Game Playing
Part 2 Alpha-Beta Pruning

Yingyu Liang

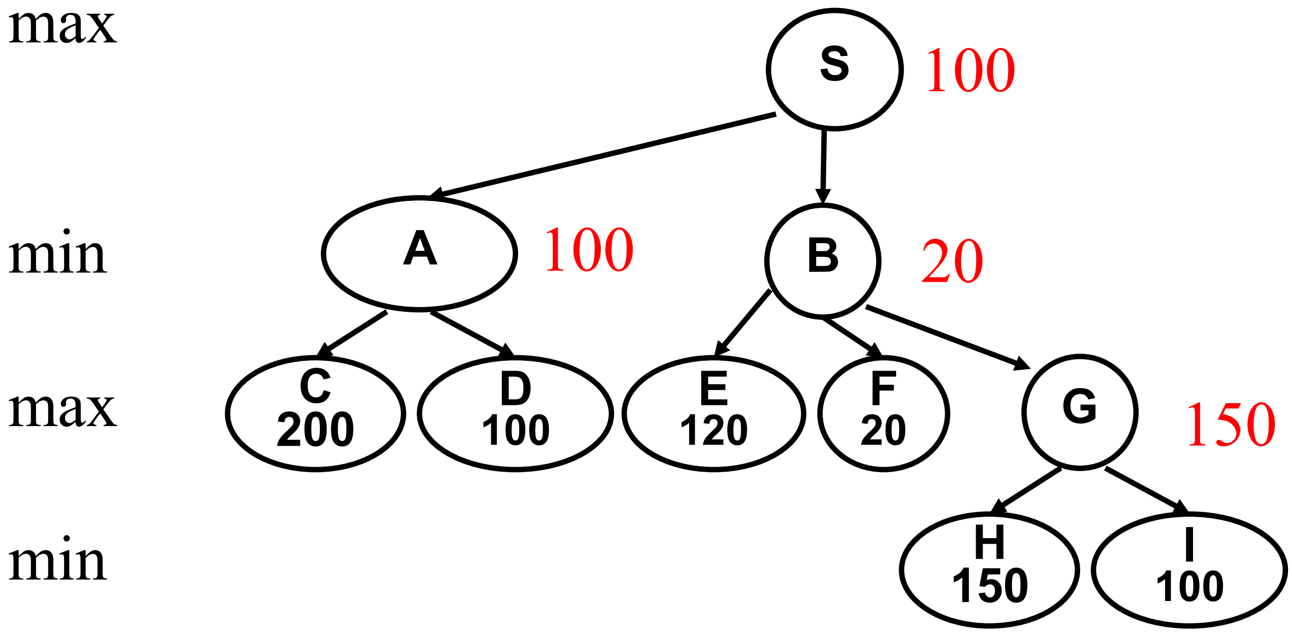
yliang@cs.wisc.edu

Computer Sciences Department

University of Wisconsin, Madison

[based on slides from A. Moore, C. Dyer, J. Skrentny, Jerry Zhu]

Review: Game Theoretic Value



The bottom-up approach: needs the whole game tree. Too much space!

The minimax algorithm: a variant of DFS to save space.

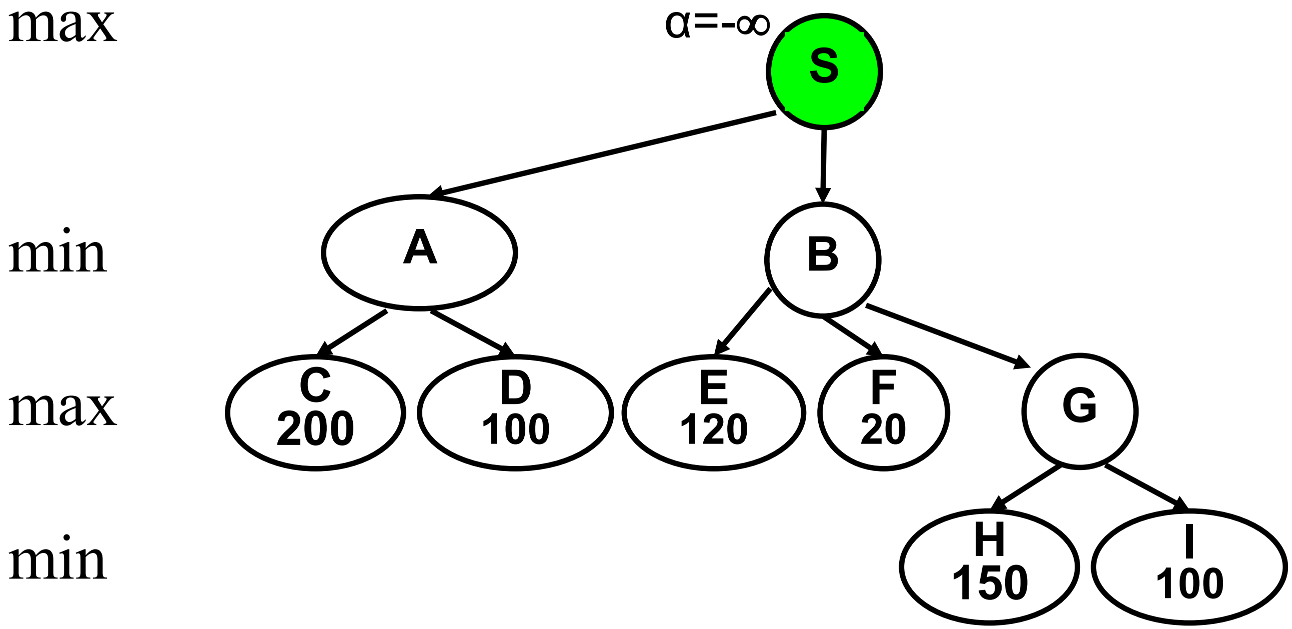
Review: Minimax algorithm

```
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
```

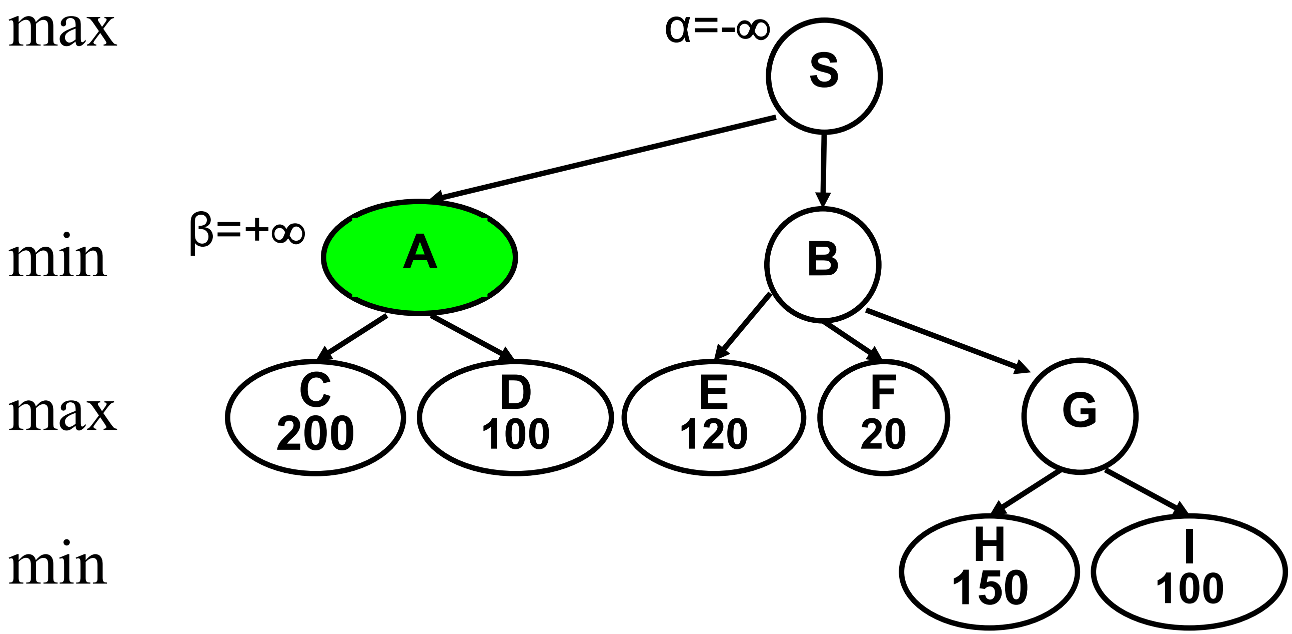
```
function Min-Value(s)
output: best-score (for Min) available from s

    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? $O(bm)$

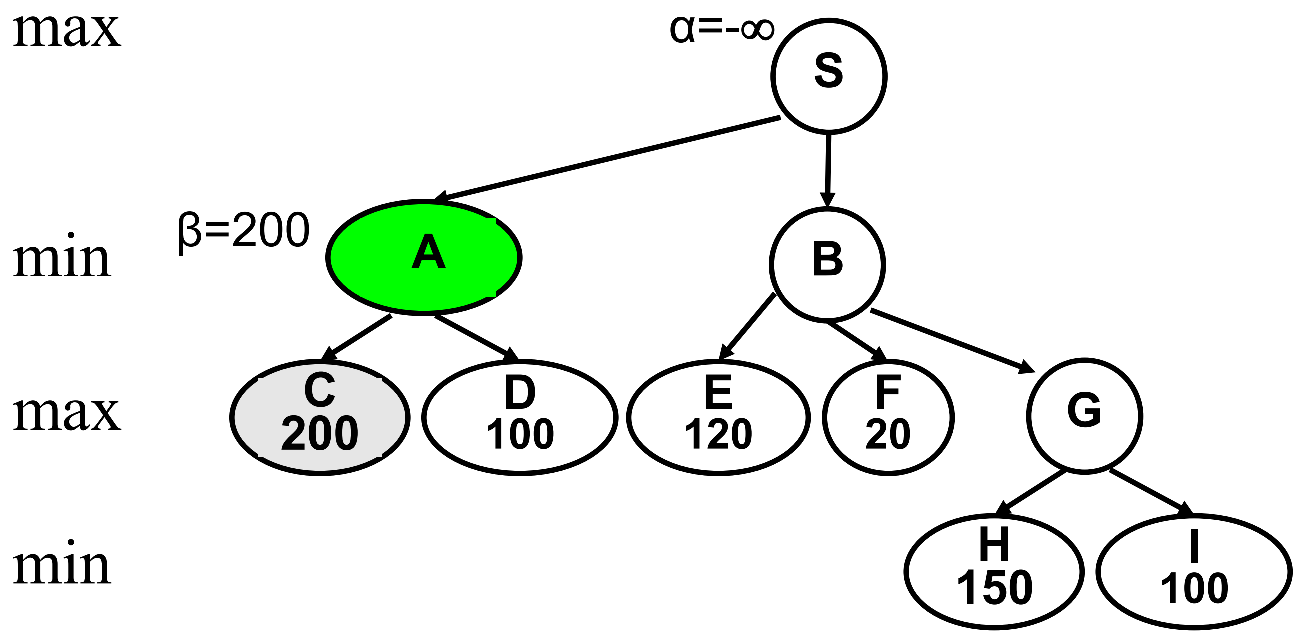
Minimax algorithm in execution



Minimax algorithm in execution

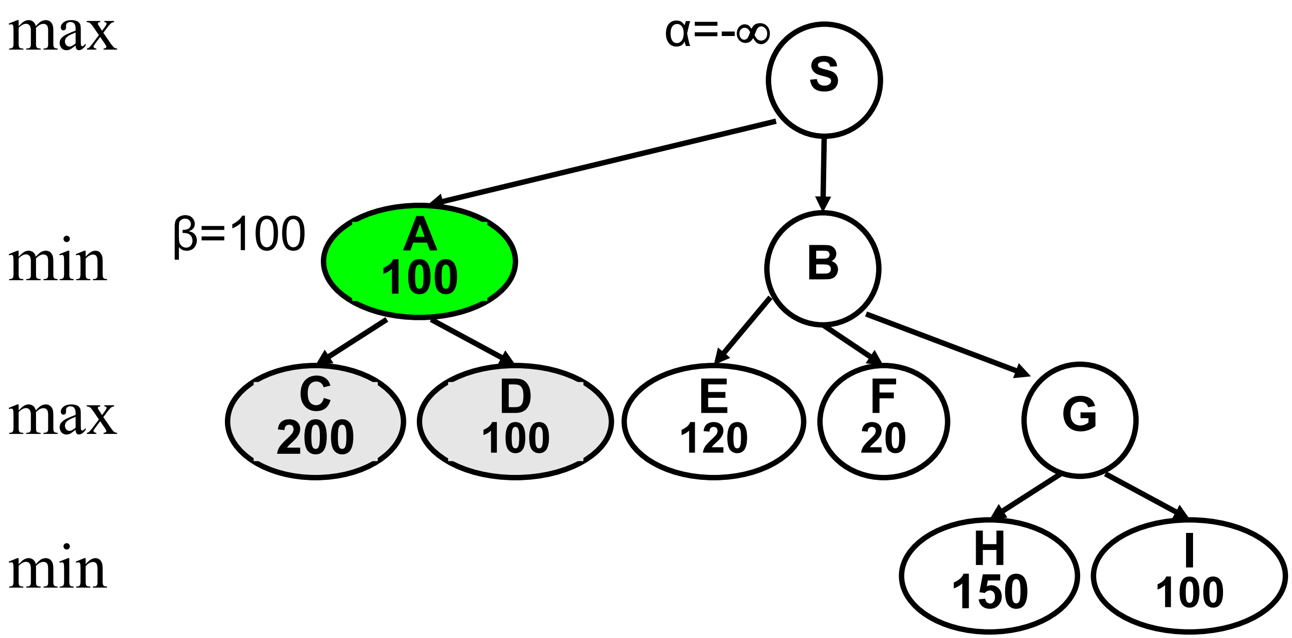


Minimax algorithm in execution

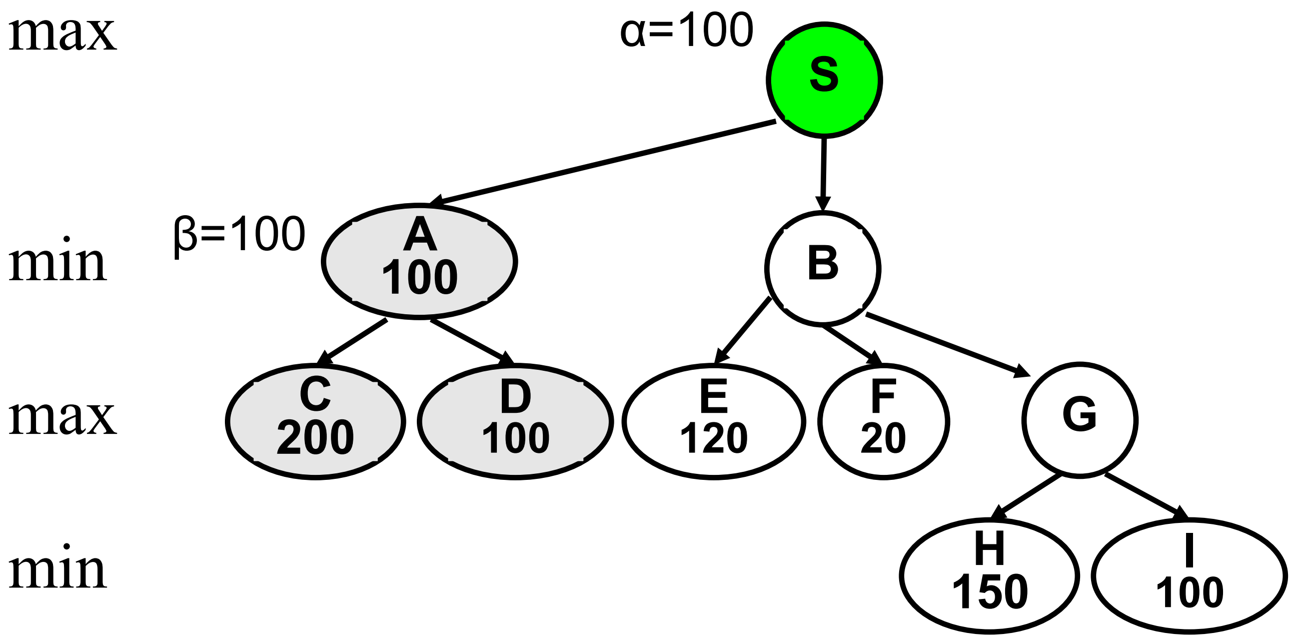


The execution on the
terminal nodes is omitted.

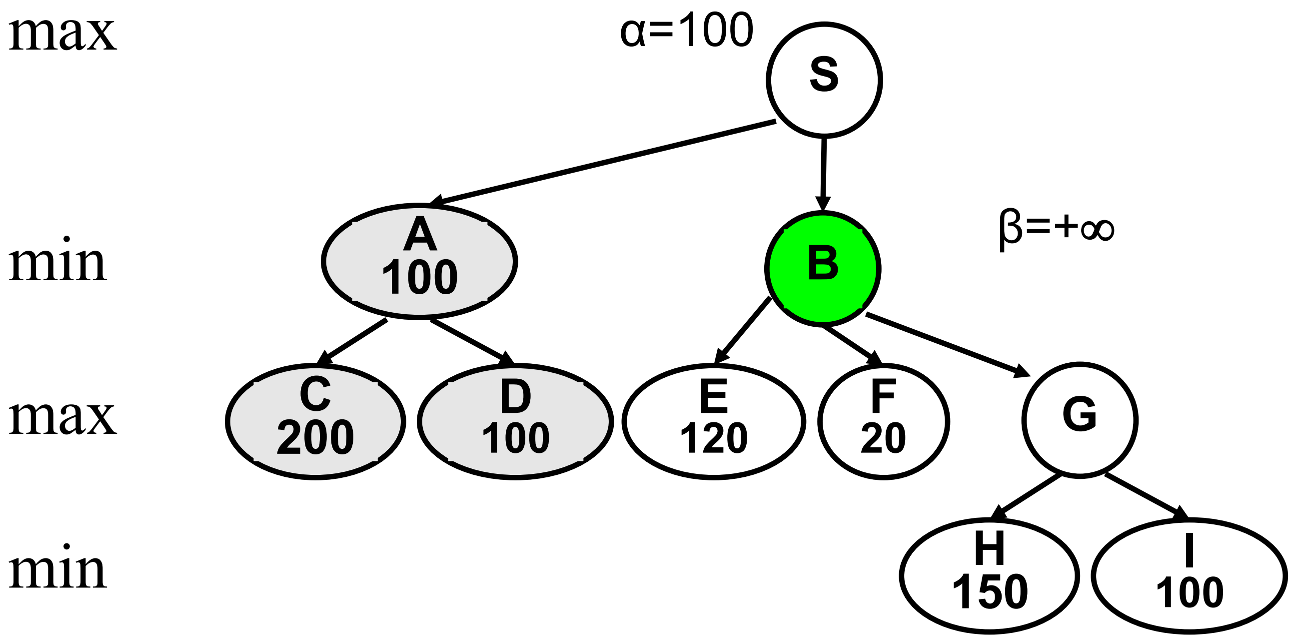
Minimax algorithm in execution



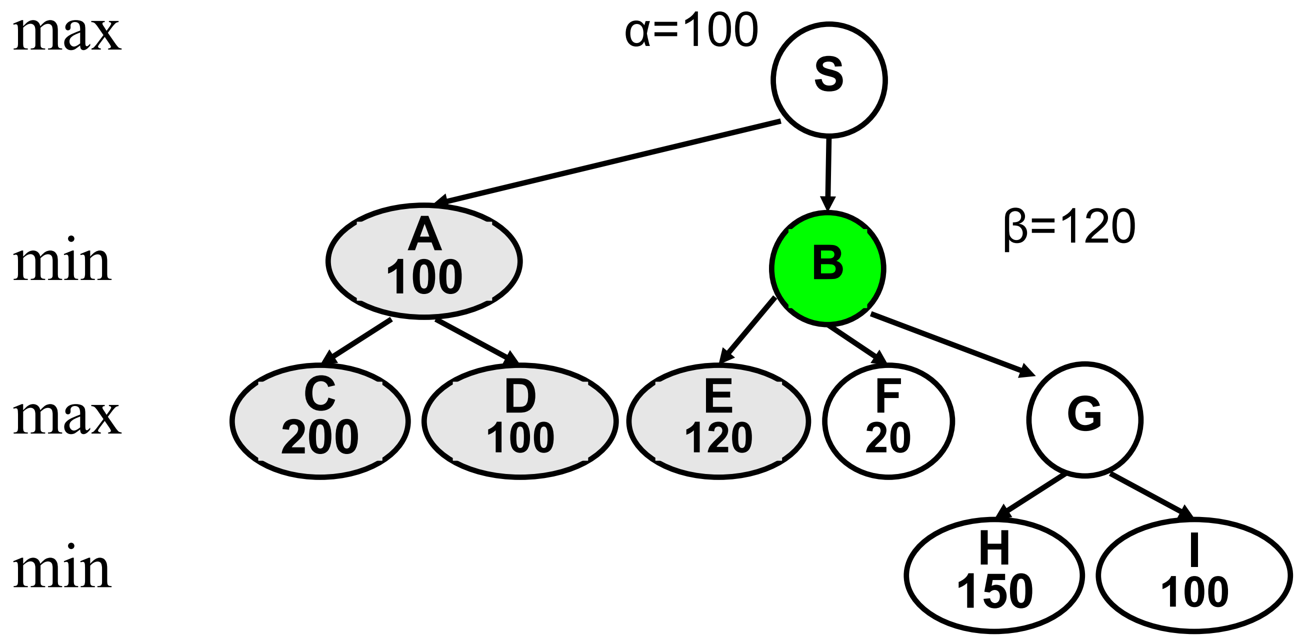
Minimax algorithm in execution



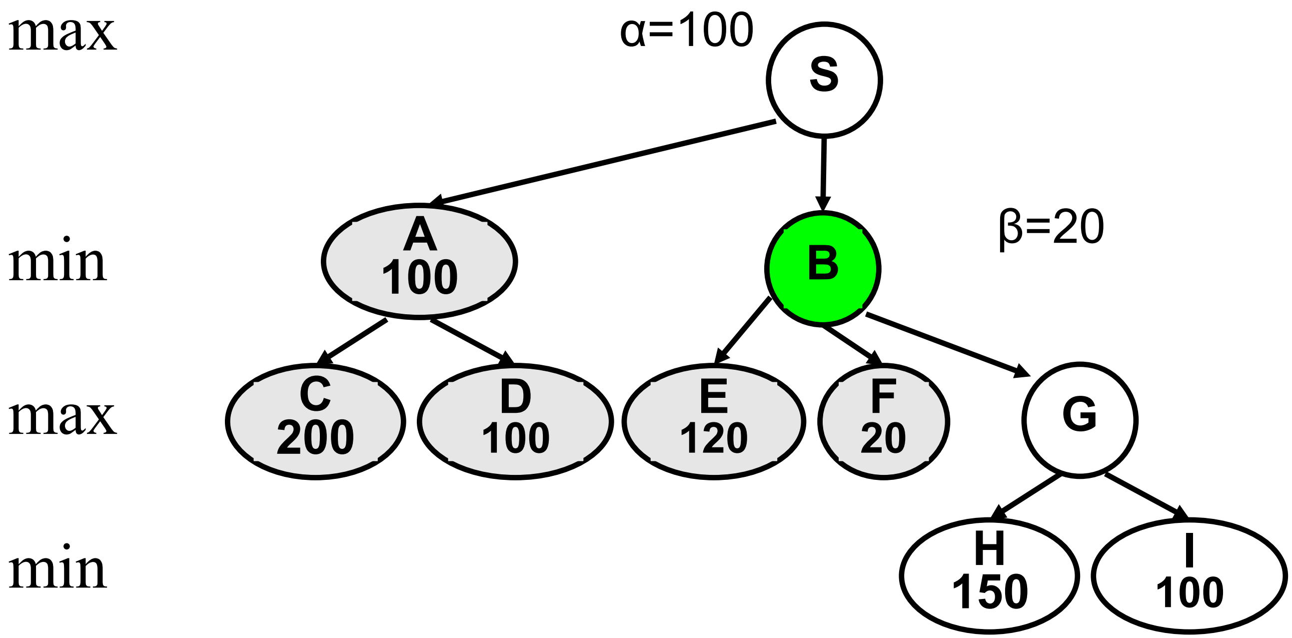
Minimax algorithm in execution



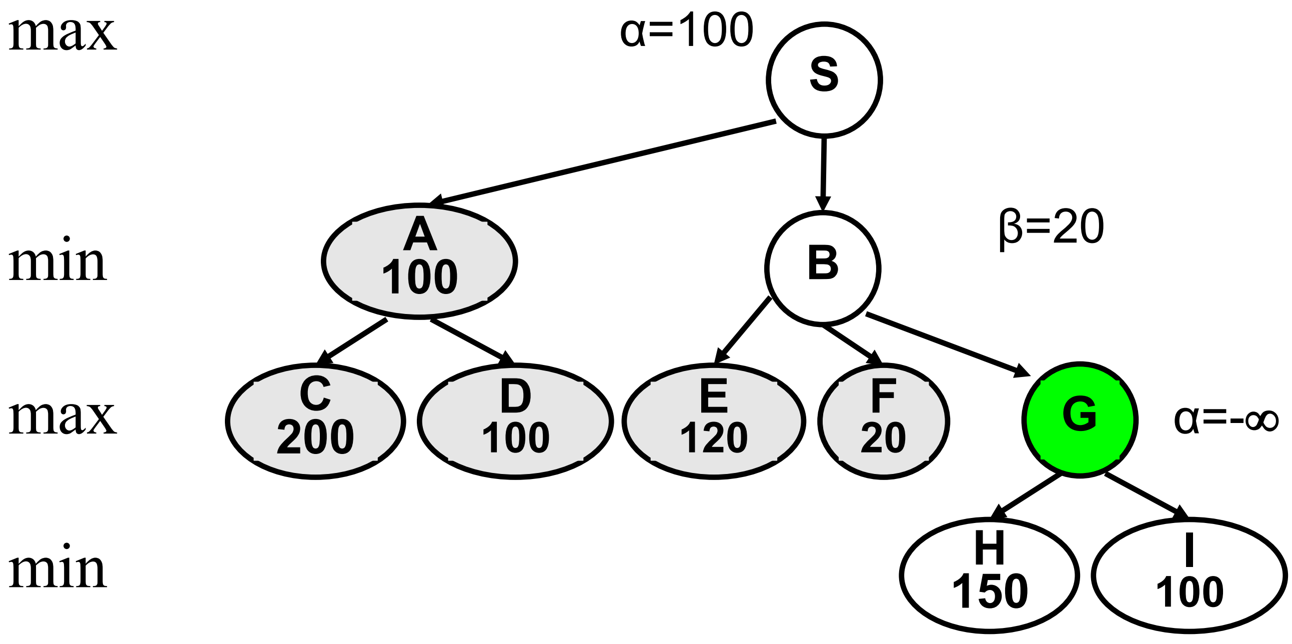
Minimax algorithm in execution



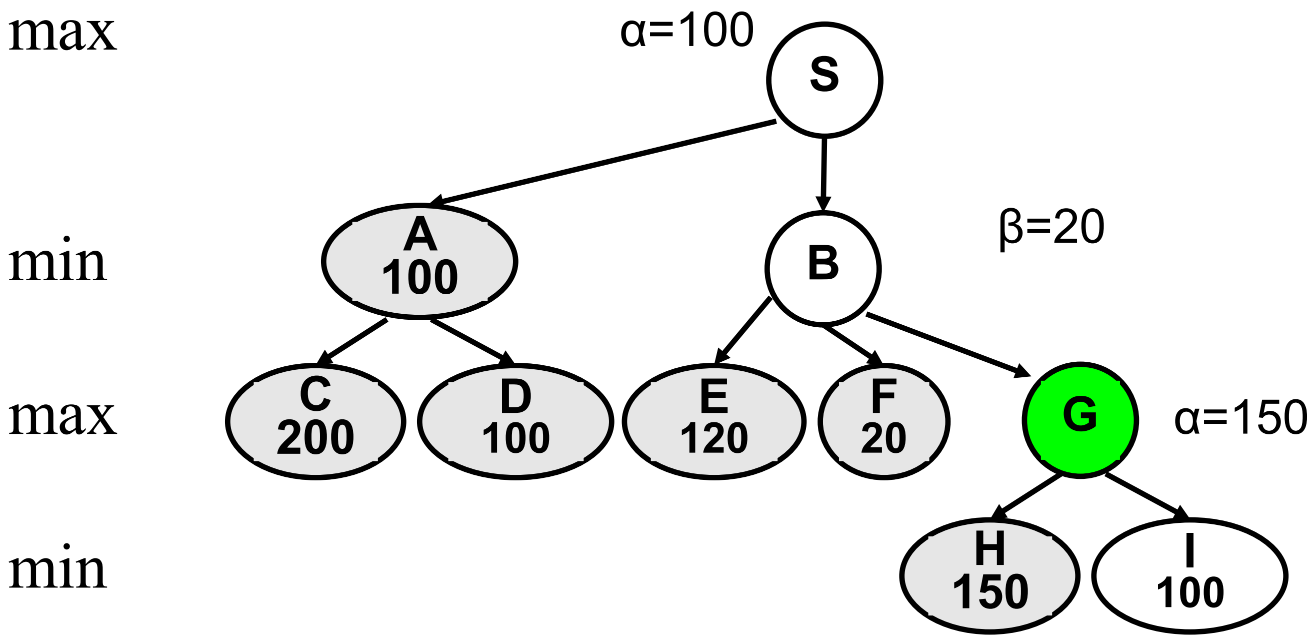
Minimax algorithm in execution



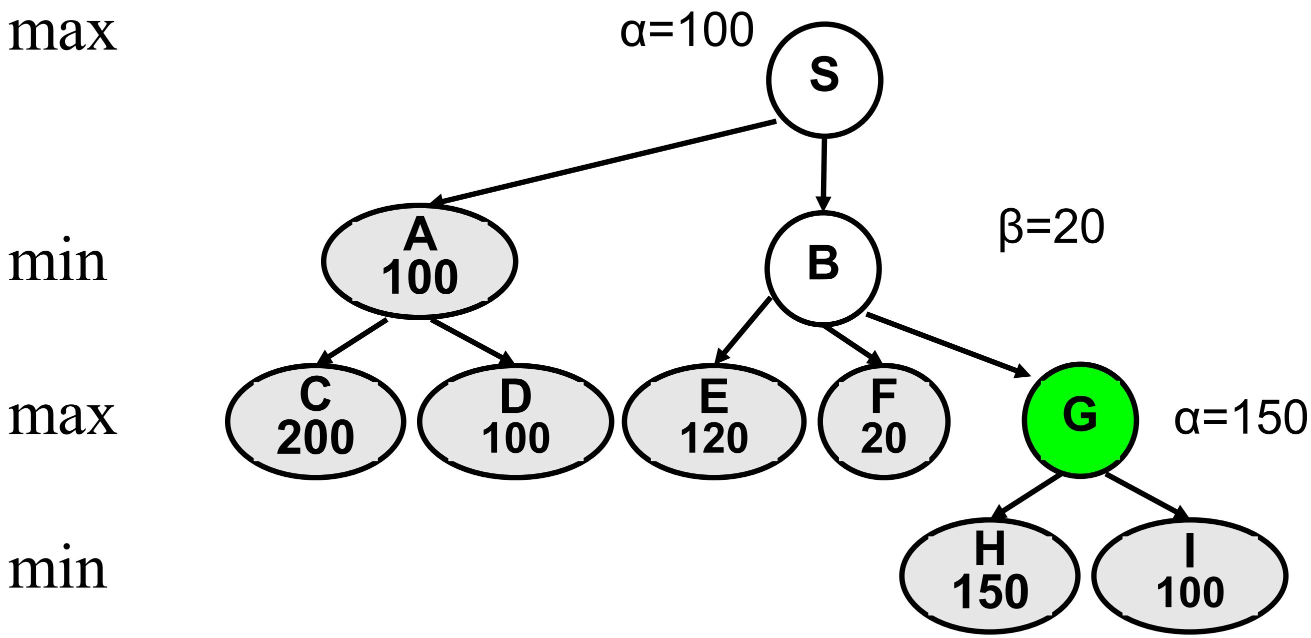
Minimax algorithm in execution



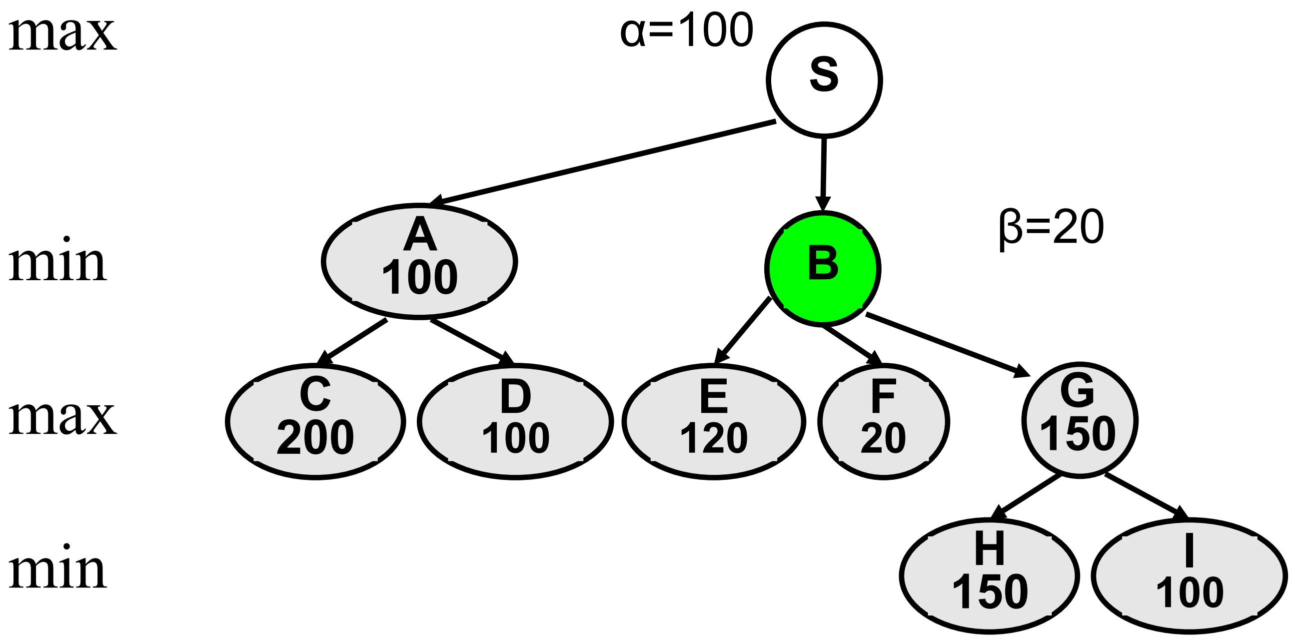
Minimax algorithm in execution



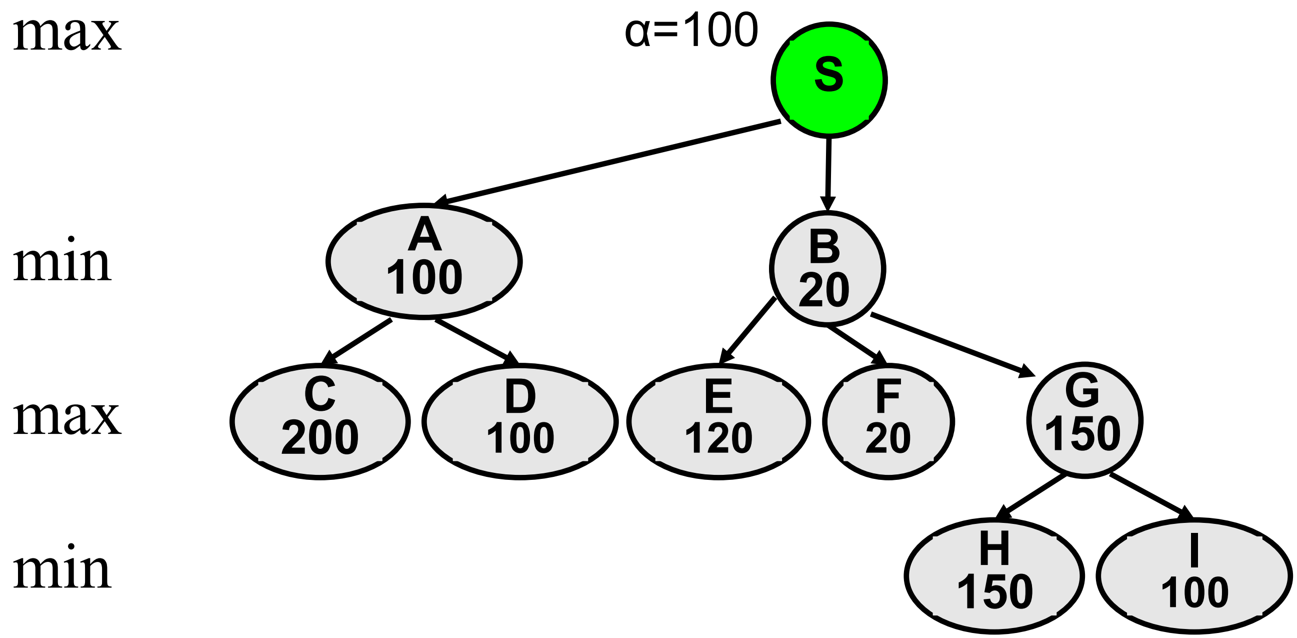
Minimax algorithm in execution



Minimax algorithm in execution



Minimax algorithm in execution

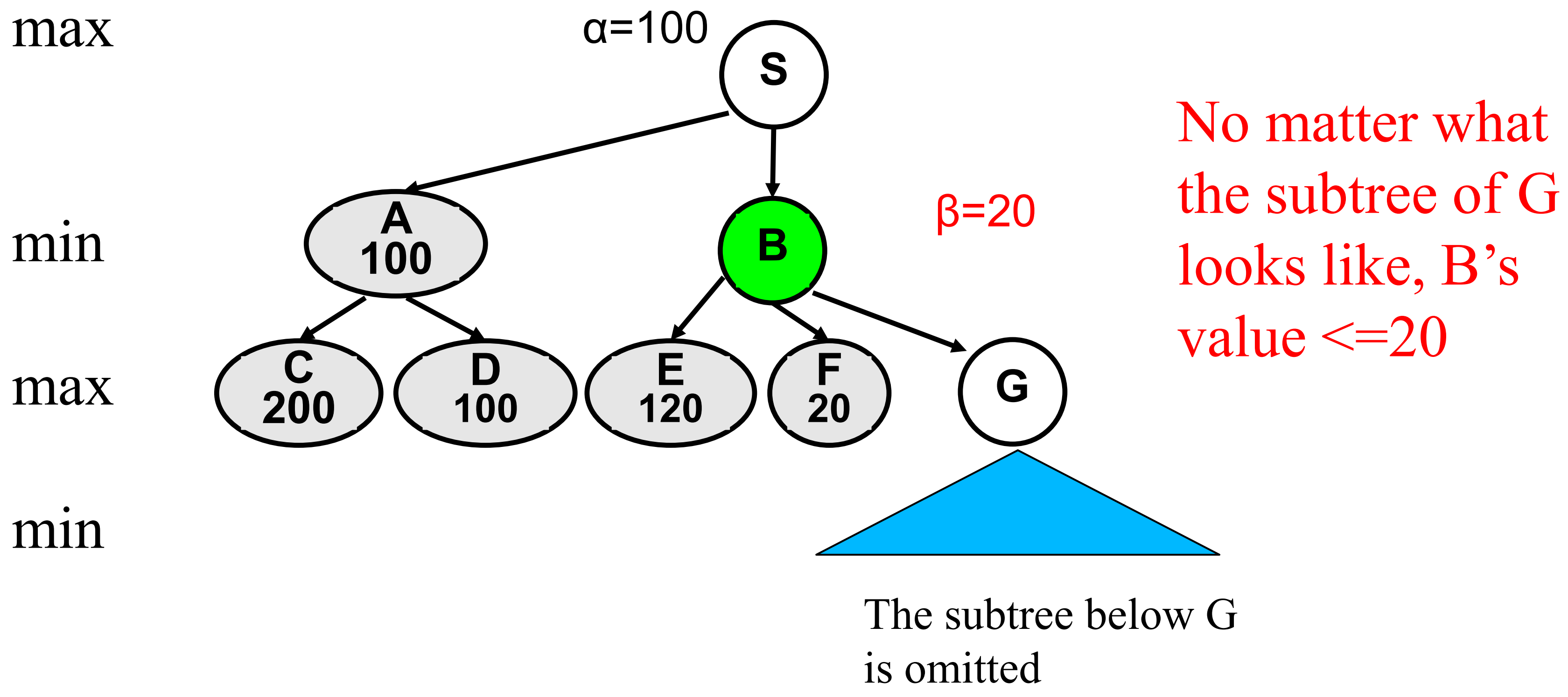


alpha-beta pruning

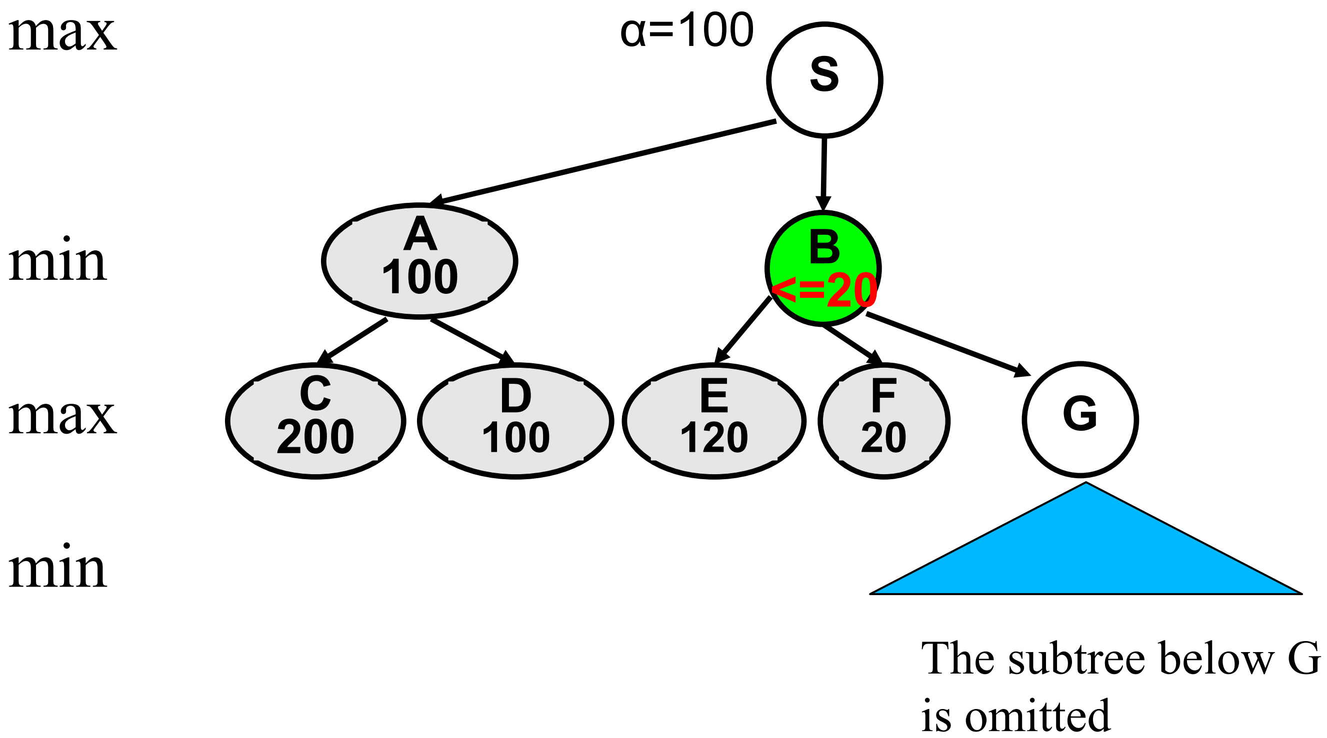
Gives the same game theoretic values as minimax, but prunes part of the game tree.

"If you have an idea that is surely bad, don't take the time to see how truly awful it is." -- Pat Winston

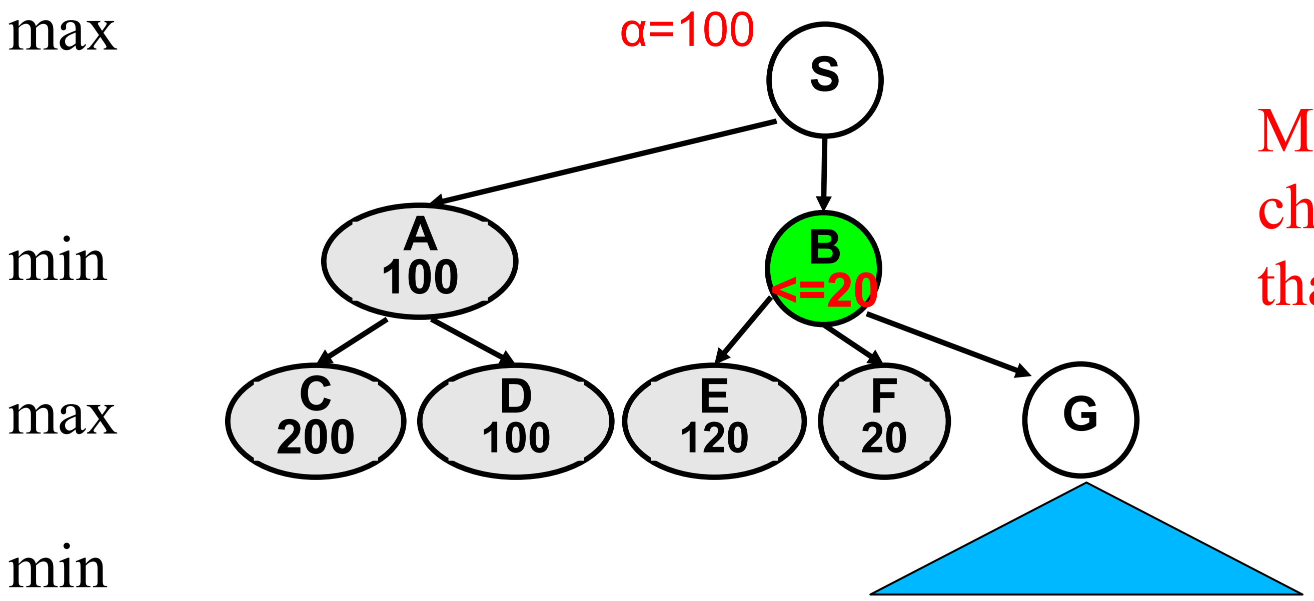
Recall: Minimax algorithm in execution



Recall: Minimax algorithm in execution



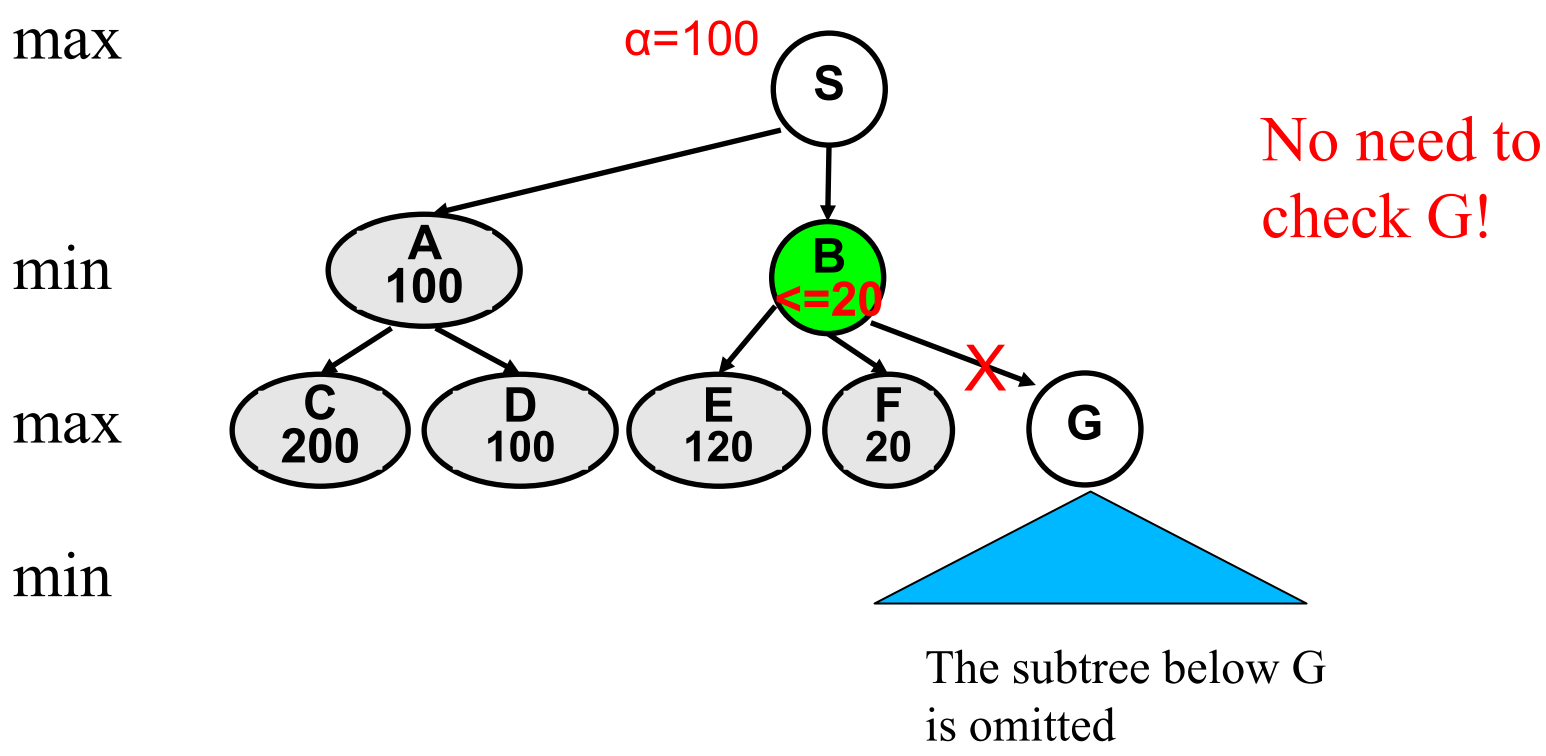
Recall: Minimax algorithm in execution



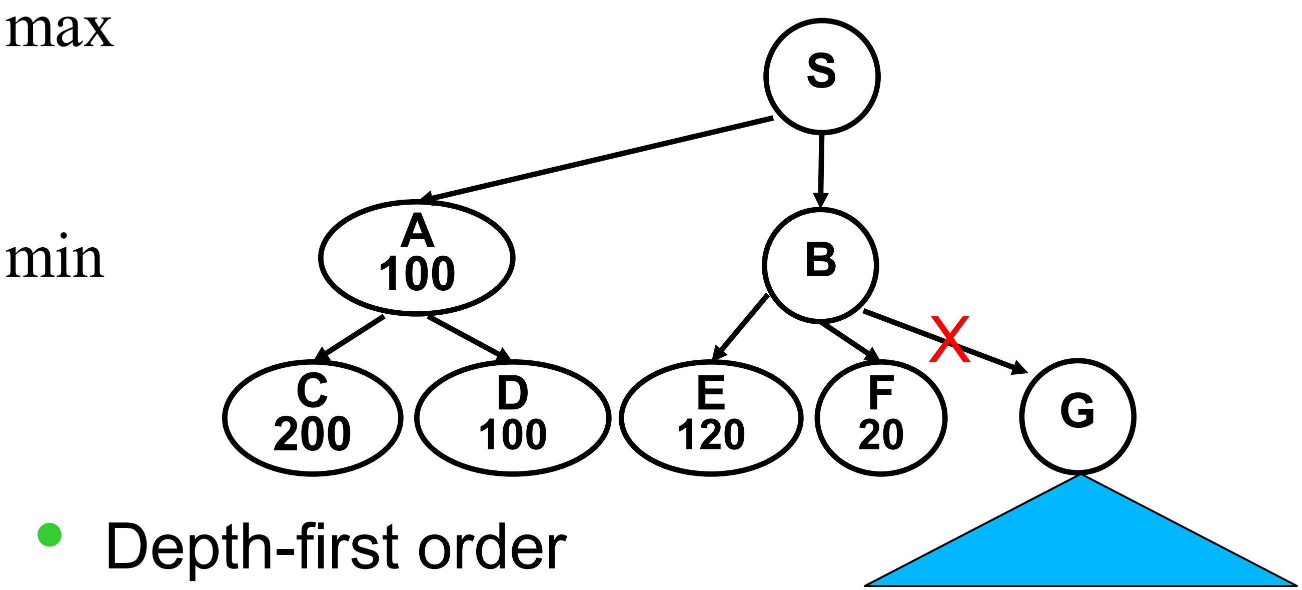
The subtree below G is omitted

Max on S must choose A rather than B.

Recall: Minimax algorithm in execution

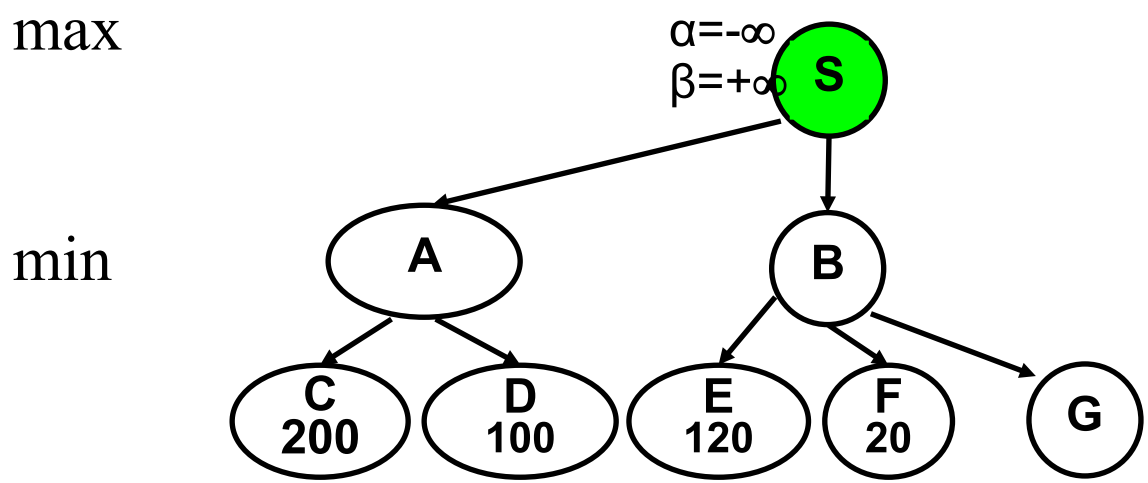


Alpha-Beta Motivation Summary



- Depth-first order
- After returning from A, Max can get at least 100 at S
- After returning from F, Max can get at most 20 at B
- At this point, Max losts interest in B
- There is no need to explore G. The subtree at G is pruned. Saves time.

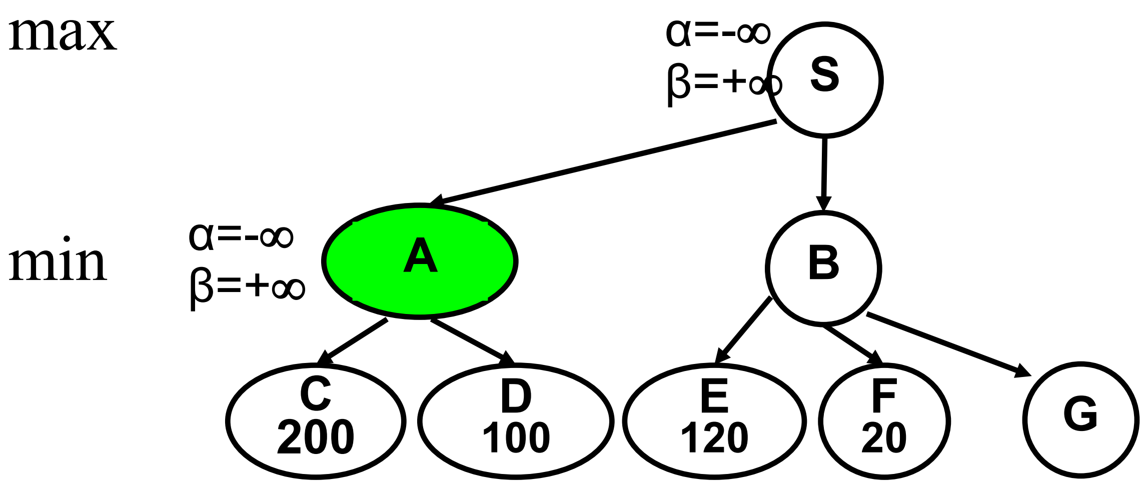
Alpha-beta pruning example 1



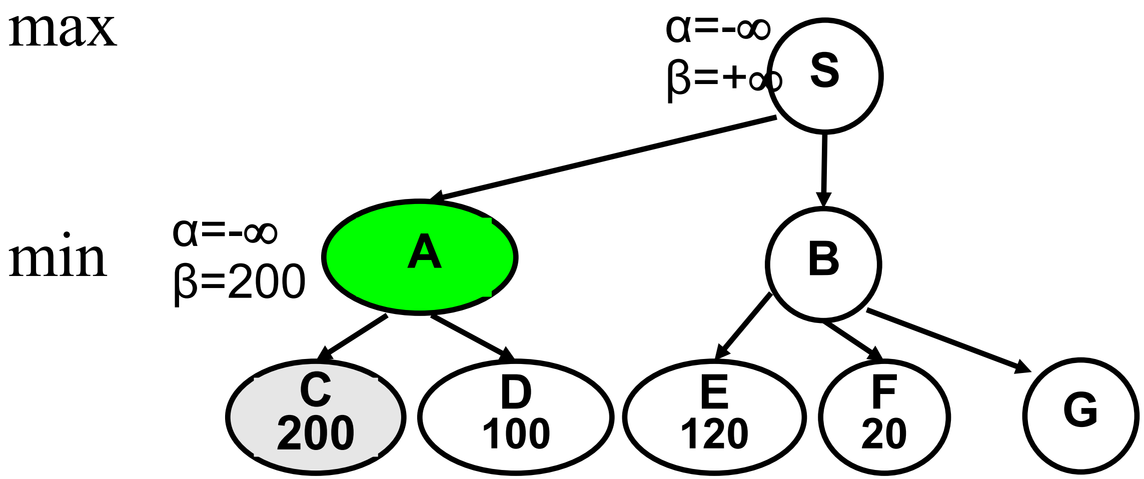
The subtree below G is omitted

- Keep two bounds along the path
  - α: the best Max can do
  - β: the best (smallest) Min can do
- If at anytime α exceeds β, the remaining children are pruned.
  - Case 1: happens on a Min node
  - Case 2: happens on a Max node

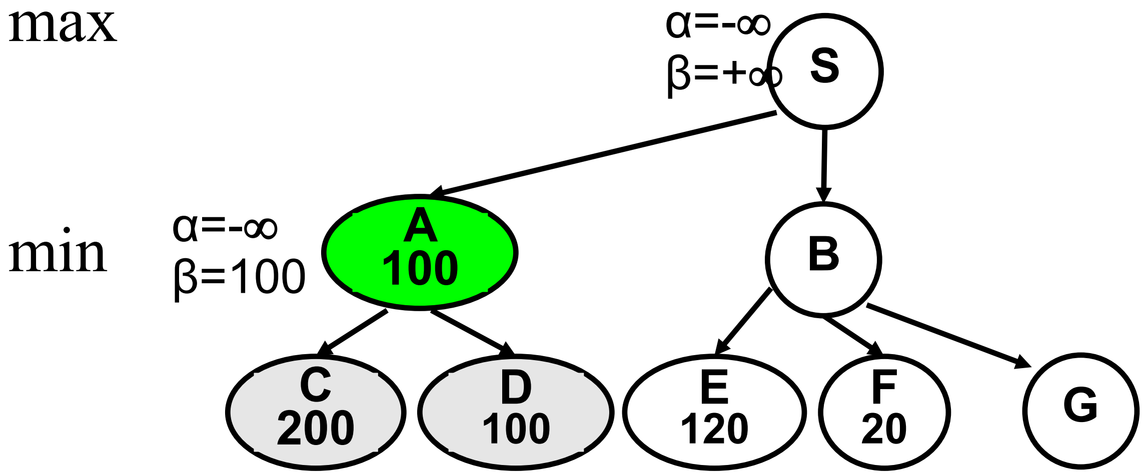
Alpha-beta pruning example 1



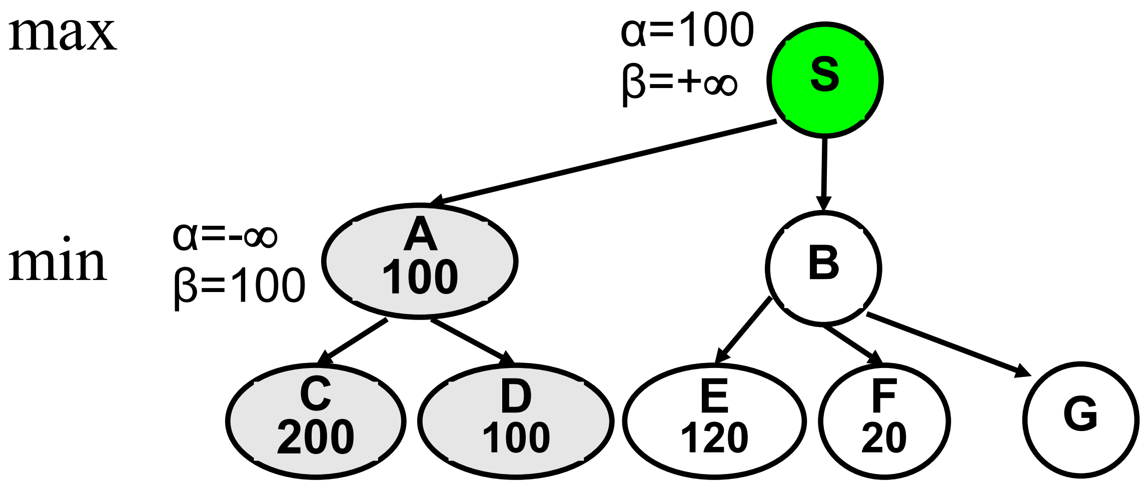
Alpha-beta pruning example 1



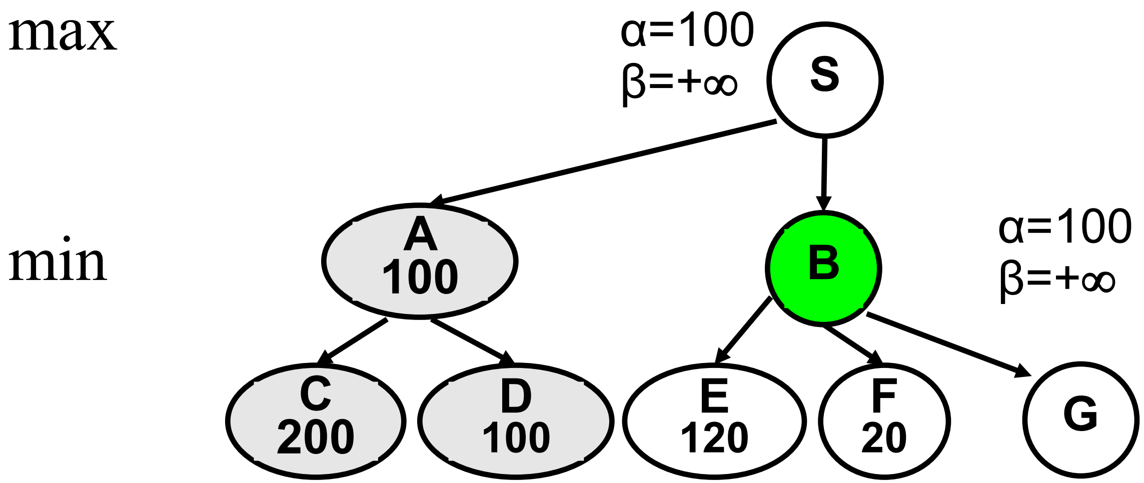
Alpha-beta pruning example 1



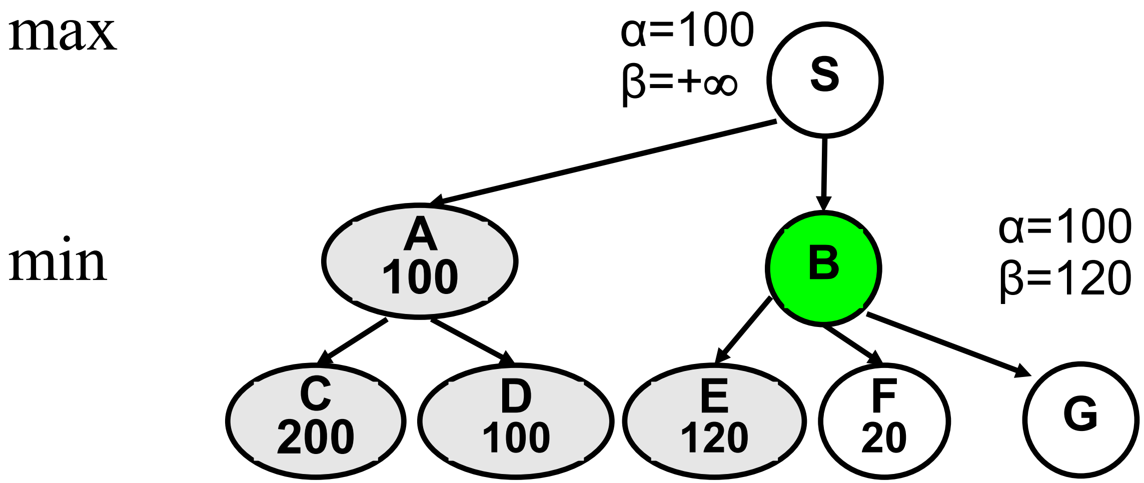
Alpha-beta pruning example 1



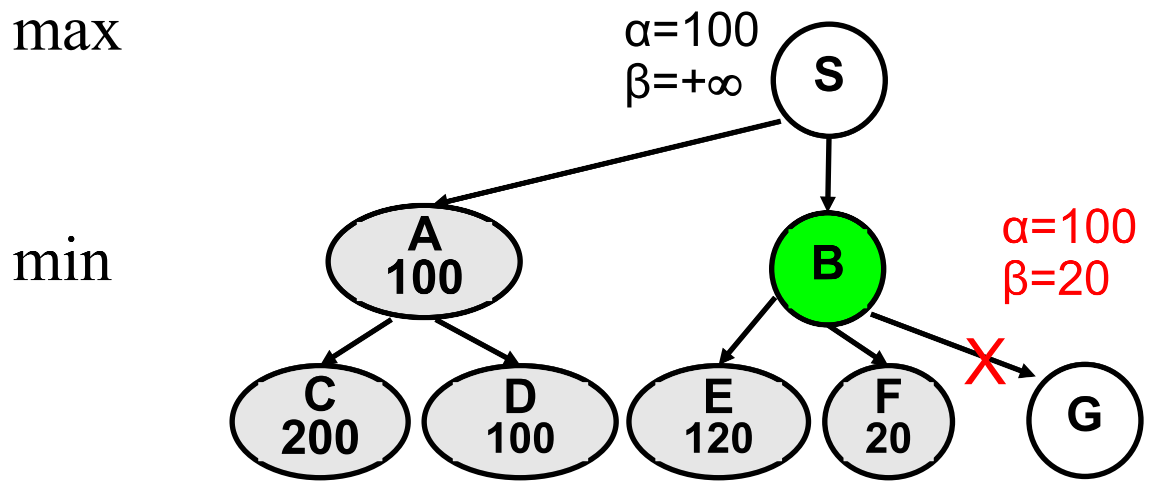
Alpha-beta pruning example 1



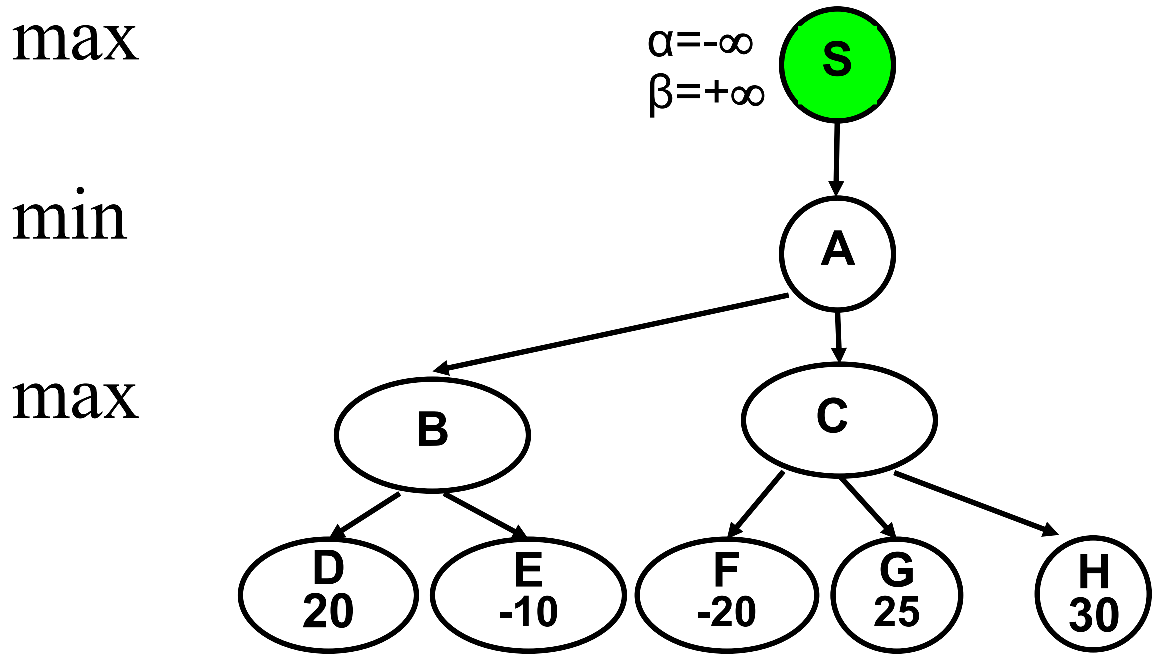
Alpha-beta pruning example 1



Alpha-beta pruning example 1

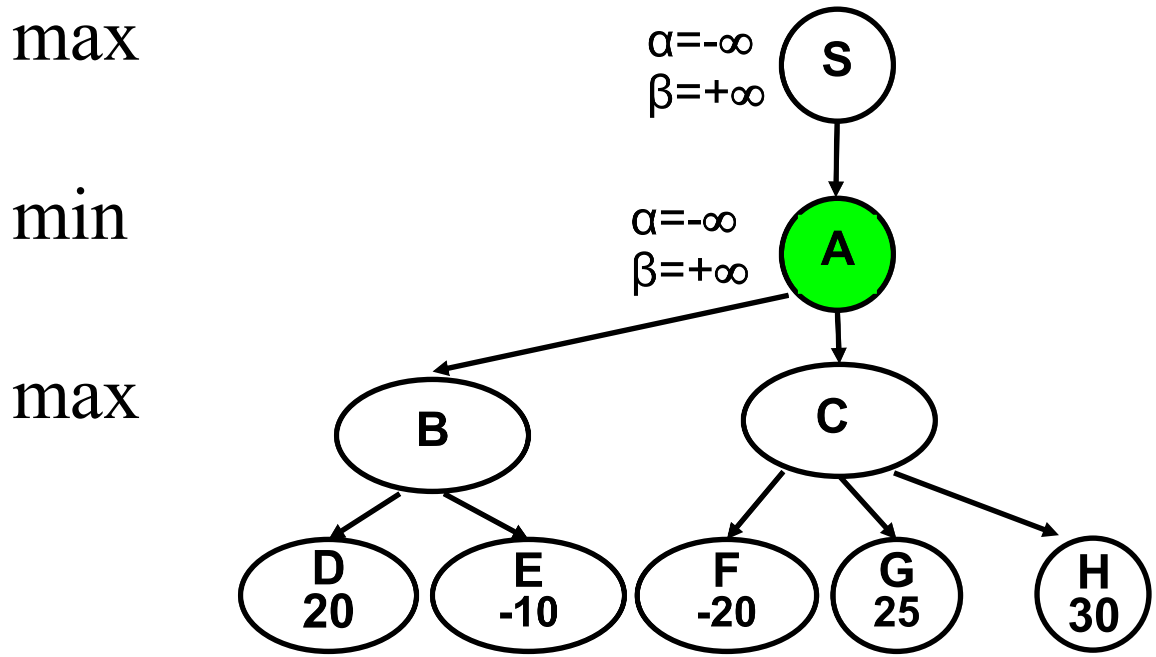


Alpha-beta pruning example 2



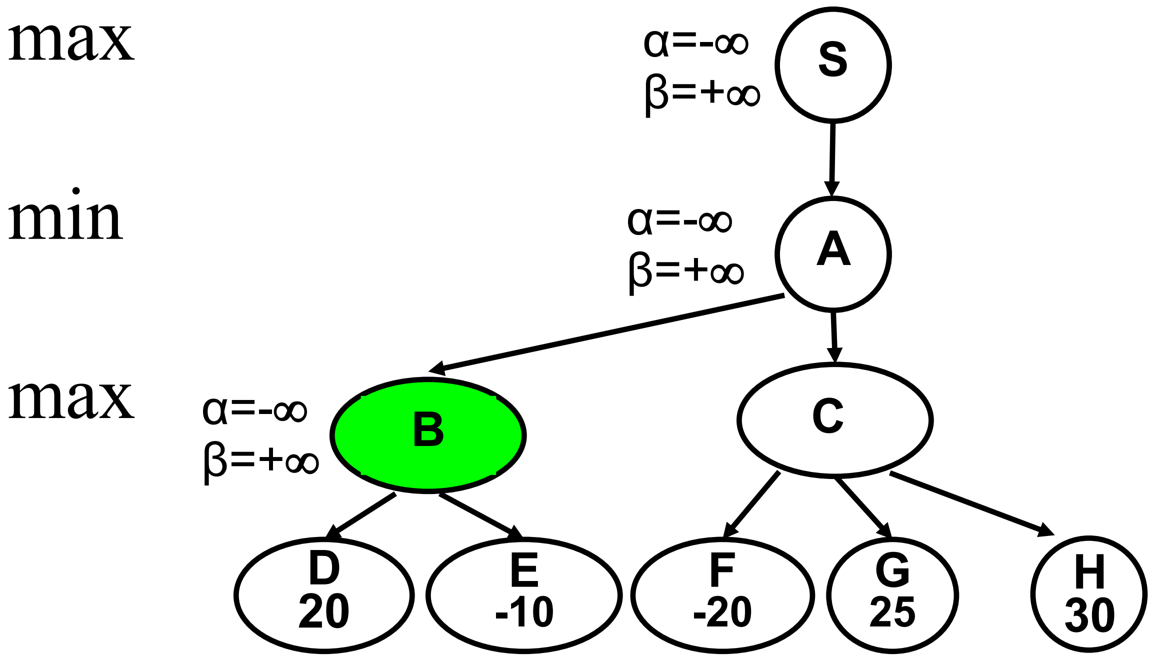
- Keep two bounds along the path
  - α: the best Max can do
  - β: the best (smallest) Min can do
- If at anytime α exceeds β, the remaining children are pruned.
  - Case 1: on a Min node; Case 2: on a Max node

Alpha-beta pruning example 2



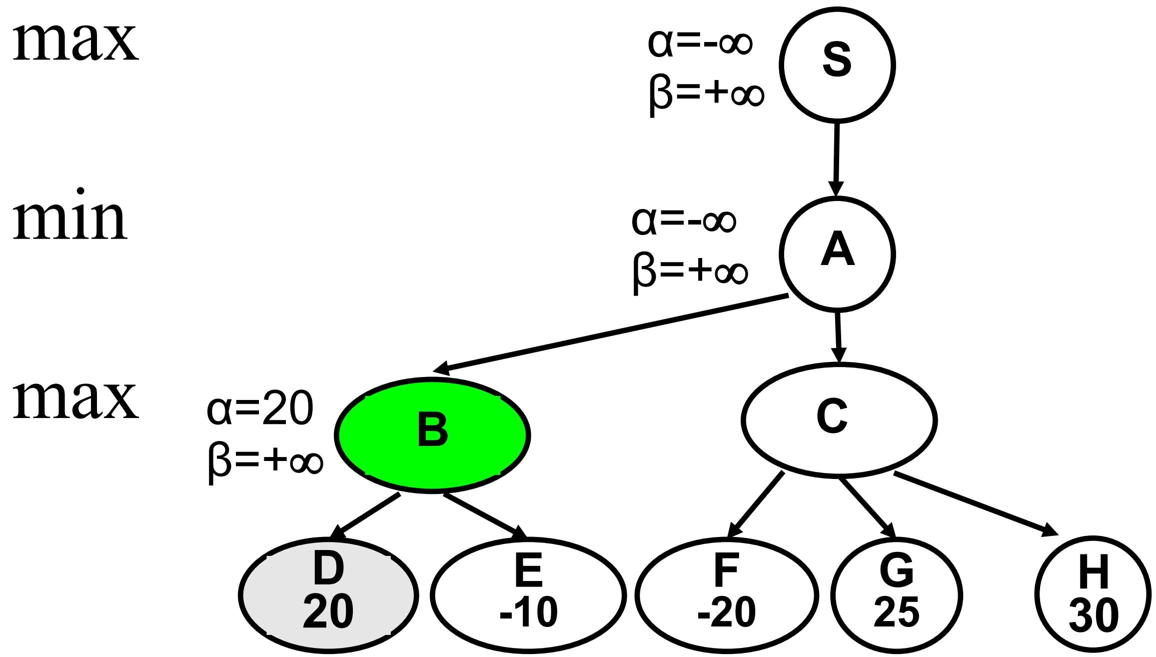
- Keep two bounds along the path
  - α: the best Max can do
  - β: the best (smallest) Min can do
- If at anytime α exceeds β, the remaining children are pruned.

Alpha-beta pruning example 2



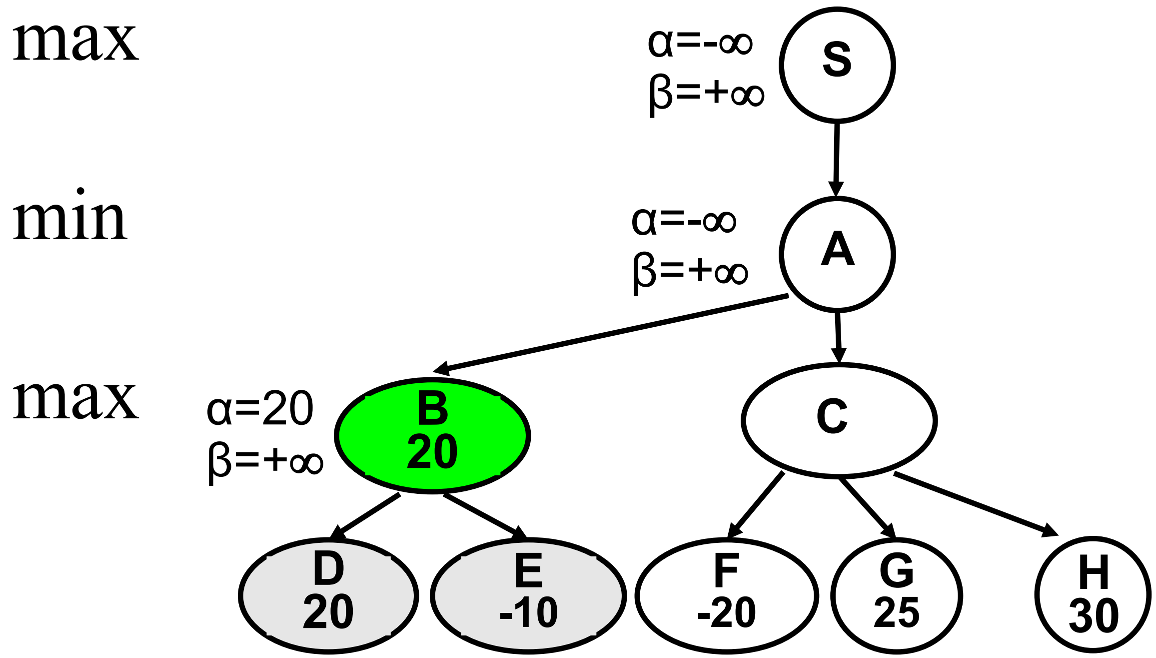
- Keep two bounds along the path
  - α: the best Max can do
  - β: the best (smallest) Min can do
- If at anytime α exceeds β, the remaining children are pruned.

Alpha-beta pruning example 2



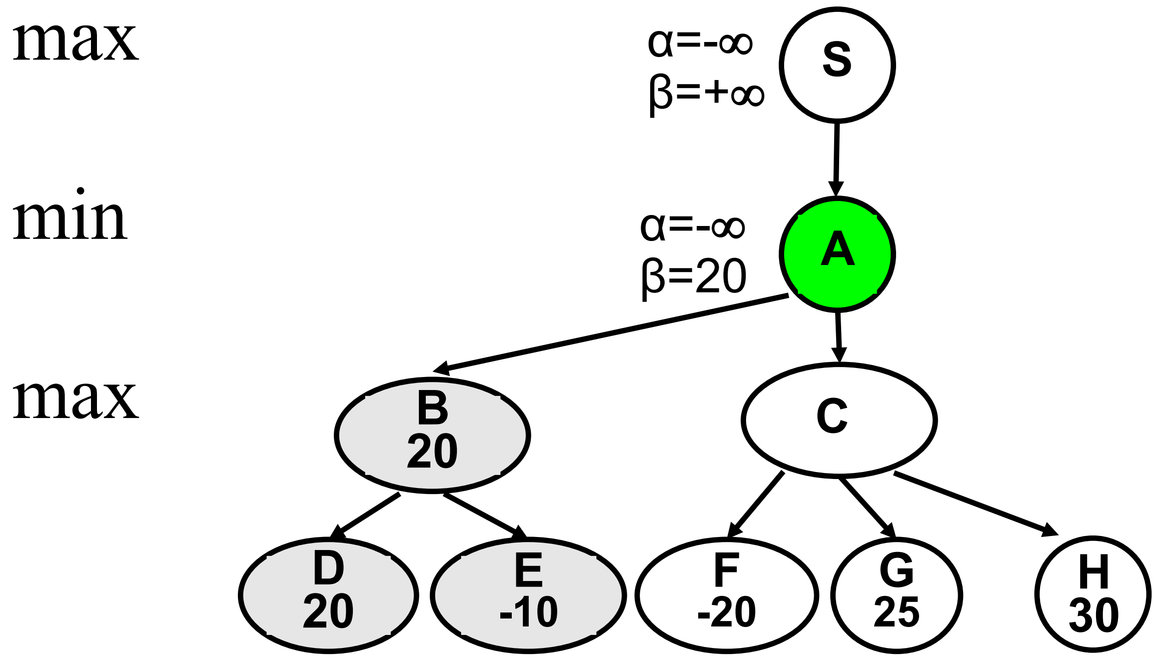
- Keep two bounds along the path
  - α: the best Max can do
  - β: the best (smallest) Min can do
- If at anytime α exceeds β, the remaining children are pruned.

Alpha-beta pruning example 2



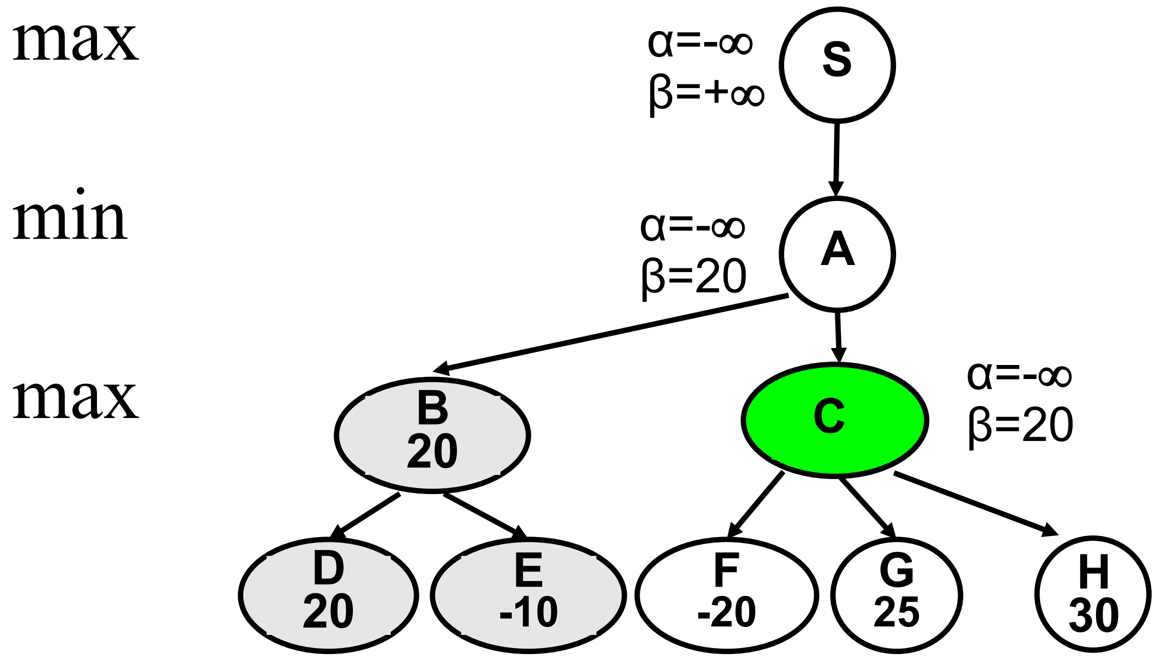
- Keep two bounds along the path
  - α: the best Max can do
  - β: the best (smallest) Min can do
- If at anytime α exceeds β, the remaining children are pruned.

Alpha-beta pruning example 2



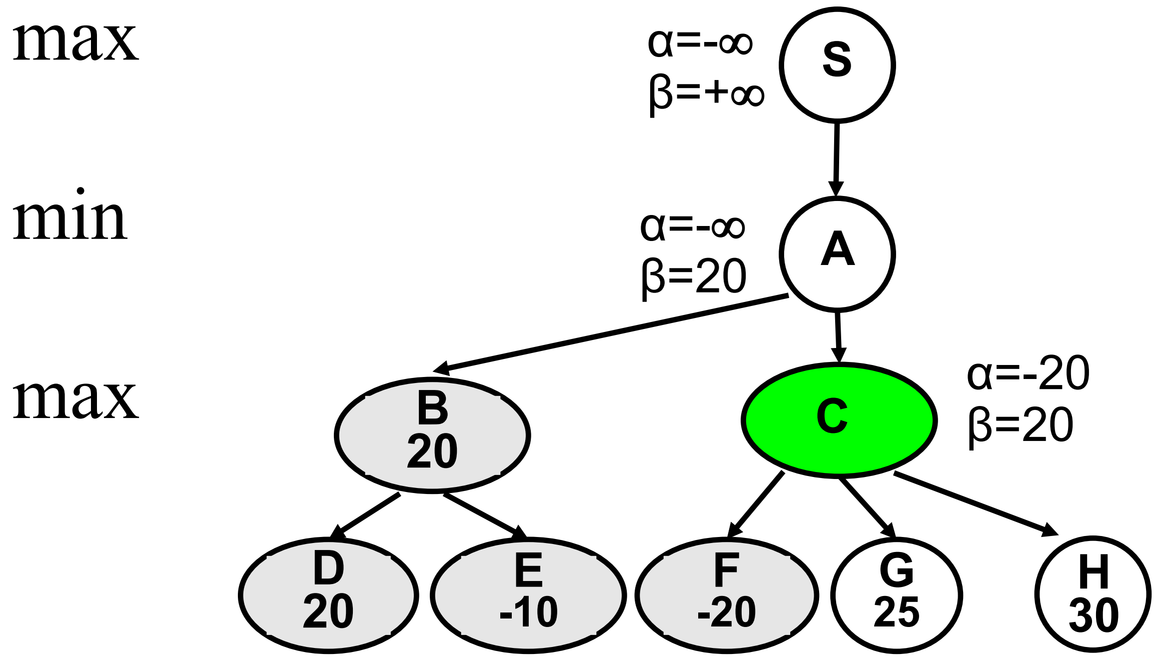
- Keep two bounds along the path
  - α: the best Max can do
  - β: the best (smallest) Min can do
- If at anytime α exceeds β, the remaining children are pruned.

Alpha-beta pruning example 2



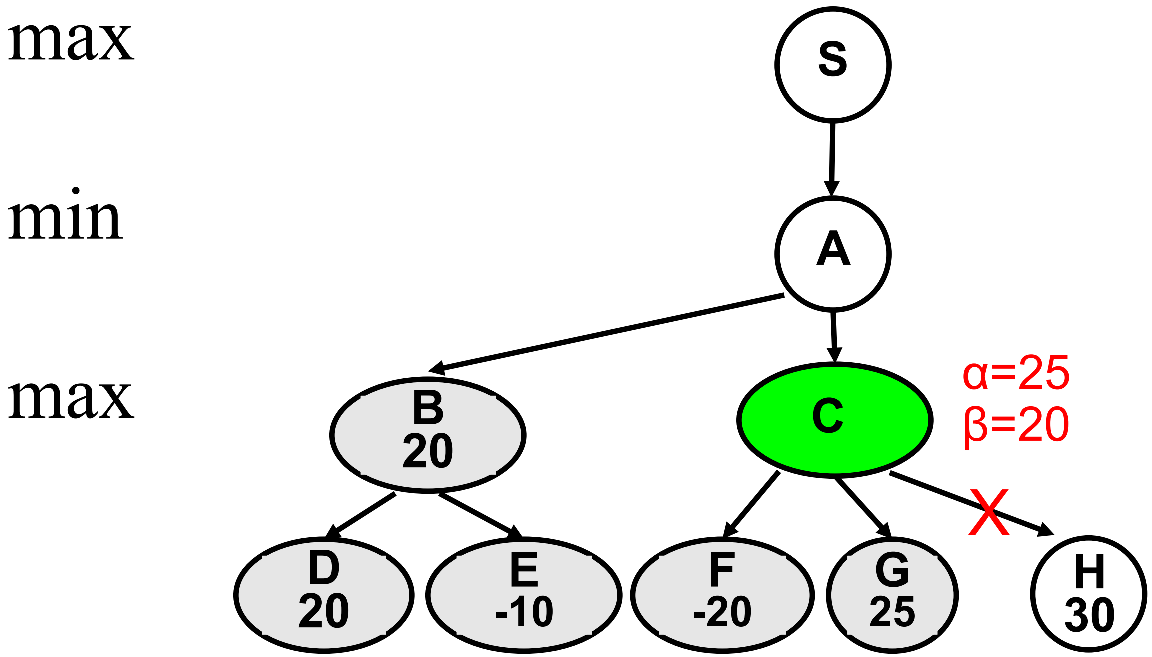
- Keep two bounds along the path
  - α: the best Max can do
  - β: the best (smallest) Min can do
- If at anytime α exceeds β, the remaining children are pruned.

Alpha-beta pruning example 2



- Keep two bounds along the path
  - α: the best Max can do
  - β: the best (smallest) Min can do
- If at anytime α exceeds β, the remaining children are pruned.

Alpha-beta pruning example 2



- Keep two bounds along the path
  - α: the best Max can do
  - β: the best (smallest) Min can do
- If at anytime α exceeds β, the remaining children are pruned.

Alpha-beta pruning

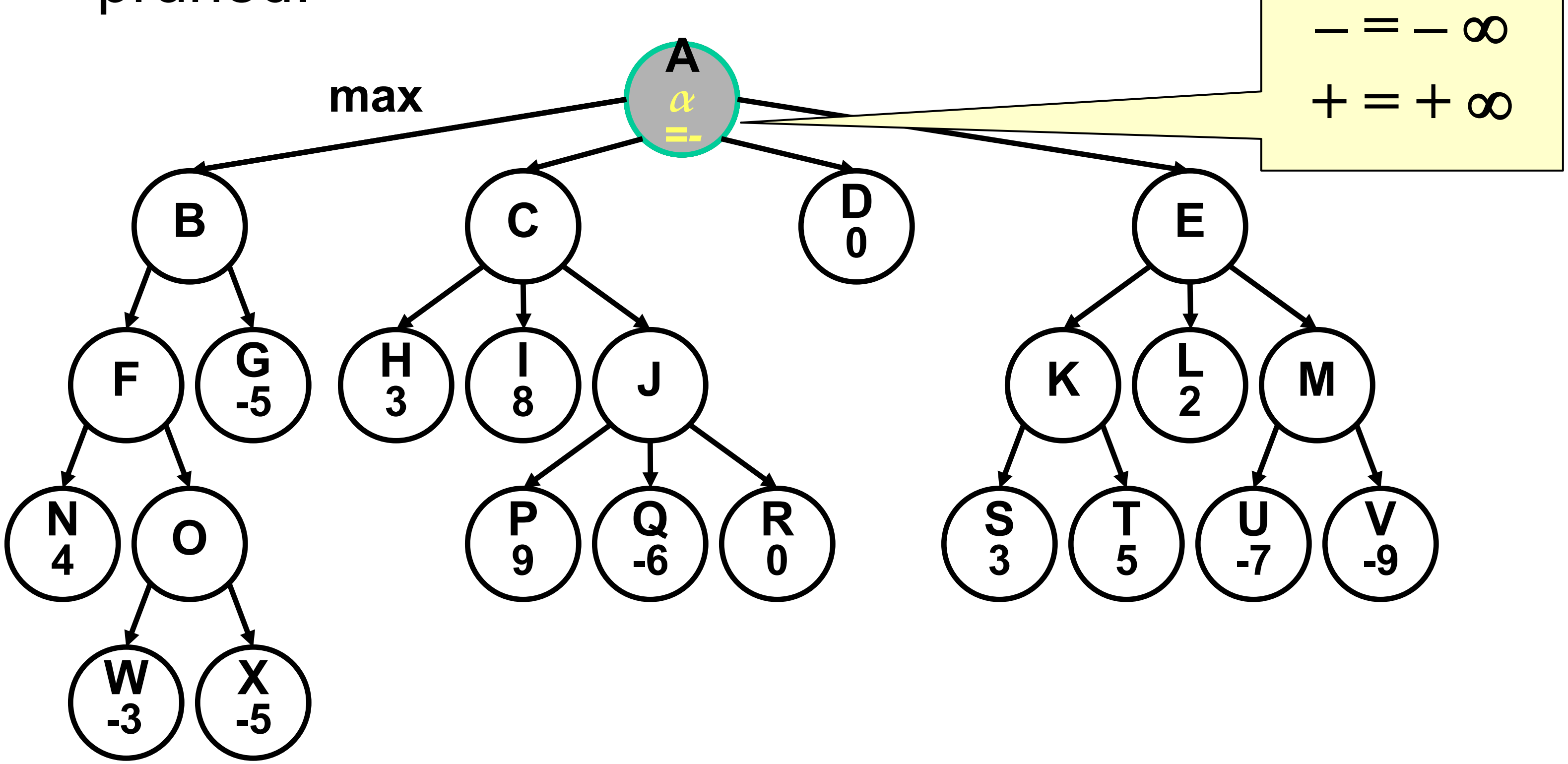
```
function Max-Value (s,α,β)
inputs:
    s: current state in game, Max about to play
    α: best score (highest) for Max along path to s
    β: best score (lowest) for Min along path to s
output: min(β , best-score (for Max) available from s)

    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   /* alpha 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 (α ≥ β ) then return α   /* beta pruning */
    return β
```

Starting from the root:
Max-Value(root, $-\infty$, $+\infty$)

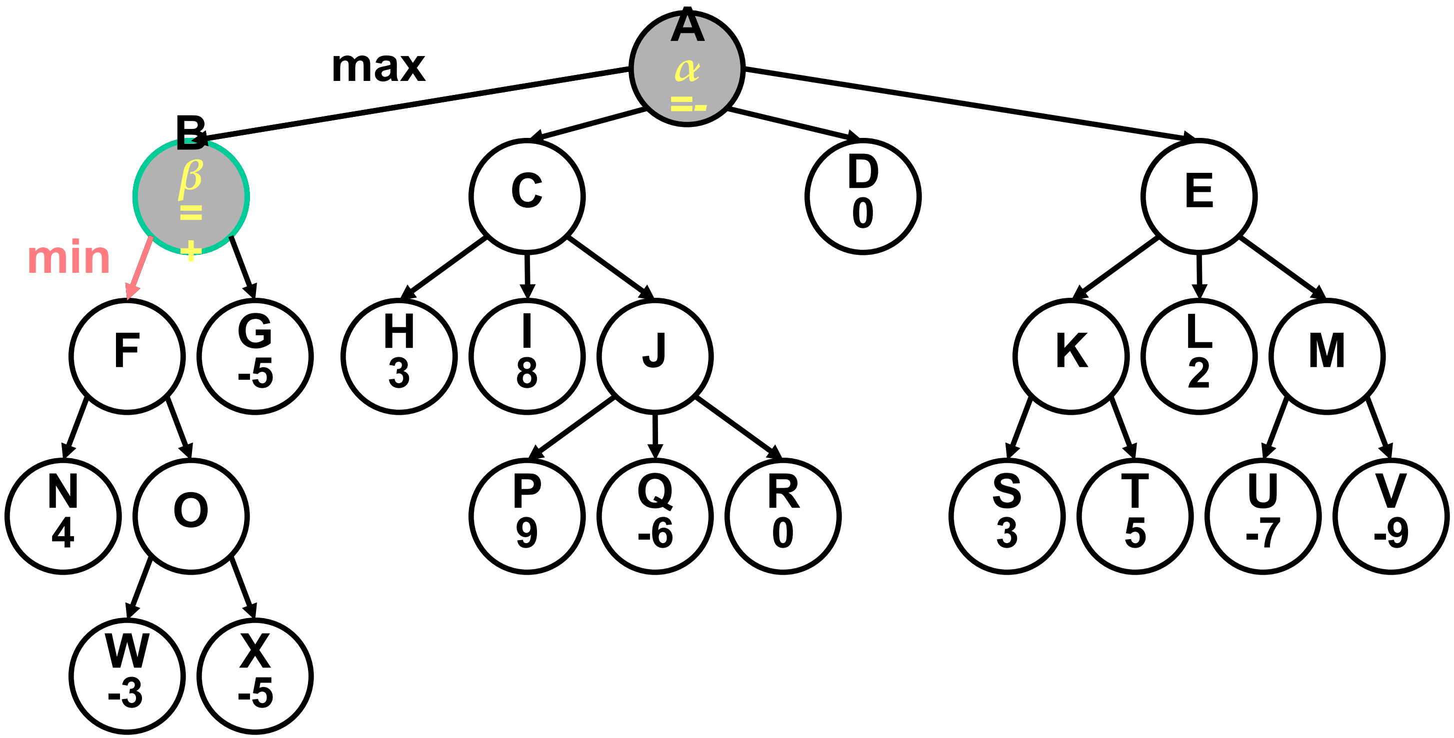
Alpha-beta pruning example

- 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 at anytime α exceeds β, the remaining children are pruned.



Alpha-beta pruning example

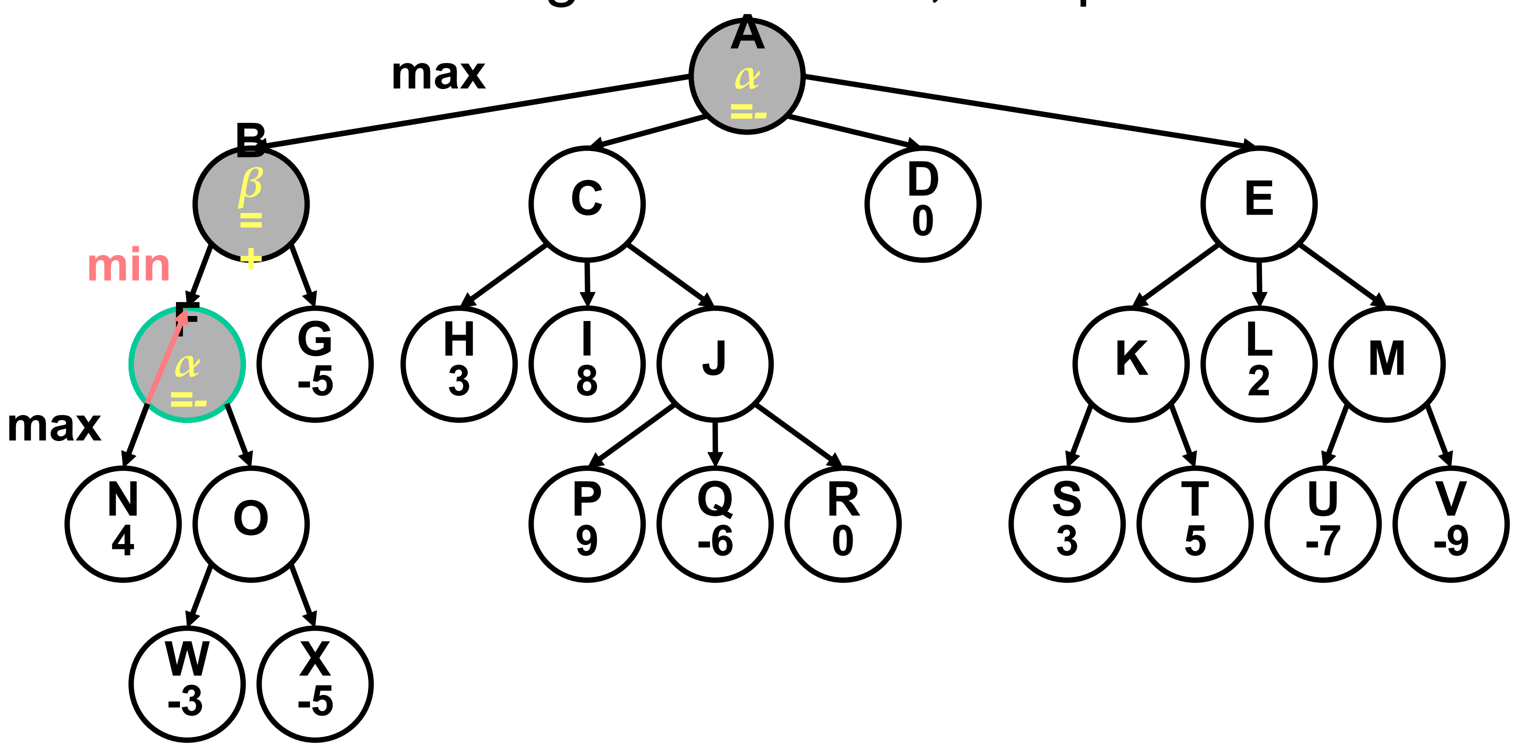
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

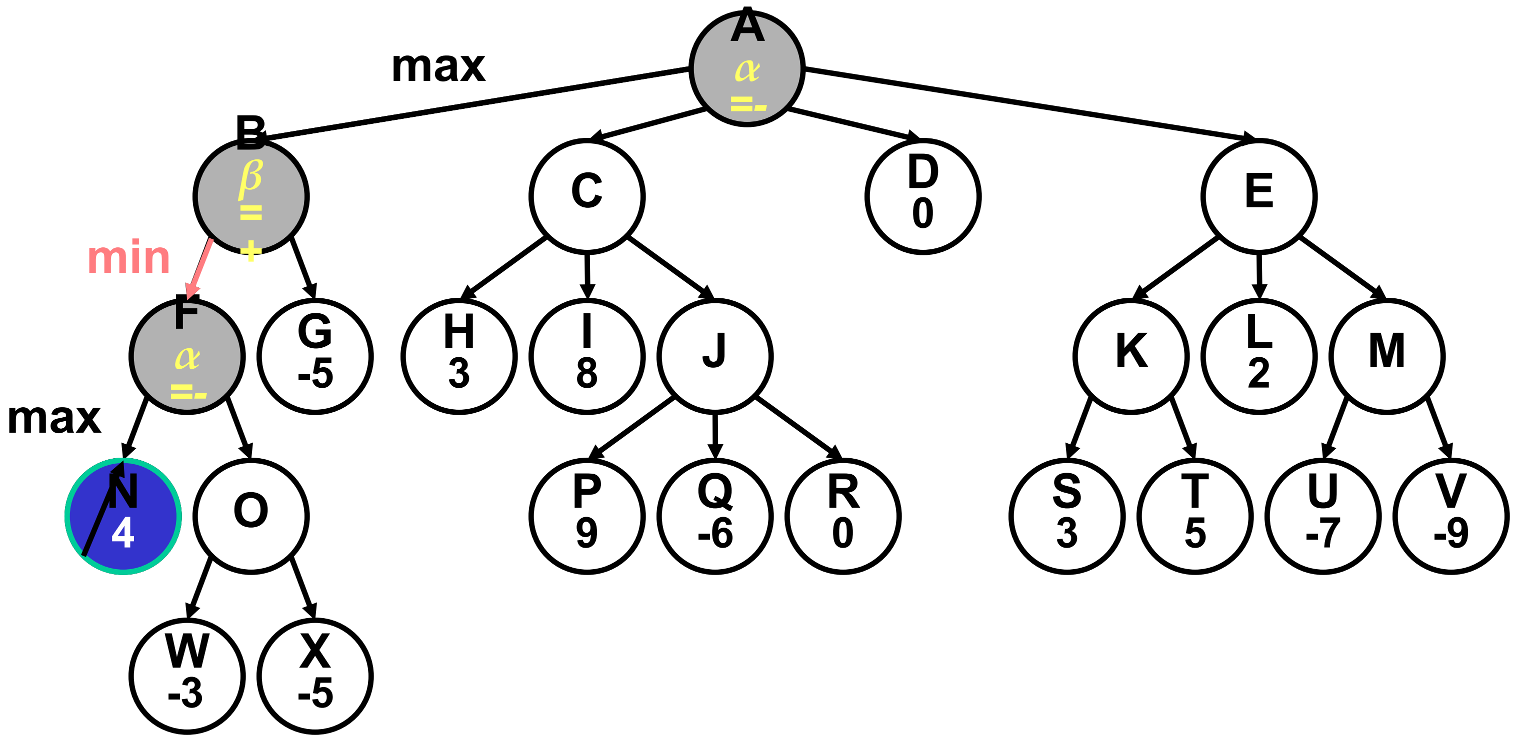
Alpha-beta pruning example

- 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.



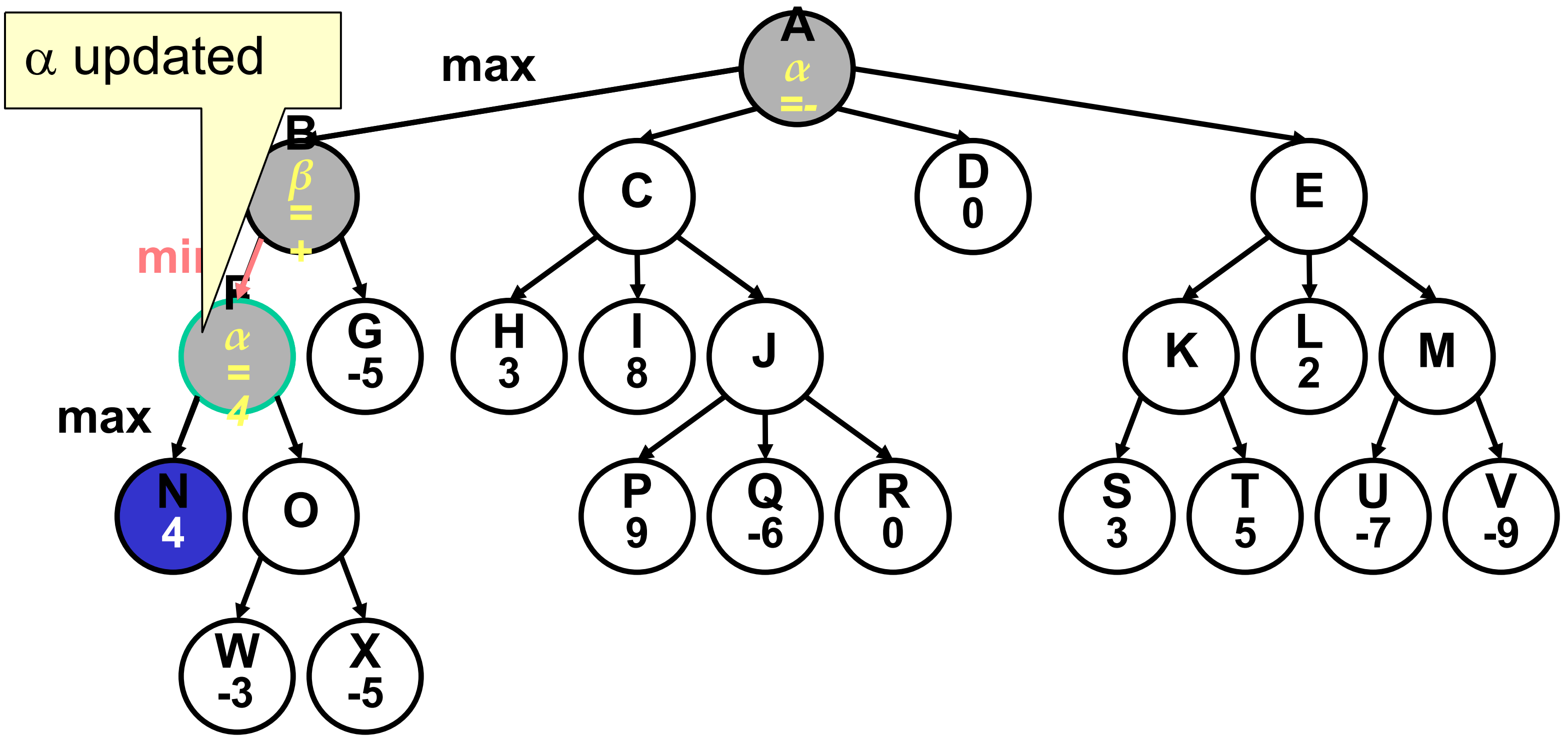
Alpha-beta pruning example

- 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.



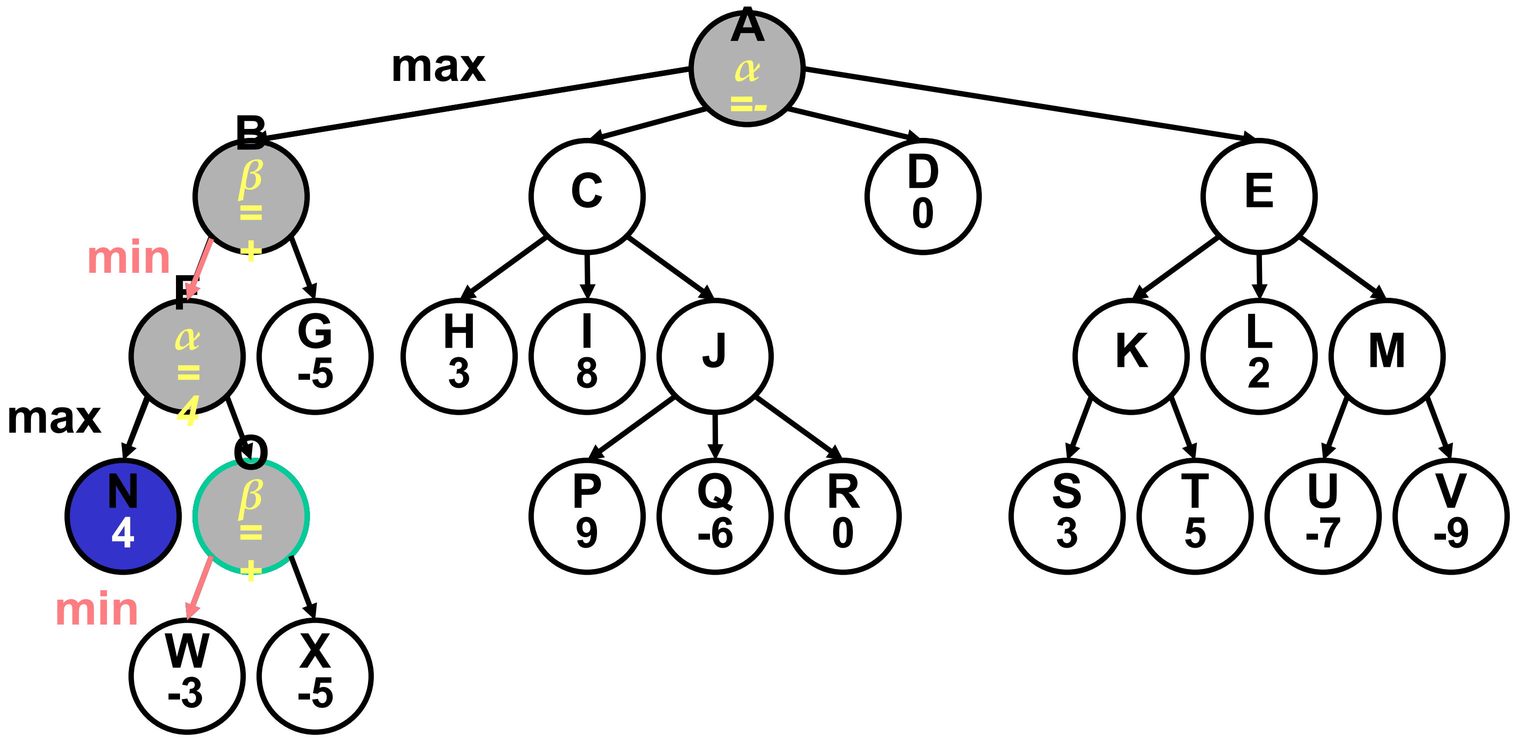
Alpha-beta pruning example

- 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

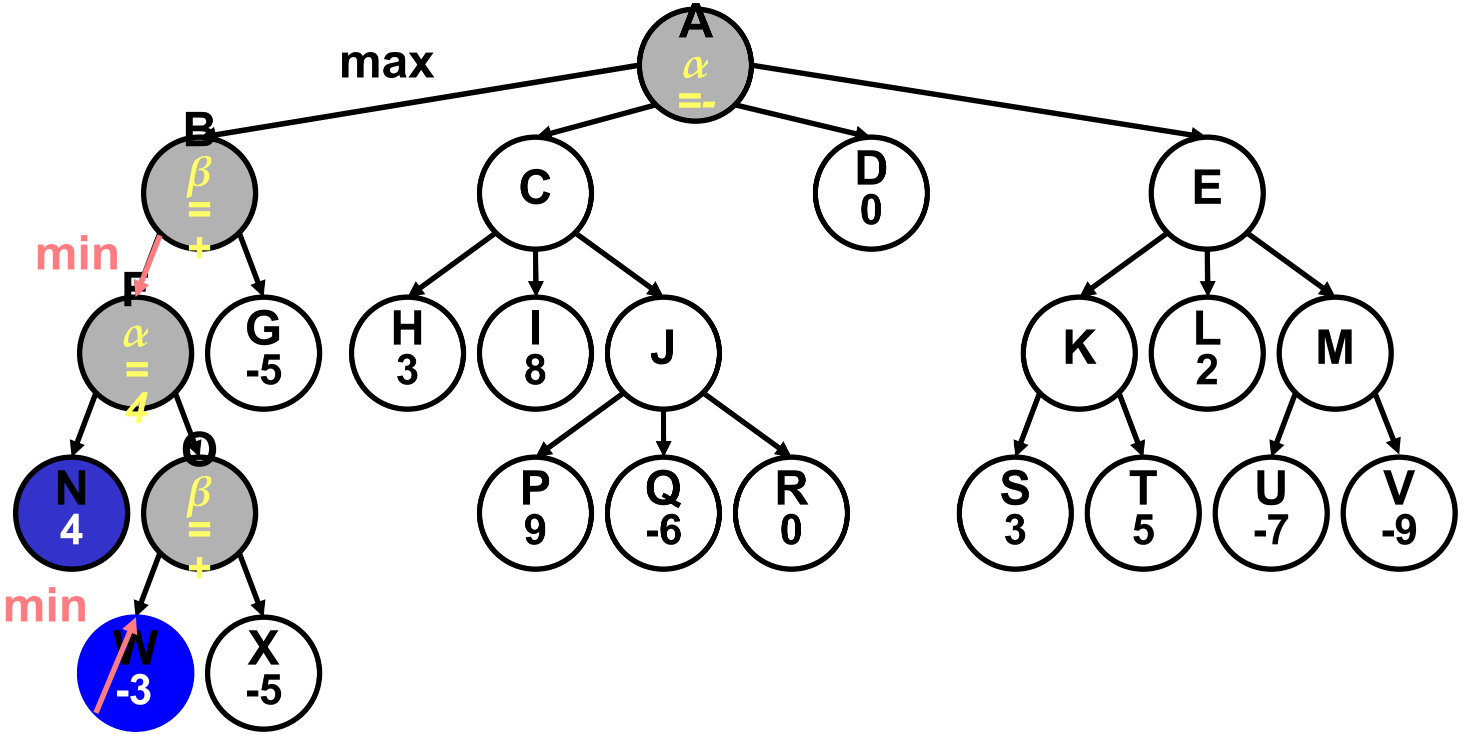
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

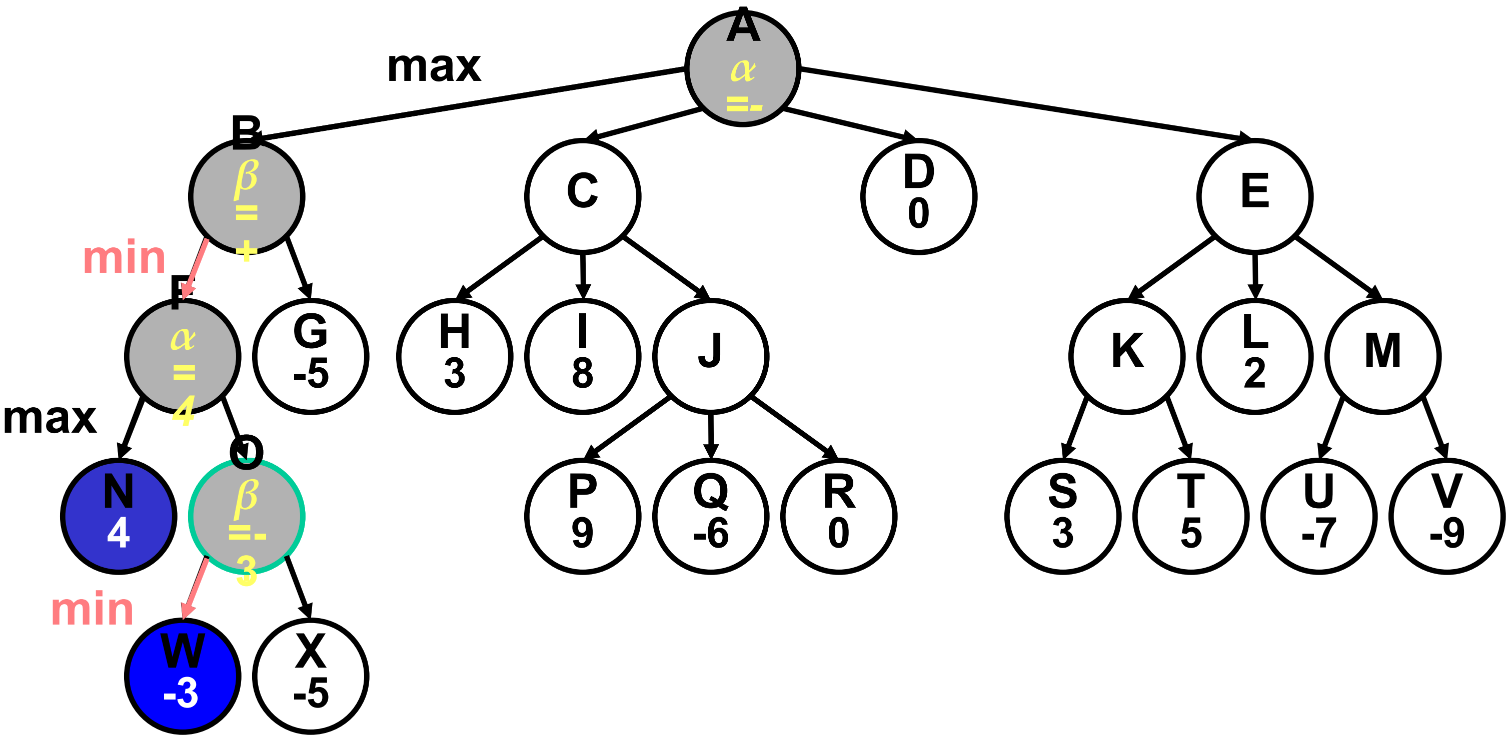
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

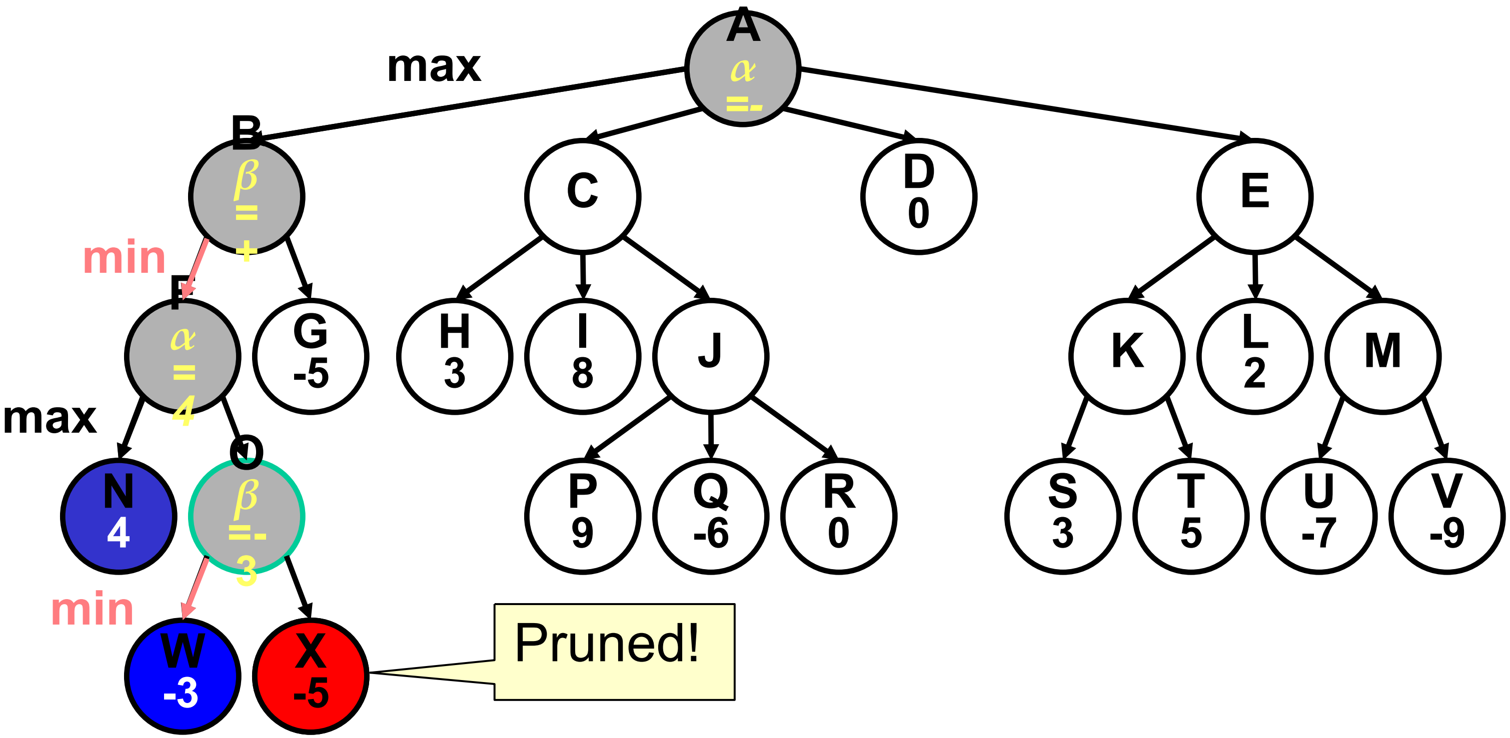
Alpha-beta pruning example

- 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.



Alpha-beta pruning example

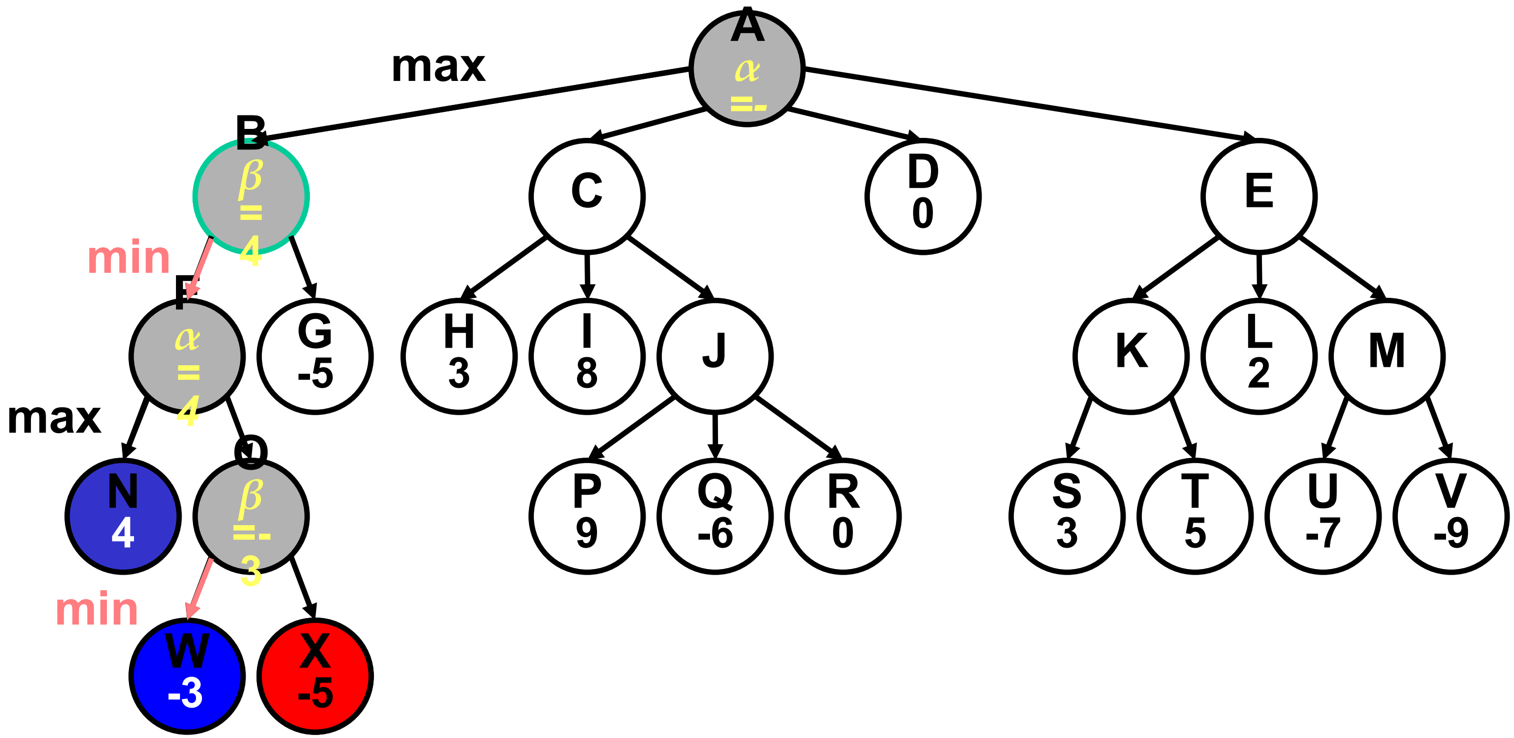
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

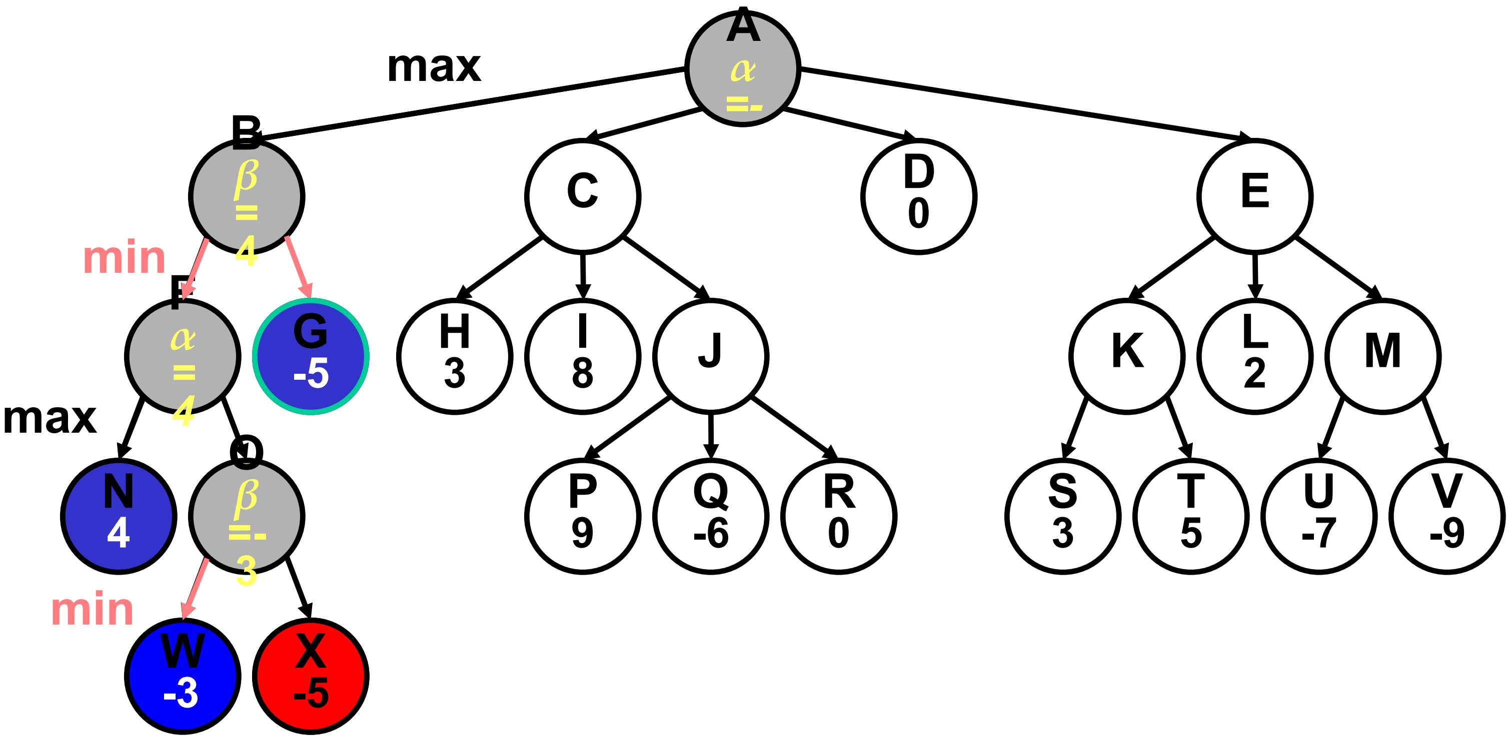
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

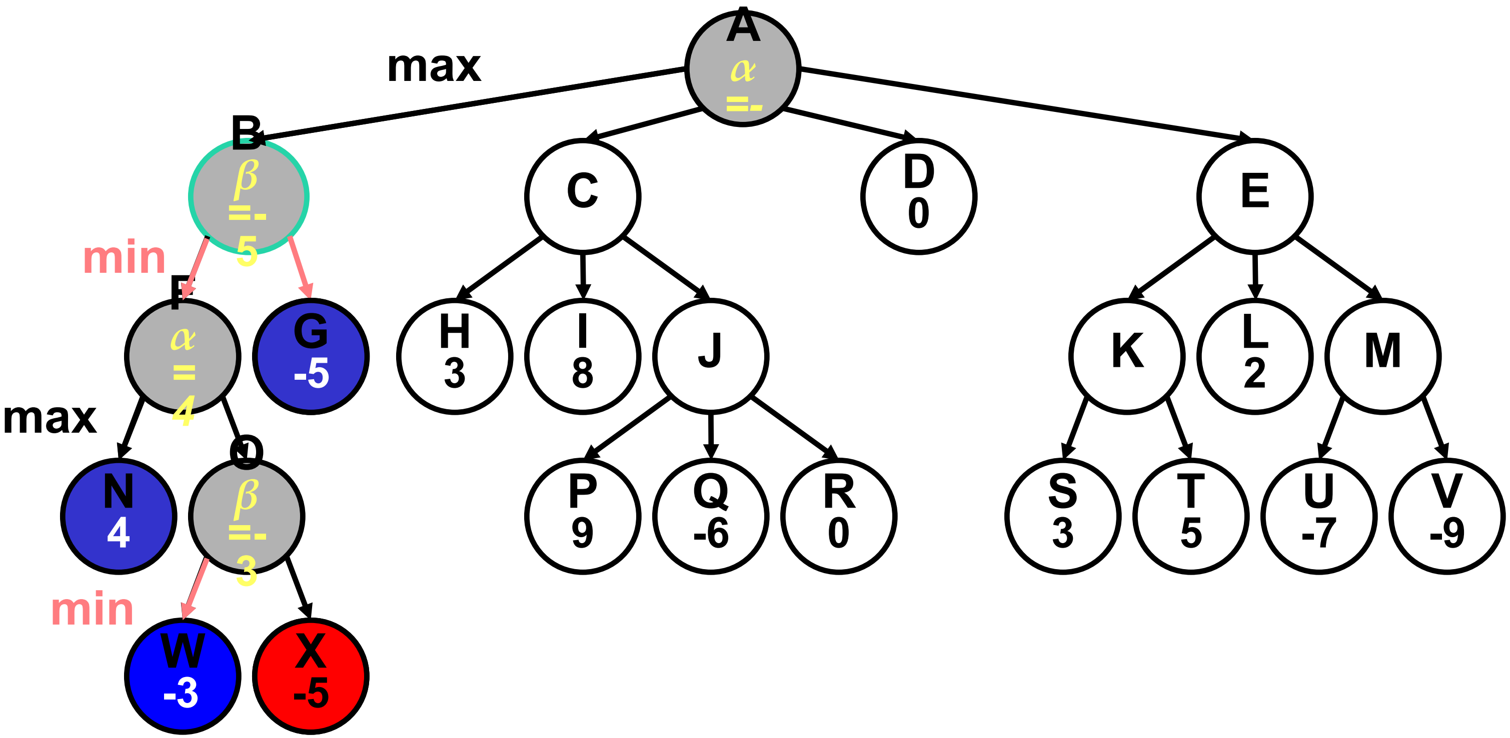
Alpha-beta pruning example

- 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

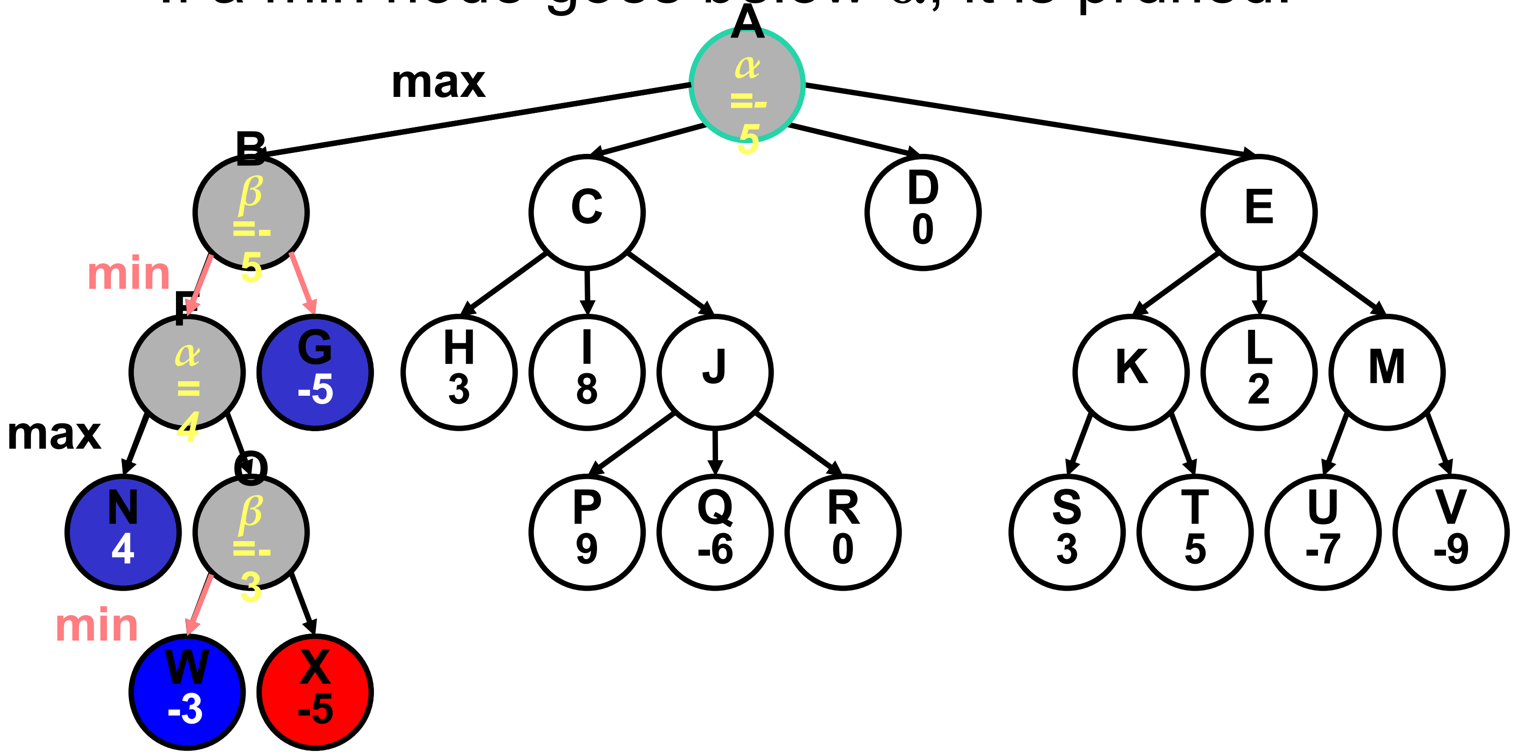
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

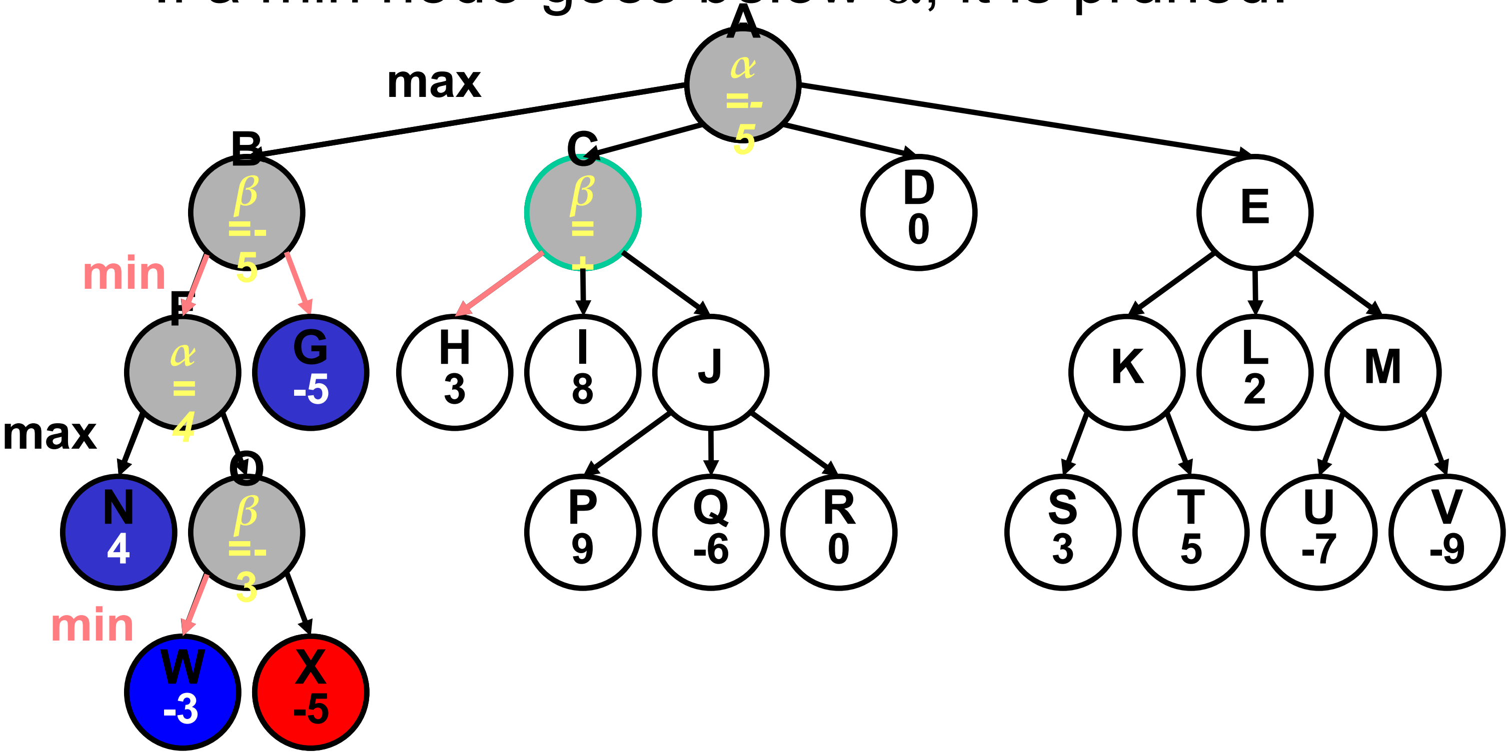
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

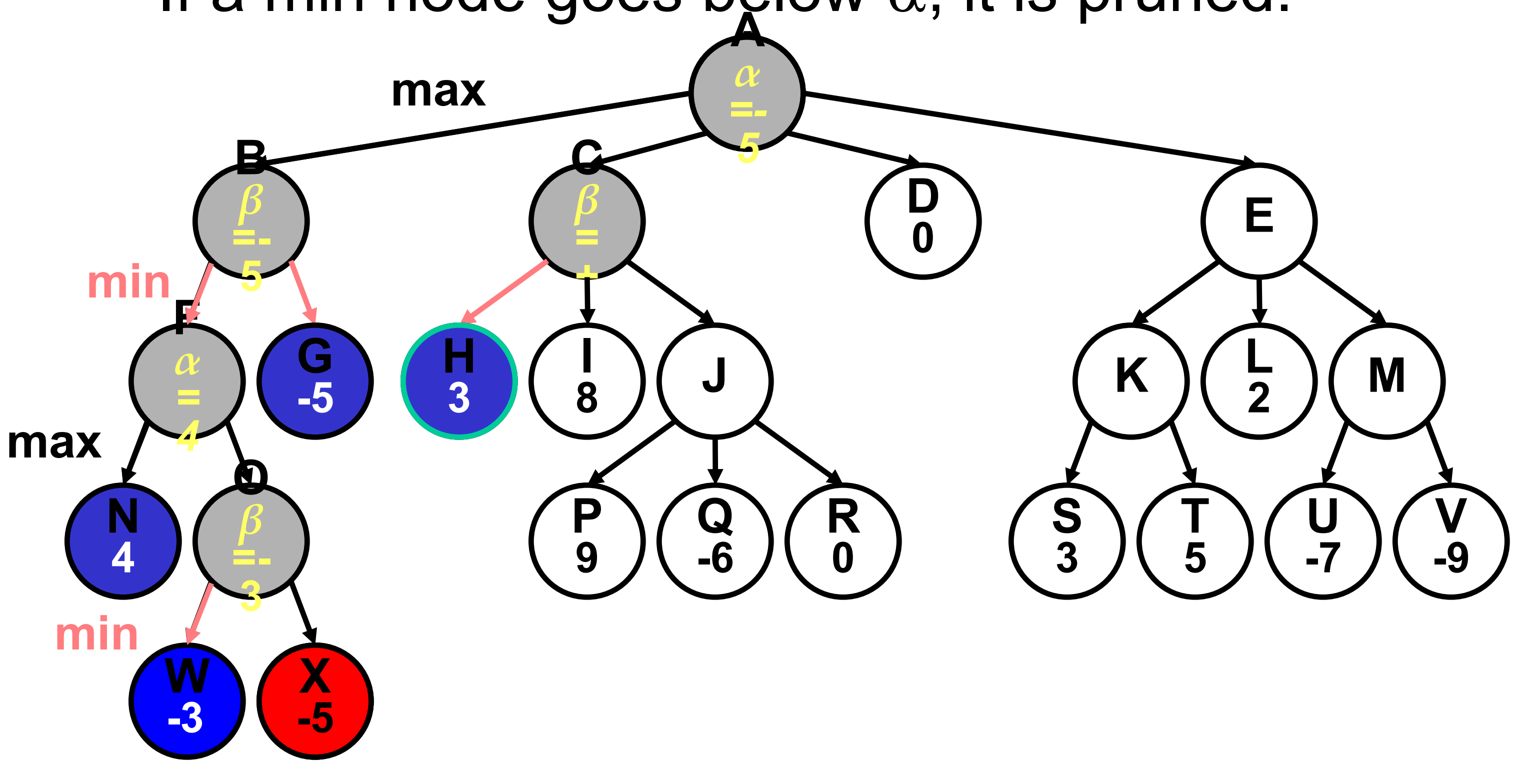
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

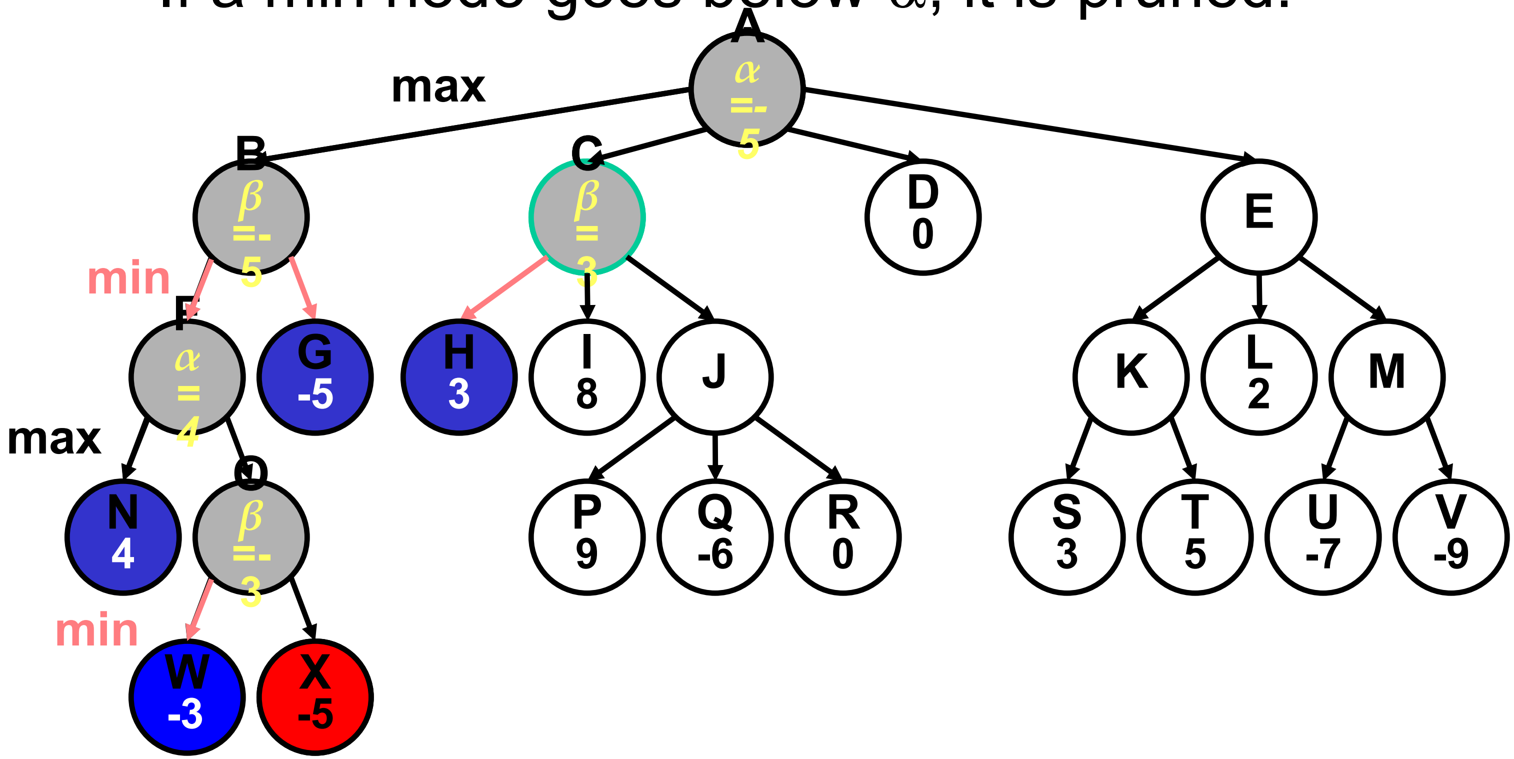
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

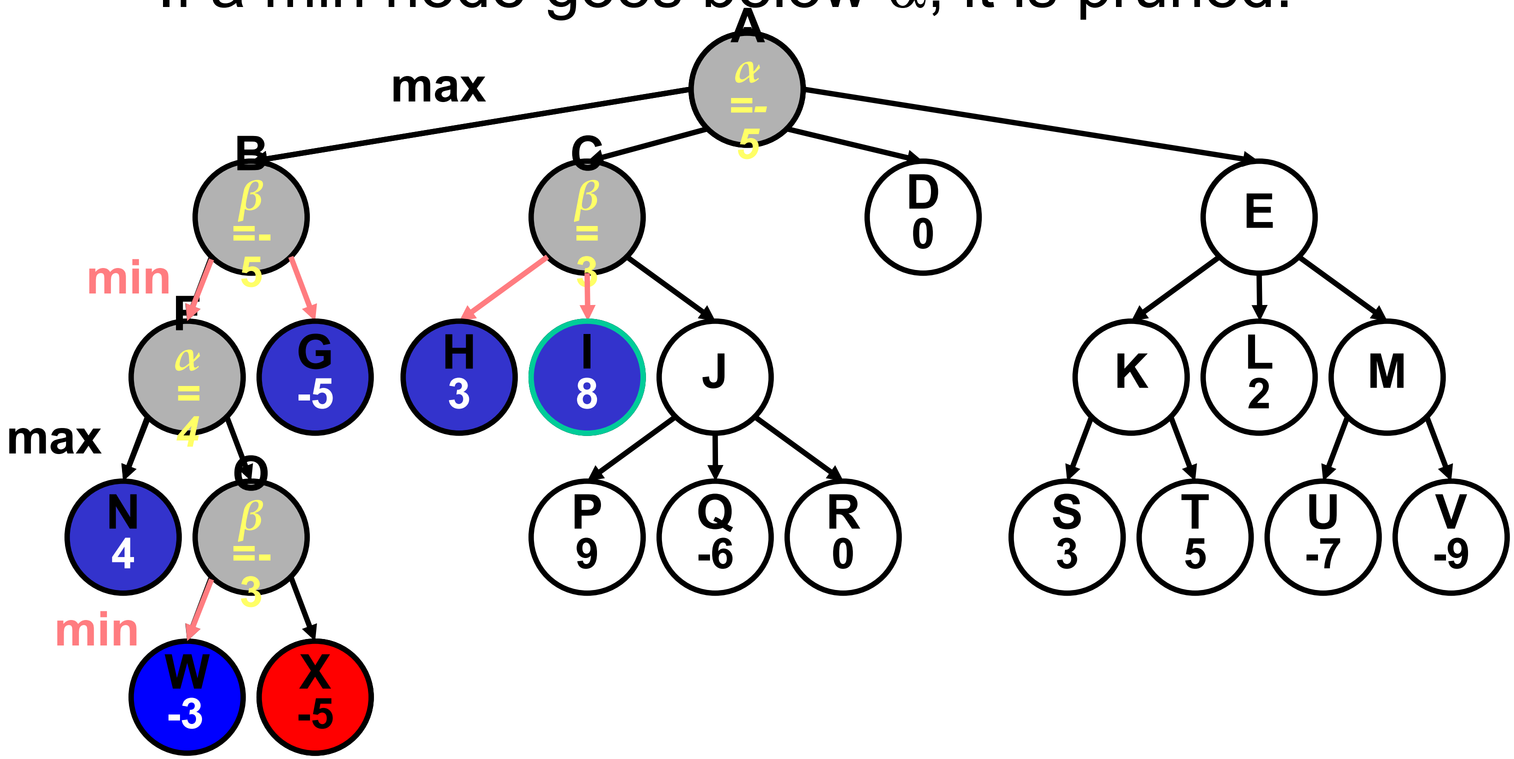
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

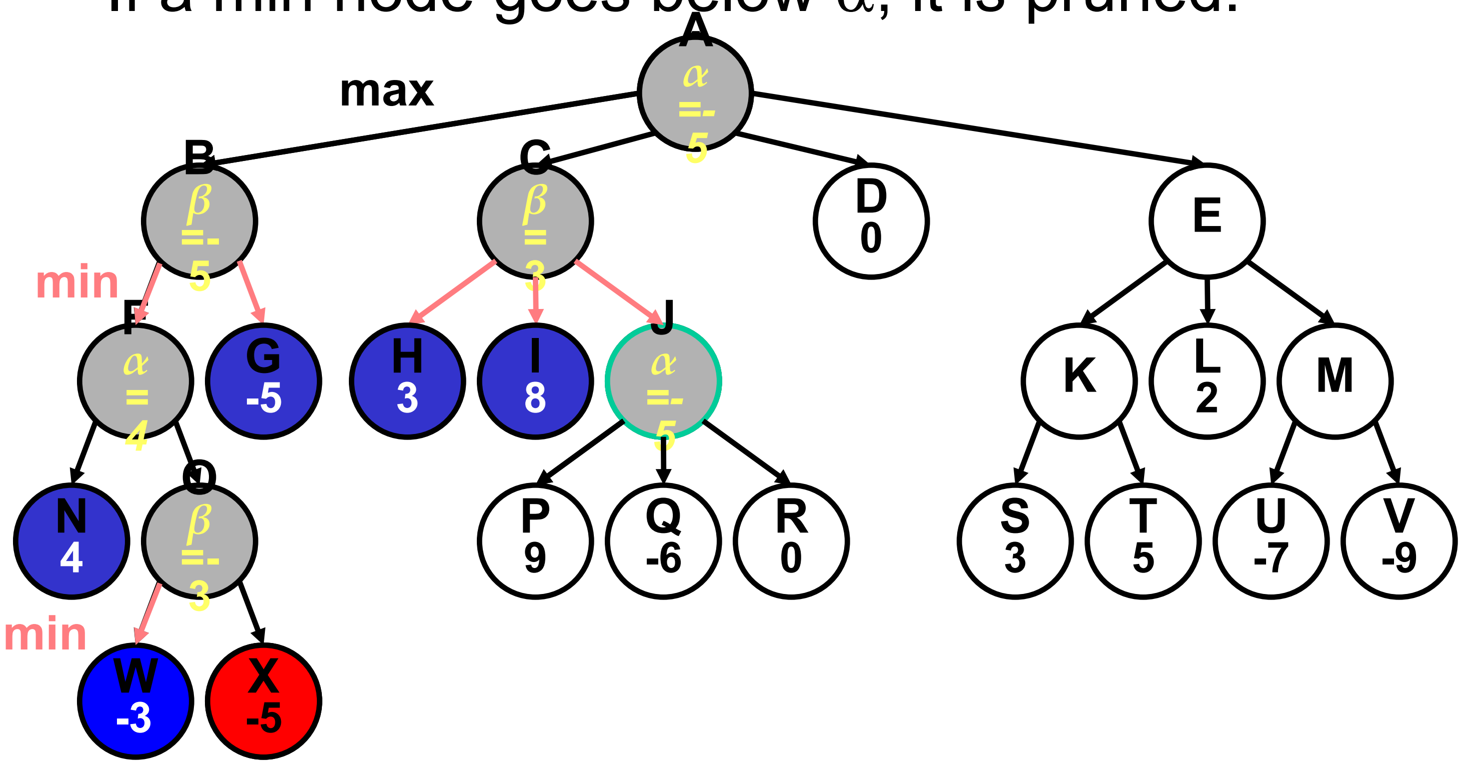
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

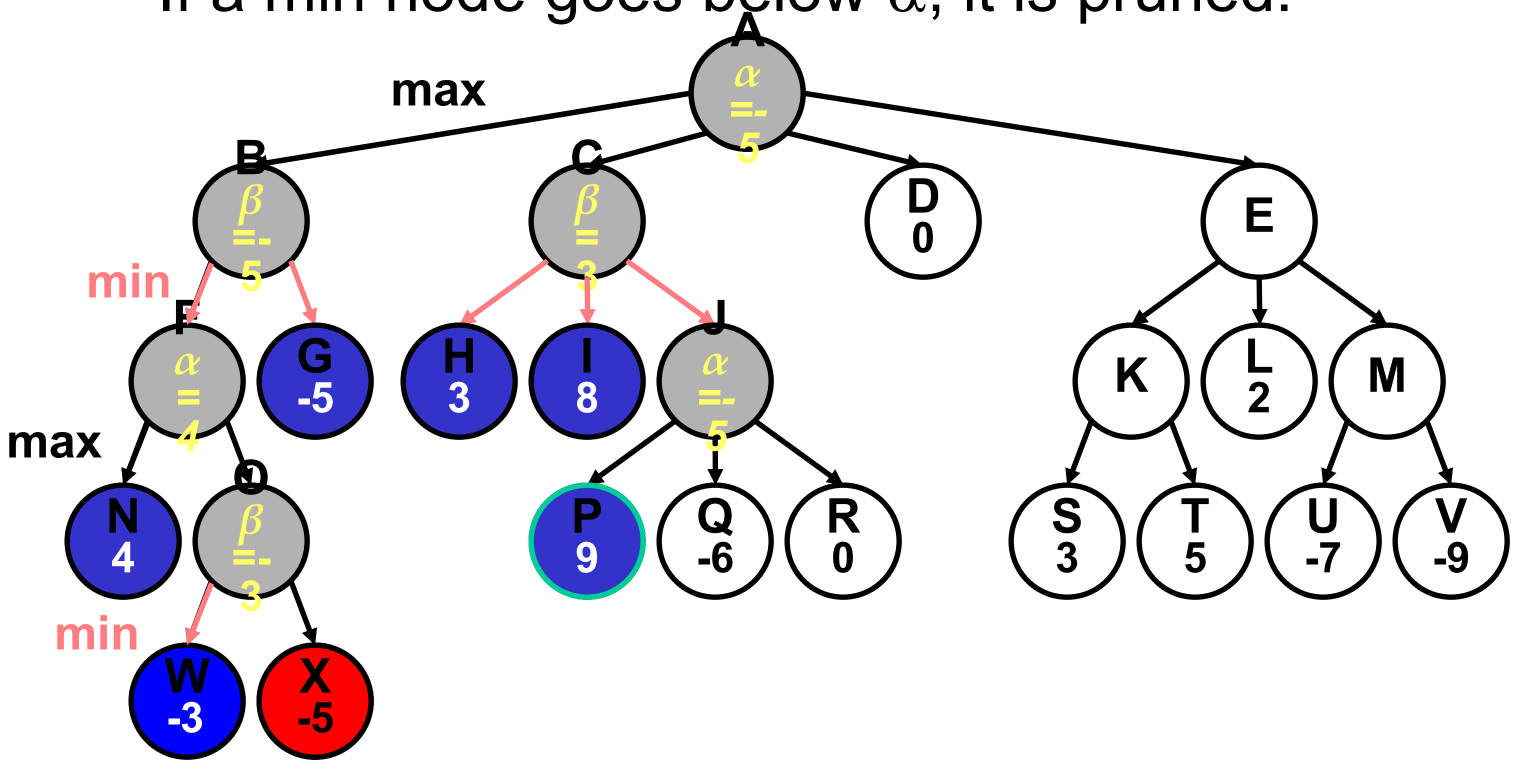
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

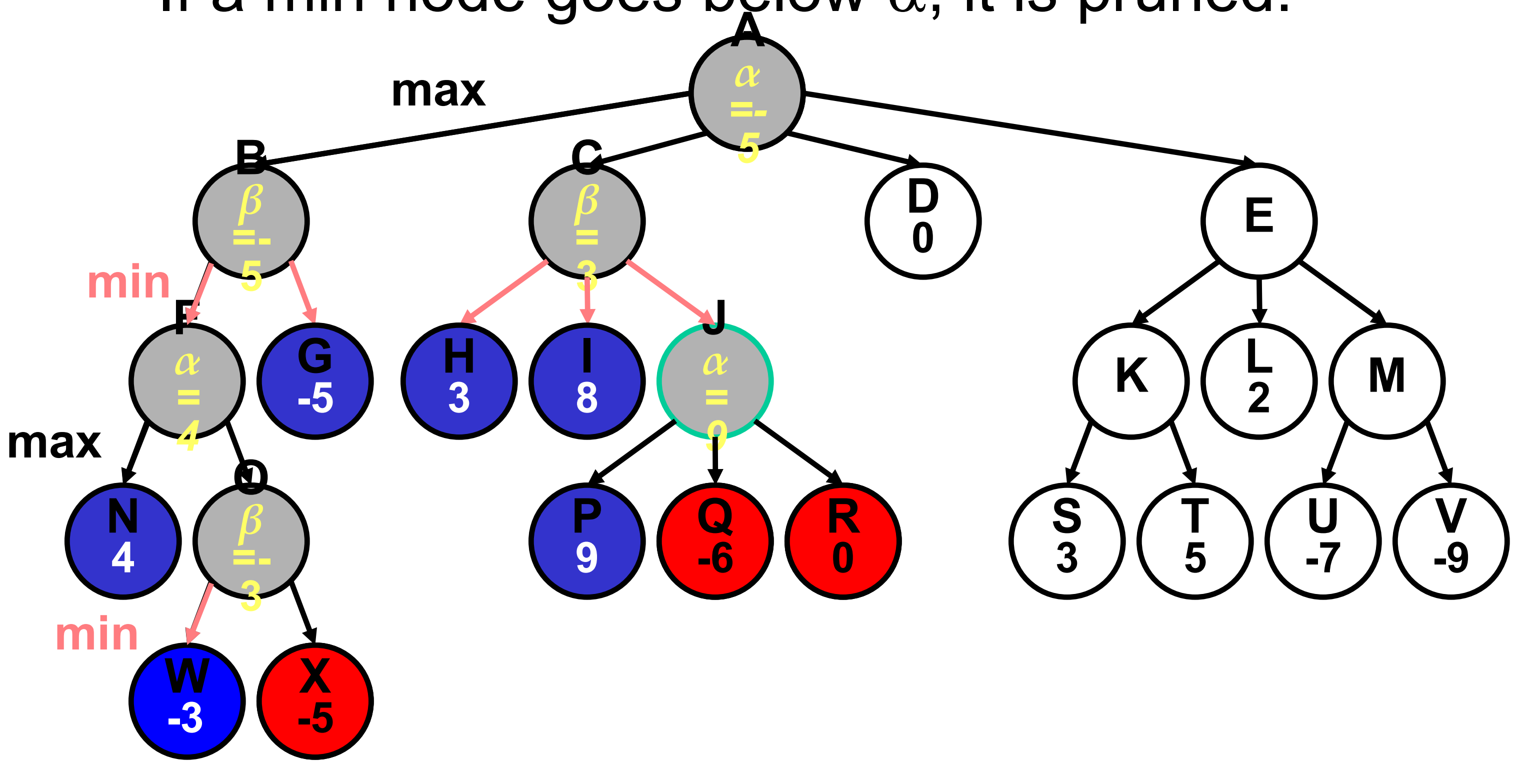
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

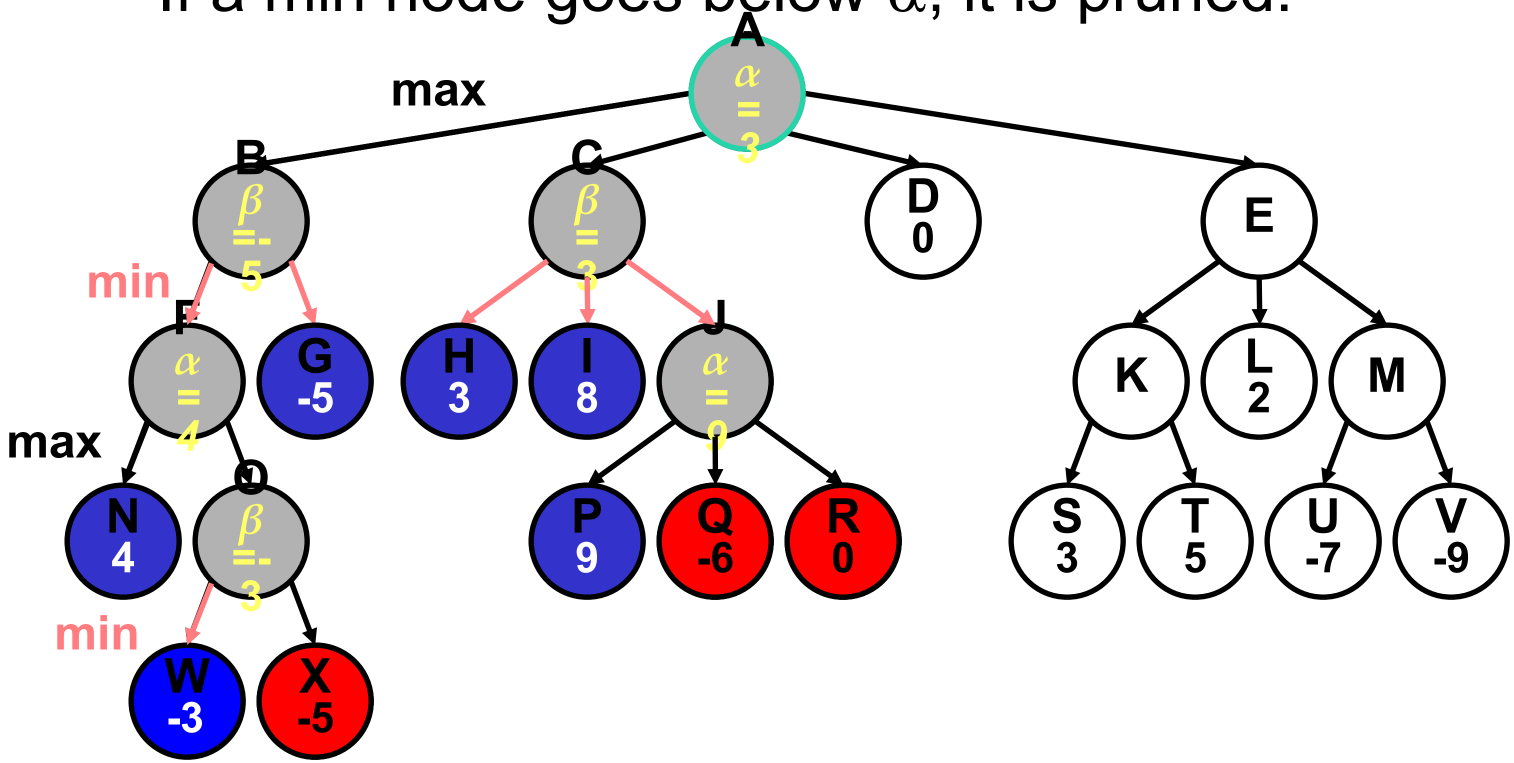
Alpha-beta pruning example

- 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.



Alpha-beta pruning example

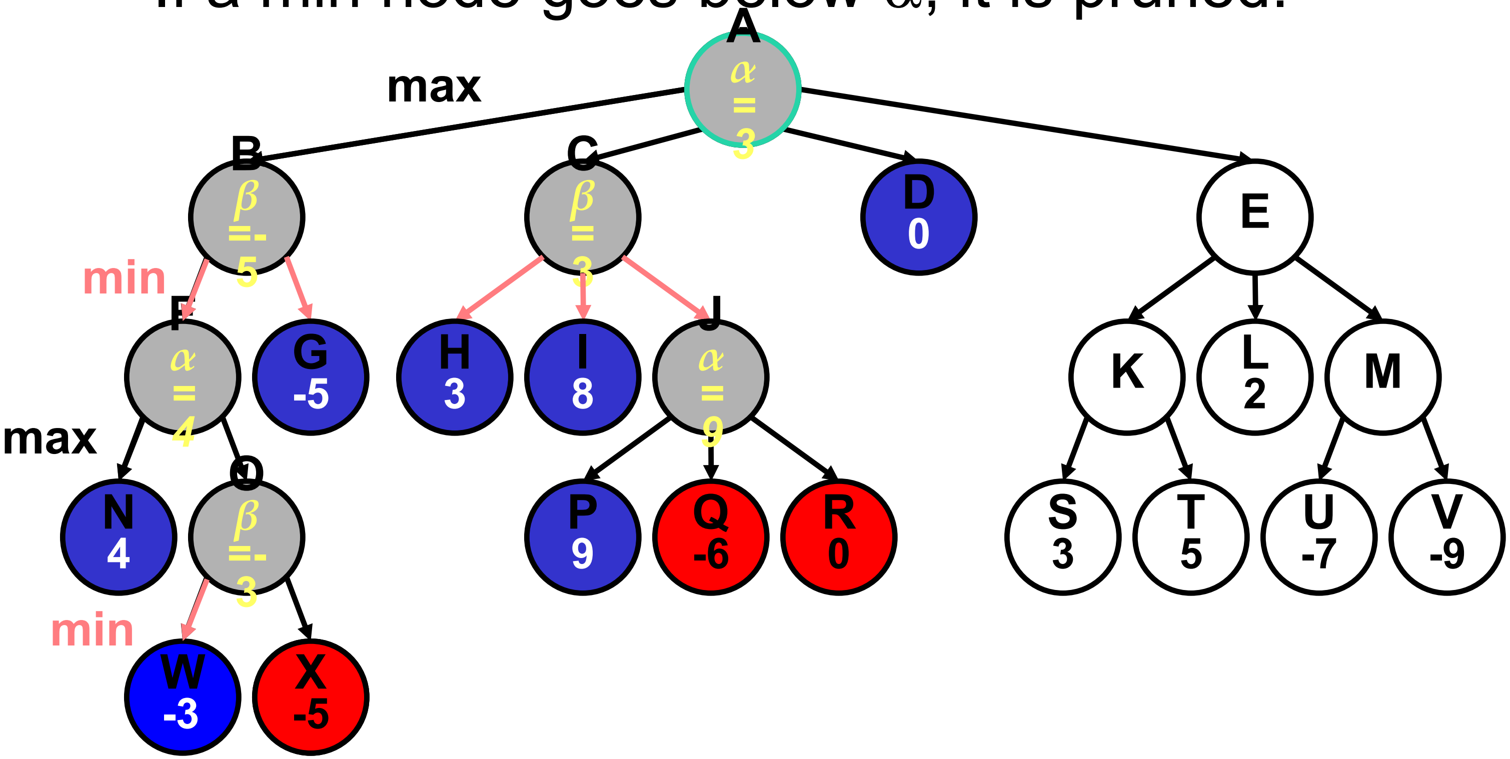
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

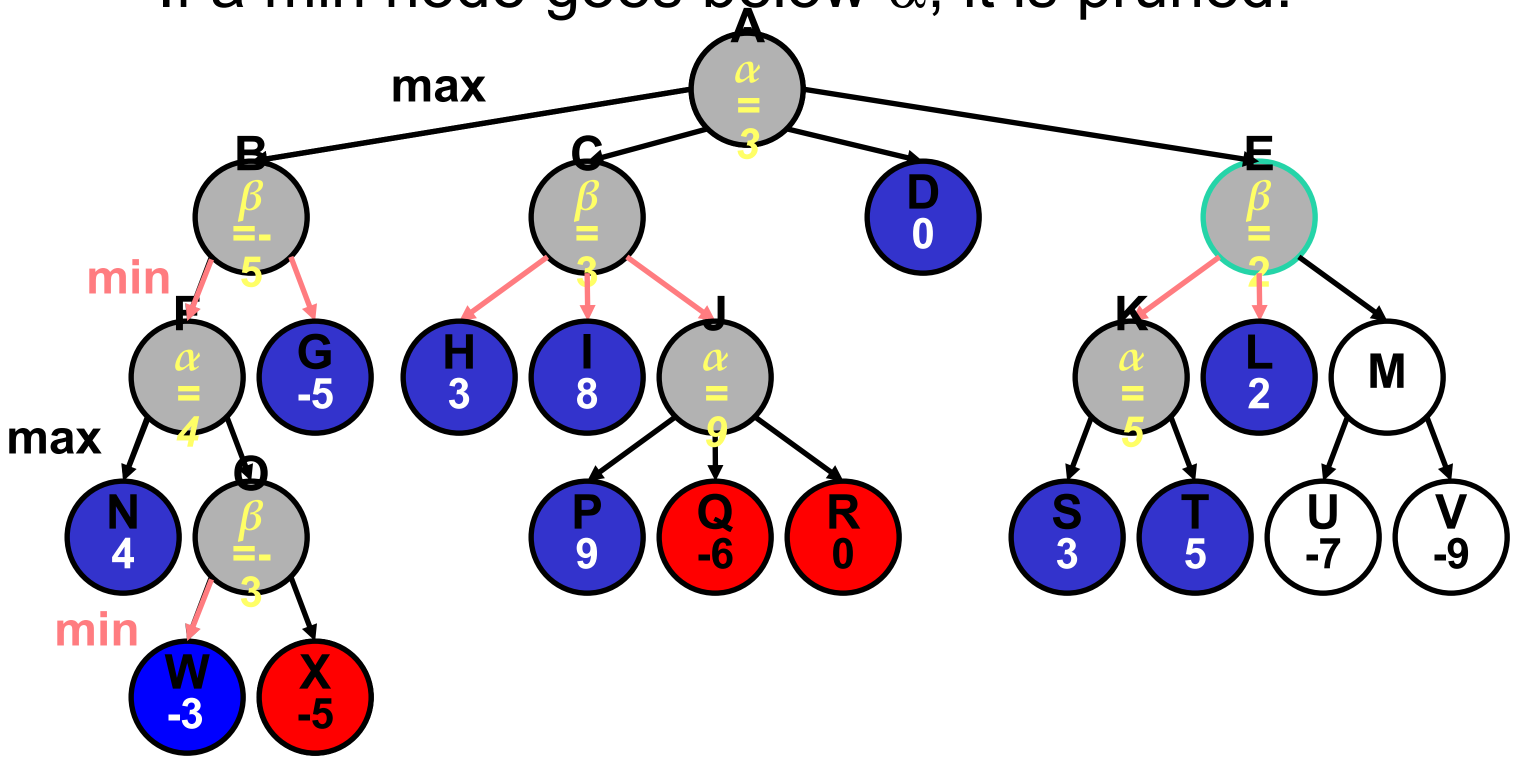
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

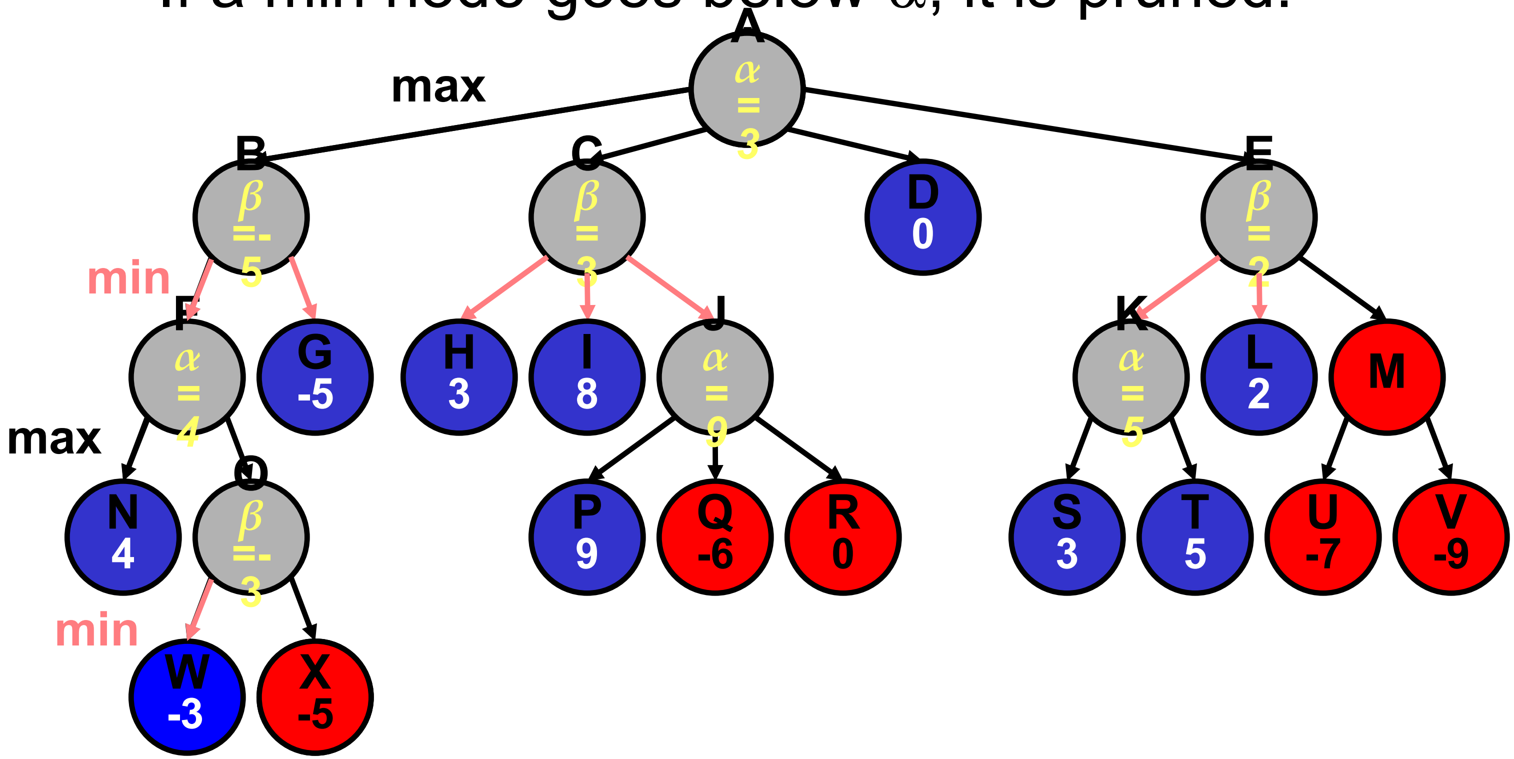
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[Example from James Skrentny]

Alpha-beta pruning example

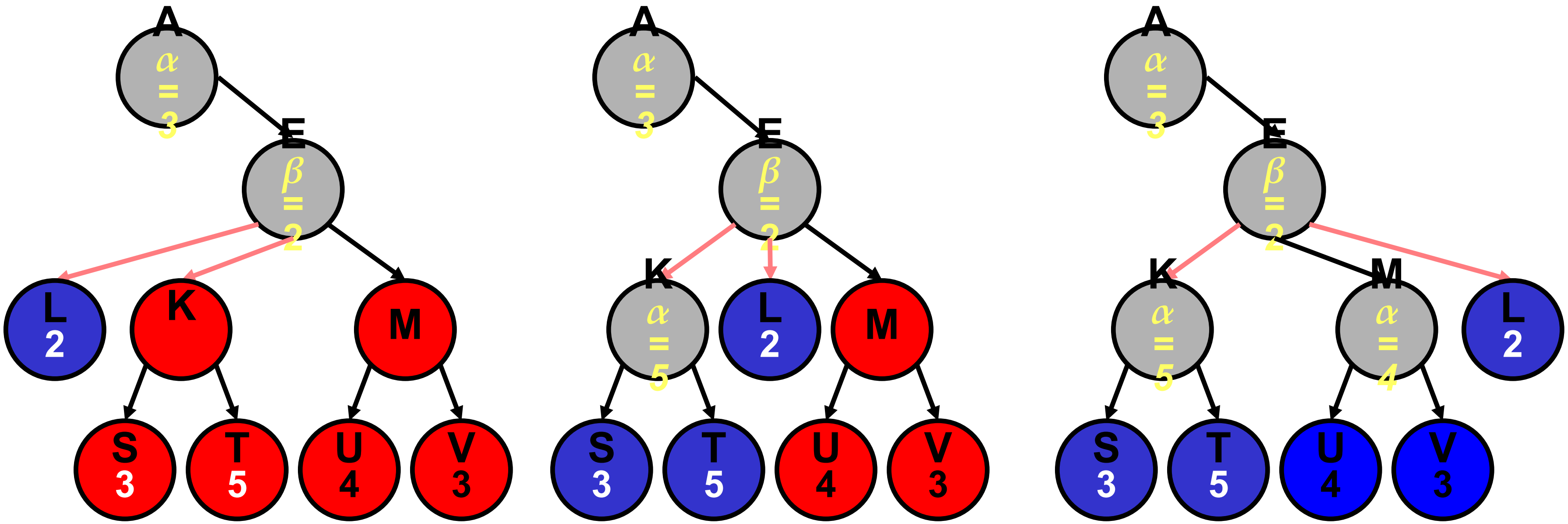
- Keep two bounds along the path
  - α: the best Max can do on the path
  - β: the best (smallest) Min can do on the path
- If a max node exceeds β, it is pruned.
- If a min node goes below α, it is pruned.



[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.
- This occurs when each player's best move is the leftmost child.
- In DeepBlue (IBM Chess), the average branching factor was about 6 with alpha-beta instead of 35-40 without.
- The worst case is no pruning at all.

Game-playing for large games

- We've seen how to find game theoretic values. But it is too expensive for large games.
- What do real chess-playing programs do?
  - They can't possibly search the full game tree
  - They must respond in limited time
  - They can't pre-compute a solution

Game-playing for large games

- The most popular solution: heuristic evaluation functions for games
  - ‘Leaves’ are intermediate nodes at a depth cutoff, not terminals
  - Heuristically estimate their values
  - Huge amount of knowledge engineering (R&N 6.4)
  - Example: Tic-Tac-Toe:

(number of 3-lengths open for me)-(number of 3-lengths open for you)

- Each move is a new depth-cutoff game-tree search (as opposed to search the complete game-tree once).

Prelude

- Heuristic estimation of the values of intermediate nodes can be done via Machine Learning