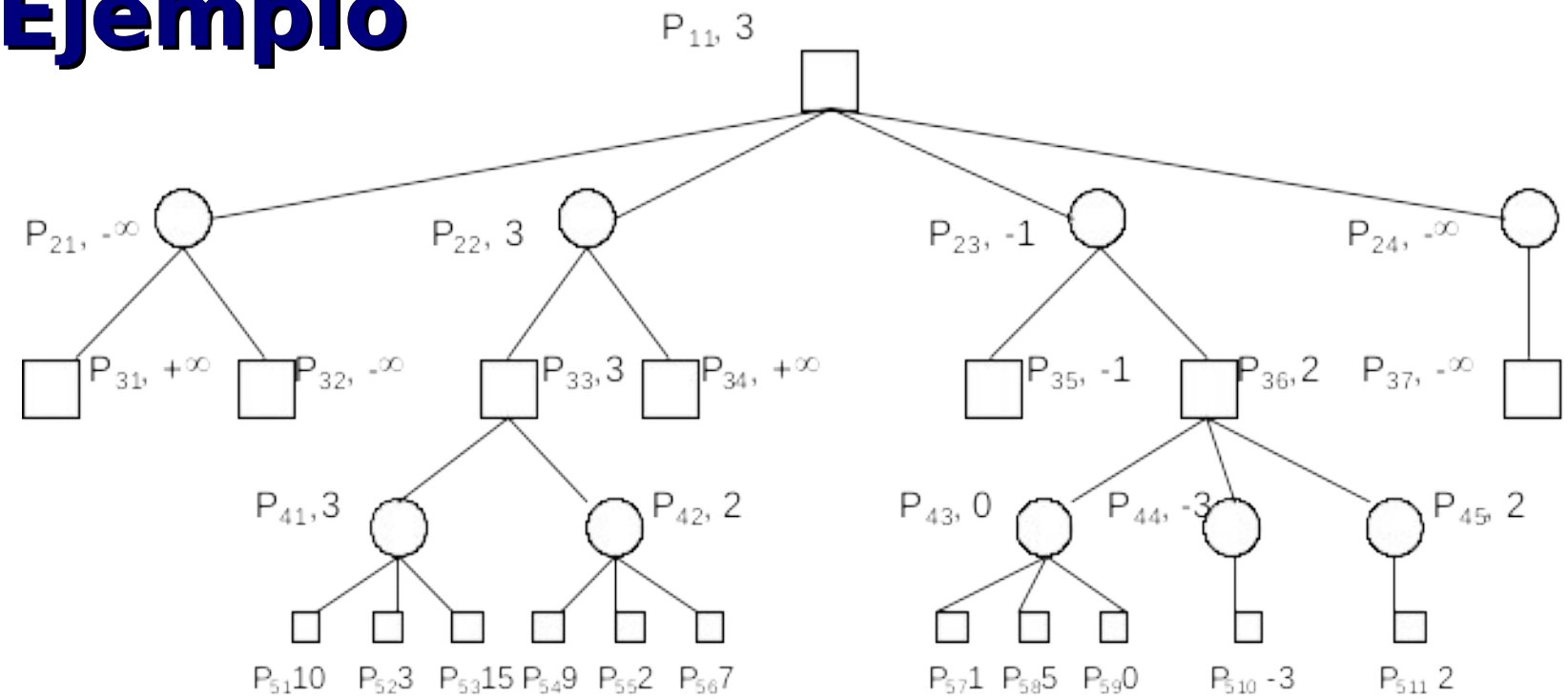
$$\text{Min } \{V[C(i)], 1 \leq i \leq d \}$$

- La ecuación define lo que se denomina **procedimiento minimax** para determinar el valor de la configuración x.

Un ejemplo: Tres en raya



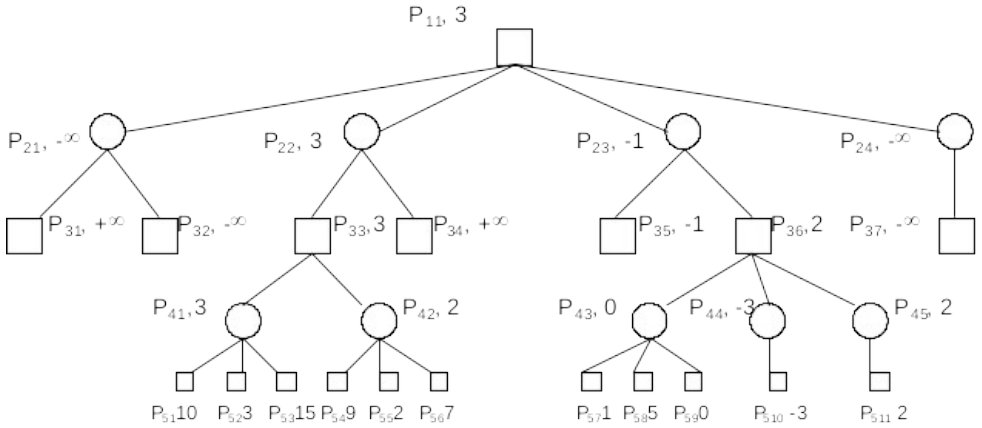
Ejemplo



- P(11) representa una configuración arbitraria desde la que A tiene que efectuar un movimiento. Los valores de los nodos hijo se obtienen evaluando la función E(x). El valor de P(11) se obtiene comenzando en los nodos del nivel 4, y usando la ecuación para calcular sus valores.
- El valor resultante para P(11) es 3, lo que significa que partiendo de P(11) lo mejor que A puede esperar es alcanzar una configuración de valor 3.

Ejemplo de desarrollo de un juego

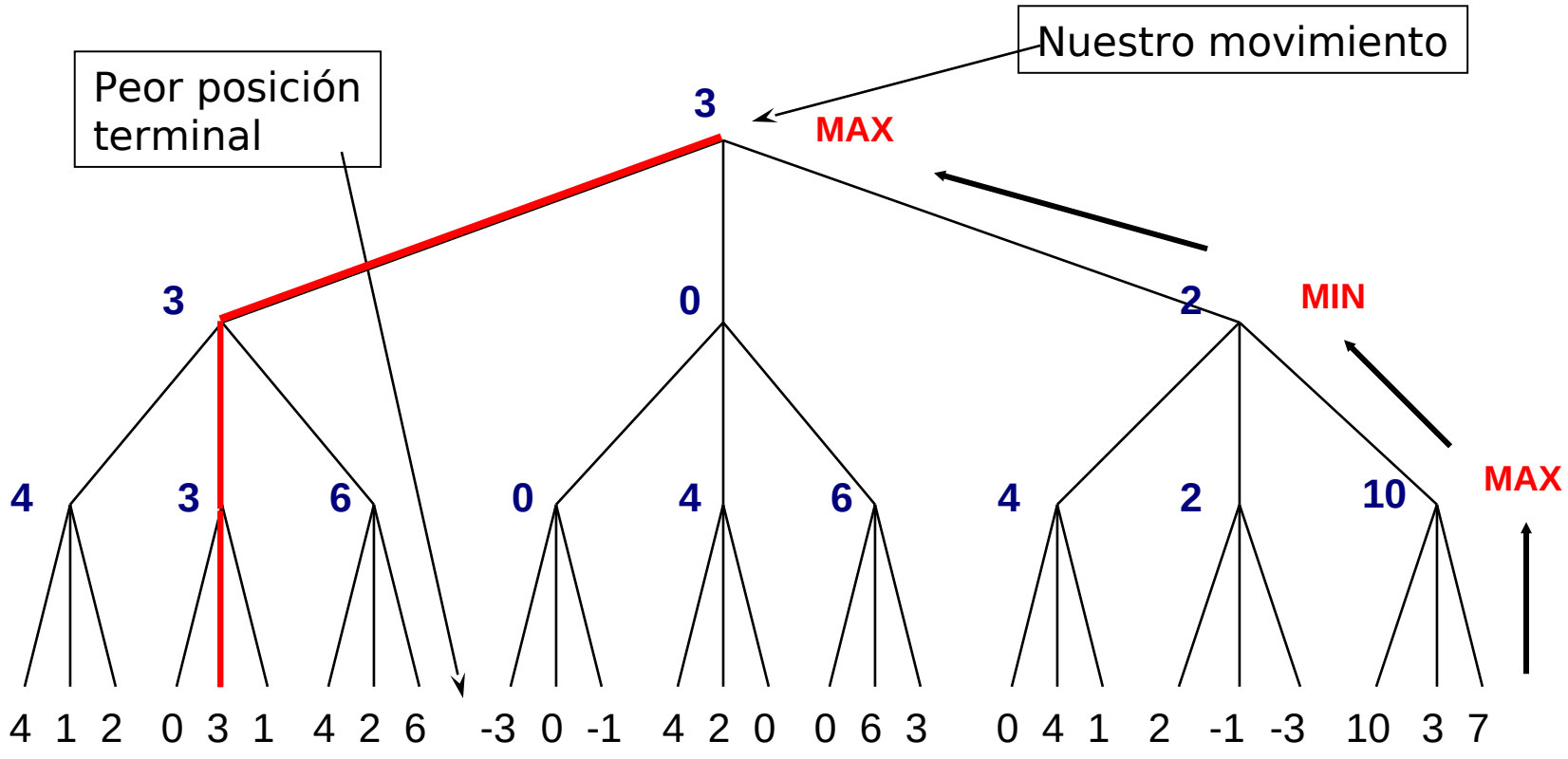
- Aunque algunos nodos tienen valores superiores a 3, nunca serán alcanzados ya que los contra-movimientos de B lo impedirán.
- Por ejemplo, si A se mueve a P(21) esperando ganar el juego en P(31), A quedaría estupefacto cuando viera que B hacia el contramovimiento a P(32), resultando una perdida para A.



Dada la función de evaluación de A y el árbol del juego de la figura, el mejor movimiento de A es hacia la configuración P(22).

Habiendo hecho este movimiento, aún podría ser que el juego no alcanzara la configuración P(52) ya que, en general, B podría usar una función de evaluación distinta, lo que podría proporcionar valores diferentes a las distintas configuraciones.

Otro ejemplo de juego



Desarrollo del juego

- Este procedimiento no es demasiado interesante en juegos con árboles pequeños. Donde realmente es potente es en juegos, como el ajedrez, en los que el correspondiente árbol es tan grande que es imposible generarlo completamente.
- Se estima que el árbol del ajedrez tiene mas de 10^{100} nodos por lo que, aunque usáramos un ordenador potentísimo, podríamos tardar mas 10 años en generarlo por completo.
- Así en juegos de características similares, la decisión de como efectuar un movimiento se realiza mediante la exploración de unos cuantos niveles.
- La función de evaluación E(X) se usa para obtener los valores de los nodos hoja del subárbol generado

Construcción de un algoritmo

- Intentemos escribir un algoritmo que pueda usar A para calcular V(X).
- El procedimiento para calcular V(X) puede usarse también para determinar el siguiente movimiento que debería hacer A.
- Puede obtenerse un procedimiento recursivo para calcular V(X) usando minimax si disponemos la definición de minimax en la siguiente forma

$$V'(X) = e(X) \quad \text{si X es una hoja del subárbol generado}$$
$$= \text{Max } \{-V'[C(i)],\ 1 \leq i \leq d\} \quad \text{si X no es una hoja del subárbol generado y los C(i), son los hijos de X}$$

- donde e(X) = E(X) si X es una posición desde la que A se tiene que mover, pero e(X) = -E(X) en caso contrario.
- Esta ecuación calcula V'(X) = V(X) como la ecuación anterior.

Algoritmo

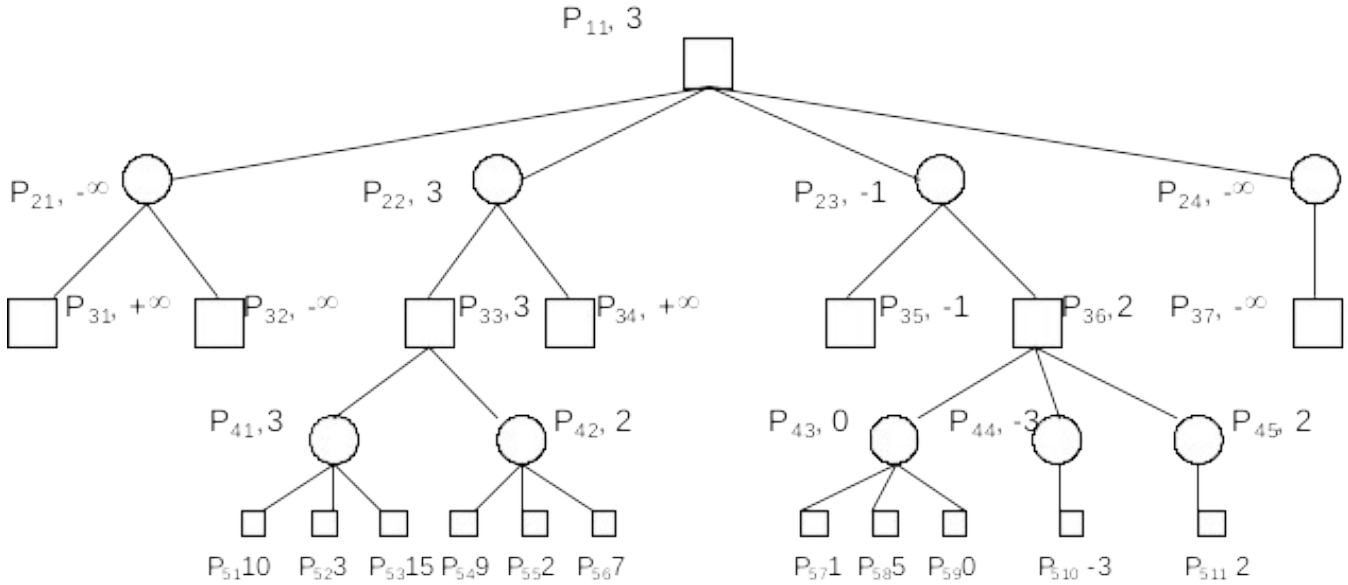
- El procedimiento recursivo para evaluar V'(X) basado en la anterior formula es el siguiente,

Procedimiento VE(X,l)

{Calcula V'(X) realizando a lo sumo l movimientos hacia adelante. e(X) es la función de evaluación de A. Se supone por conveniencia que partiendo de una configuración X, los movimientos legales del juego solo permiten transiciones a las configuraciones C(1),...,C(d) si X no es una configuración terminal}

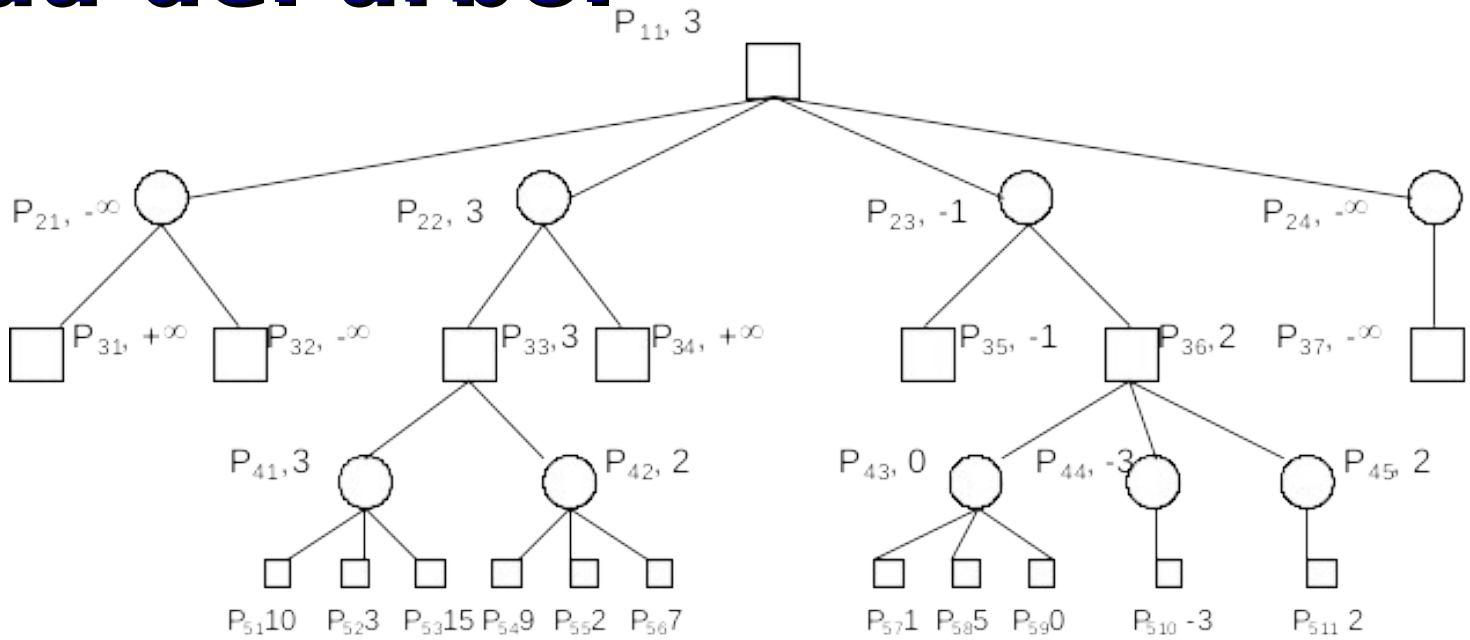
```
Begin
    If X es terminal o l = 0 Then return (e(X))
       ans = -VE(C(1), l-1)
       For i = 2 to d do
     ans = Max(ans, -VE(C(i), l-1))
       Return (ans)
end
```

Funcionamiento del algoritmo



- Sobre el juego de la figura, una llamada a este algoritmo con X = P(11) y l = 4 produciría la generación completa del árbol del juego.
- Los valores de las diversas configuraciones se determinarían en el orden P(31), P(32), P(21), P(51), P(52), P(53), P(41), P(54), P(55), P(56), P(42), ..., P(37), P(24) y P(11).

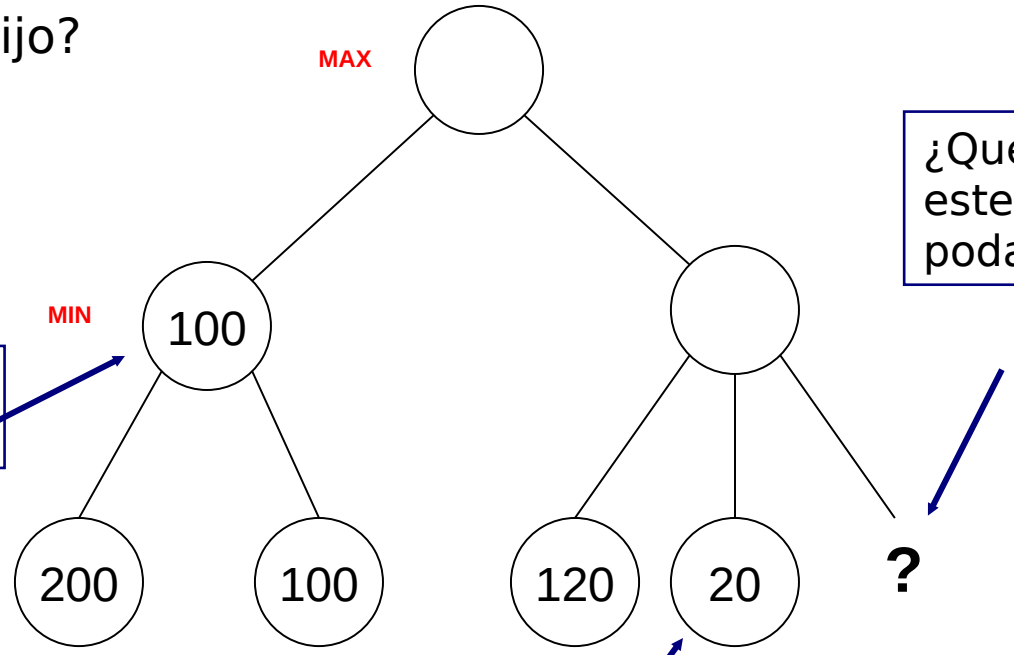
Poda del árbol



- Después de haber calculado V[P(41)], se sabe que V[P(33)] es al menos V[P(41)] = 3.
- A continuación, cuando se ve que V[P(55)] es 2, sabemos que V[P(42)] es a lo mas igual a 2. Como P(33) es una posición Max, V[P(42)] no puede afectar a V[P(33)].
- Independientemente de los valores de los restantes hijos de P(42), el valor de P(33) no se determina por V[P(42)] ya que V[P(42)] no puede ser mas que V[P(41)].

La poda aún mas clara

Deberiamos evaluar el siguiente hijo?



¿Que valor debe tener este nodo para que lo podamos seleccionar?

El primer hijo "min" es 100

> 100

Pero el "min" seleccionará el 20!!!

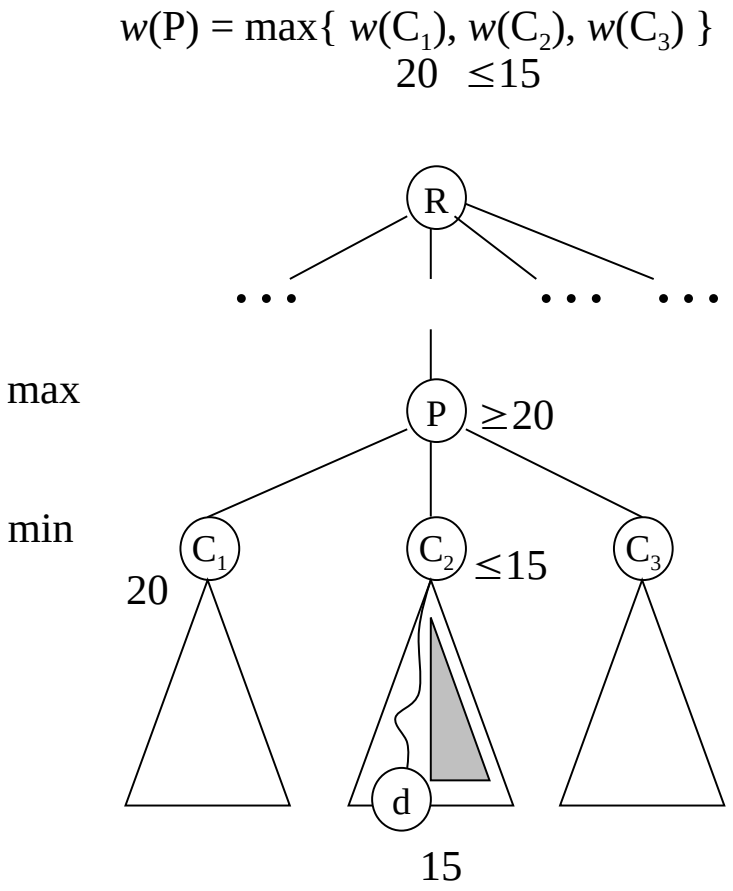
Poda alfa

- Def'n : El valor alfa de una nodo max. Es el minimo valor posible para ese nodo.

  El max. Valor posible

- Si el valor de un nodo min es $\leq$ el valor alfa de su padre, entonces podemos parar el examen de los restantes hijos de este nodo min.

  Poda alfa !!!



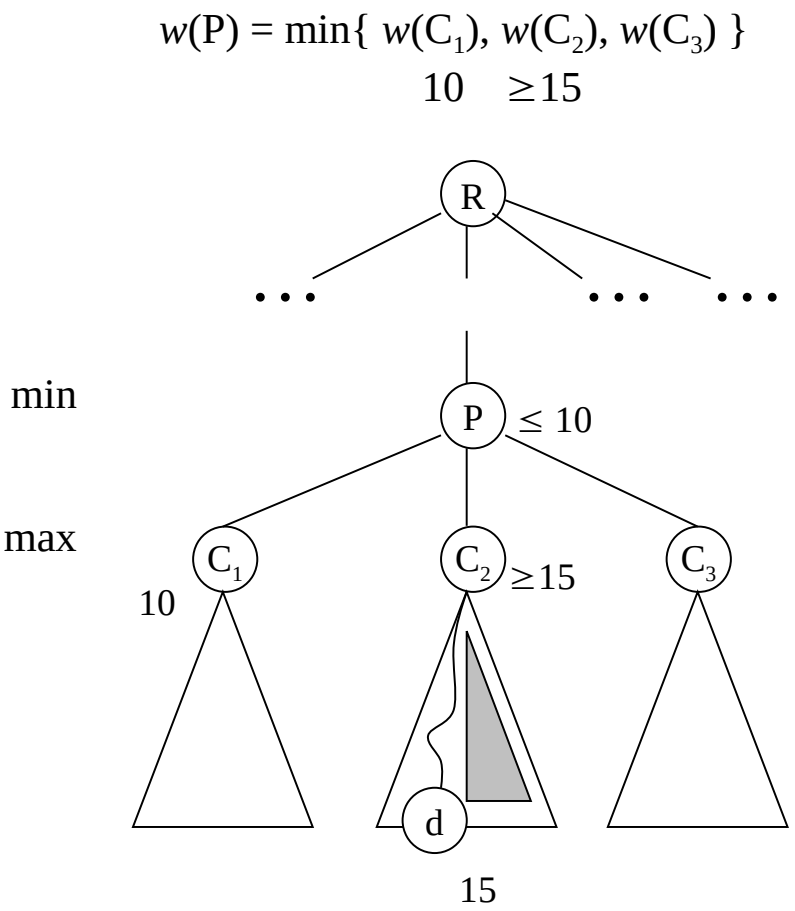
Poda beta

- Def'n : El valor beta de un nodo min. Se define como el maximo Valor posible para ese nodo.

  El min. Valor posible

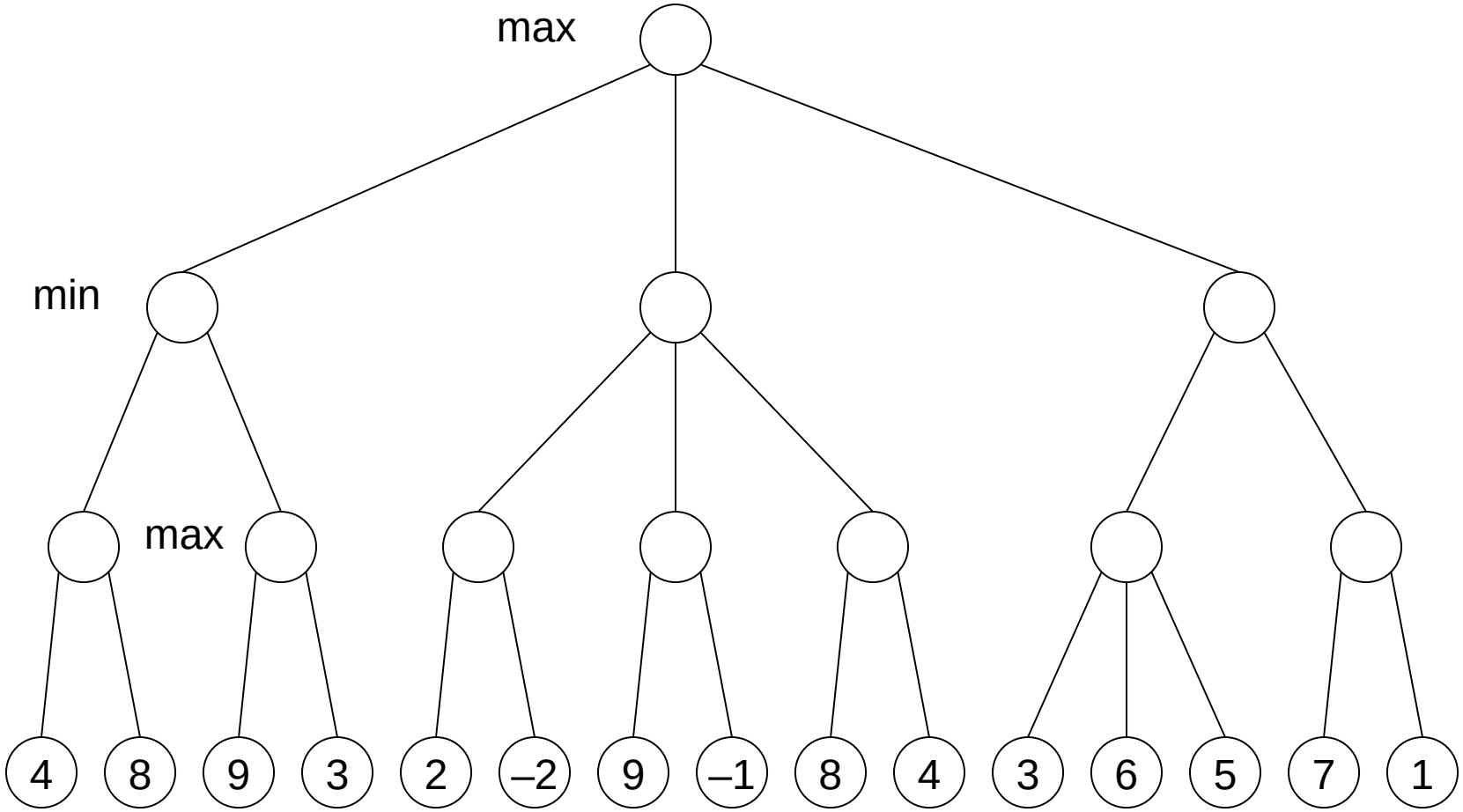
- Si el valor de una nodo max. es ≥ que el valor beta de su padre, entonces podemos parar de examinar los restantes hijos de es nodo max.

  Poda Beta!!!

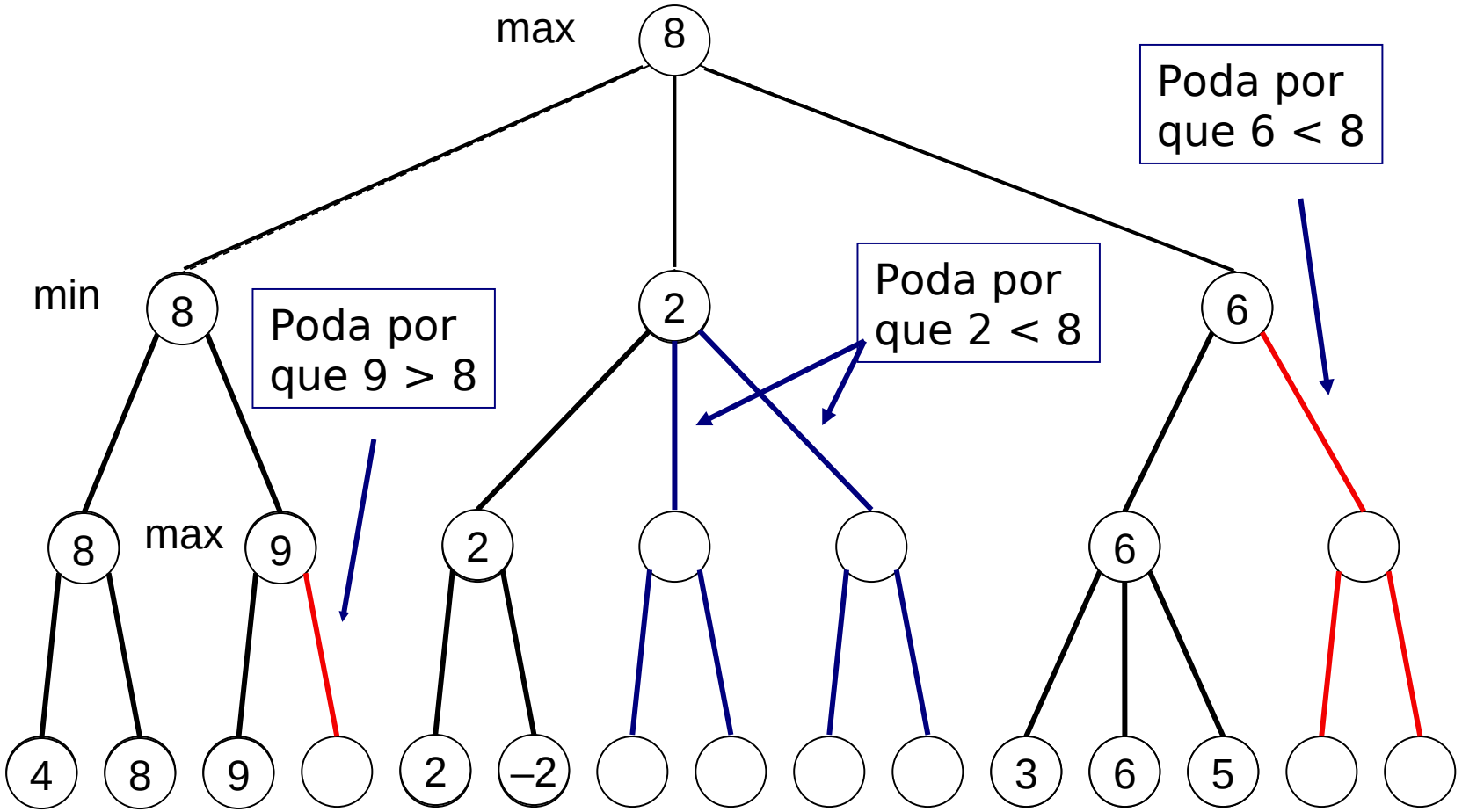


Ejemplo

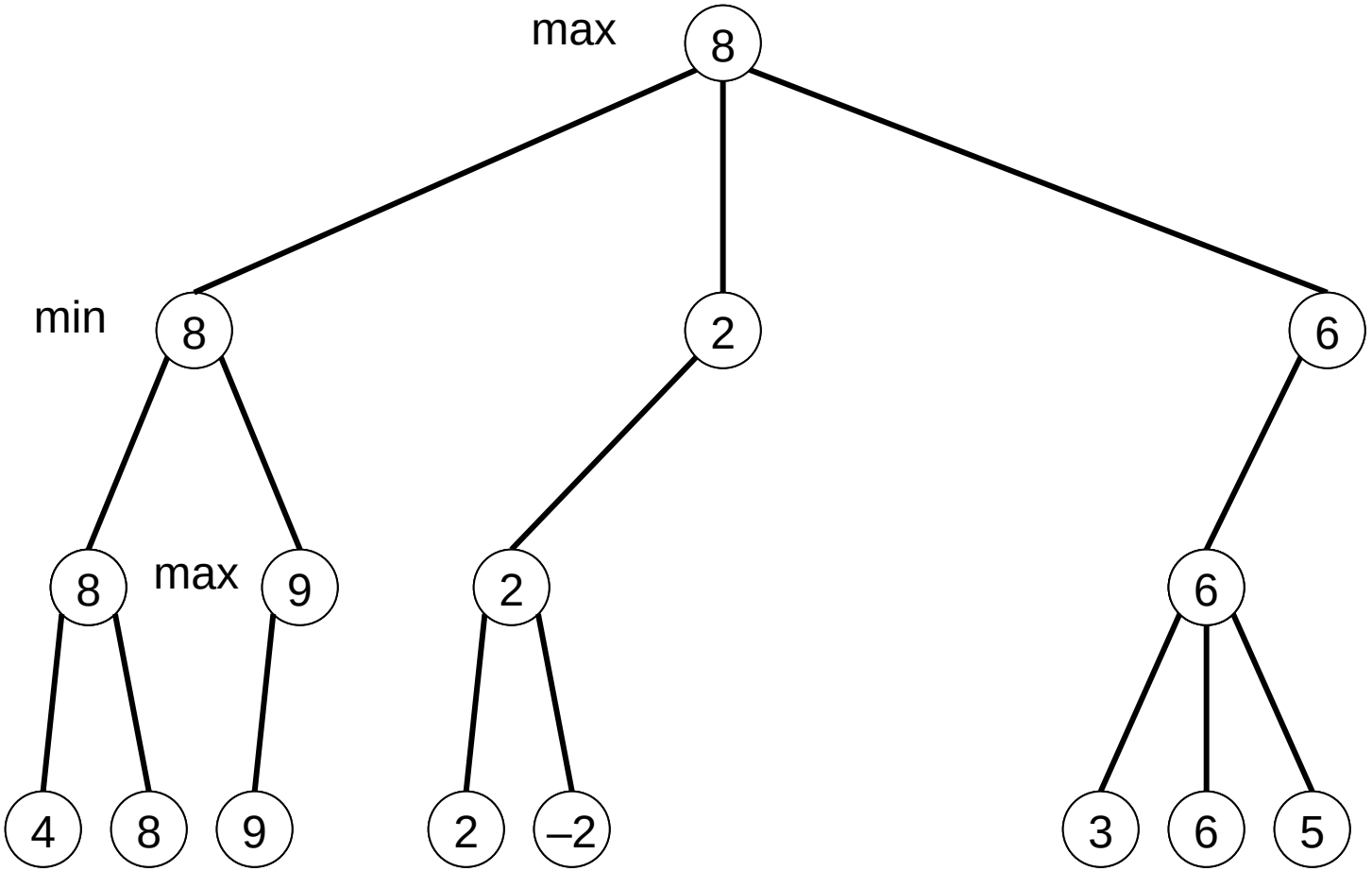
- Arbol Minimax



Poda Alfa-Beta



Generacion real



Poda alfa-beta

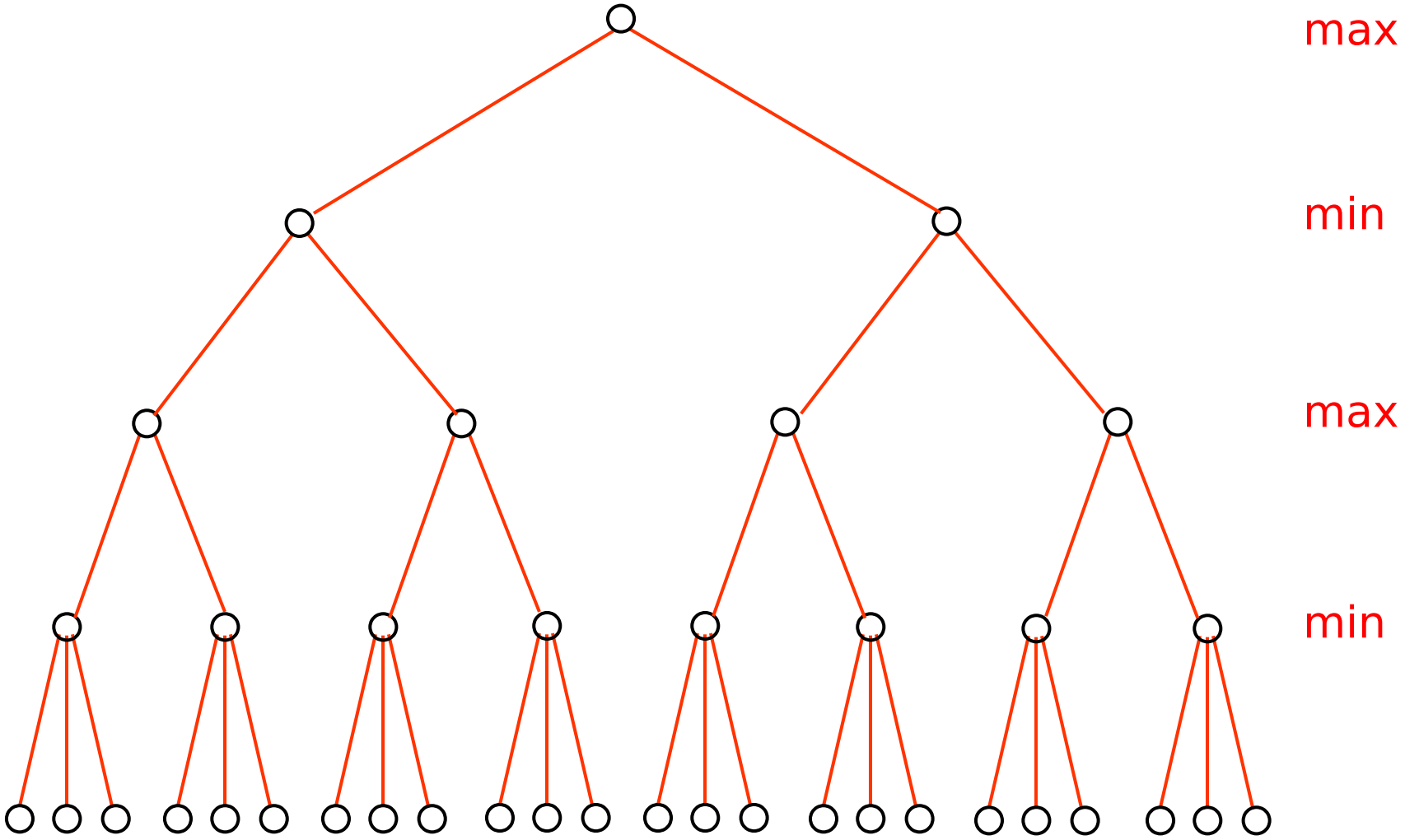
- Las dos reglas anteriores pueden combinarse juntas para dar lo que se denomina Poda Alfa-Beta.
- Para introducir la poda alfa-beta en el algoritmo VE es necesario establecer esta regla en los términos de la ecuación para V'(x), ya que bajo ese esquema todas las posiciones son posiciones max ya que los valores de las posiciones min se multiplicaron por -1.
- La regla de la poda alfa-beta se reduce entonces a lo siguiente:
- Llamemos B-valor al mínimo valor que una cierta posición puede tener. Para cualquier posición X, sea B el B-valor de su padre y D = -B. Entonces si el valor de X es mayor o igual que D, podemos terminar la generación de los restantes hijos de X.
- La incorporación de esta regla en el algoritmo VE es sencilla, y produce el algoritmo VEB siguiente

Poda alfa-beta

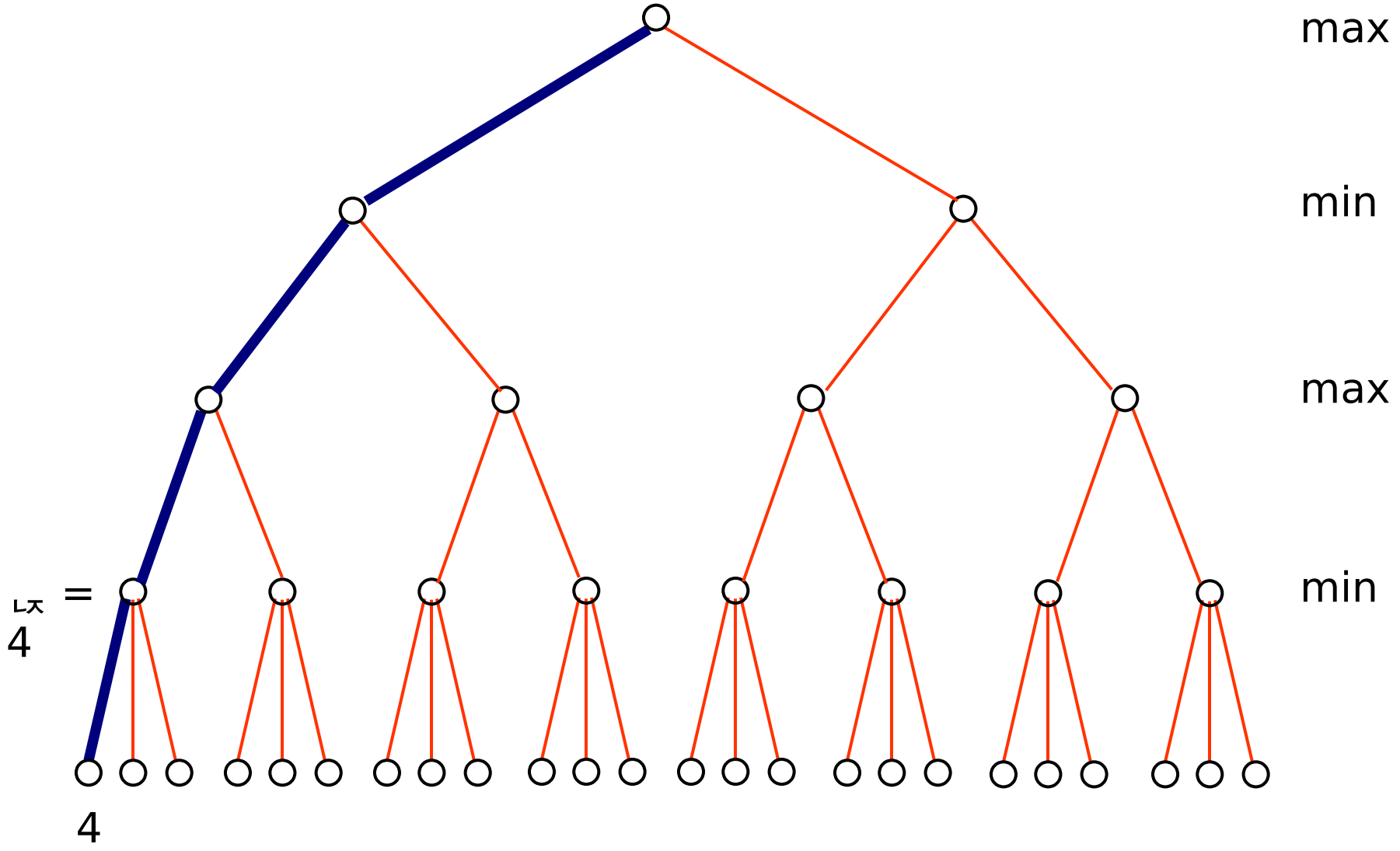
- D es un parámetro adicional D que es el valor negativo del B-valor del padre de X

```
Procedimiento VEB(X,l,D)
  {Determina V'(X)  usando la regla B  y  haciendo  solo
l movimientos hacia adelante. Las restantes hipótesis y
notaciones son  las  mismas que en el algoritmo VE}
 Begin
    Si X es terminal o l = 0 Entonces return (e(X))
    Temp = -VEB(C(1), l-1, ∞)
    Para i = 2 hasta d hacer
              Si Temp ≥ D entonces return (Temp)
         Temp = Max(Temp, -VEB(C(i), l-1, -Temp))
    Return (Temp)
 end
```

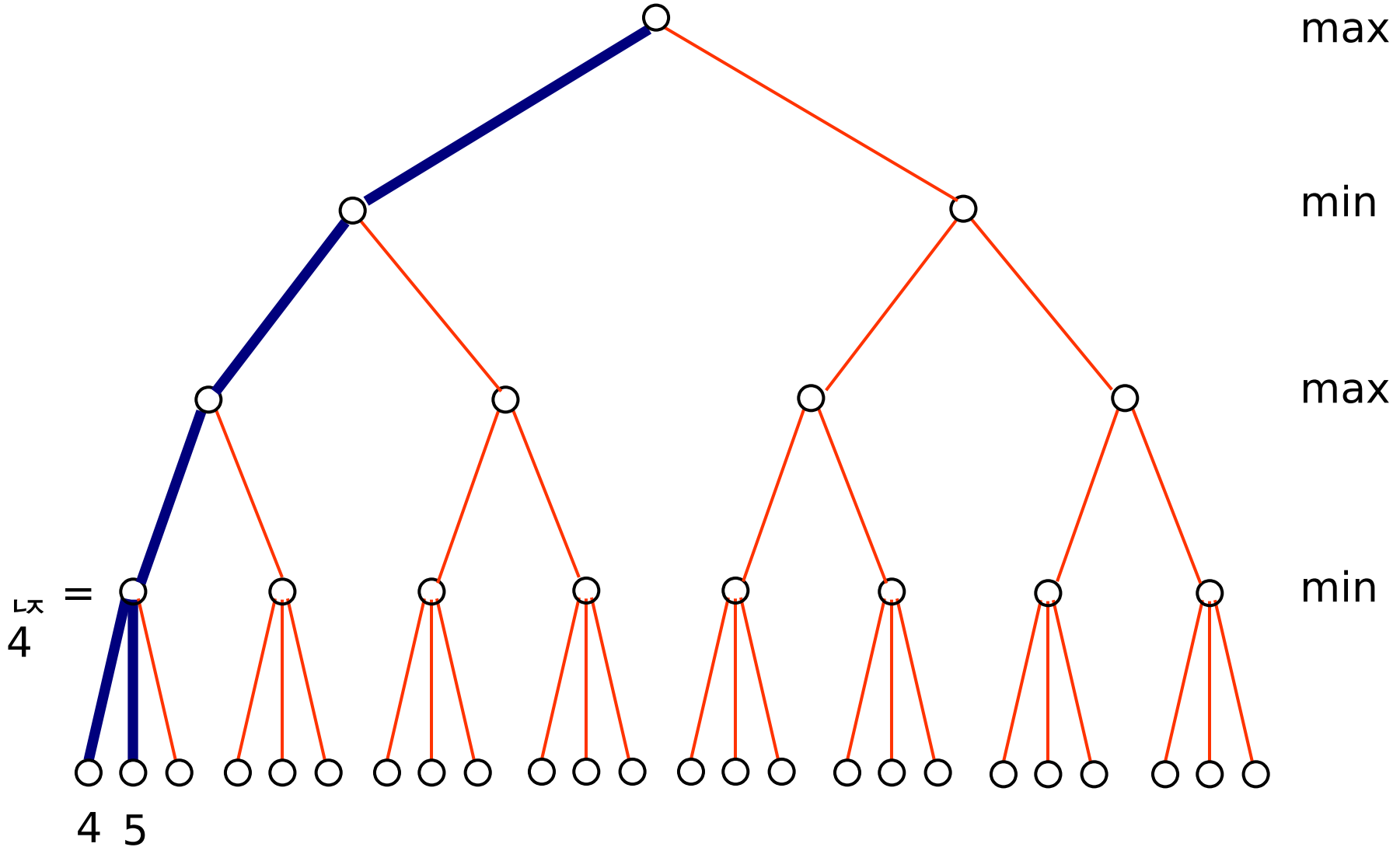
Ejemplo de Poda Alfa-Beta



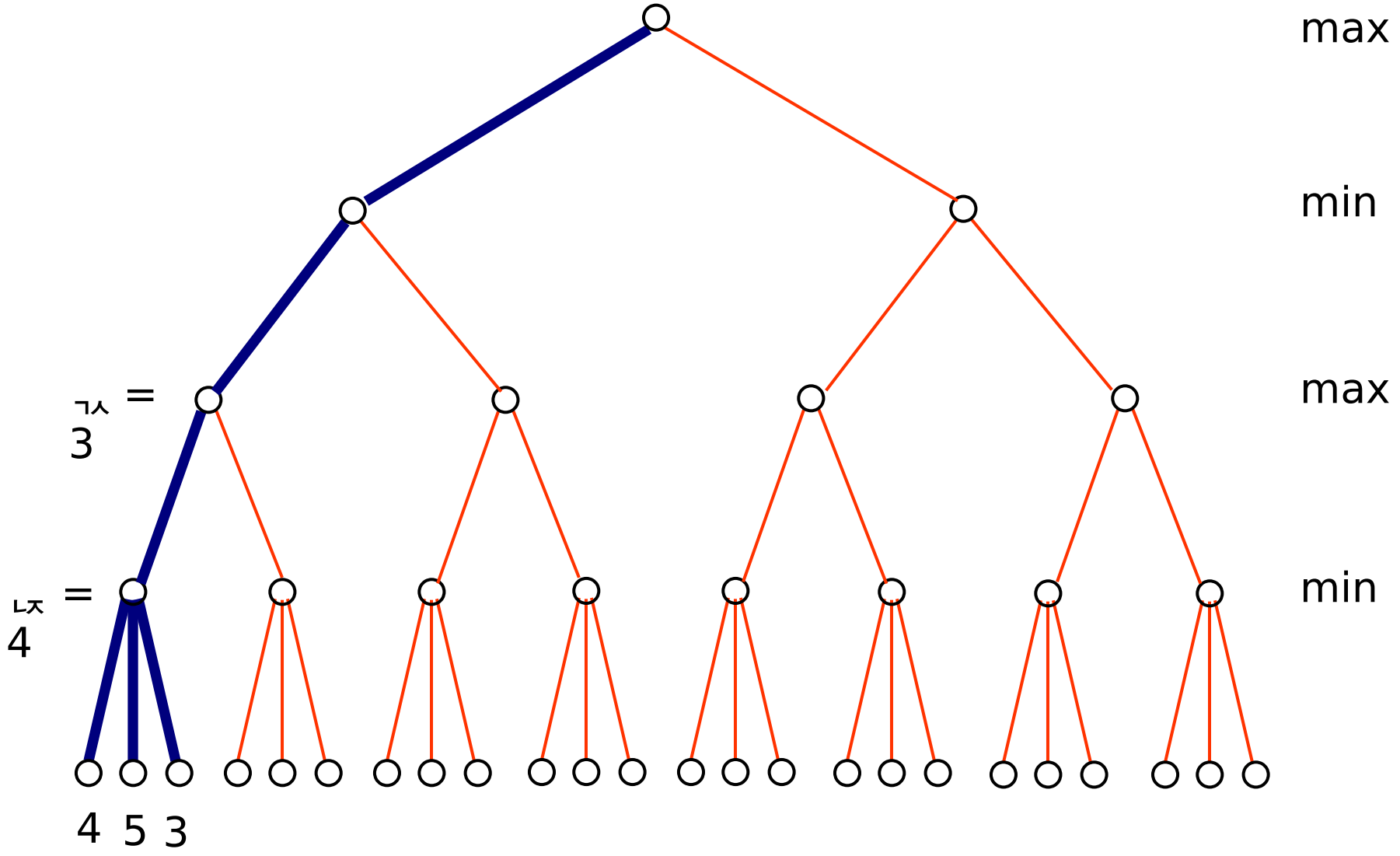
Ejemplo de Poda Alfa-Beta



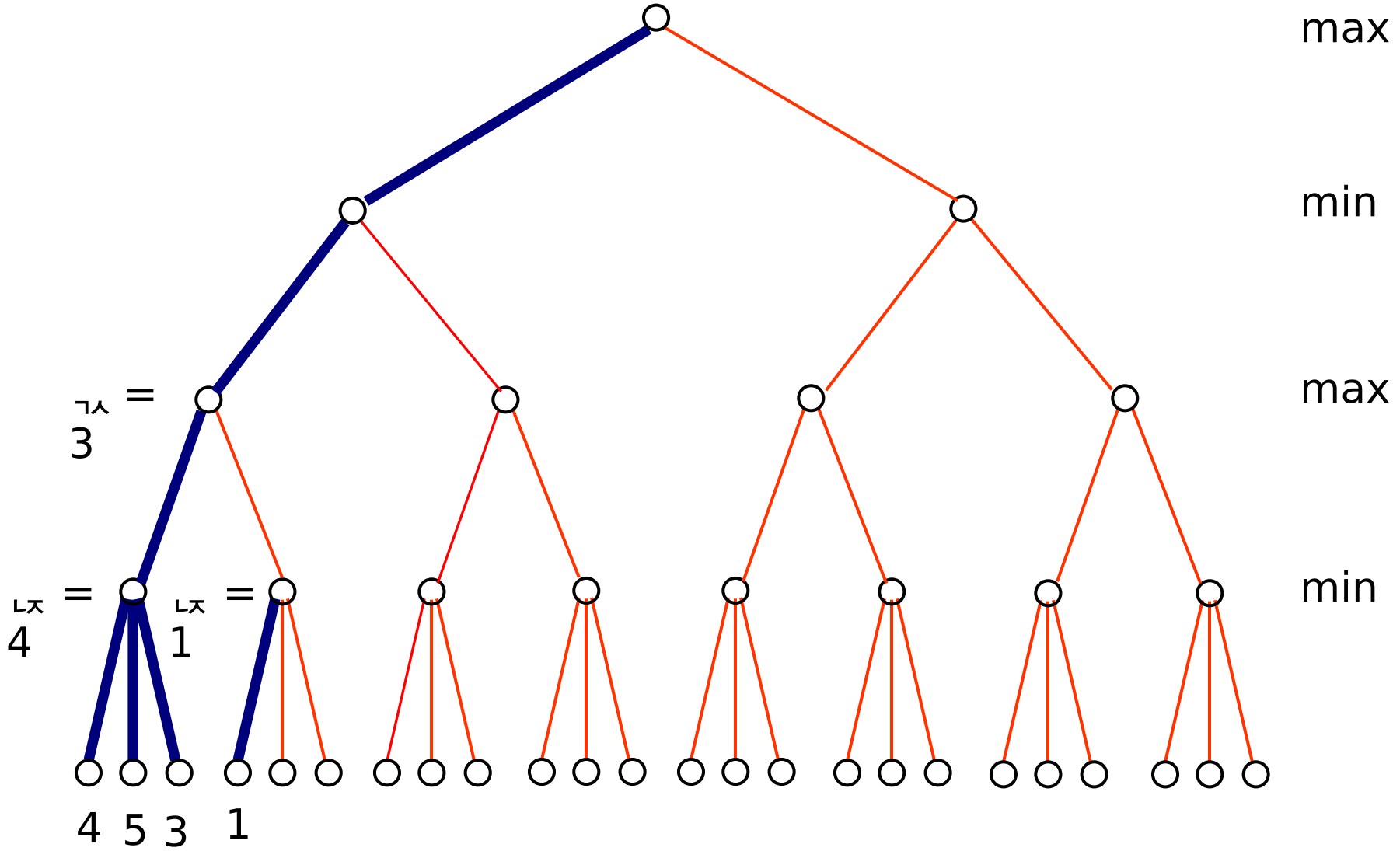
Ejemplo de Poda Alfa-Beta



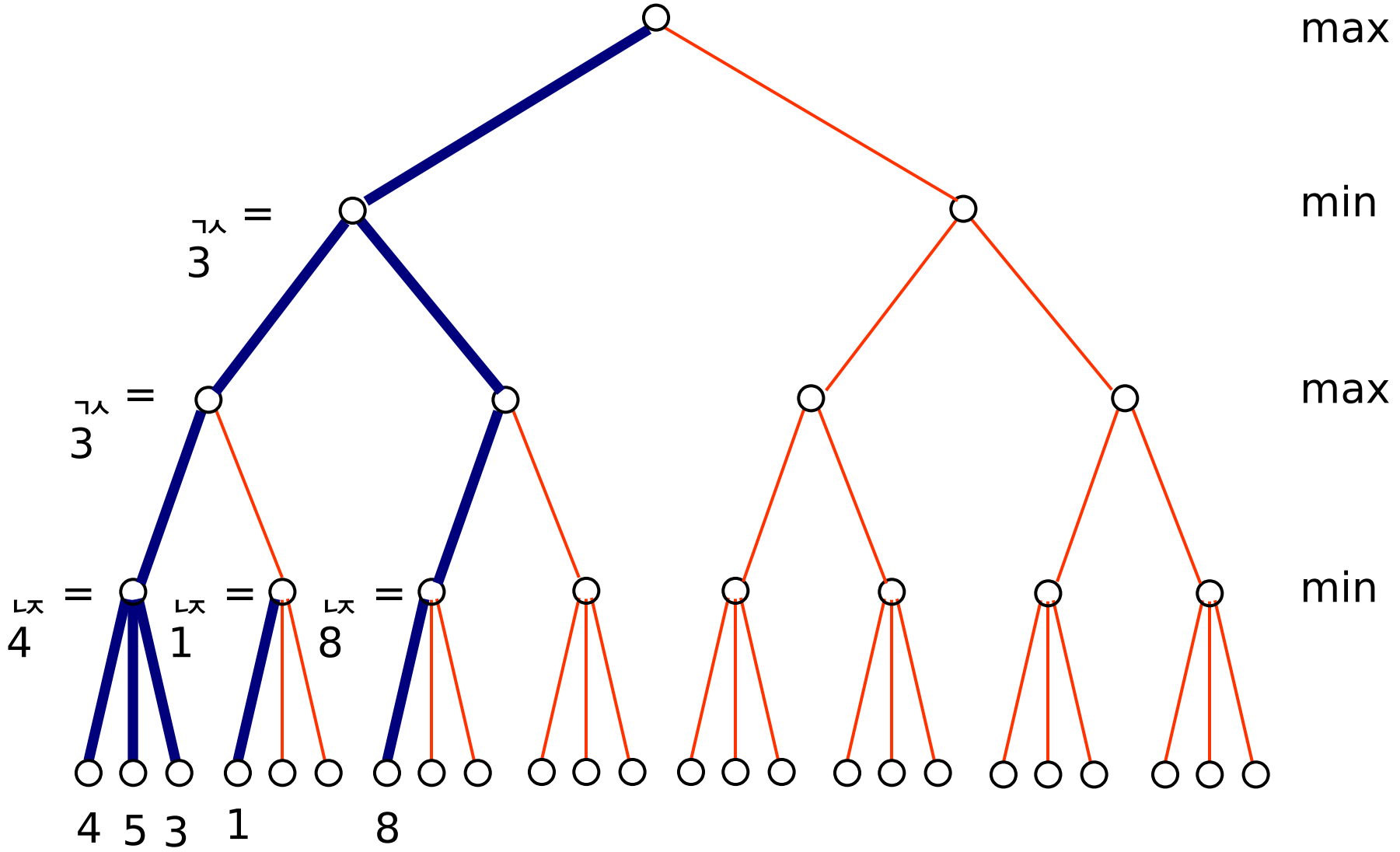
Ejemplo de Poda Alfa-Beta



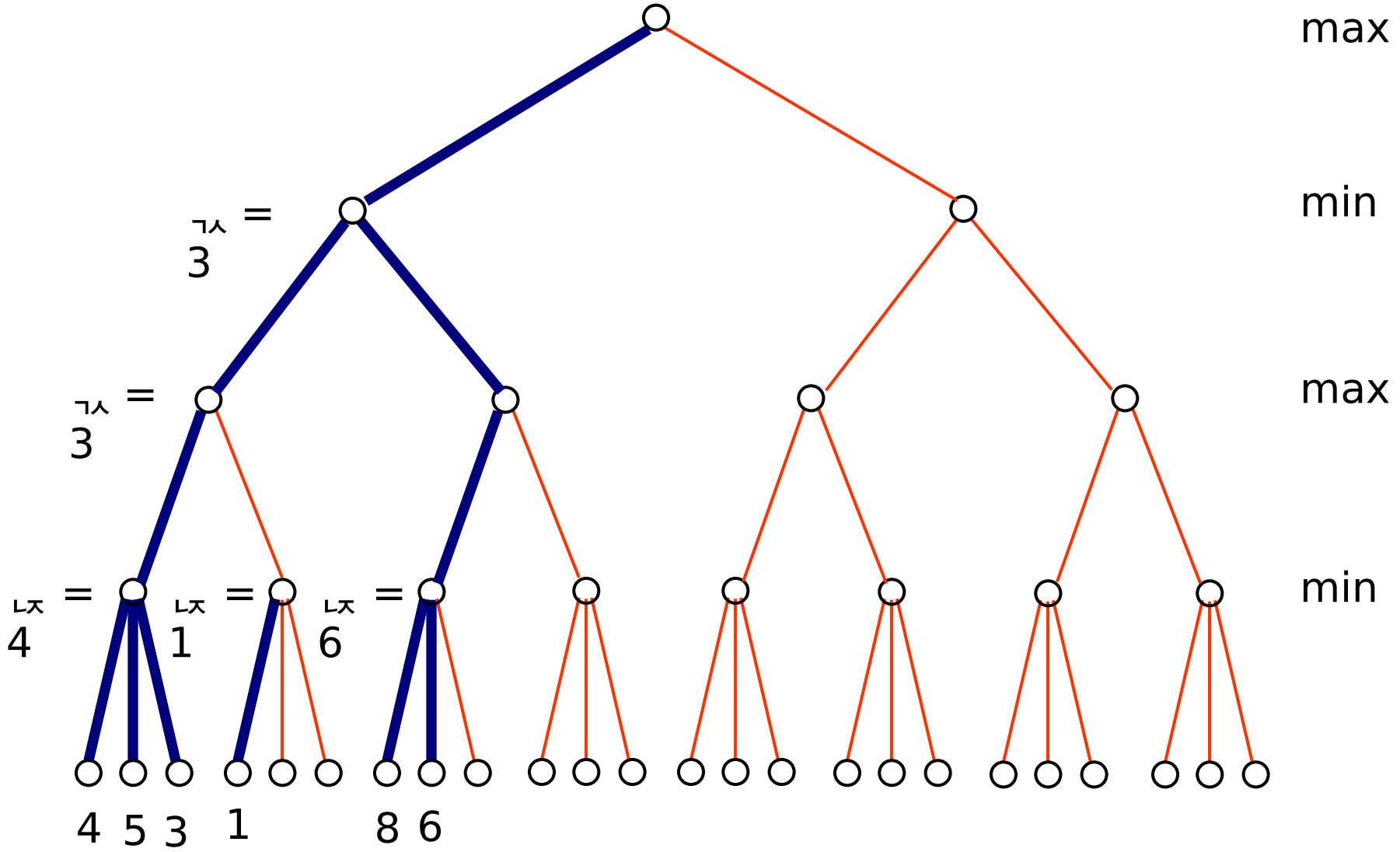
Ejemplo de Poda Alfa-Beta



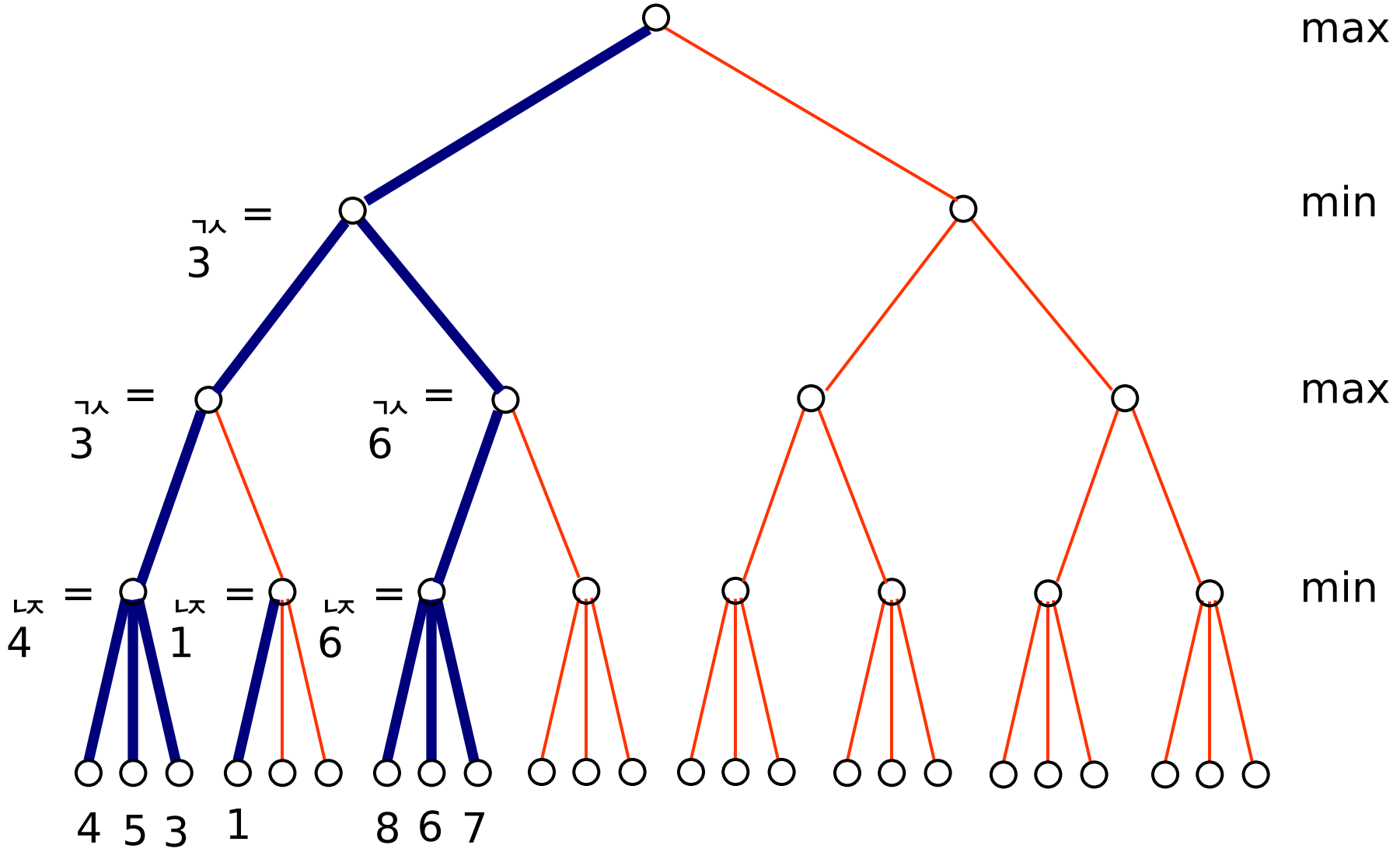
Ejemplo de Poda Alfa-Beta



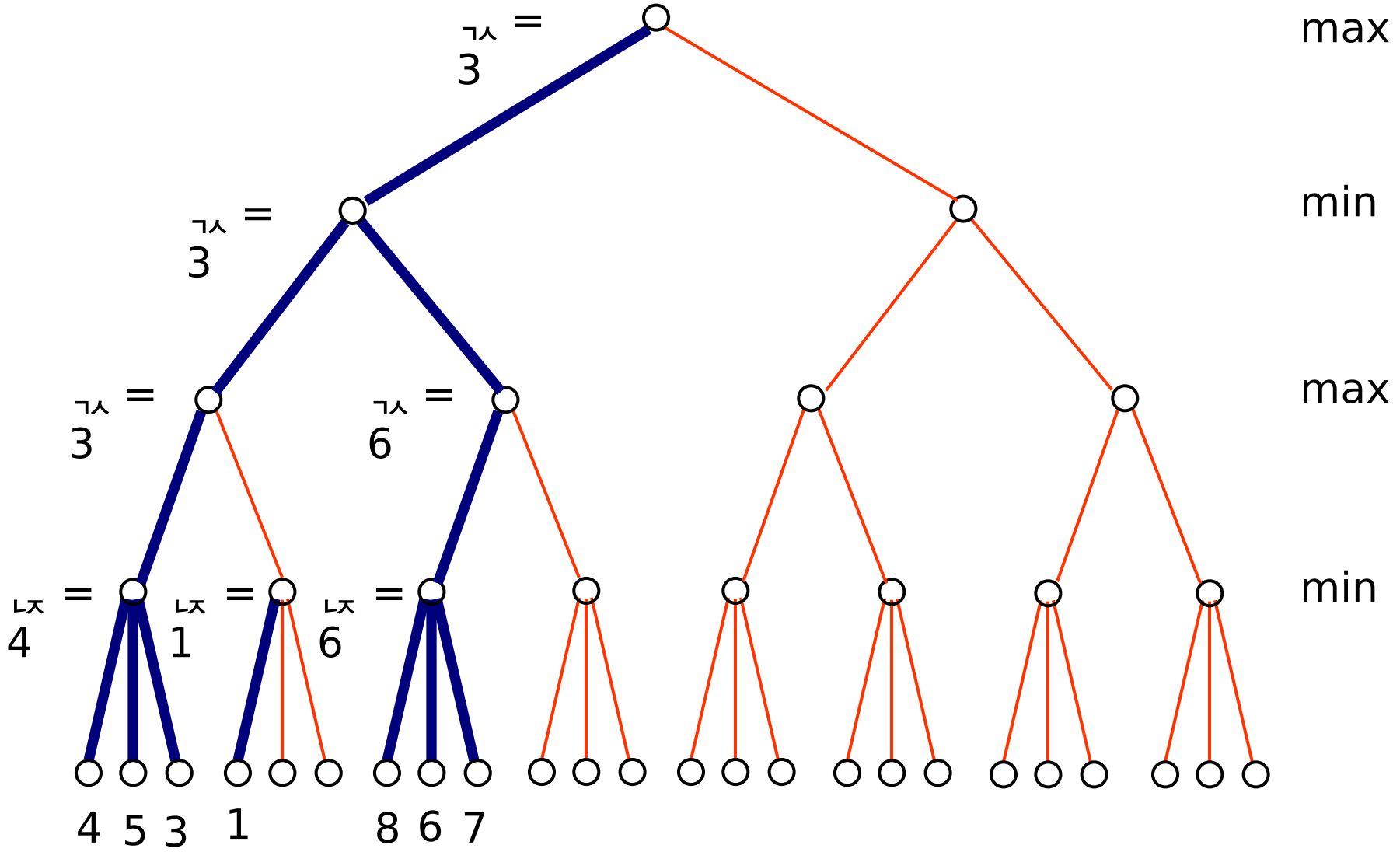
Ejemplo de Poda Alfa-Beta



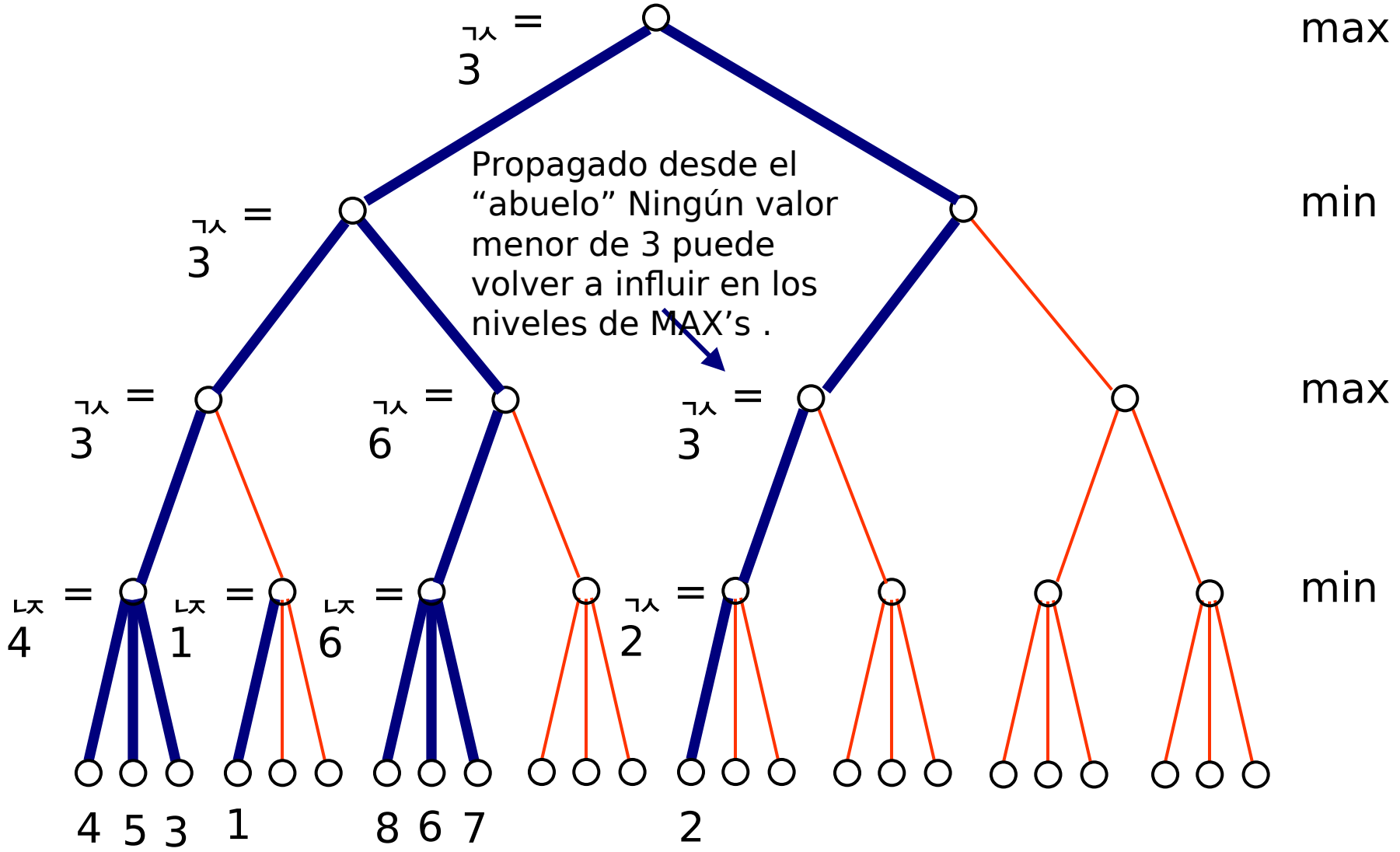
Ejemplo de Poda Alfa-Beta



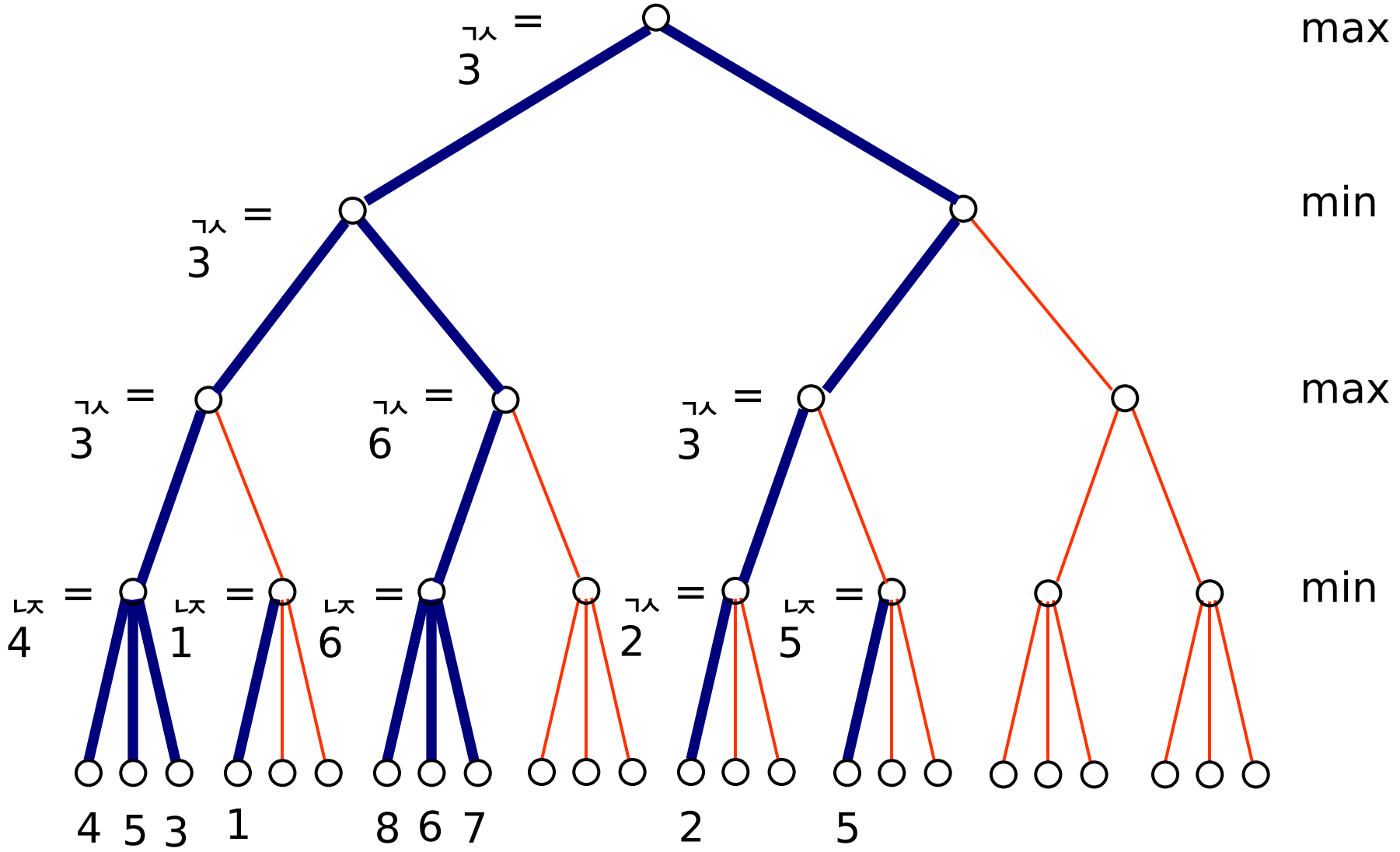
Ejemplo de Poda Alfa-Beta



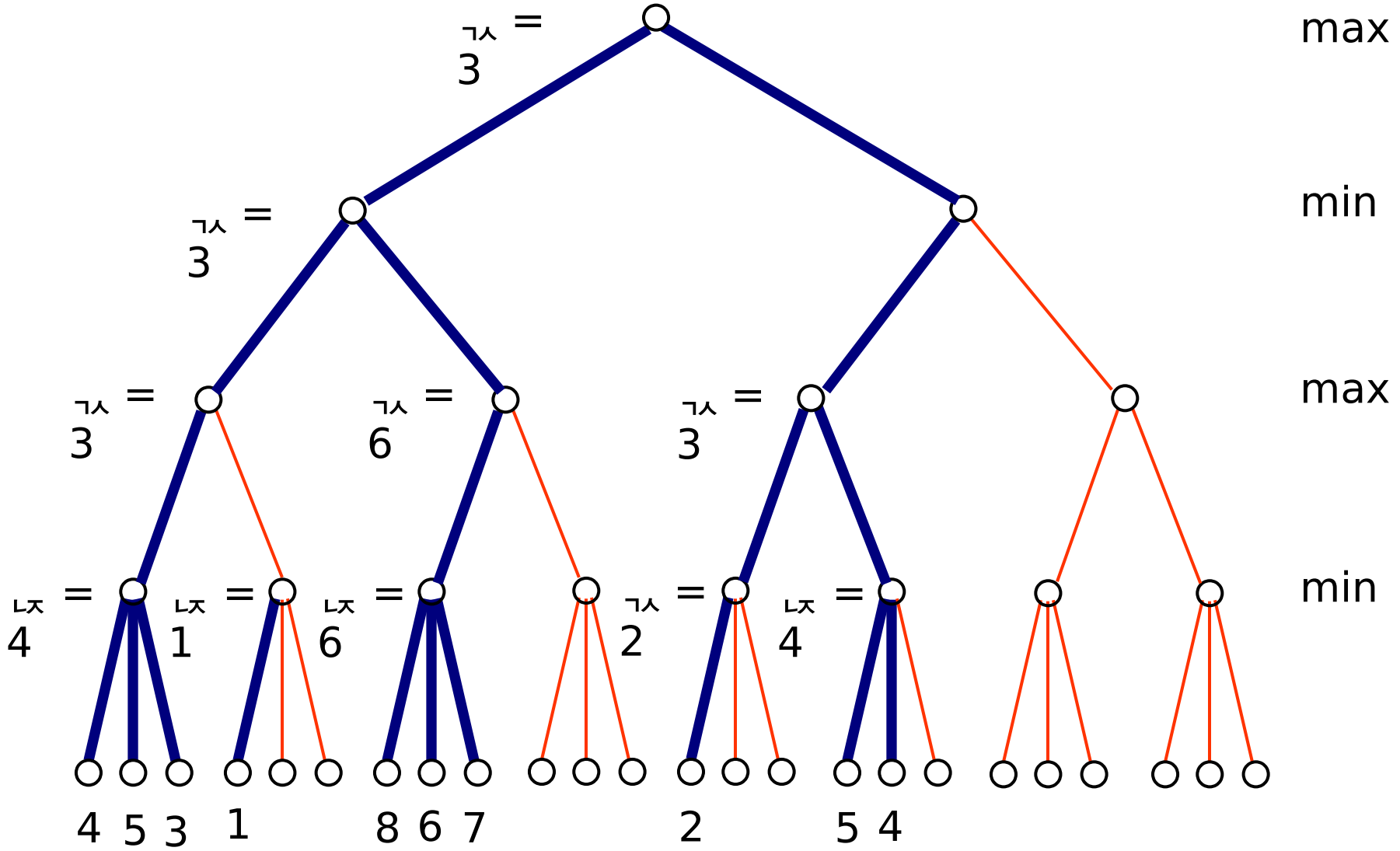
Ejemplo de Poda Alfa-Beta



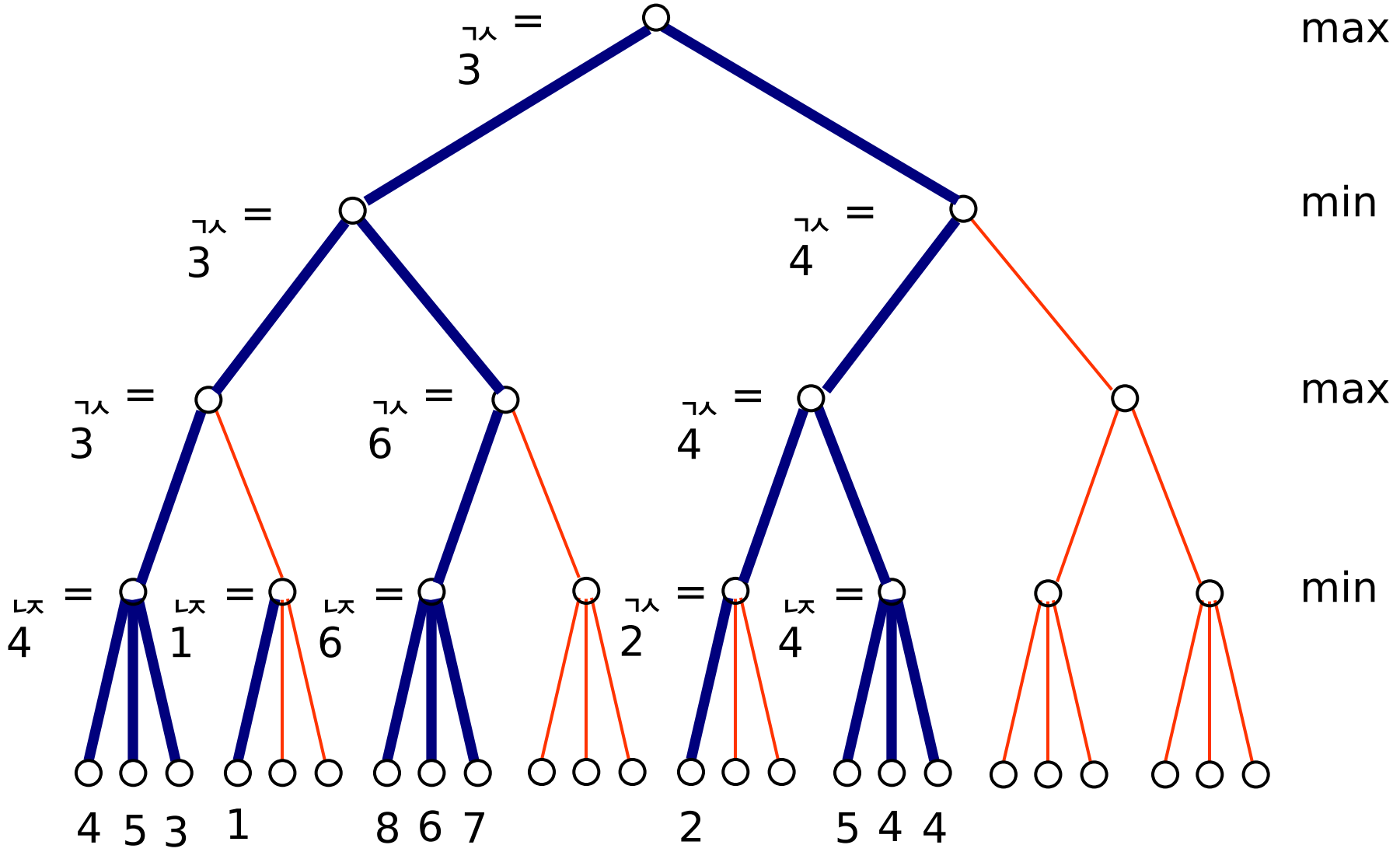
Ejemplo de Poda Alfa-Beta



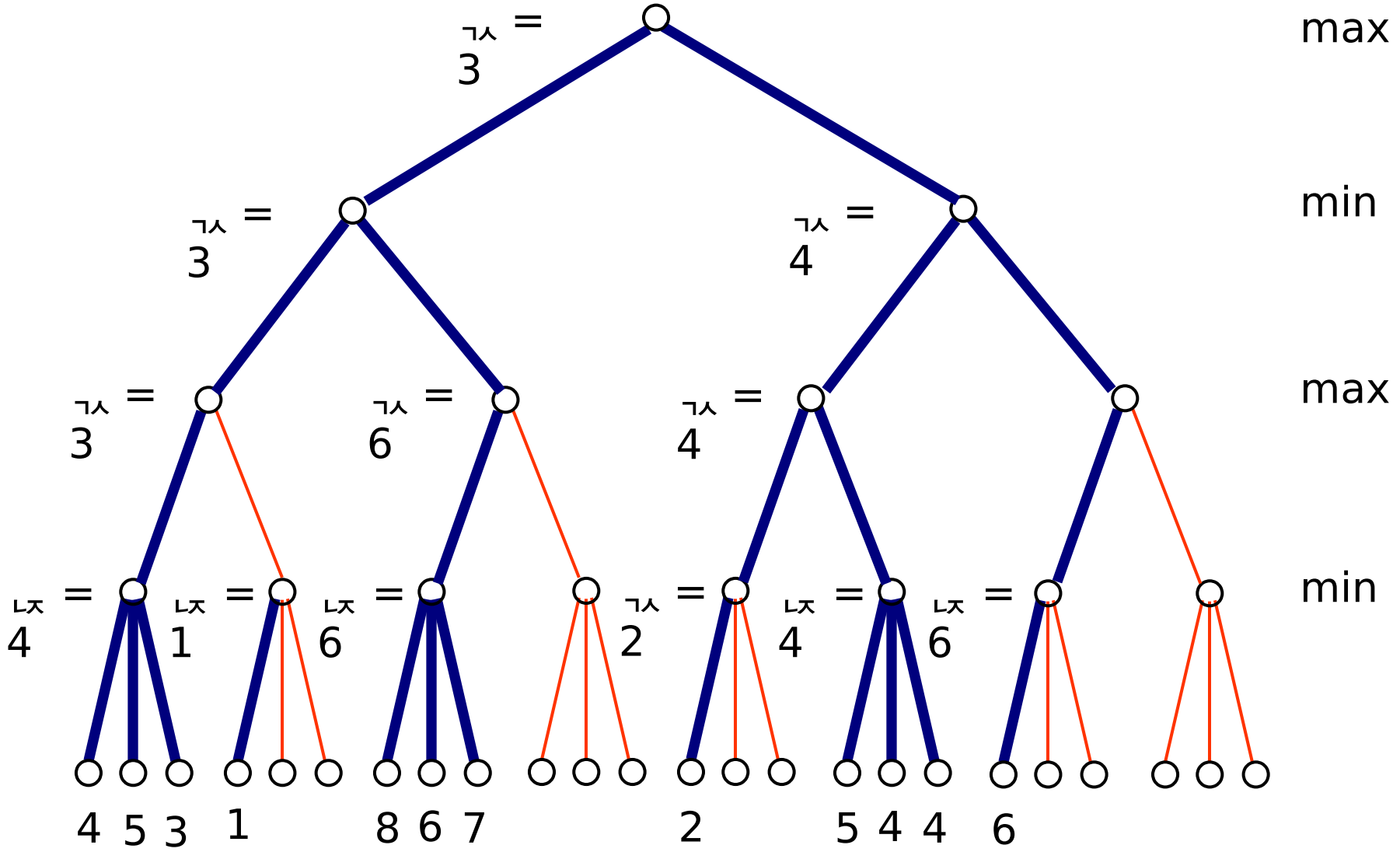
Ejemplo de Poda Alfa-Beta



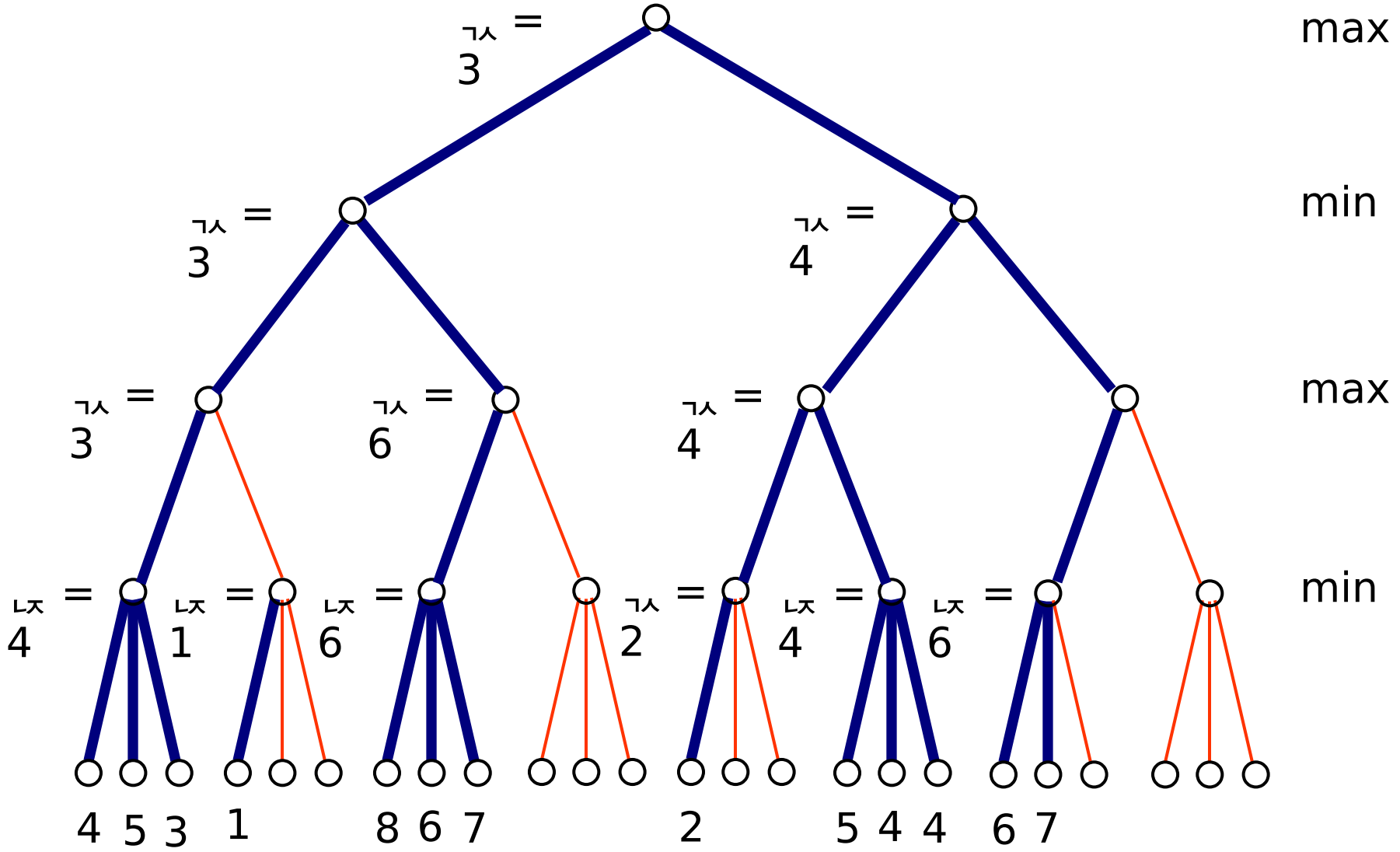
Ejemplo de Poda Alfa-Beta



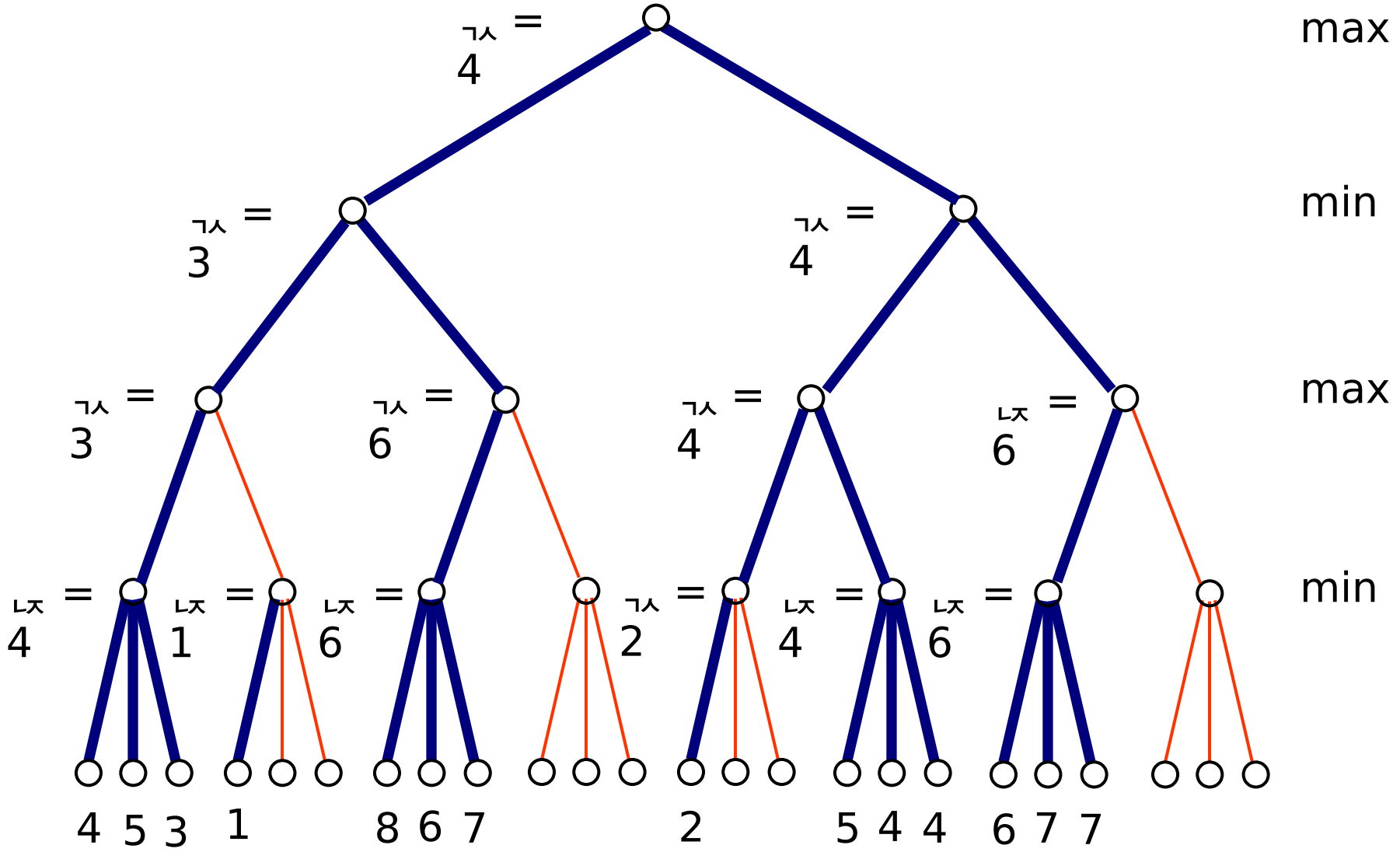
Ejemplo de Poda Alfa-Beta



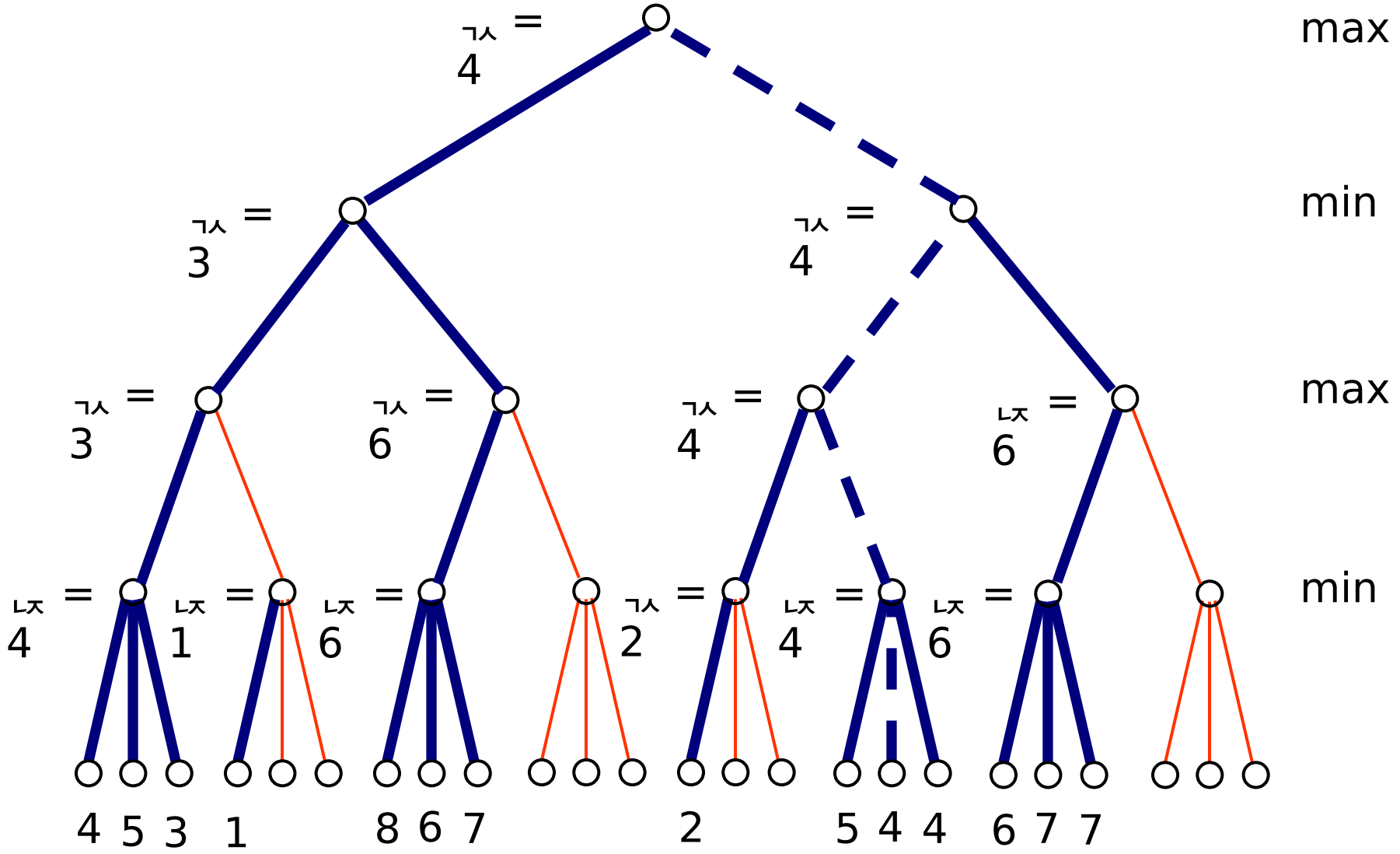
Ejemplo de Poda Alfa-Beta



Ejemplo de Poda Alfa-Beta



Ejemplo de Poda Alfa-Beta



Análisis del procedimiento VEB

- Knuth y Moore han analizado el procedimiento VEB para ciertos tipos de arboles de juegos, y algunos de sus resultados son los siguientes.
- **Definición.** Un árbol de juego uniforme de grado d y altura h es un árbol de juego en el que cualquier nodo en los niveles 1,2, ..., h-1 tiene exactamente d hijos. Además, cualquier nodo en el nivel h es terminal.
- **Definición.** Un árbol de juego uniforme aleatorio es un árbol de juego uniforme en el que los nodos terminales tienen valores aleatorios independientes.

Análisis del procedimiento VEB

- **Teorema.** El numero esperado T(d,h) de posiciones terminales examinadas por el procedimiento alfa-beta VEB), en un árbol de juego uniforme aleatorio de grado d y altura h, es menor que $c(d)r(d)^h$, donde r(d) es el mayor autovalor de la matriz M_d cuyos términos están dados por

$$M_d(i,j) = (\mathbf{C}_{(i-1)+(j-1)/d,\,(i-1)})^{-1/2} \qquad 1 \le i \le d \text{ y } 1 \le j \le d$$

siendo c(d) una constante apropiada.

- **Teorema.** T(d,h) para un árbol de juego uniforme aleatorio de grado d y altura h+1 satisface la igualdad,

$$\lim_{h\to\infty} T(d,h)^{1/h} = r(d)$$

donde

$$c_1 d (\log d)^{-1} \le r(d) \le c_2 d (\log d)^{-1}$$

para ciertas constantes positivas c_1 y c_2.

Ejemplo: ¿Cómo jugar el juego de NIM?

- Recordemos que,
  - Una configuración terminal es aquella que representa una situación de ganar, de perder o de empate. Todas las demás configuraciones son no terminales.
  - En el juego de Nim solo hay una configuración terminal: Cuando no quedan palillos en el panel.
  - Esta configuración es de perdida para A, si B hizo el último movimiento o viceversa.
  - Este es un juego para el que existe una forma óptima de jugar: Nunca se pierde
  - El juego de Nim es importante porque es un juego de **Información Perfecta**, porque es un juego equitativo y porque cualquier otro juego equitativo es equivalente a uno de Nim.

El juego de NIM en general

- Se disponen varios montones de palillos
- Representemos la configuración de los montones por una sucesión monótona de enteros

  * * * * * 5

  * * * 3 -> (1,3,5)

  * 1
- En su turno, un jugador puede quitar cualquier número de palillos del montón que escoja, asi (1,3,5) podría pasar a ser (1,3,1) si el jugador quita cuatro palillos de uno de los montones

  * 1

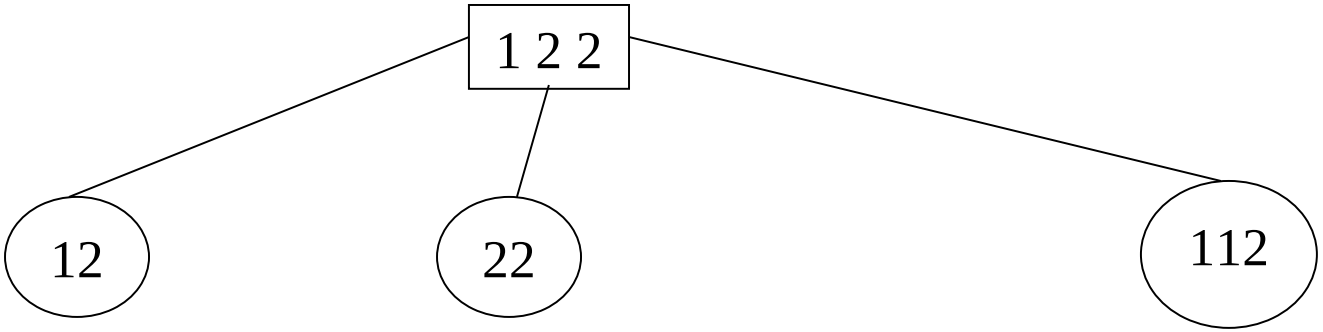
  * * * 3 -> (1,3,1)

  * 1
- El jugador que coge el último palillo pierde

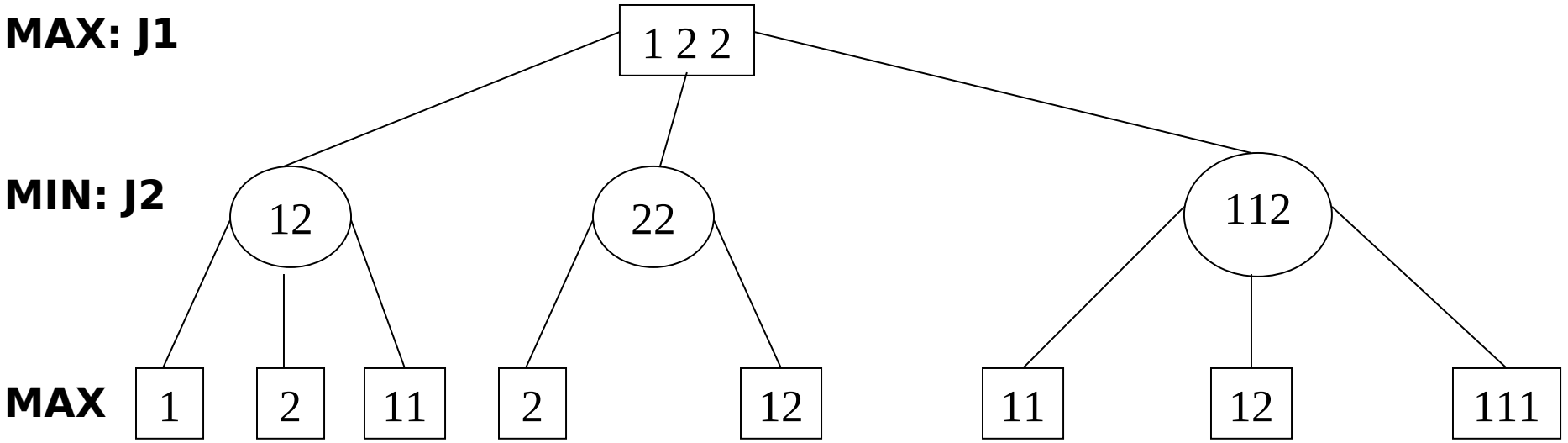
El juego de NIM: Arbol del juego (1,2,2)

MAX: J1

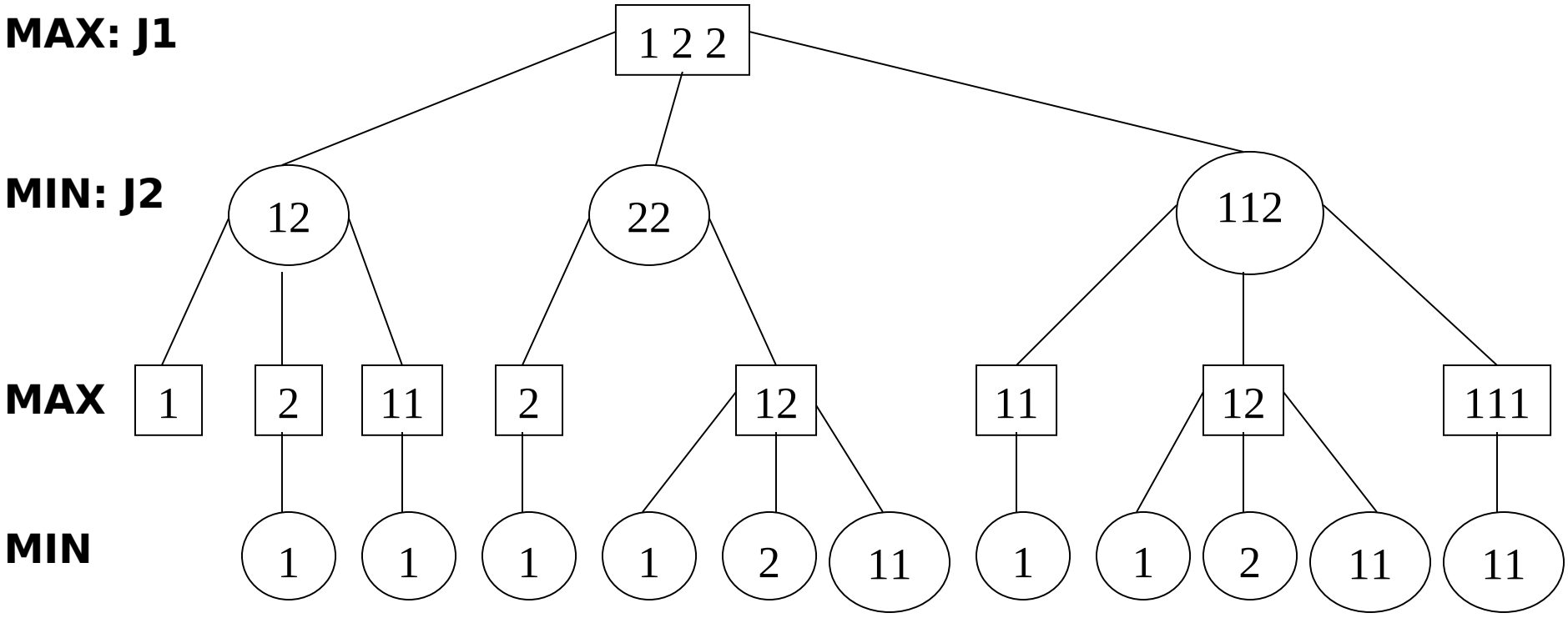
MIN: J2



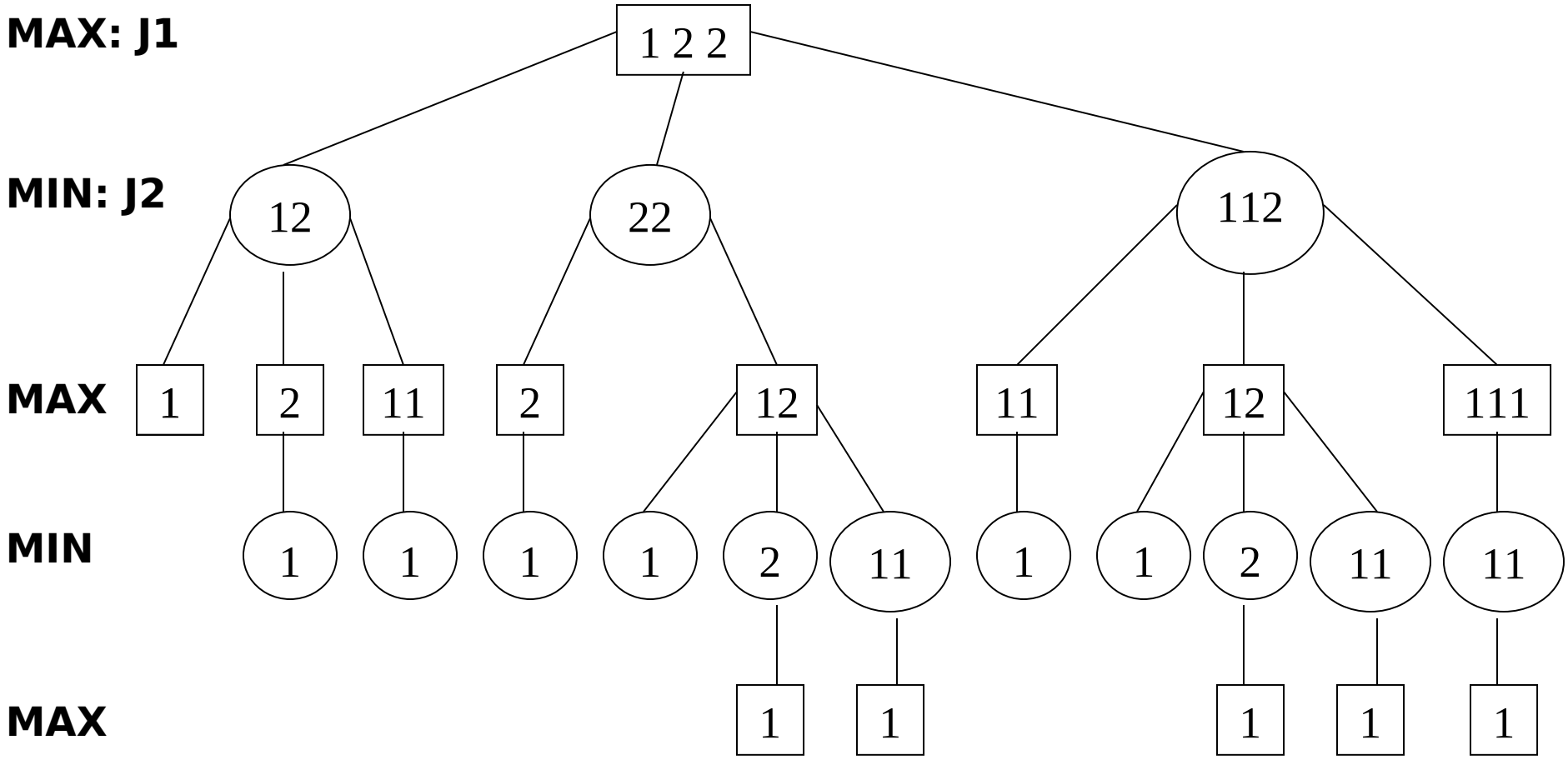
El juego de NIM: Arbol del juego (1,2,2)



El juego de NIM: Arbol del juego (1,2,2)

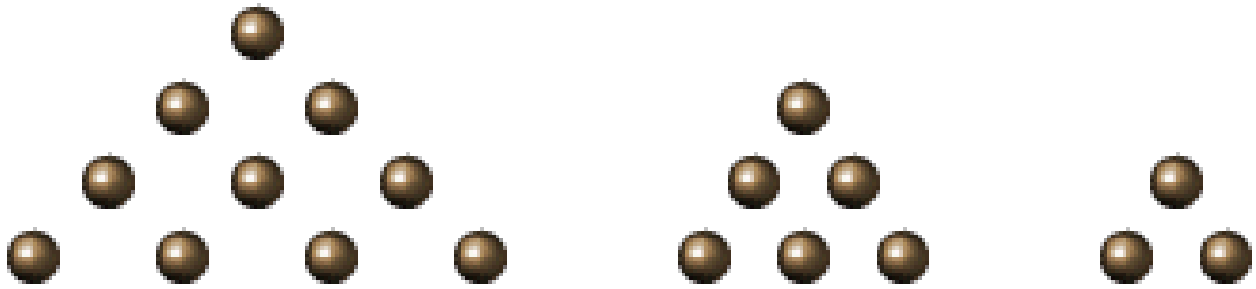


El juego de NIM: Arbol del juego (1,2,2)



¿Cómo ganar al juego de NIM?

Para construir una estrategia ganadora en este juego se usa una técnica llamada Suma de Nim



¿Cómo ganar al juego de NIM?

Suma de Nim:

La suma de Nim de dos enteros no negativos es su suma en base 2.

Base 2:

Cualquier entero puede escribirse como una serie de 1’s y 0’s de la forma:

$$x = x_m 2^m + x_{m-1} 2^{m-1} + \ldots + x_1 2 + x_0$$

Por ejemplo

$$\begin{aligned} 27 &= 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 \\ &= 1\times 16 + 1\times 8 + 0\times 4 + 1\times 2 + 1\times 1 \\ &= (11011)_{base\,2} \end{aligned}$$

¿Cómo ganar al juego de NIM?

Para calcular la Suma de Nim de dos enteros:

1. Pasarlos a base 2
2. Calcular su suma directa modulo 2

$$27 \oplus 51 = 40$$

$$(11011)_2 \oplus (110011)_2 = (101000)_2$$

Por ejemplo

$$11011 = 27$$

$$110011 = 51$$

$$- - - - - -$$

$$\text{Nim-Sum} = 101000 = 40$$

¿Cómo ganar al juego de NIM?

Calcular la Suma de Nim es la única técnica posible para ganar este juego

Teorema:

Una posición (x_1, x_2, x_3) en Nim es una posición ganadora si y solo si la Suma de Nim de sus componentes es cero:

$$x_1 \oplus x_2 \oplus x_3 = 0$$

Nota: La Suma de Nim es asociativa y conmutativa, asi que el orden en el que se haga no es problema.

Ejemplo

Supongamos tres montones de palillos con 5, 7 y 9 piezas respectivamente.

Parte I:

Calculamos la Suma de Nim para determinar si el primer o el segundo jugador tiene una estrategia ganadora.

Respuesta:

$$\begin{aligned} 101 &= 5 \\ 111 &= 7 \\ 1001 &= 9 \\ - \; - \; - \; - \; - & \\ \text{Sum} &= 1011 = 11 \end{aligned}$$

Como la Suma de Nim no es igual a 0, la posición (5,7,9) no es una posición ganadora. Por tanto, el primer jugador puede mover a una posición ganadora y formular una estrategia ganadora.

¡Estamos suponiendo que el jugador que quita el último palillo es el ganador!

Ejemplo

Parte II:

¿Cual es el primer movimiento que hay que hacer para asegurar que el primer jugador se mueve a una posición ganadora?

Respuesta:

$$
\begin{aligned}
101 &= 5 \\
111 &= 7 \\
10 &= 2 \\
- \; - \; - \; - \; - & \\
\text{Nim-Sum} = 000 &= 0
\end{aligned}
$$

Solo hay un movimiento posible:

Hay que quitar 7 palillos del montón en el que hay 9 para dejar solo 2 en ese montón.