Game Playing

Overview

- two-player zero-sum discrete finite deterministic game of perfect information
- Minimax search
- Alpha-beta pruning

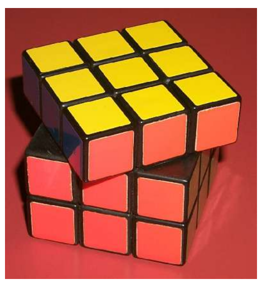
Two-player zero-sum discrete finite deterministic games of perfect information

Definitions:

- Zero-sum: one player's gain is the other player's loss.

- Discrete: states and decisions have discrete values
- Finite: finite number of states and decisions
- Deterministic: no coin flips, die rolls – no chance
- Perfect information: each player can see the complete game state. No simultaneous decisions.

Which of these are: Two-player zero-sum discrete finite deterministic games of perfect information?



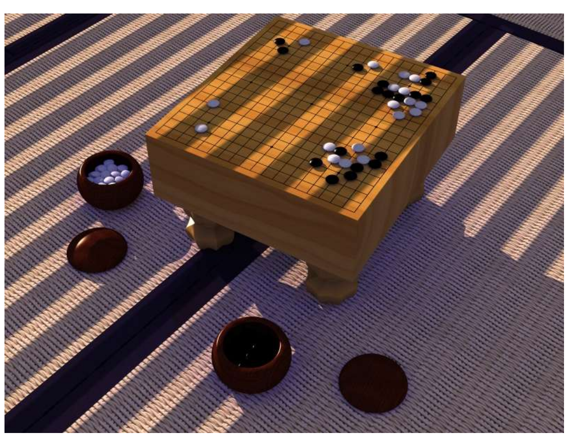
Zero-sum: one player's gain is the other player's loss. Does not mean *fair*.

Discrete: states and decisions have discrete values

Finite: finite number of states and decisions

Deterministic: no coin flips, die rolls – no chance

Perfect information: each player can see the complete game state. No simultaneous decisions.



Which of these are: Two-player zero-sum discrete finite deterministic games of perfect information?

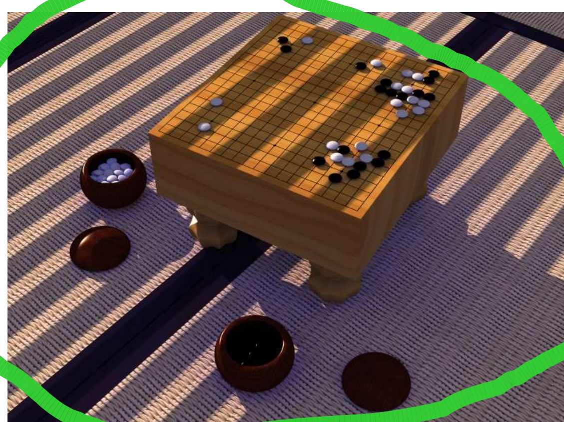
Zero-sum: one player's gain is the other player's loss. Does not mean *fair*.

Discrete: states and decisions have discrete values

Finite: finite number of states and decisions

Deterministic: no coin flips, die rolls – no chance

Perfect information: each player can see the complete game state. No simultaneous decisions.



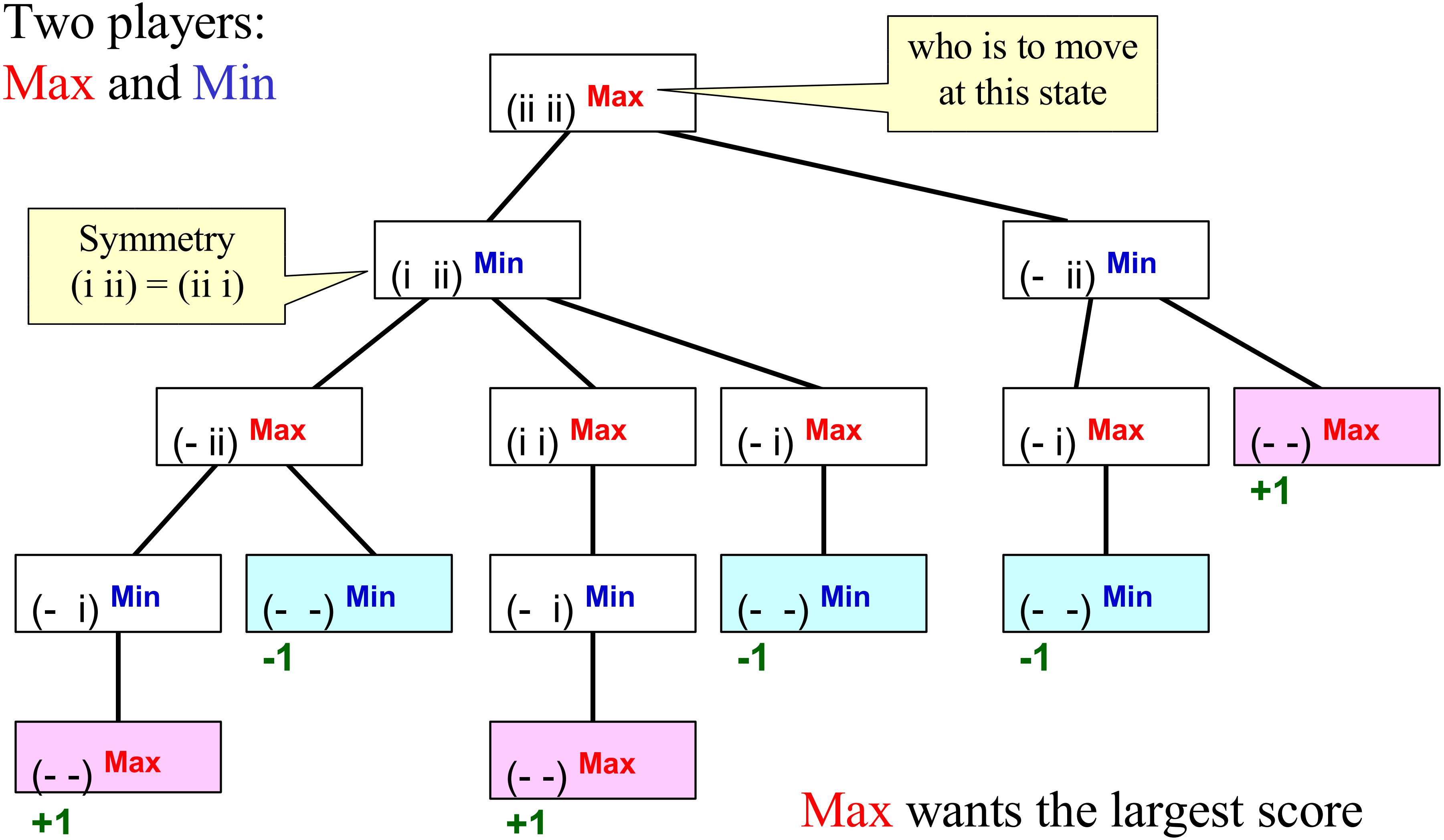
II-Nim: Max simple game

- There are 2 piles of sticks. Each pile has 2 sticks.
- Each player takes one or more sticks from one pile.
- The player who takes the last stick loses.

(ii, ii)

The game tree for II-Nim

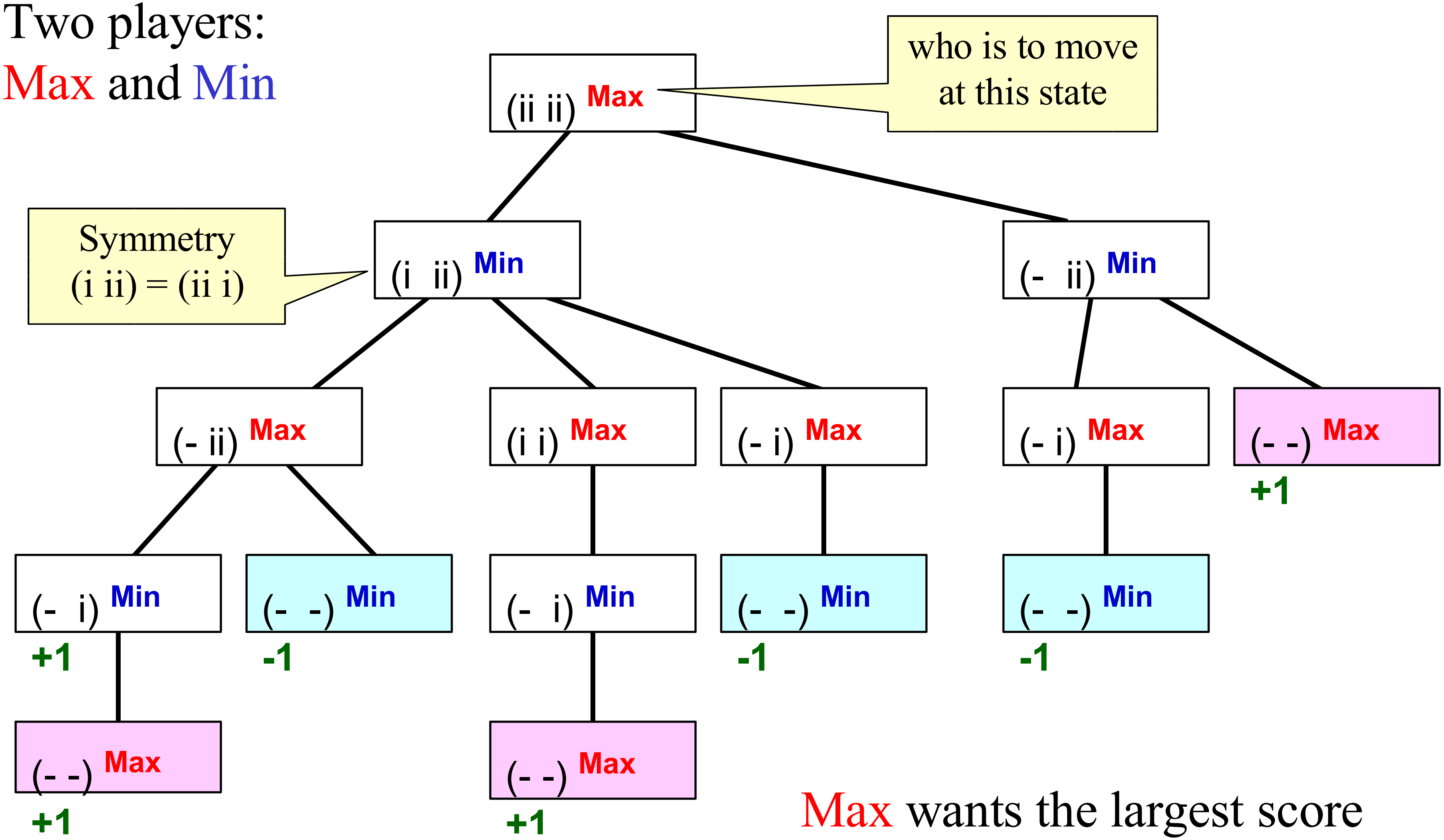
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

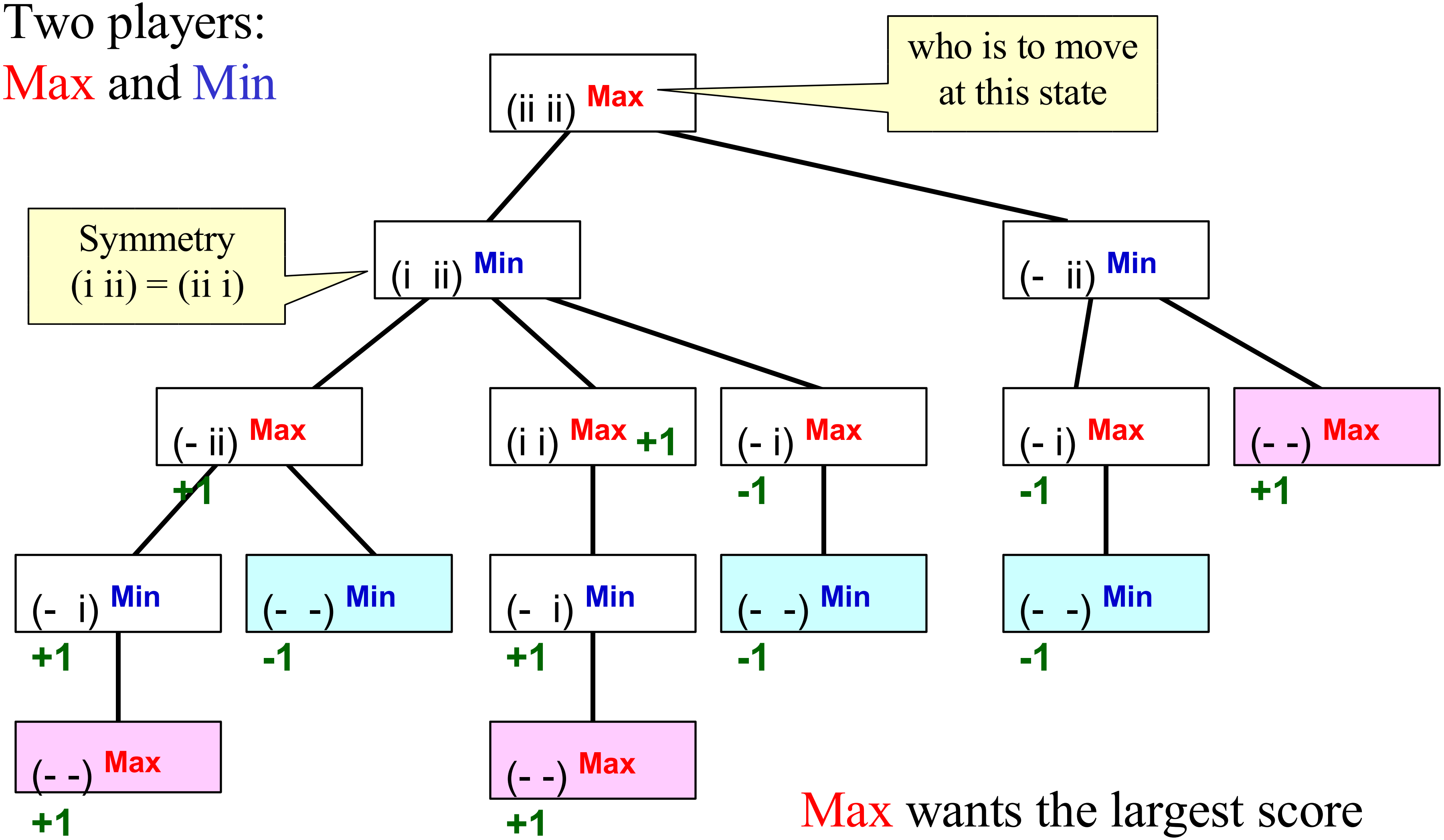
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

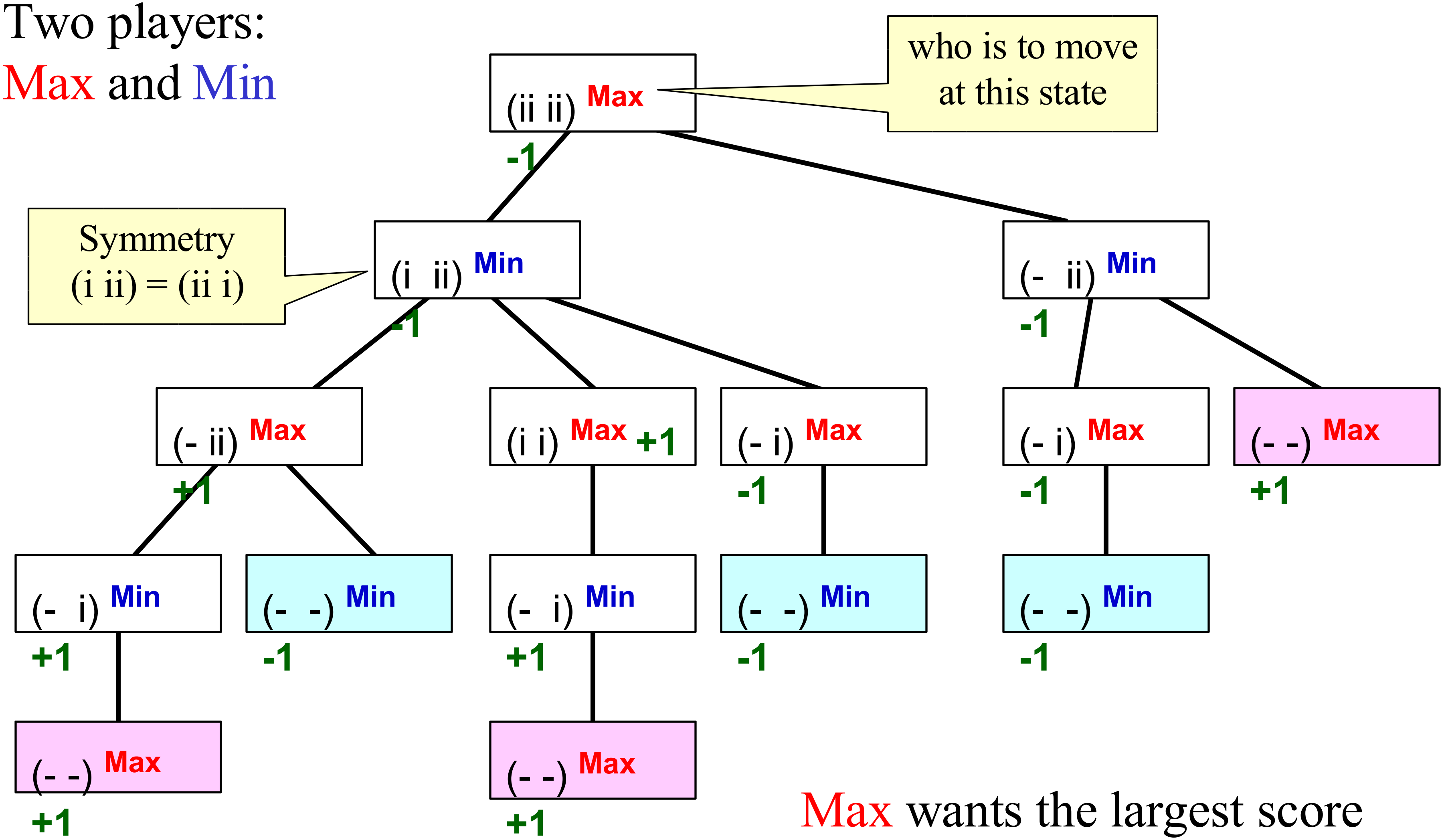
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

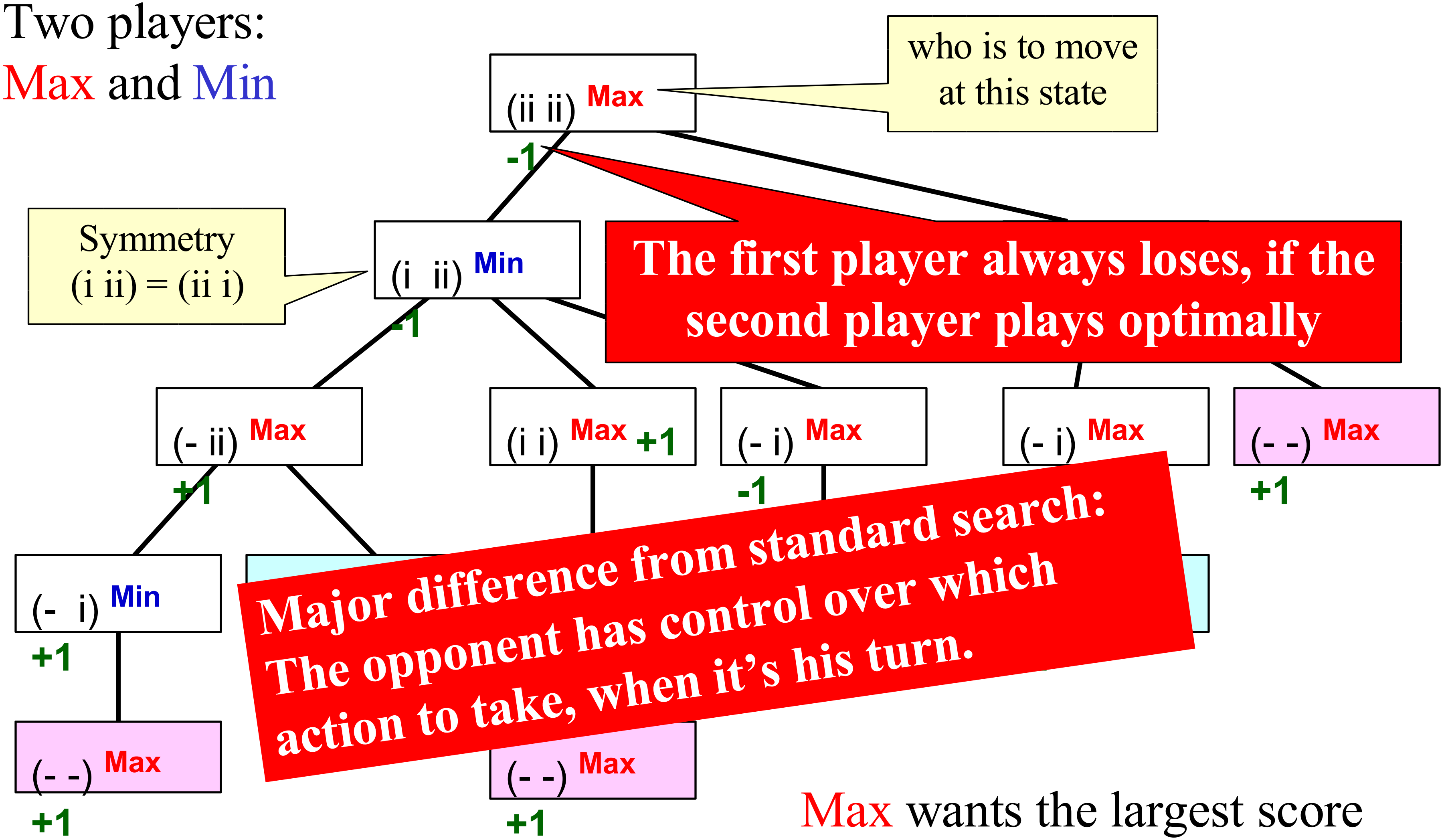
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game theoretic value

- Game theoretic value (a.k.a. minimax value) of a node = the score of the terminal node that will be reached if both players play optimally.
- = The numbers we filled in.
- Computed bottom up
  - In Max's turn, take the max of the children (Max will pick that maximizing action)
  - In Min's turn, take the min of the children (Min will pick that minimizing action)
- Implemented as a modified version of DFS: minimax algorithm

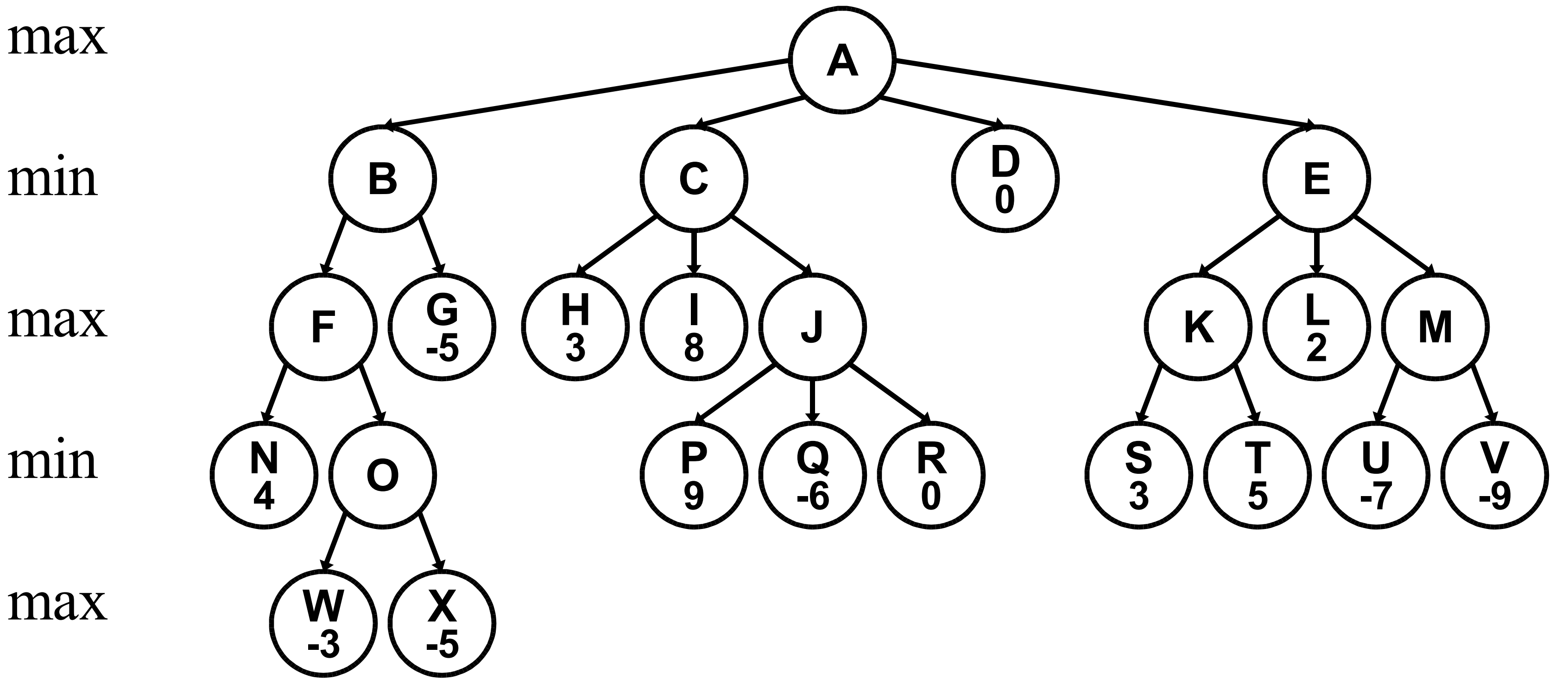
Minimax algorithm

```
function Max-Value(s)
inputs:
    s: current state in game, Max about to play
output: best-score (for Max) available from s

    if ( s is a terminal state )
    then return ( terminal value of s )
    else
            α := – ∞
            for each s’ in Succ(s)
                α := max( α , Min-value(s’))
    return α
```

```
function Min-Value(s)
output: best-score (for Min) available from s

    if ( s is a terminal state )
    then return ( terminal value of s)
    else
            β := ∞
            for each s’ in Succs(s)
                β := min( β , Max-value(s’))
    return β
```

- Time complexity?
- Space complexity?

Minimax example



Tic-Tac-Toe

Evaluation Function

Tic-Tac-Toe -1

If p is not a winning position for either player,

$e(p)$ = (number of complete rows, columns, or diagonals that are still open for *MAX*) – (number of complete rows, columns, or diagonals that are still open for *MIN*)

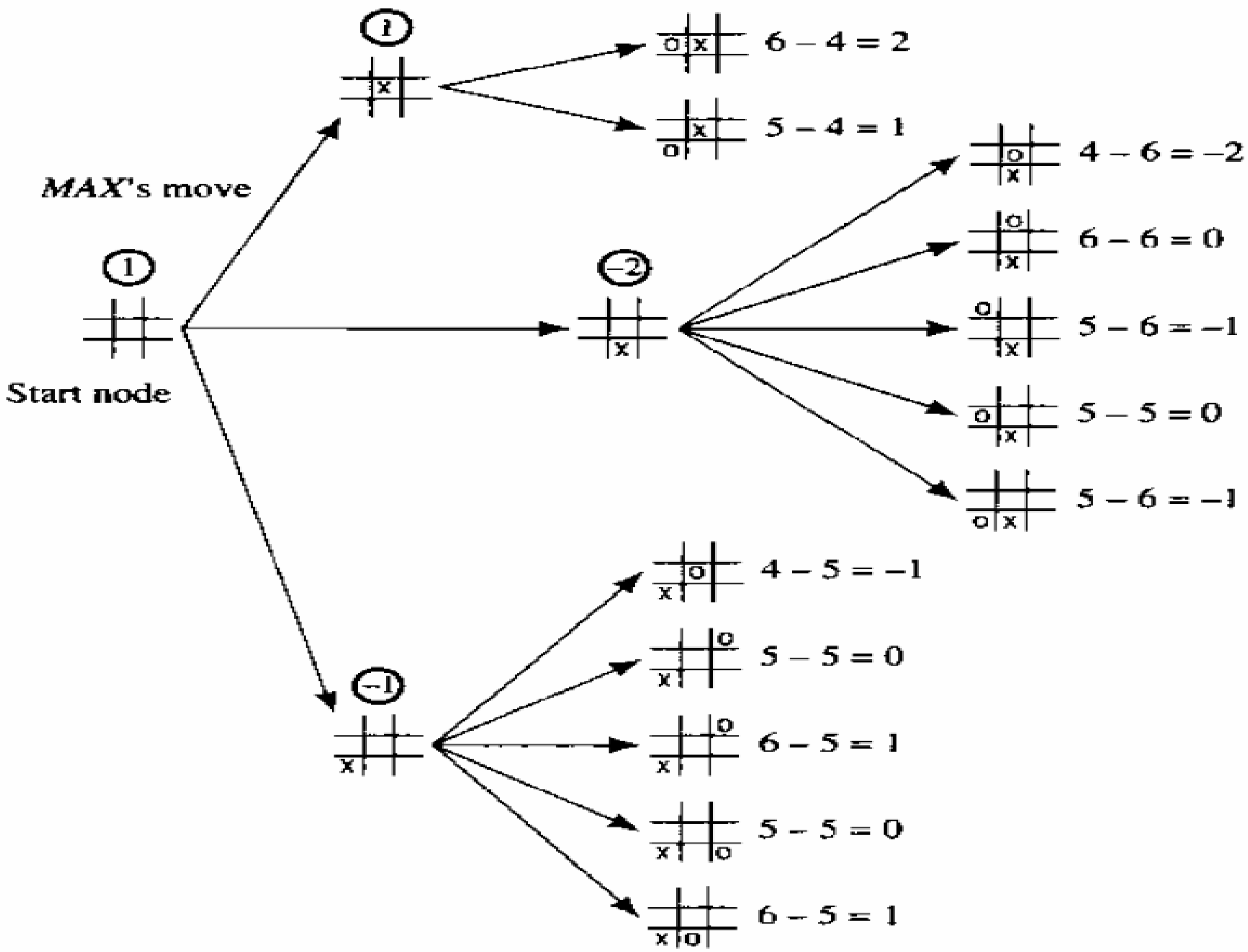
If p is a win for *MAX*,

$e(p) = \infty$(I use ∞ here to denote a very large positive number)

If p is a win for *MIN*,

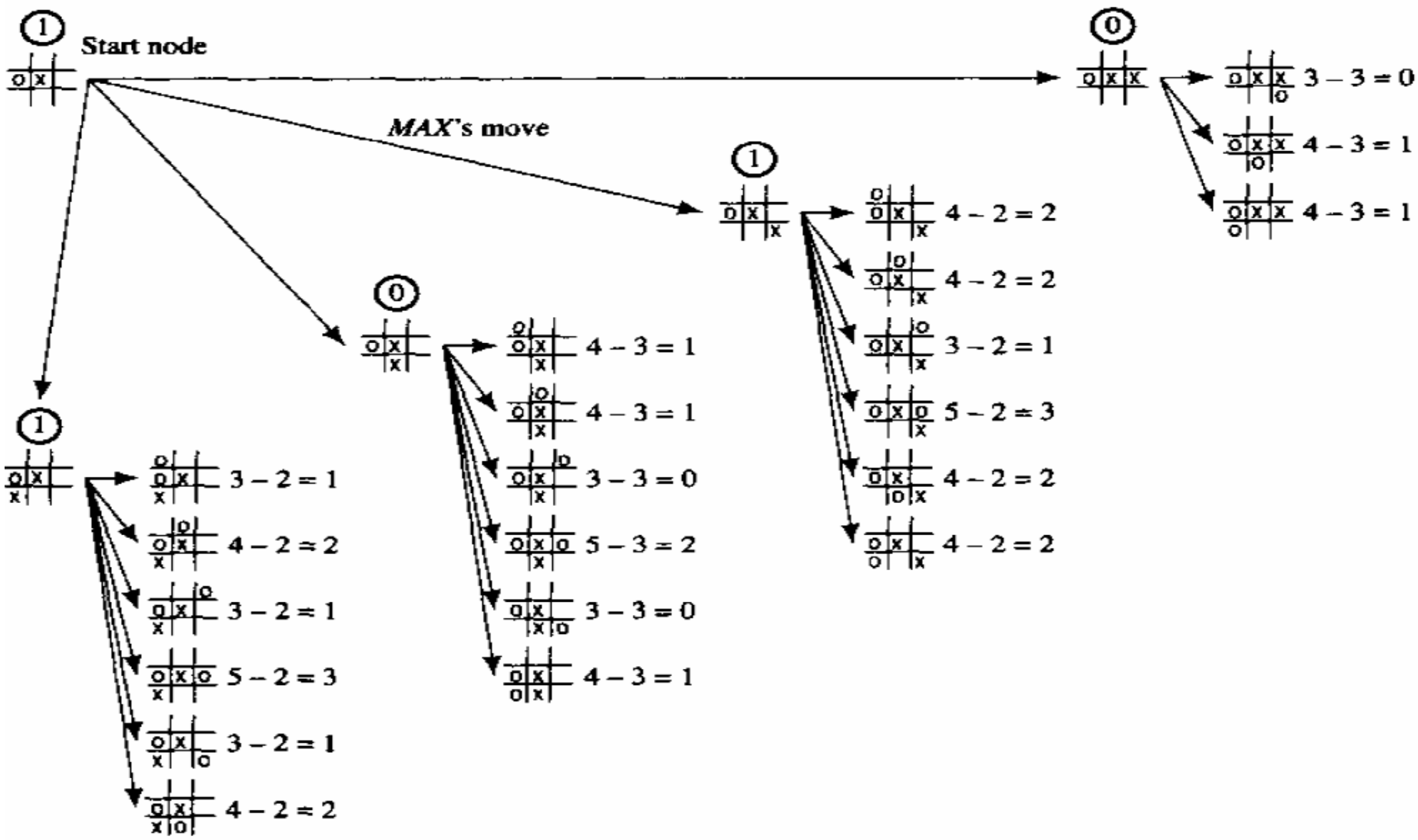
$e(p) = -\infty$

Tic-Tac-Toe -1



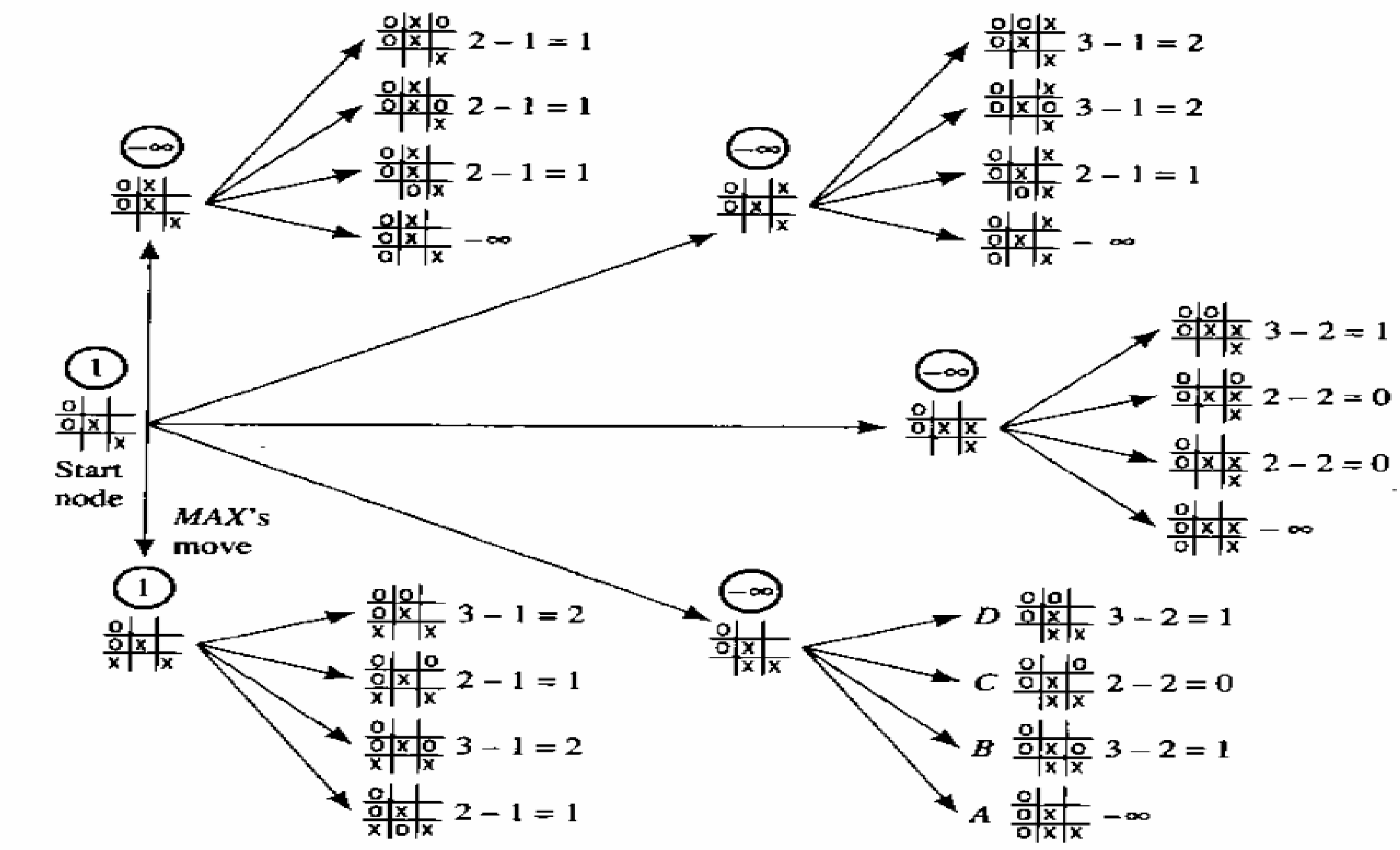
MAX's move
Start node
1
1
-2
-1
6 − 4 = 2
5 − 4 = 1
4 − 6 = −2
6 − 6 = 0
5 − 6 = −1
5 − 5 = 0
5 − 6 = −1
4 − 5 = −1
5 − 5 = 0
6 − 5 = 1
5 − 5 = 0
6 − 5 = 1

Tic-Tac-Toe -2



1
Start node
MAX's move
0
1
0
1
3 – 3 = 0
4 – 3 = 1
4 – 3 = 1
4 – 2 = 2
4 – 2 = 2
3 – 2 = 1
5 – 2 = 3
4 – 2 = 2
4 – 2 = 2
4 – 3 = 1
4 – 3 = 1
3 – 3 = 0
5 – 3 = 2
3 – 3 = 0
4 – 3 = 1
3 – 2 = 1
4 – 2 = 2
3 – 2 = 1
5 – 2 = 3
3 – 2 = 1
4 – 2 = 2

Tic-Tac-Toe -3



2 − 1 = 1
2 − 1 = 1
2 − 1 = 1
−∞
3 − 1 = 2
3 − 1 = 2
2 − 1 = 1
− ∞
3 − 2 = 1
2 − 2 = 0
2 − 2 = 0
− ∞
Start node
MAX's move
3 − 1 = 2
2 − 1 = 1
3 − 1 = 2
2 − 1 = 1
D 3 − 2 = 1
C 2 − 2 = 0
B 3 − 2 = 1
A −∞

Minimax algorithm

```
function Max-Value(s)
inputs:
    s: current state in game, Max about to play
output: best-score (for Max) available from s
    if ( s is a terminal state )
    then return ( terminal value of s )
    else
        α := – ∞
        for each s’ in Succ(s)
            α := max( α , Min-value(s’))
    return α
```

```
function Min-Value(s)
output: best-score (for Min) available from s
    if ( s is a terminal state )
    then return ( terminal value of s)
    else
        β := ∞
        for each s’ in Succs(s)
            β := min( β , Max-value(s’))
    return β
```

- Time complexity? O((b^m)) ← bad
- Space complexity? O((bm))

Next: alpha-beta pruning

Gives the same game theoretic values as minimax, but prunes part of the game tree.

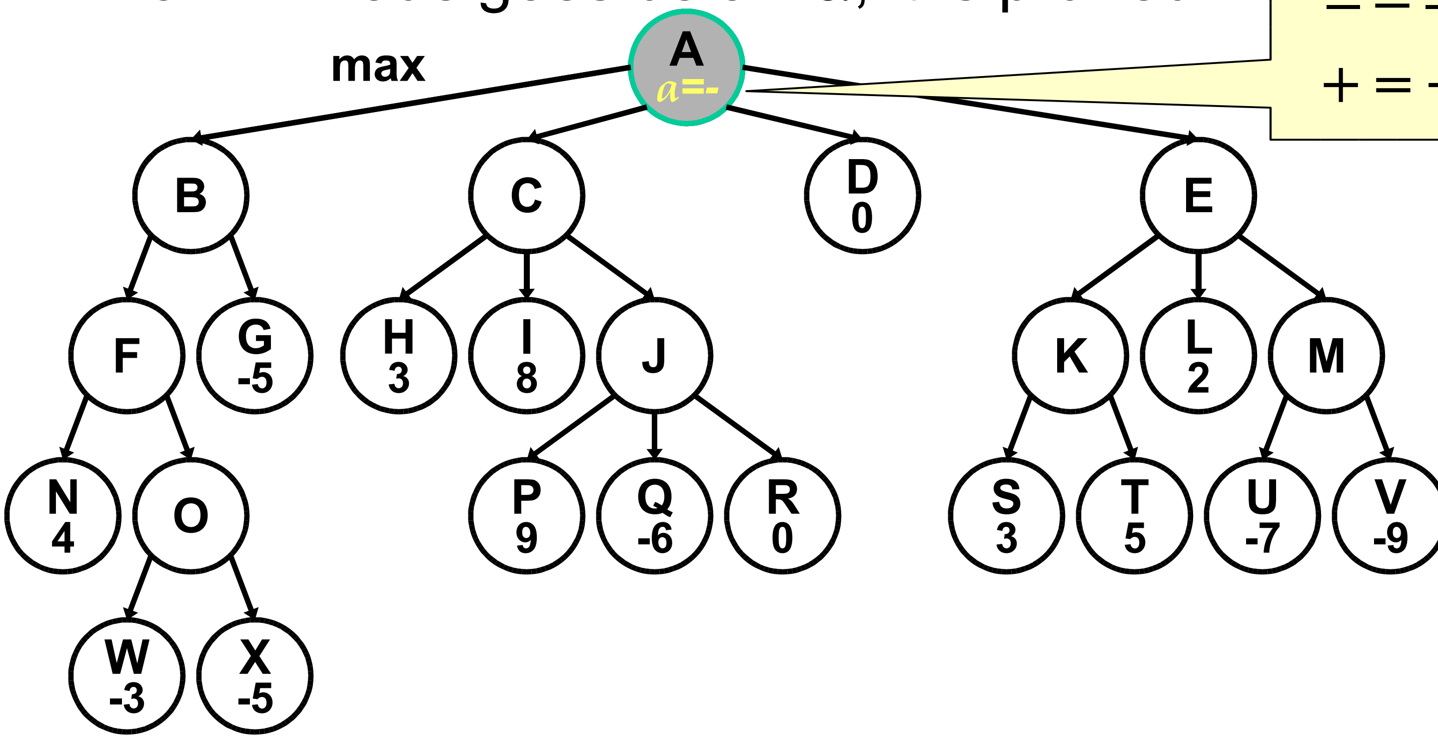
Alpha-beta pruning

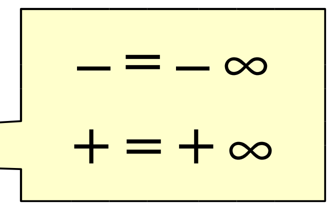
```
function Max-Value (s,α,β)
inputs:
    s: current state in game, Max about to play
    α: best score (highest) for Max along path to s
    β: best score (lowest) for Min along path to s
output: min(β , best-score (for Max) available from s)

    if ( s is a terminal state )
    then return ( terminal value of s )
    else for each s' in Succ(s)
            α := max( α , Min-value(s',α,β))
            if ( α ≥ β ) then return β   /* pruning */
    return α
```

```
function Min-Value(s,α,β)
output: max(α , best-score (for Min) available from s )

    if ( s is a terminal state )
    then return ( terminal value of s)
    else for each s' in Succs(s)
            β := min( β , Max-value(s',α,β))
            if (β ≤ α ) then return α   /* pruning */
    return β
```

Alpha-beta pruning example

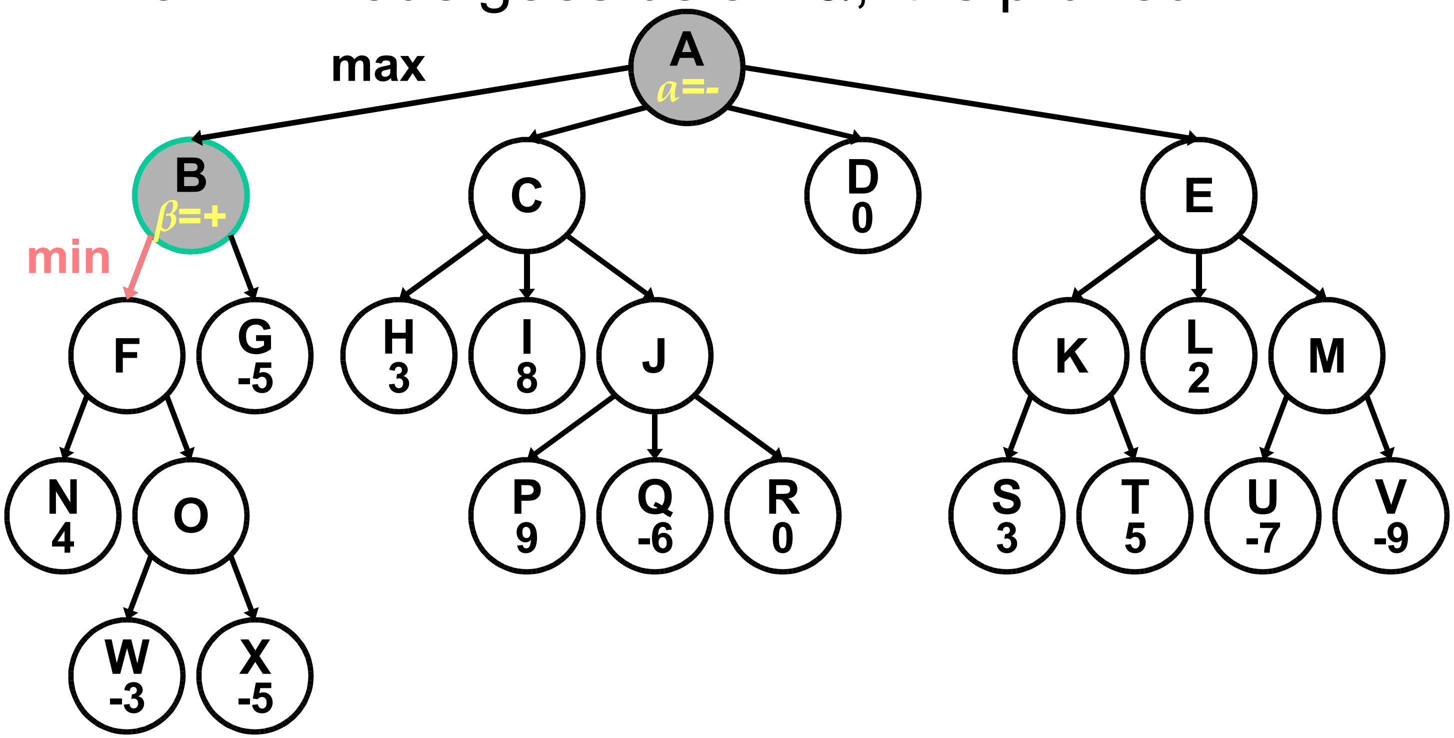
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.





Alpha-beta pruning example

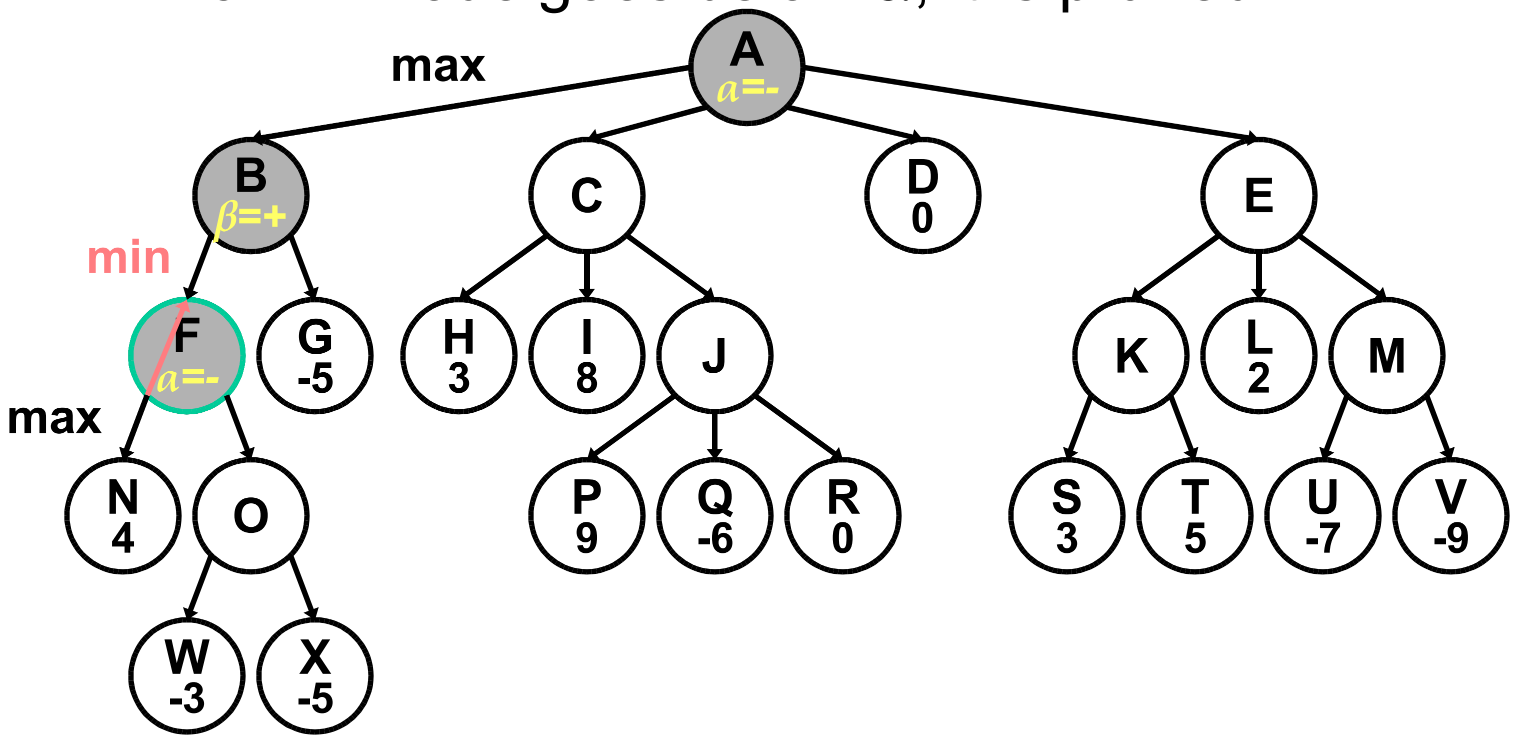
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

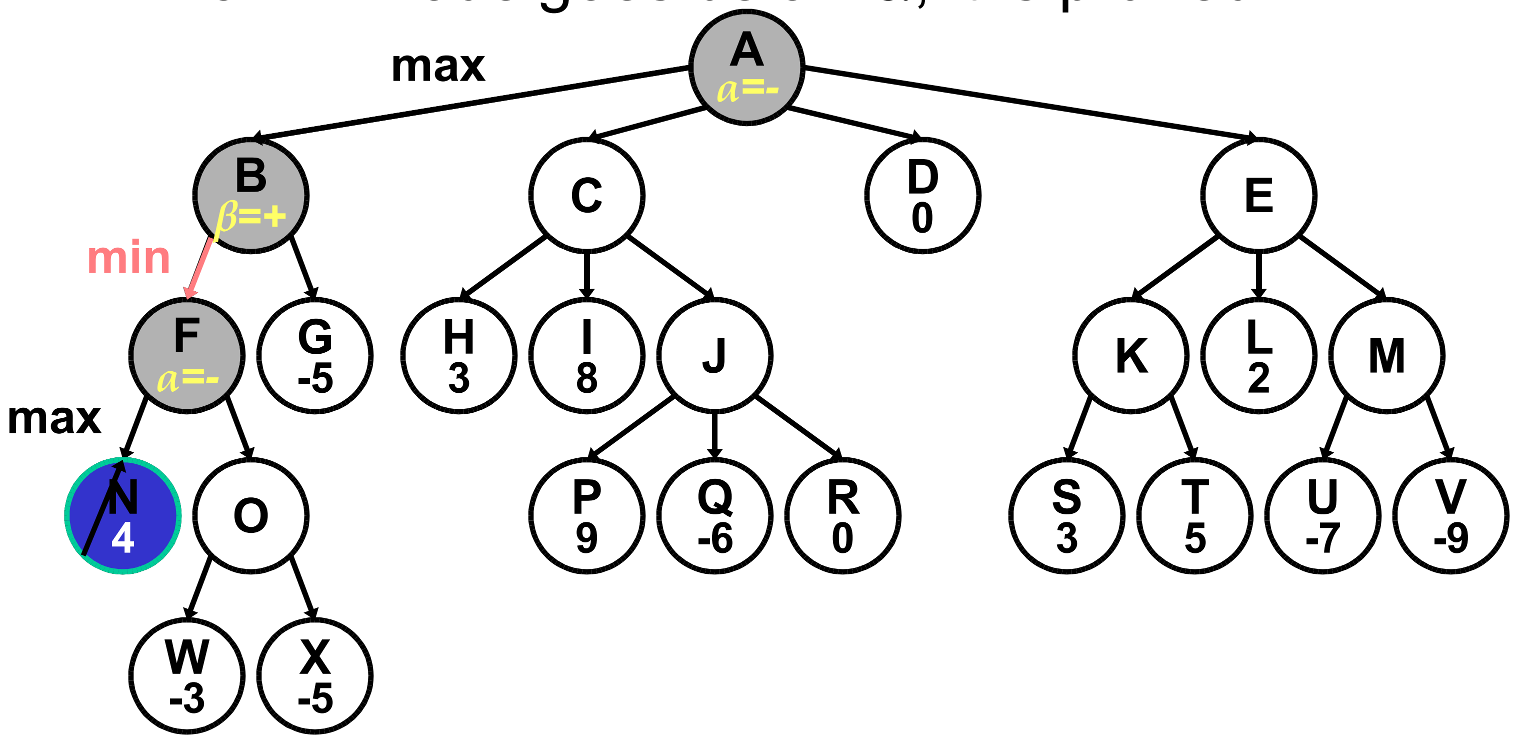
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

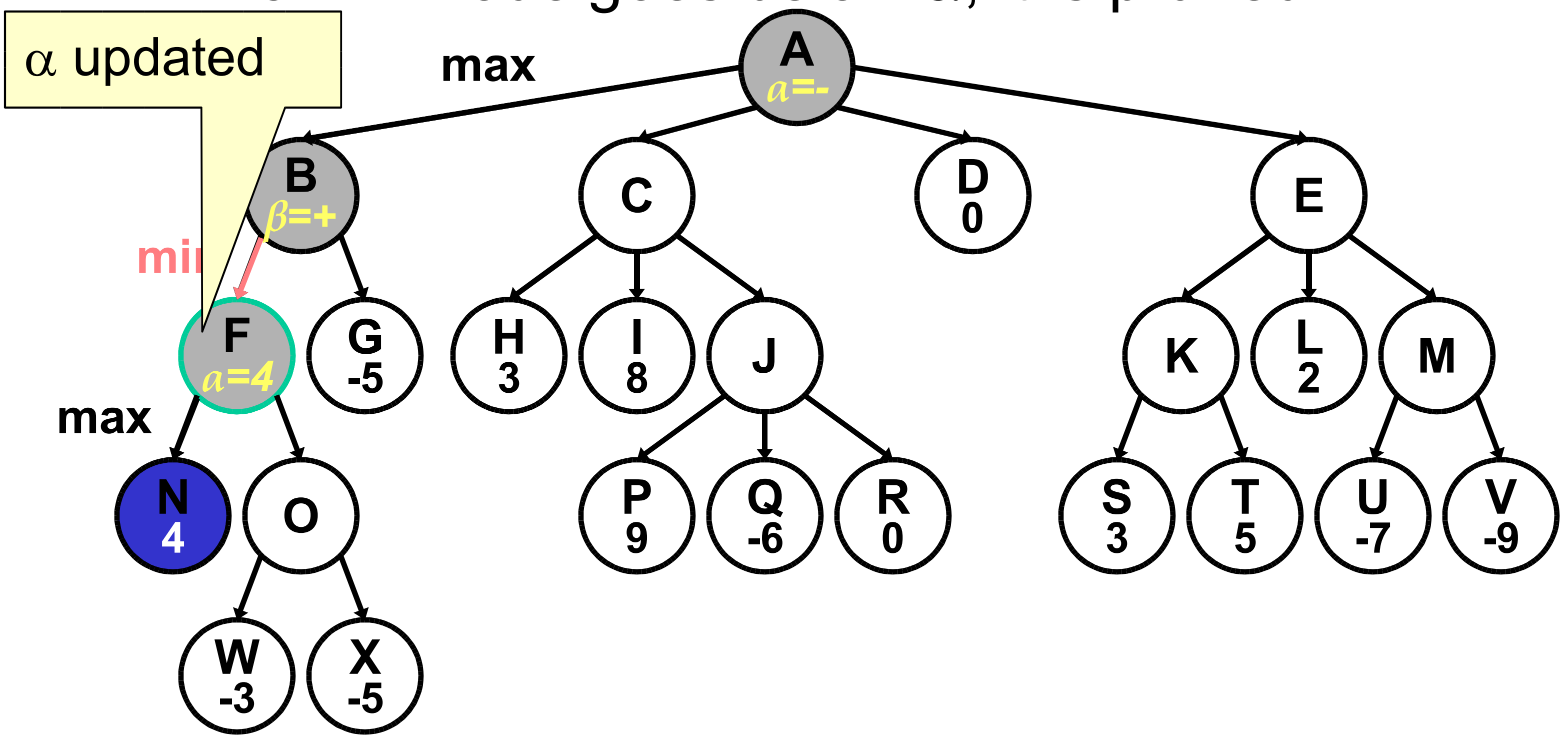
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

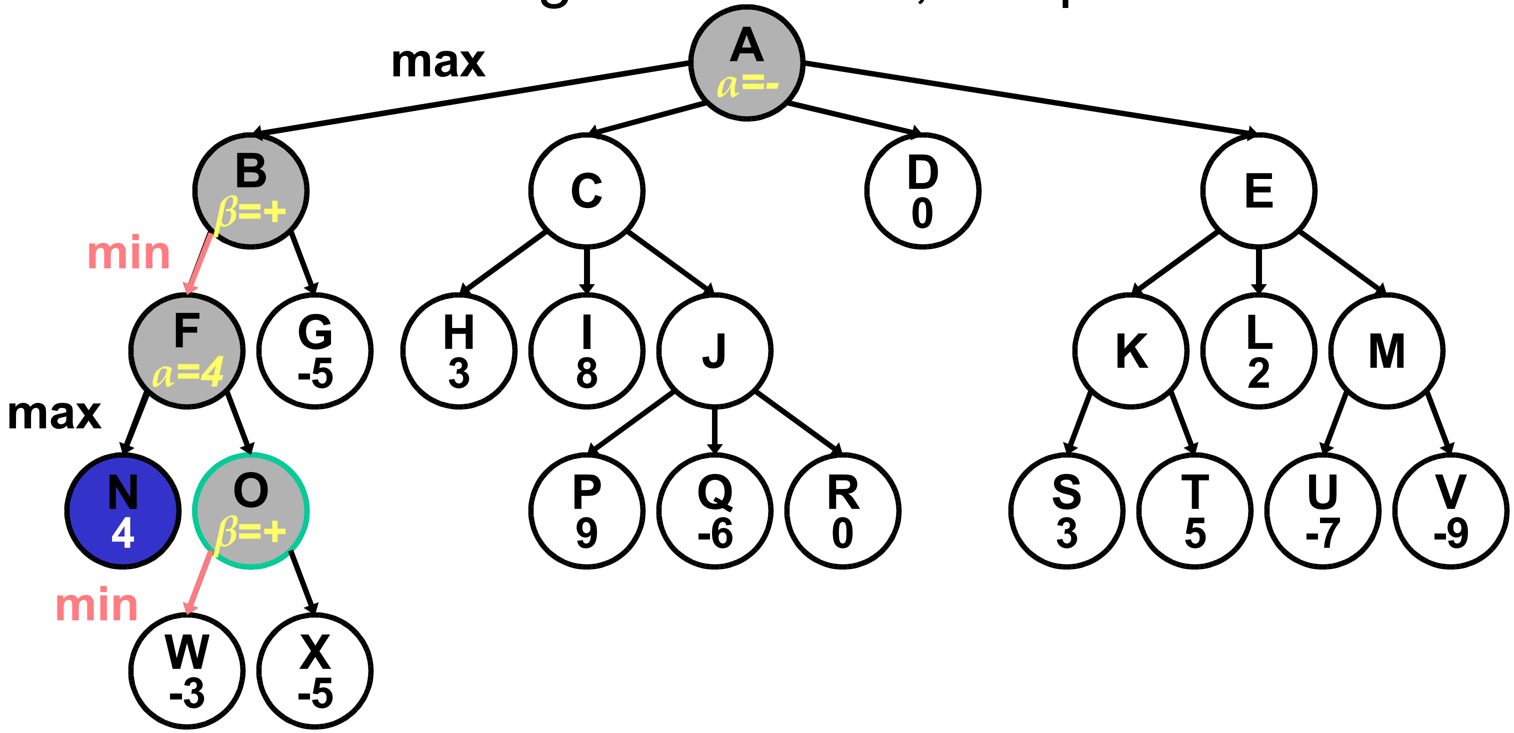
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

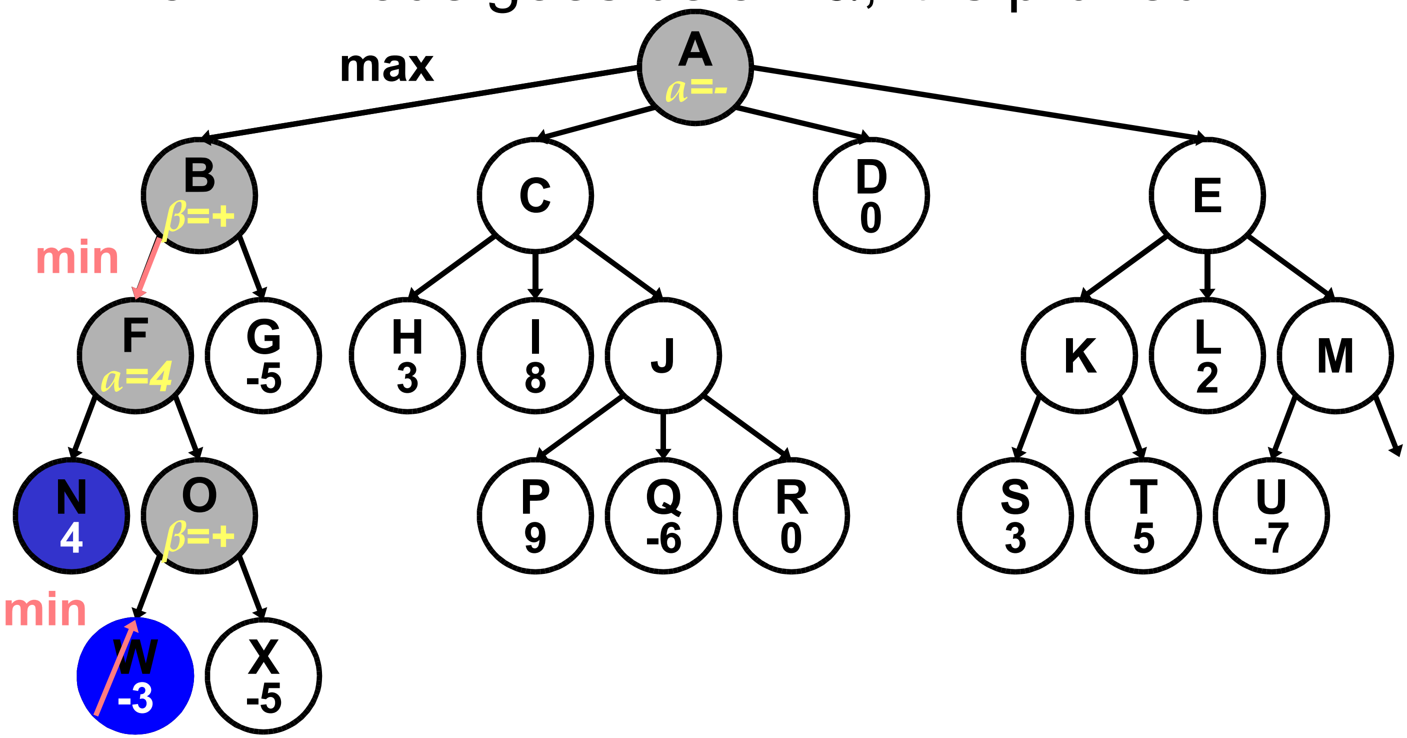
Alpha-beta pruning example

- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



Alpha-beta pruning example

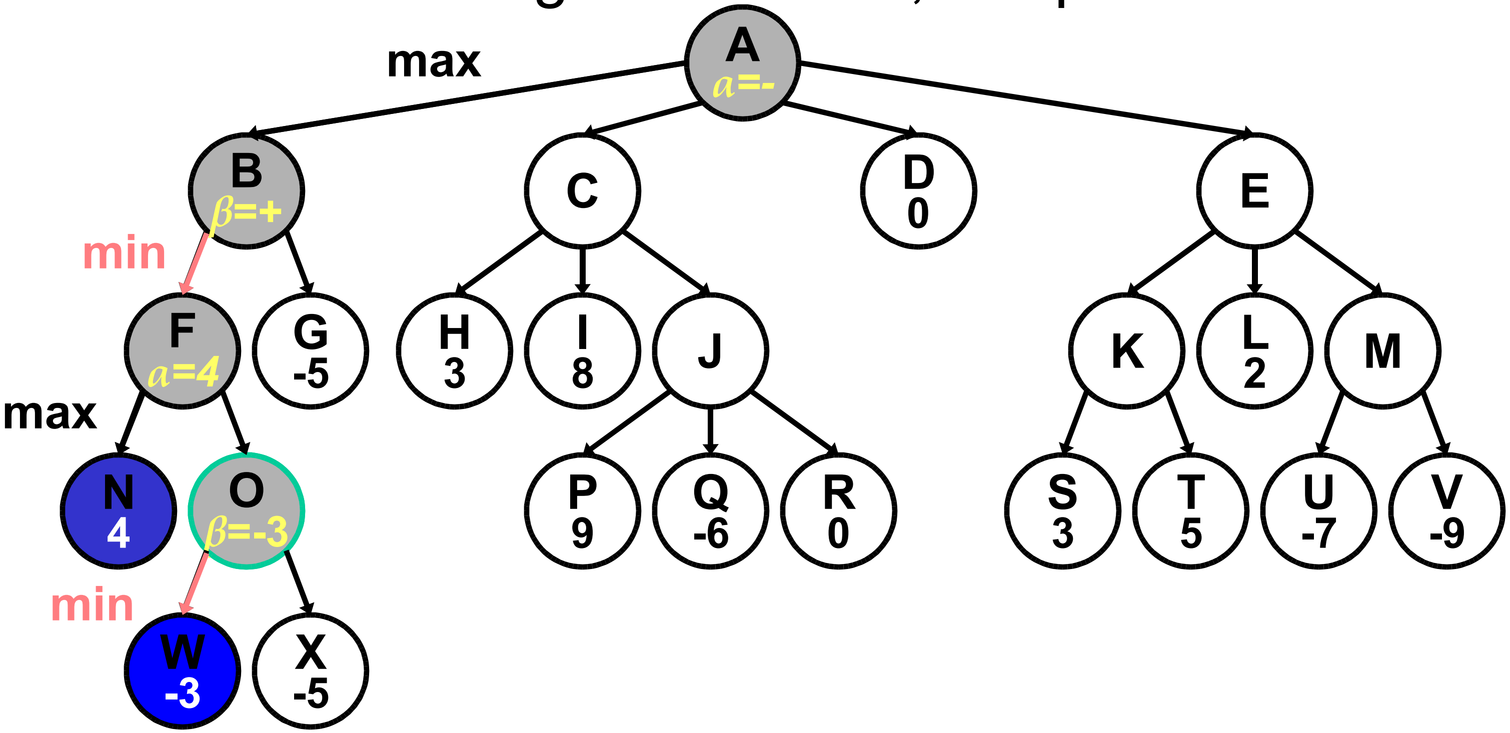
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

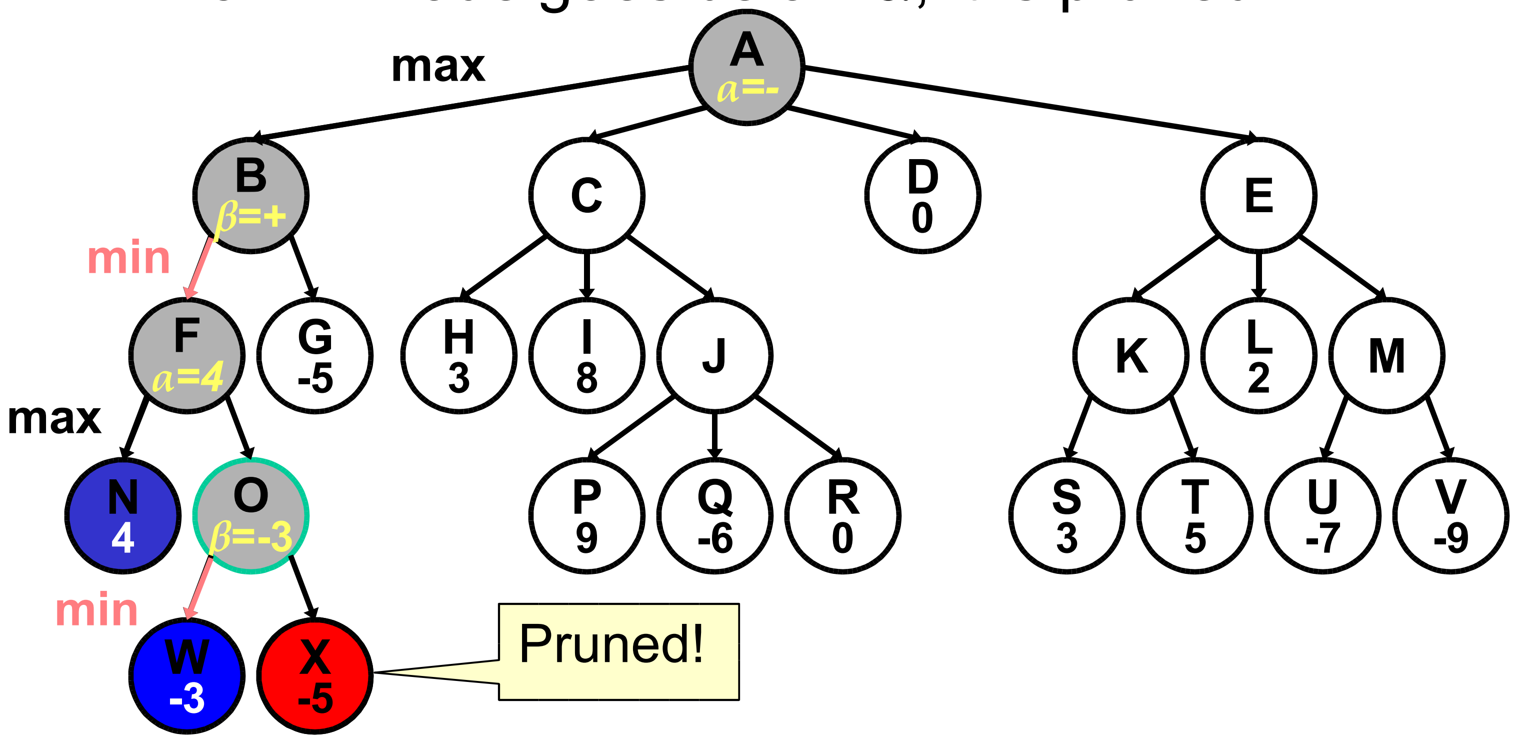
Alpha-beta pruning example

- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



Alpha-beta pruning example

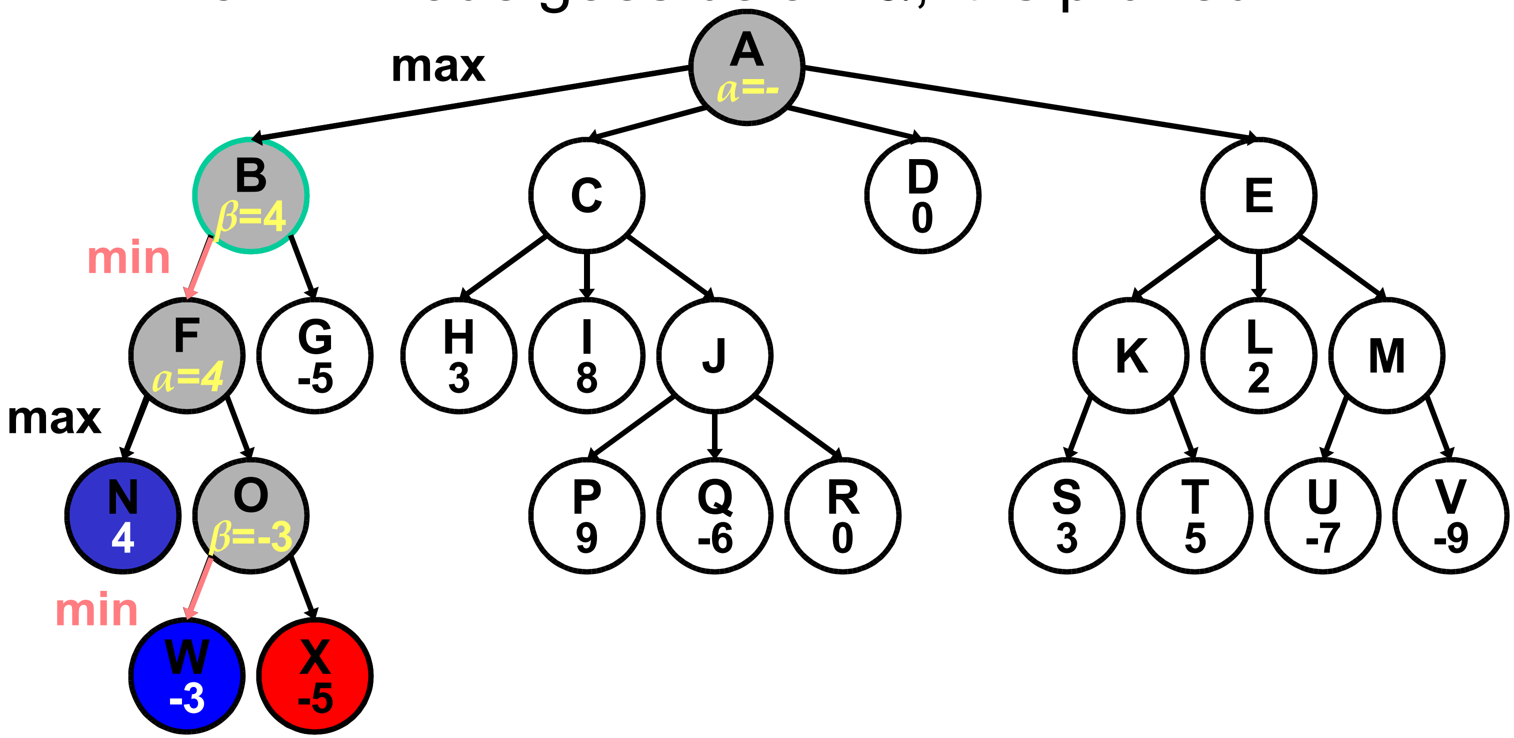
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

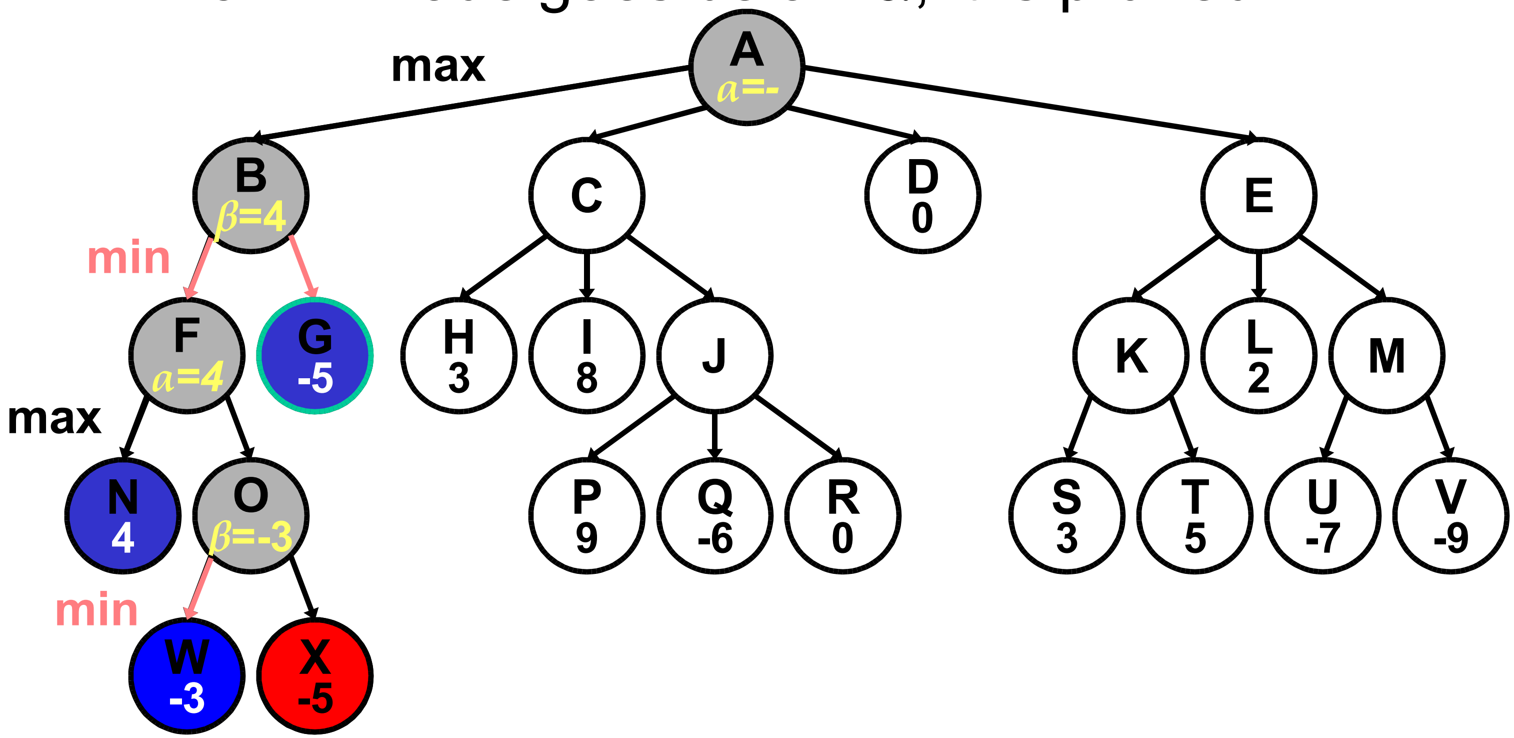
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

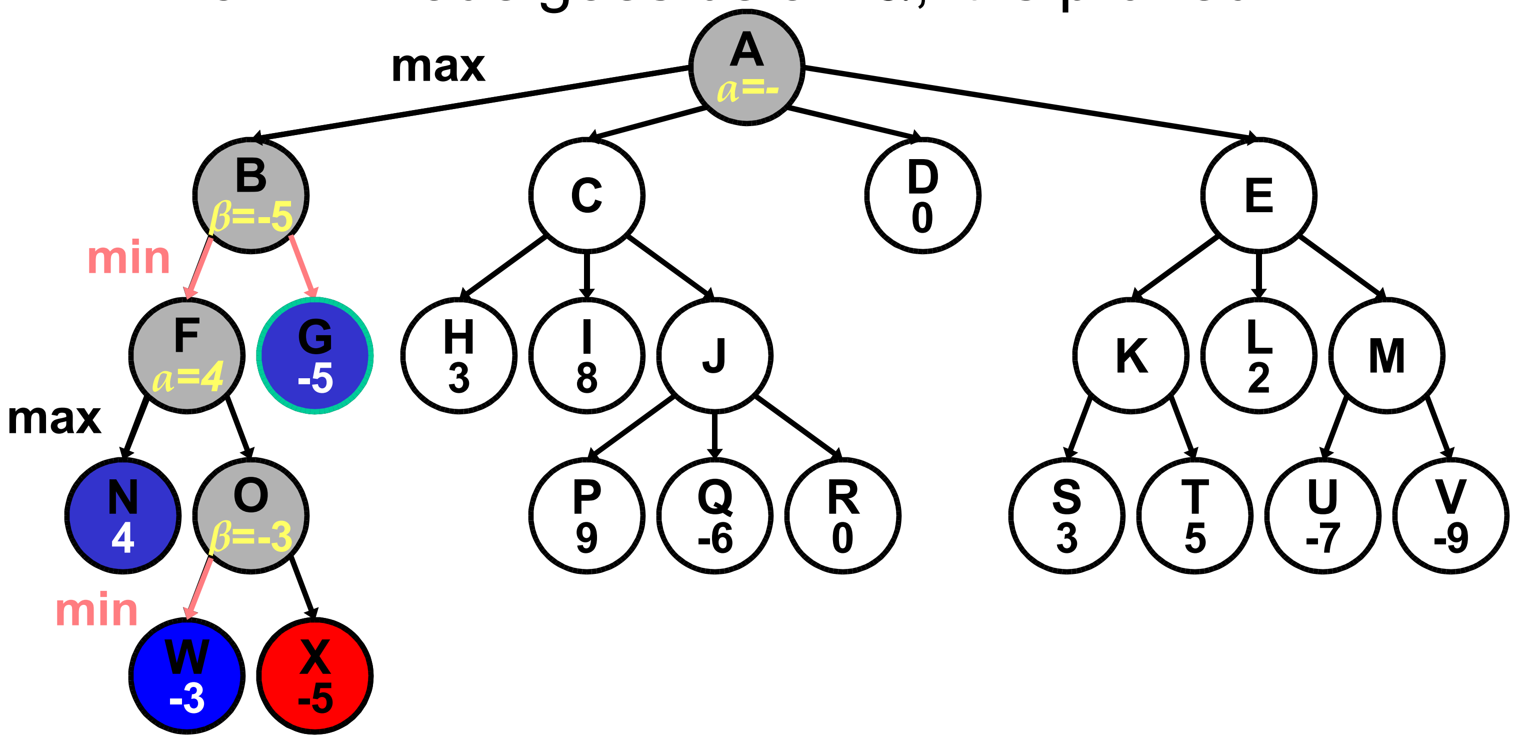
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

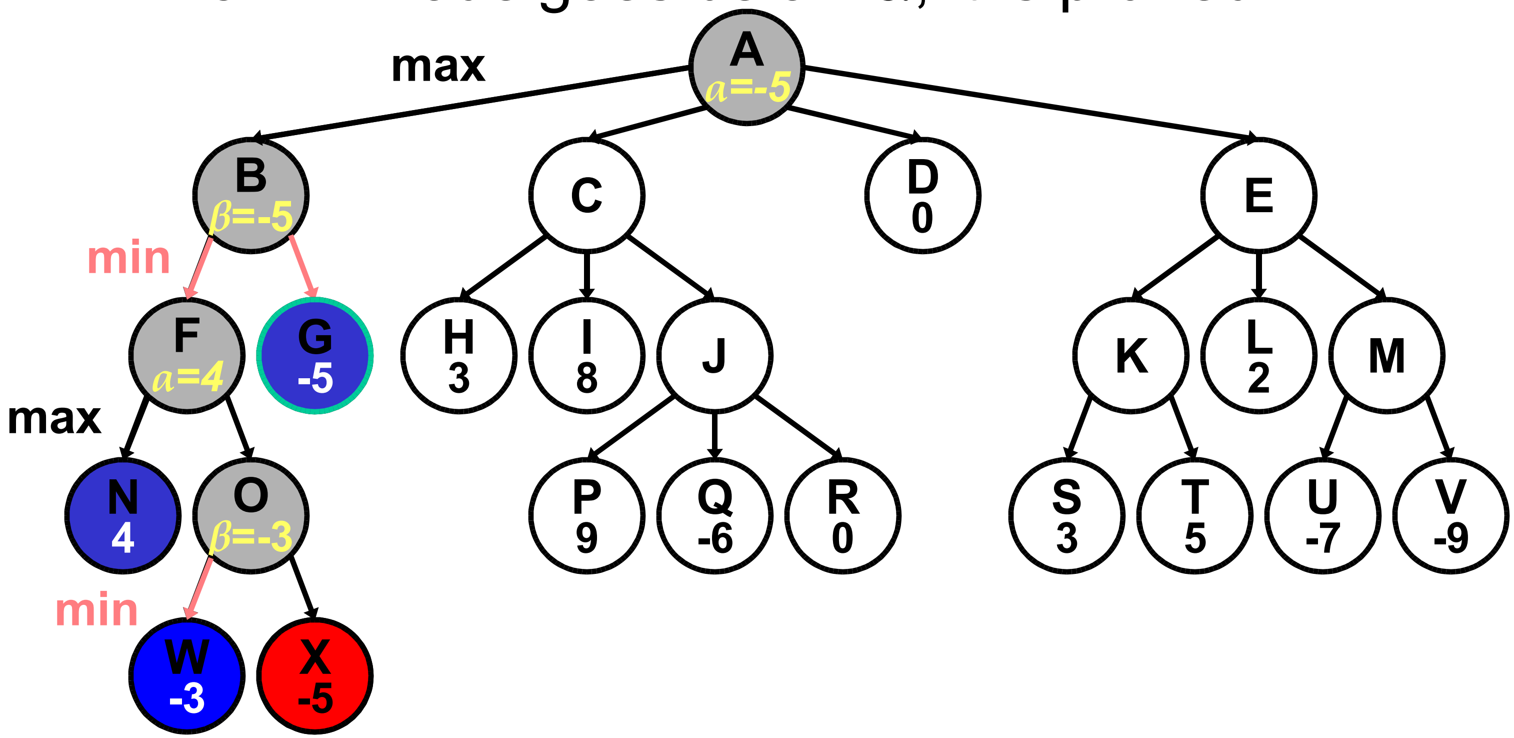
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

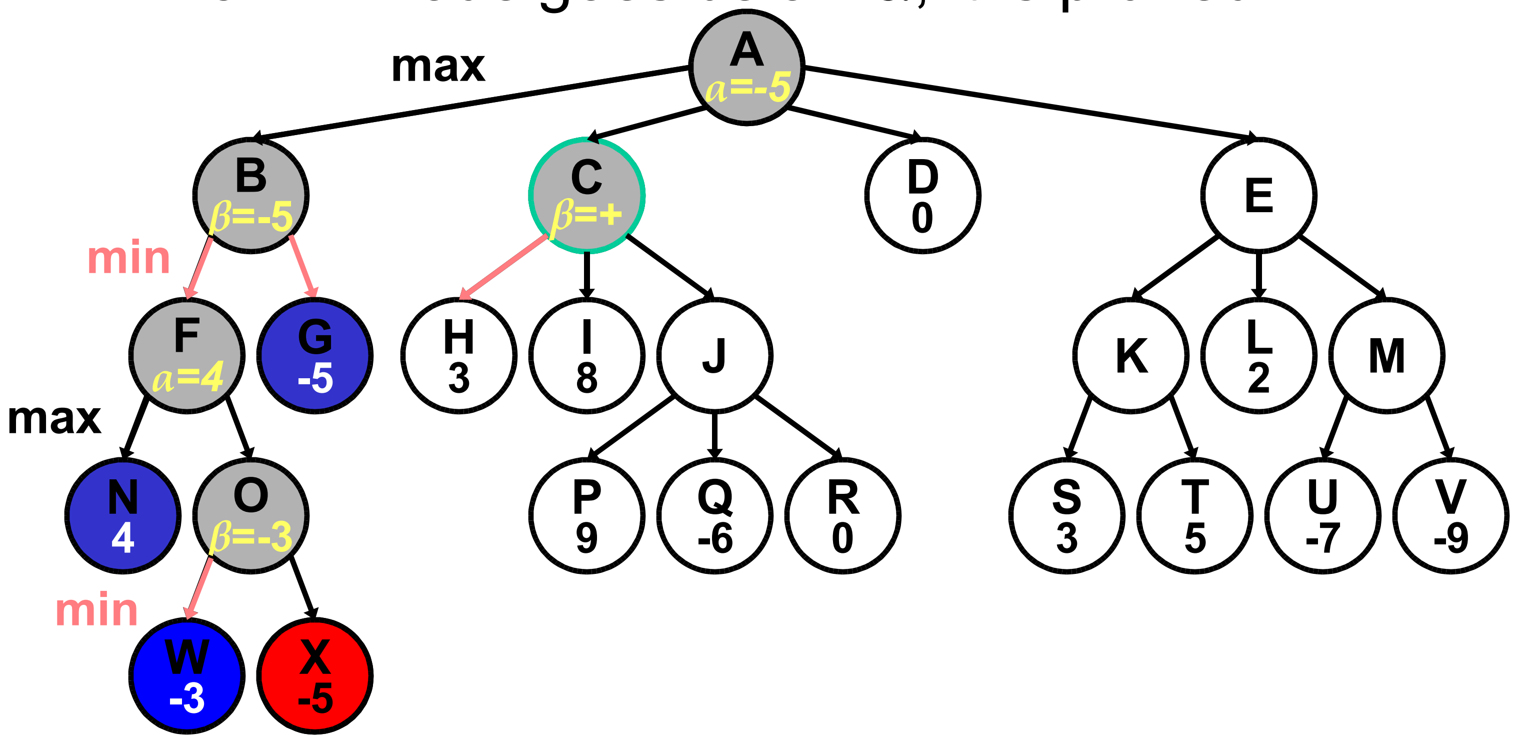
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

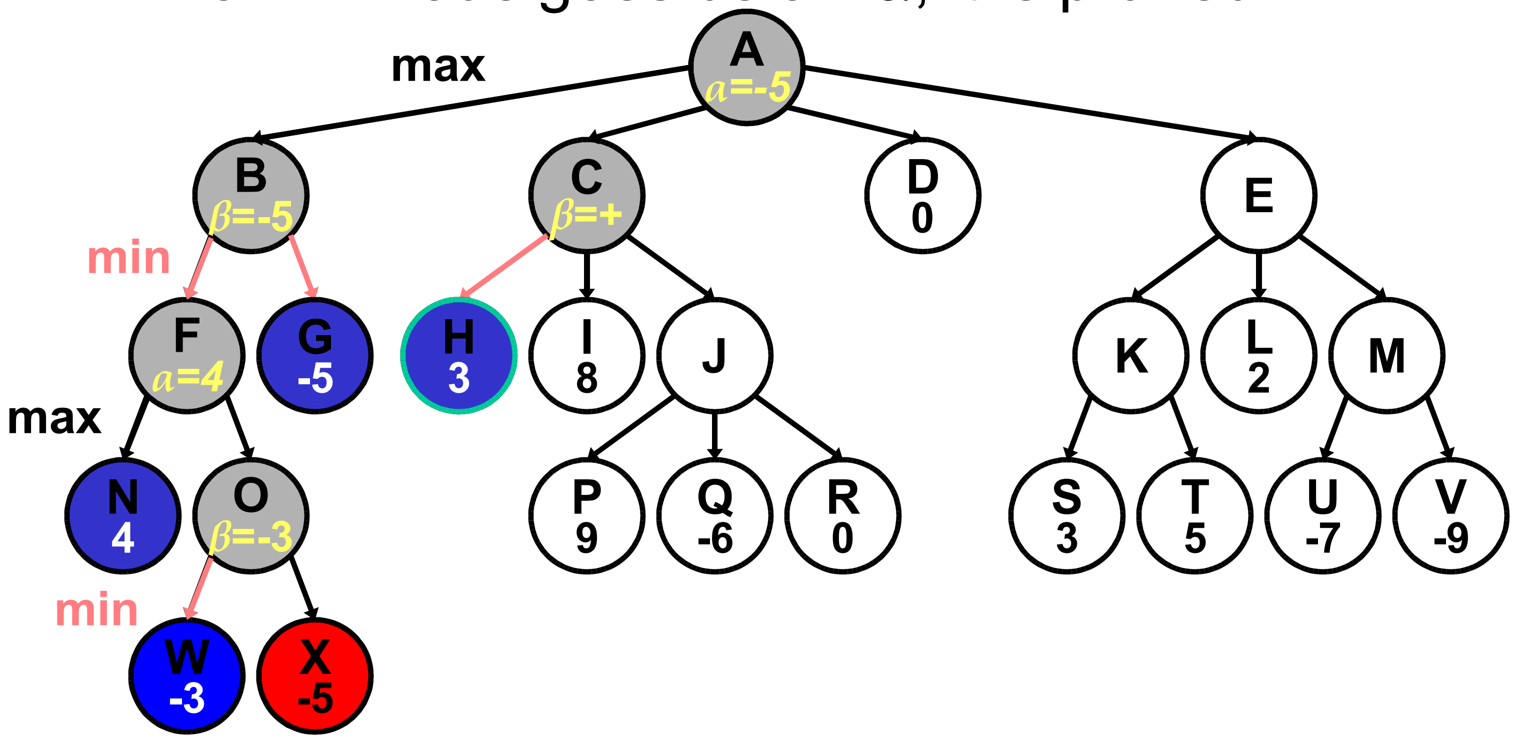
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

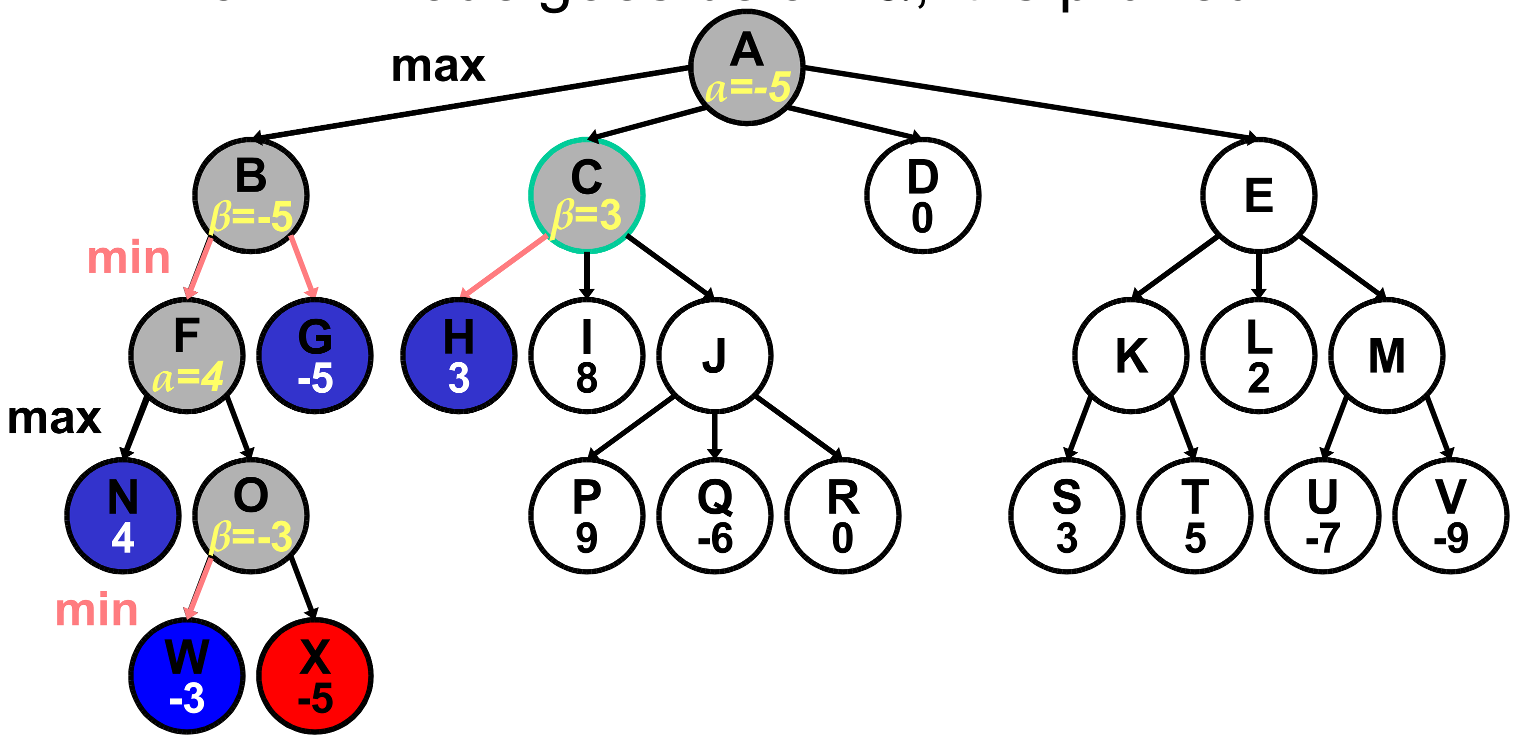
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

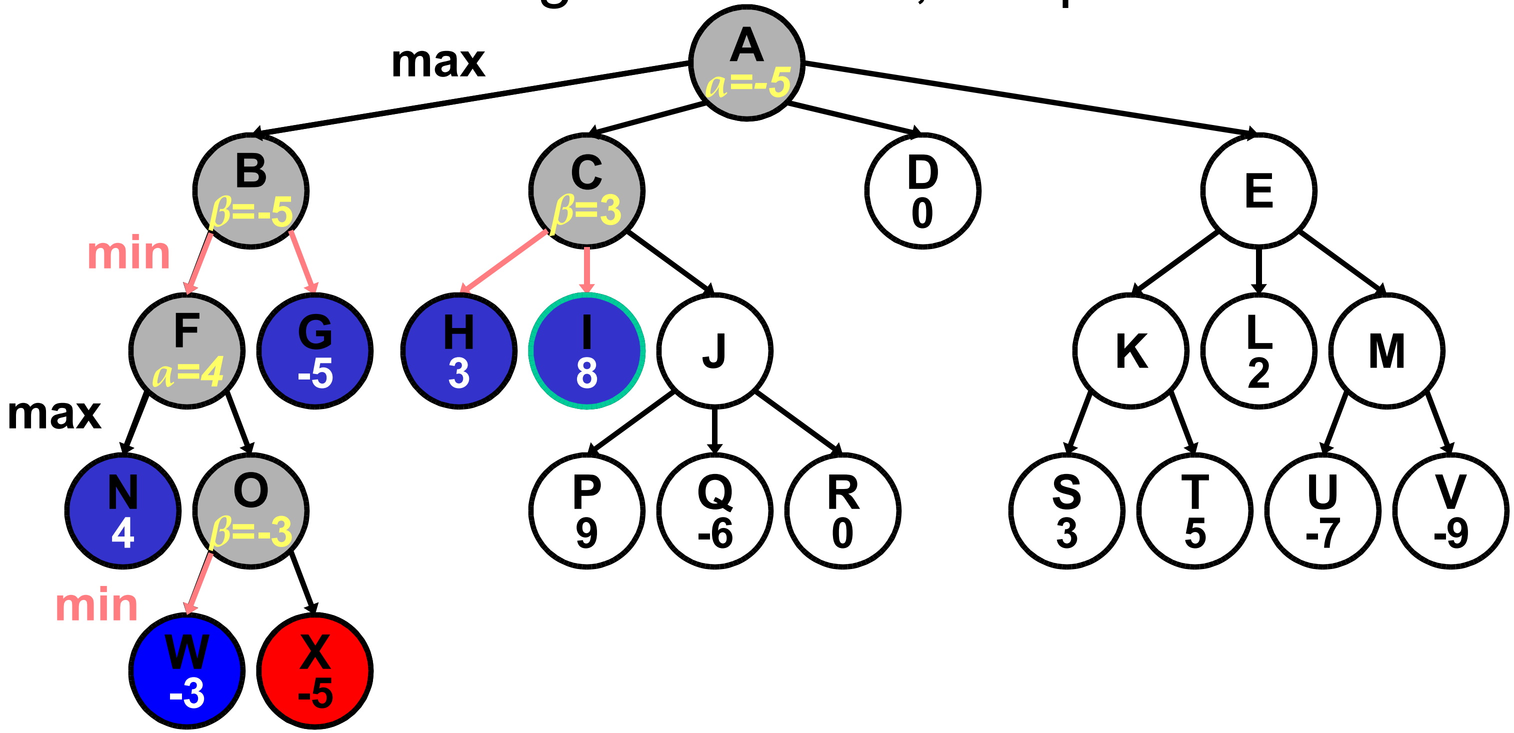
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

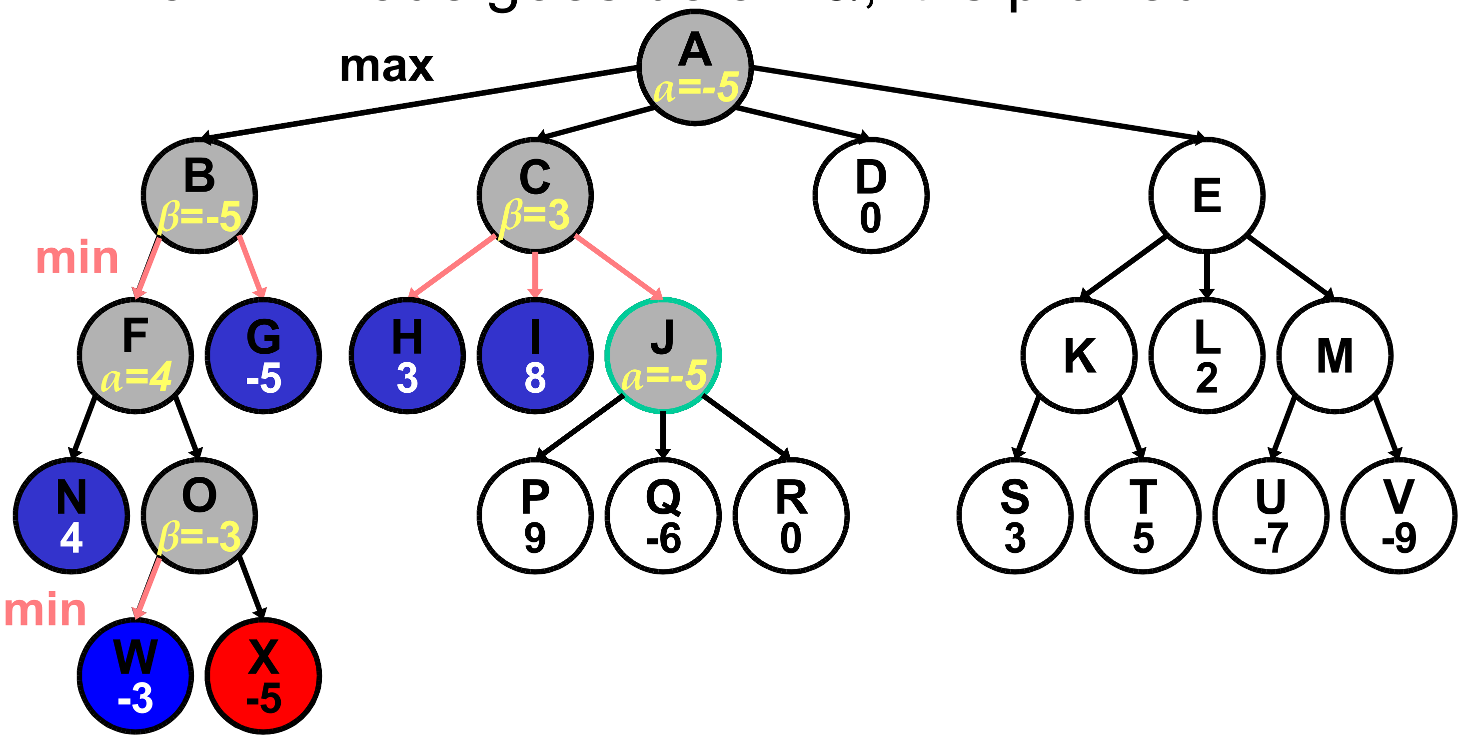
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

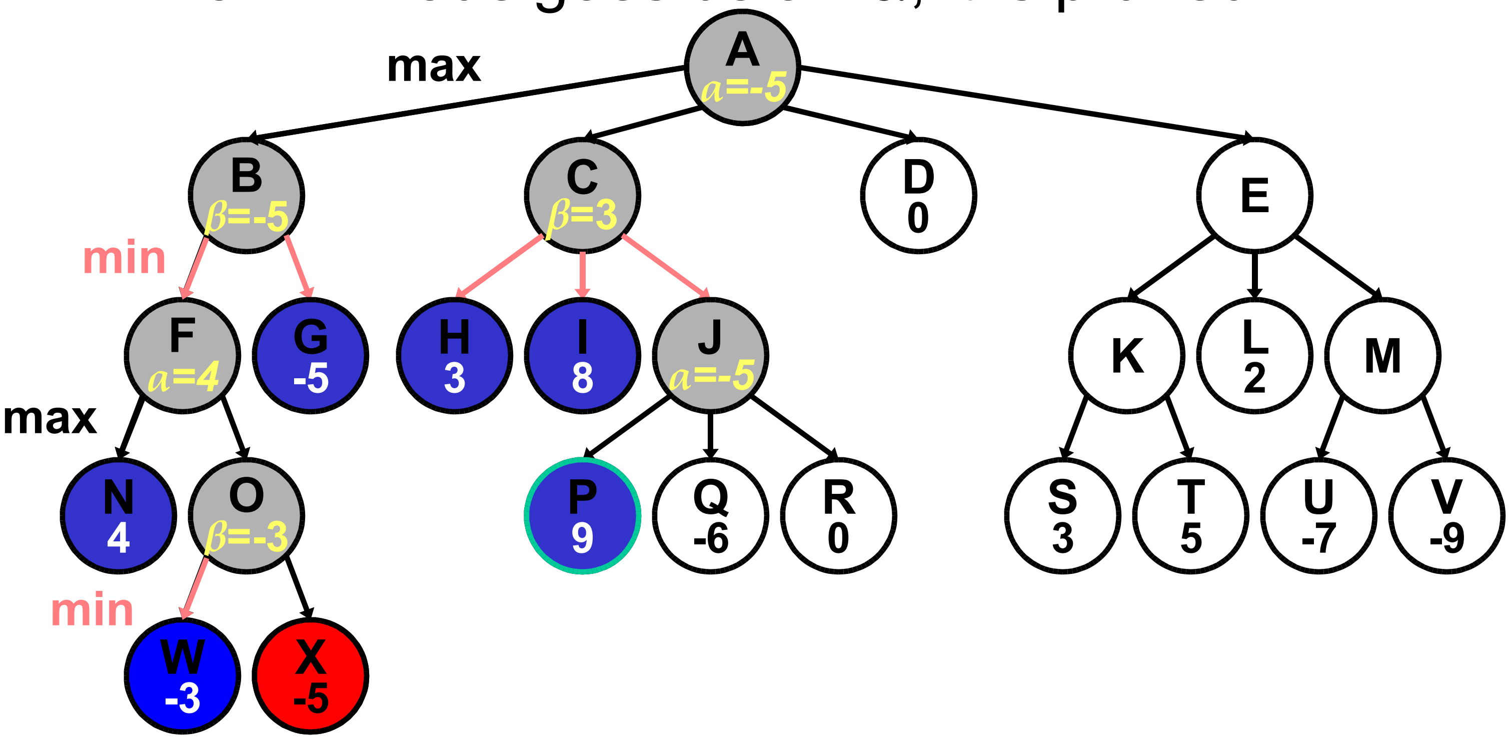
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

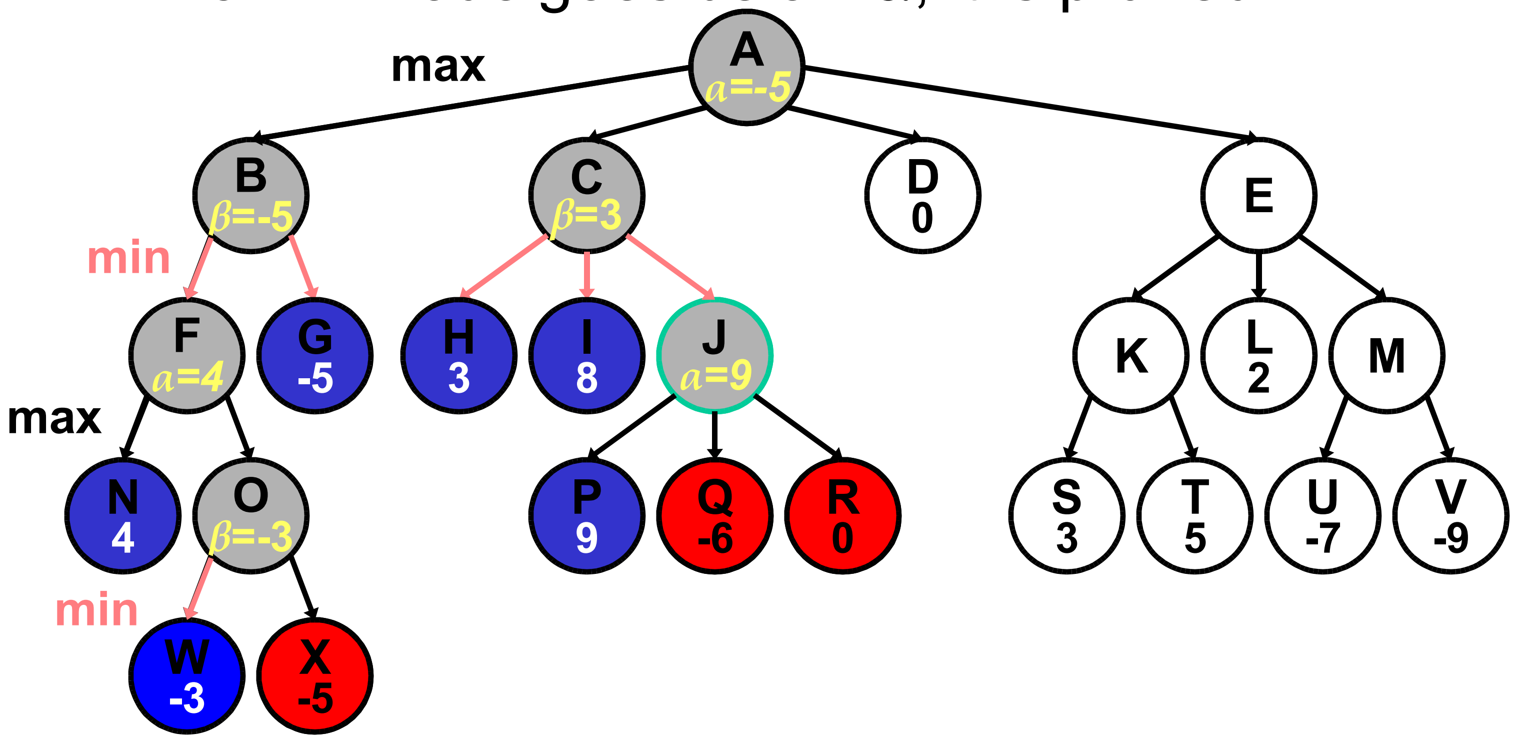
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

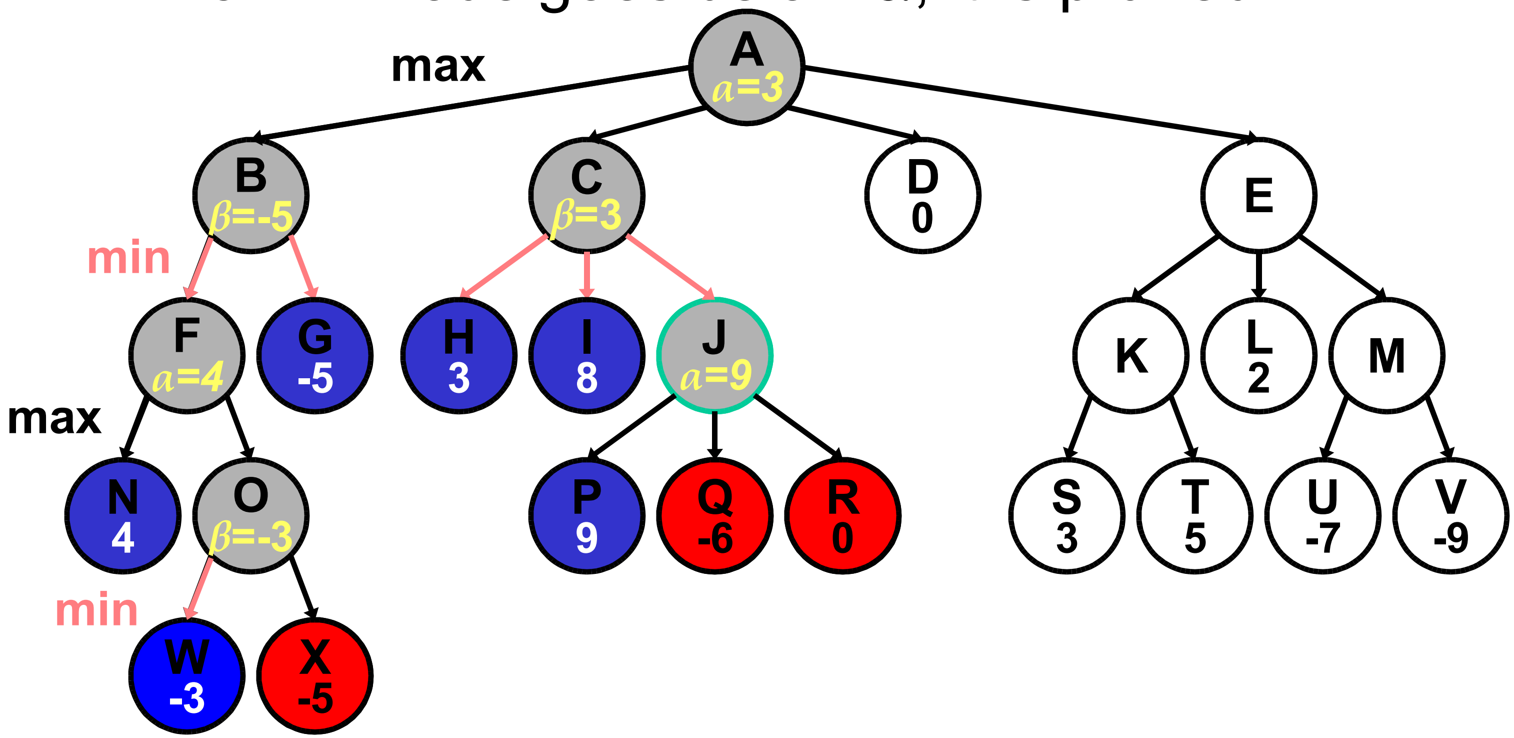
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

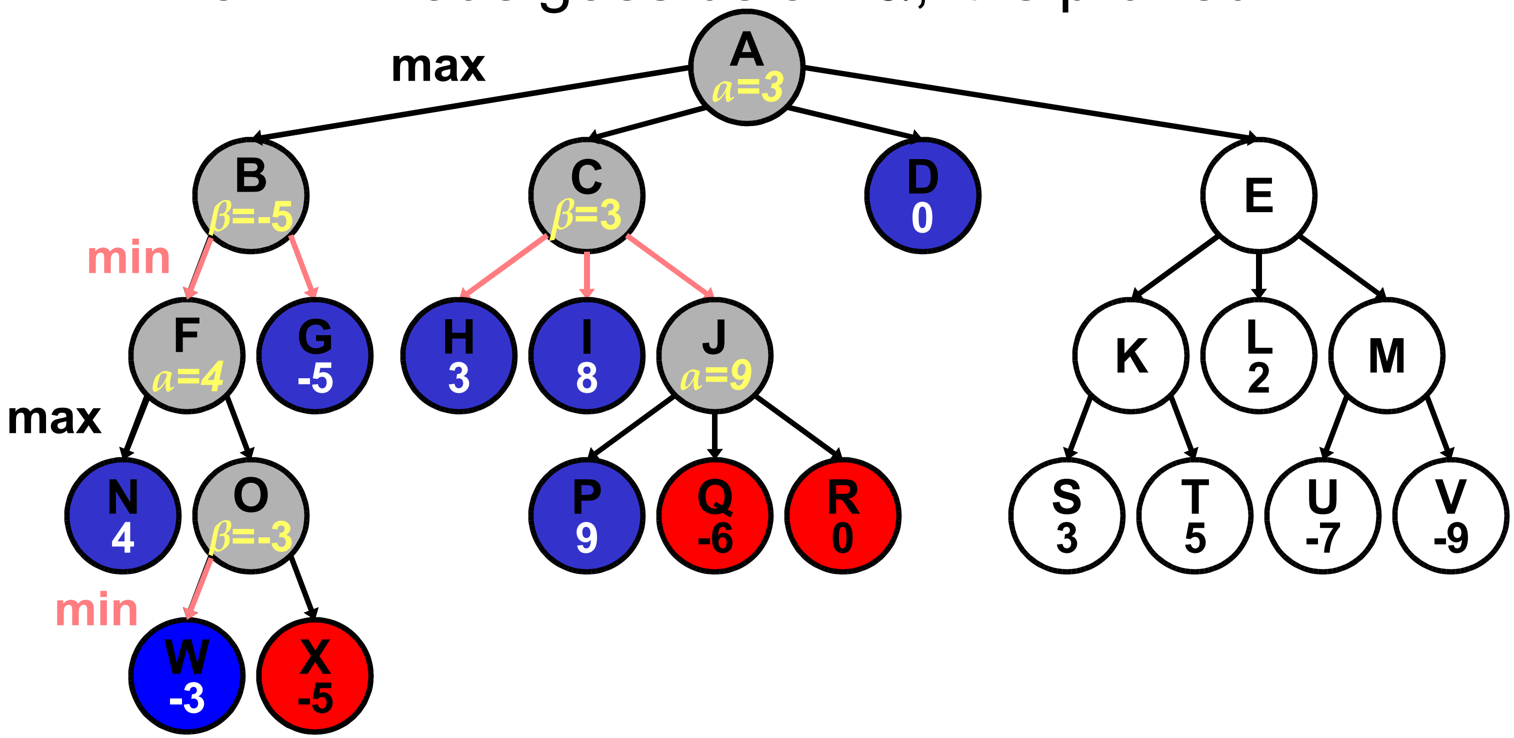
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

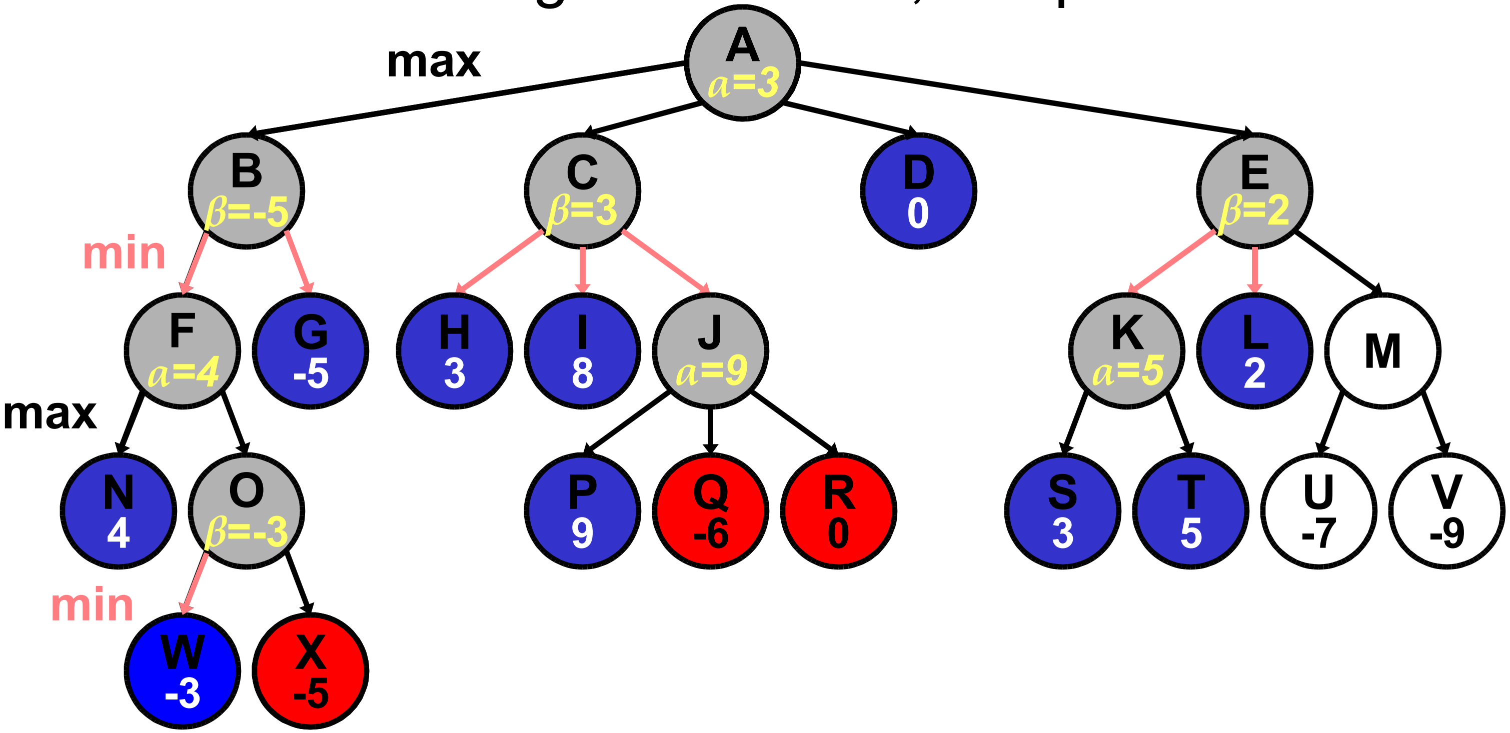
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

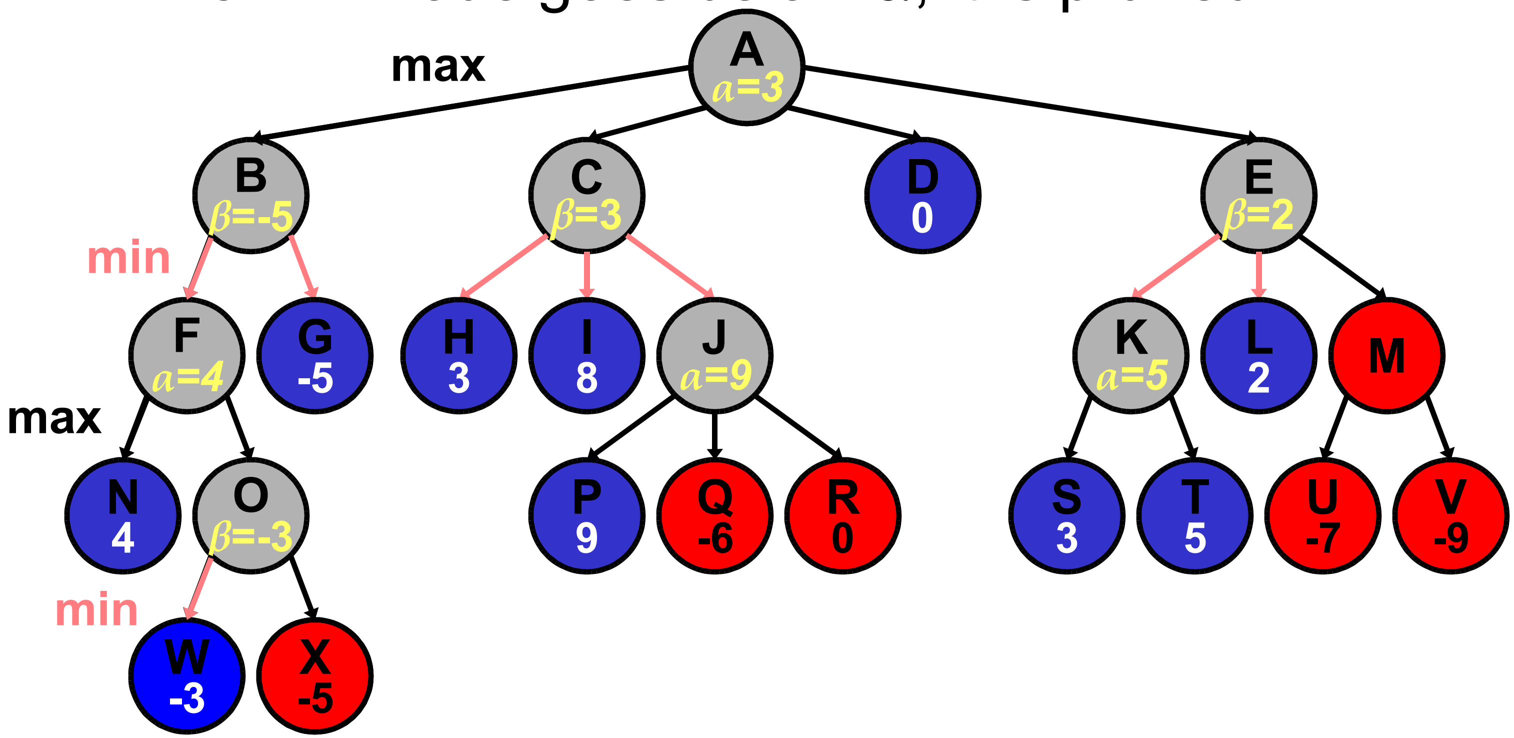
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

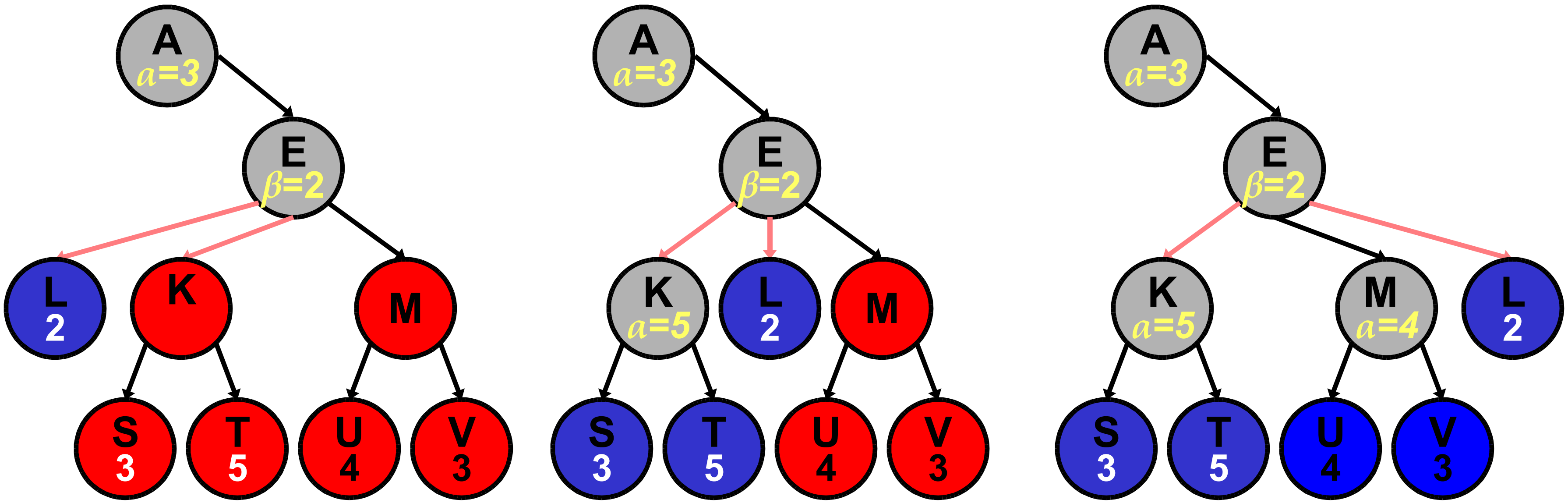
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

How effective is alpha-beta pruning?

- Depends on the order of successors!



- In the best case, the number of nodes to search is O ($b^{m/2}$), the square root of minimax's cost
- Still not practical for large games like chess

What you should know

- What is a two-player zero-sum discrete finite deterministic game of perfect information
- What is a game tree
- What is the minimax value of a game
- Minimax search
- Alpha-beta pruning