Adversarial Search

A-B PRUNING

Practical problem with minimax search

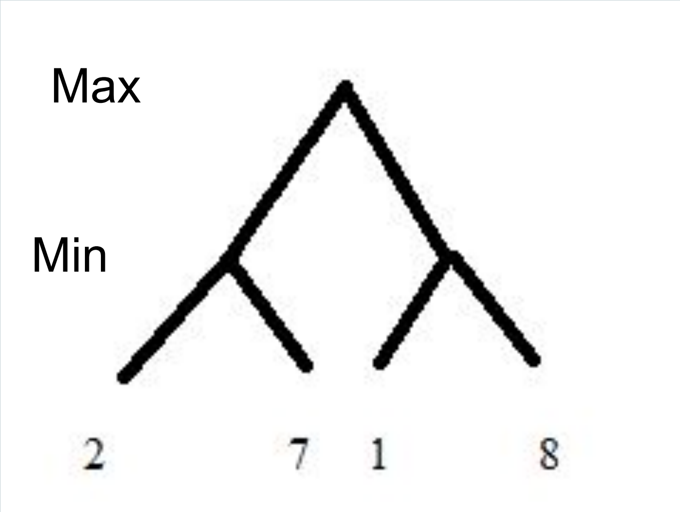
- Number of game states is exponential in the number of moves.
  - Solution: Do not examine every node

  => pruning
    - Remove branches that do not influence final decision

- Revisit example …

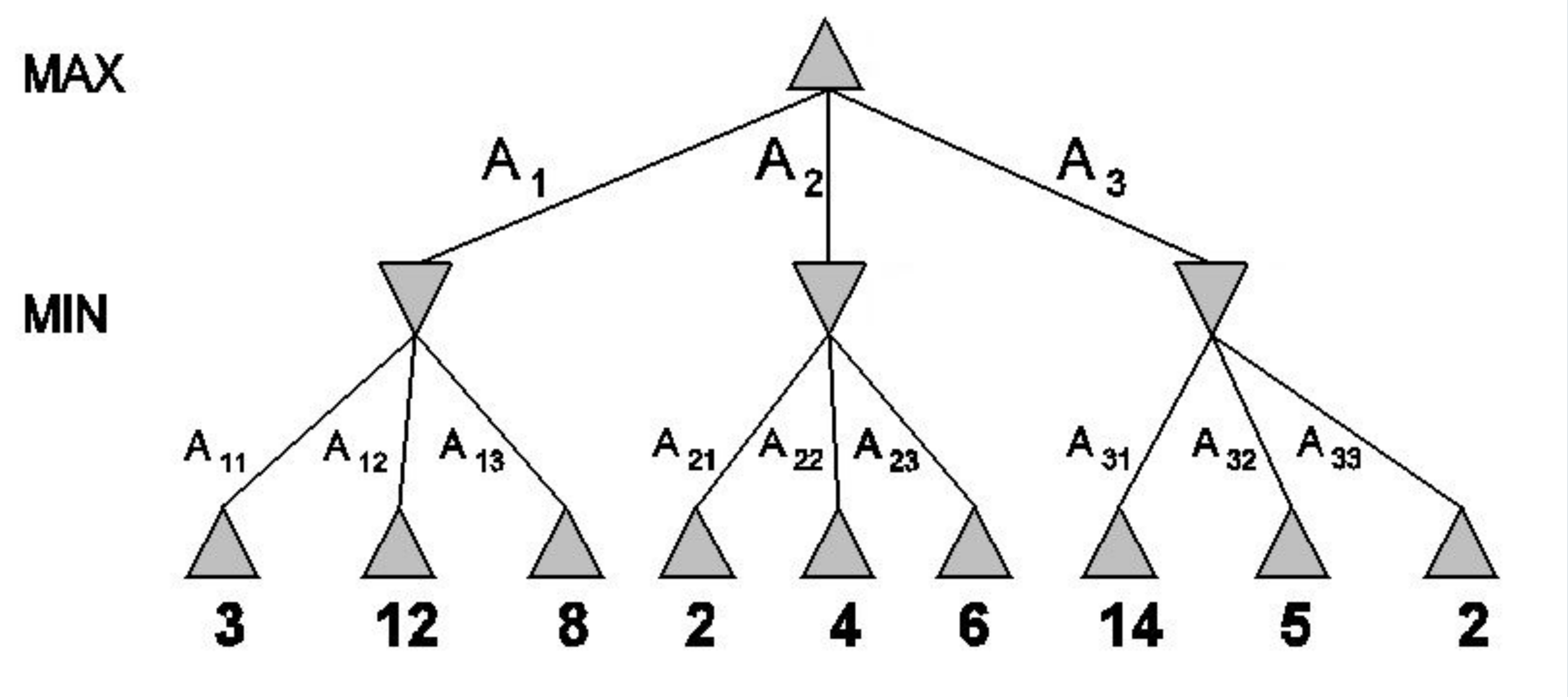
Pruning example

Minimax

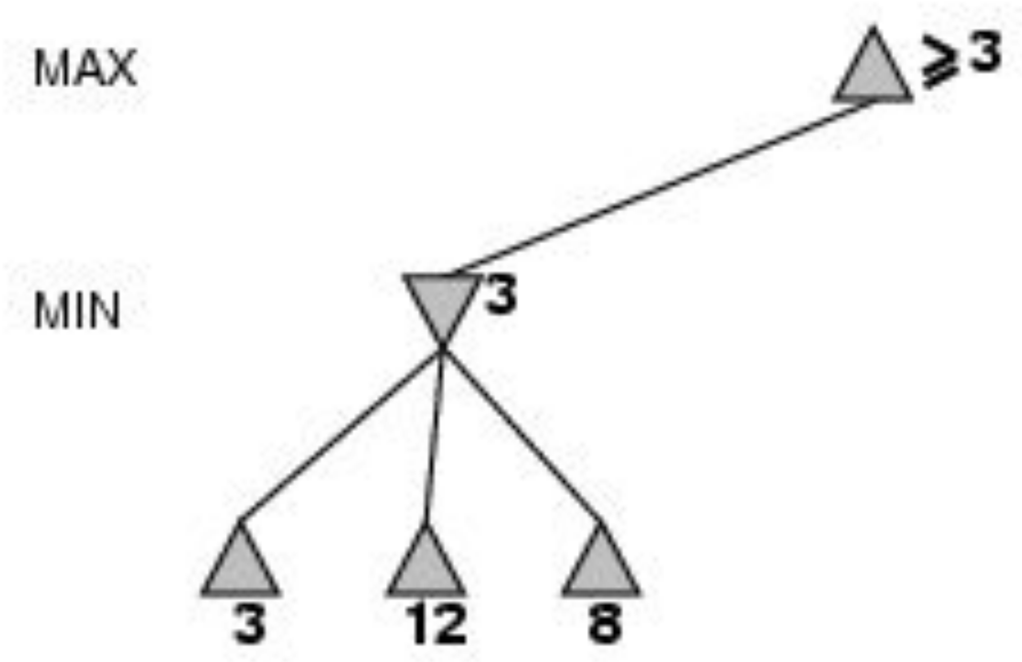


Minimax with pruning

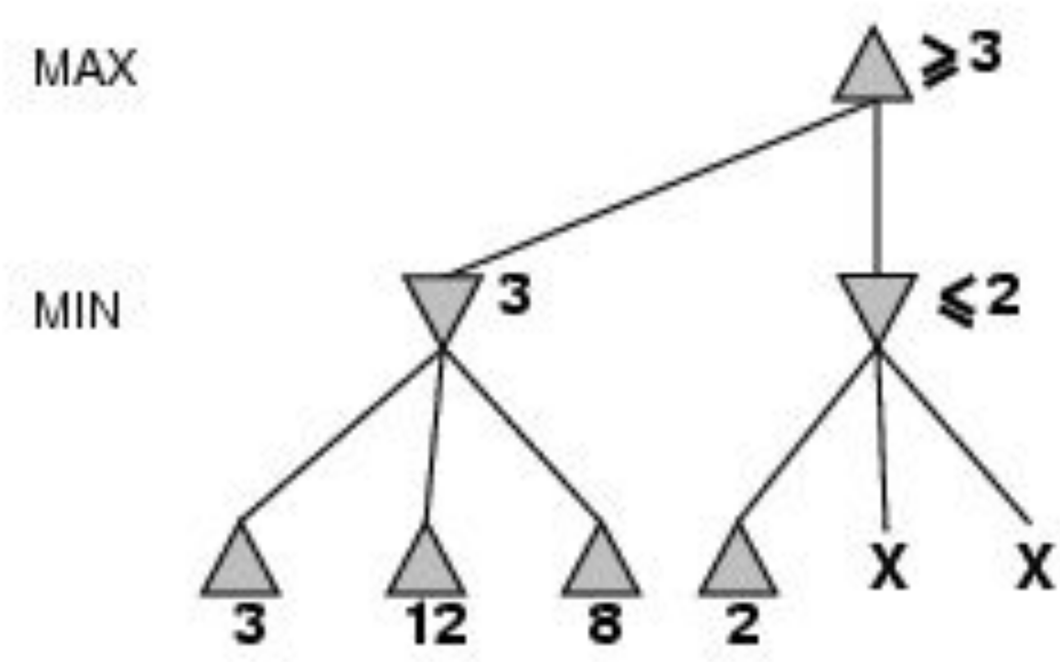
Pruning example



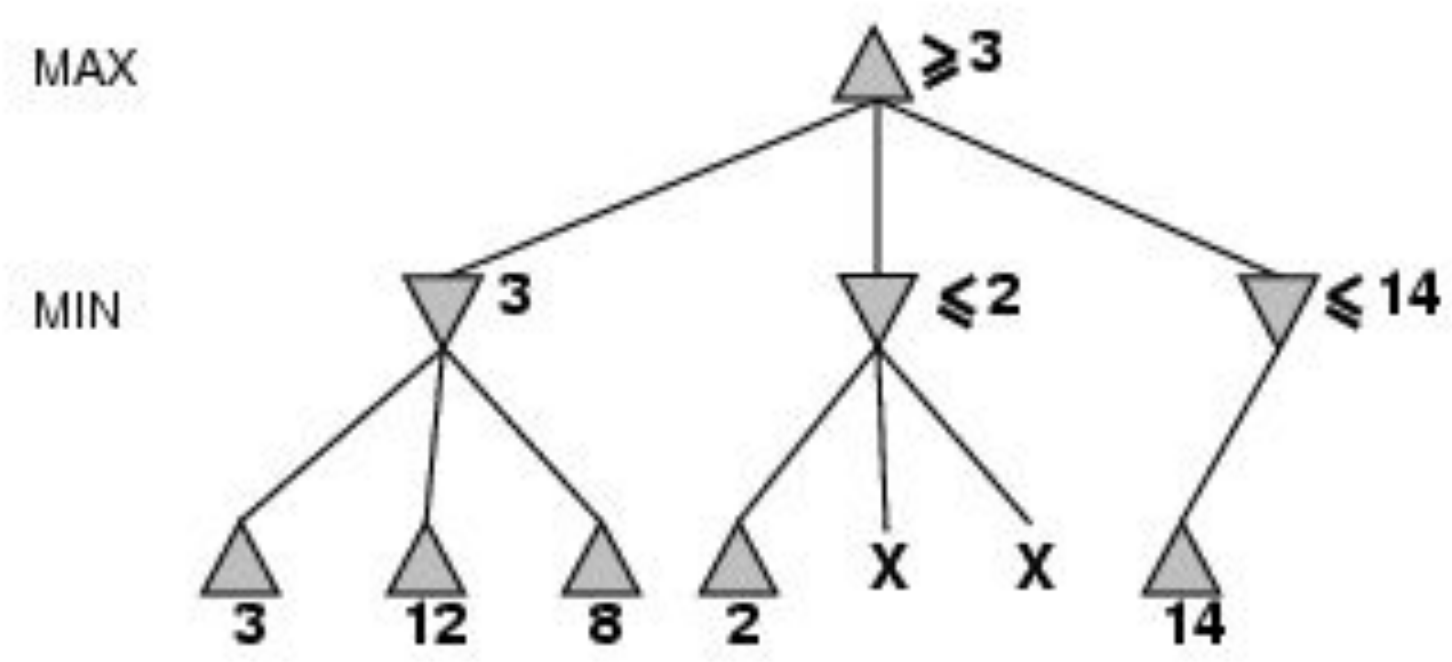
Pruning example



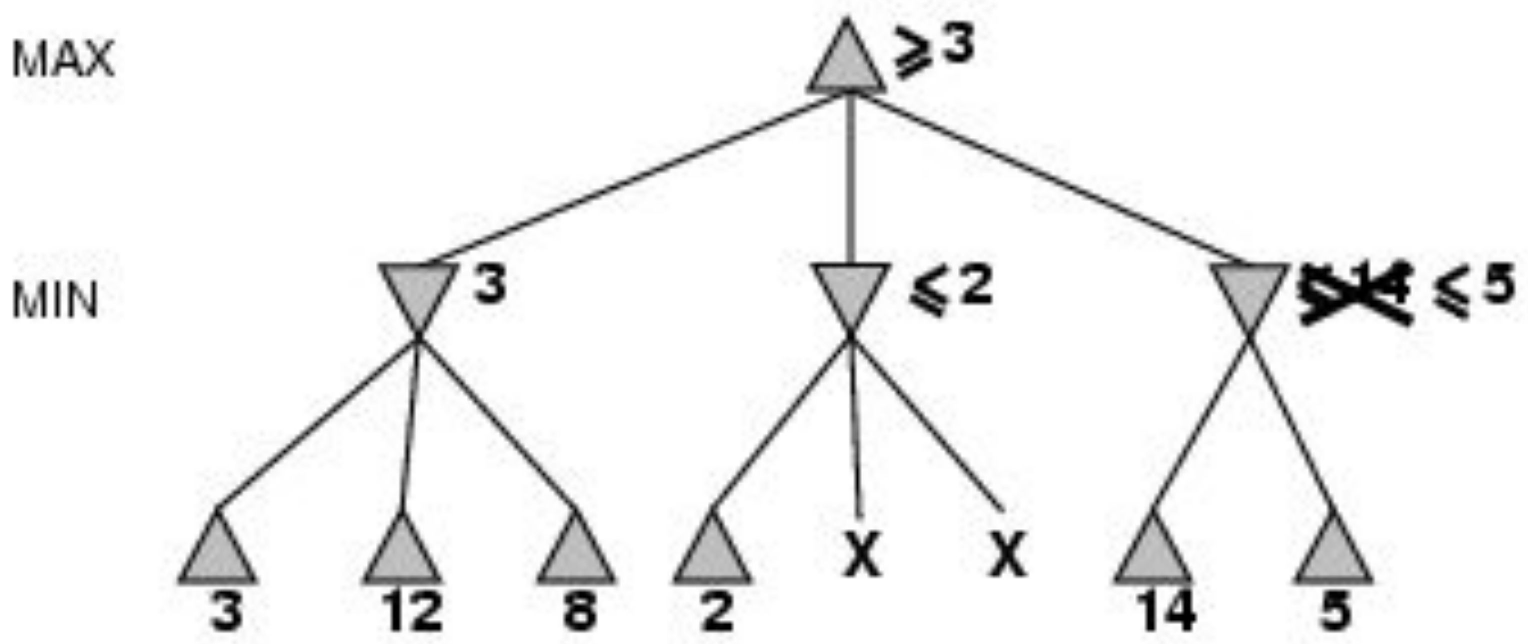
Pruning example



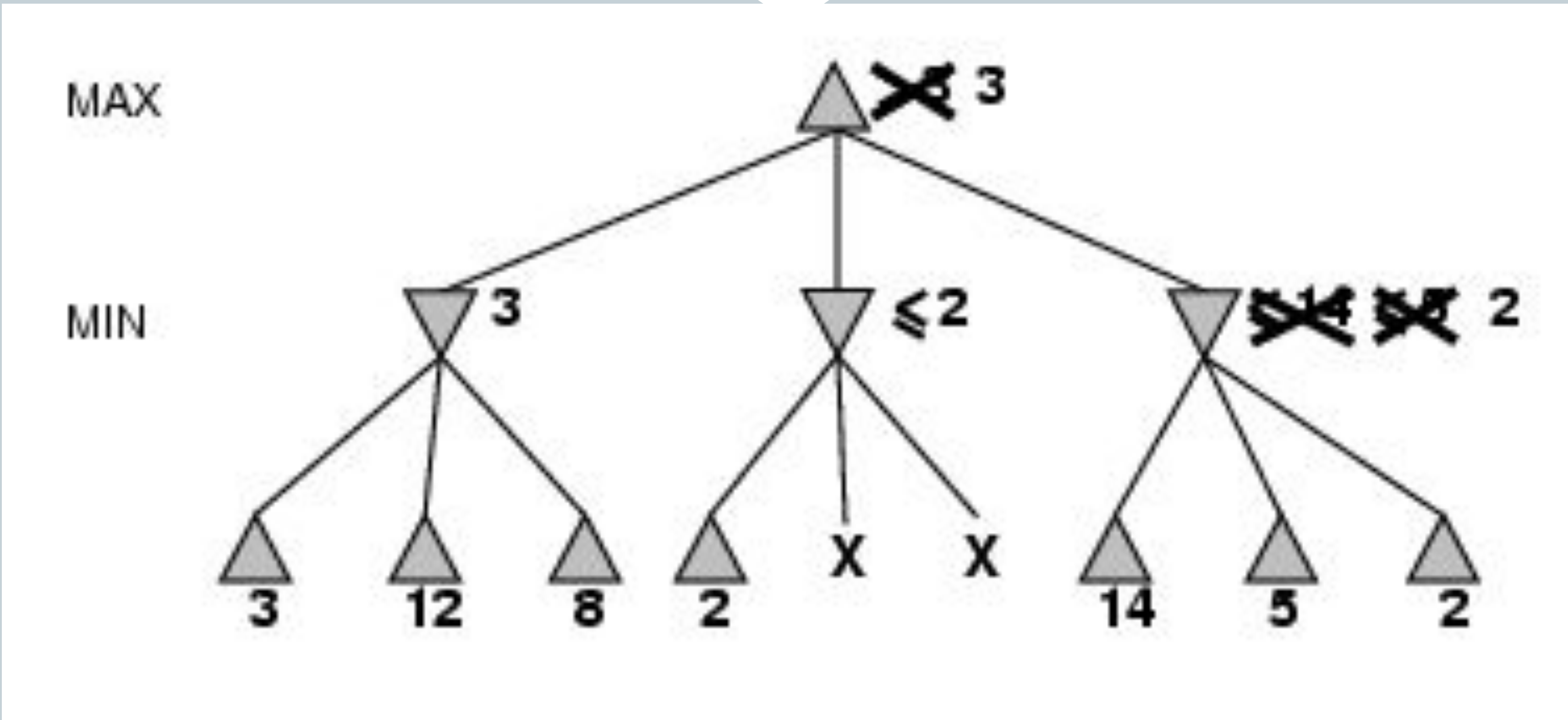
Pruning example



Pruning example

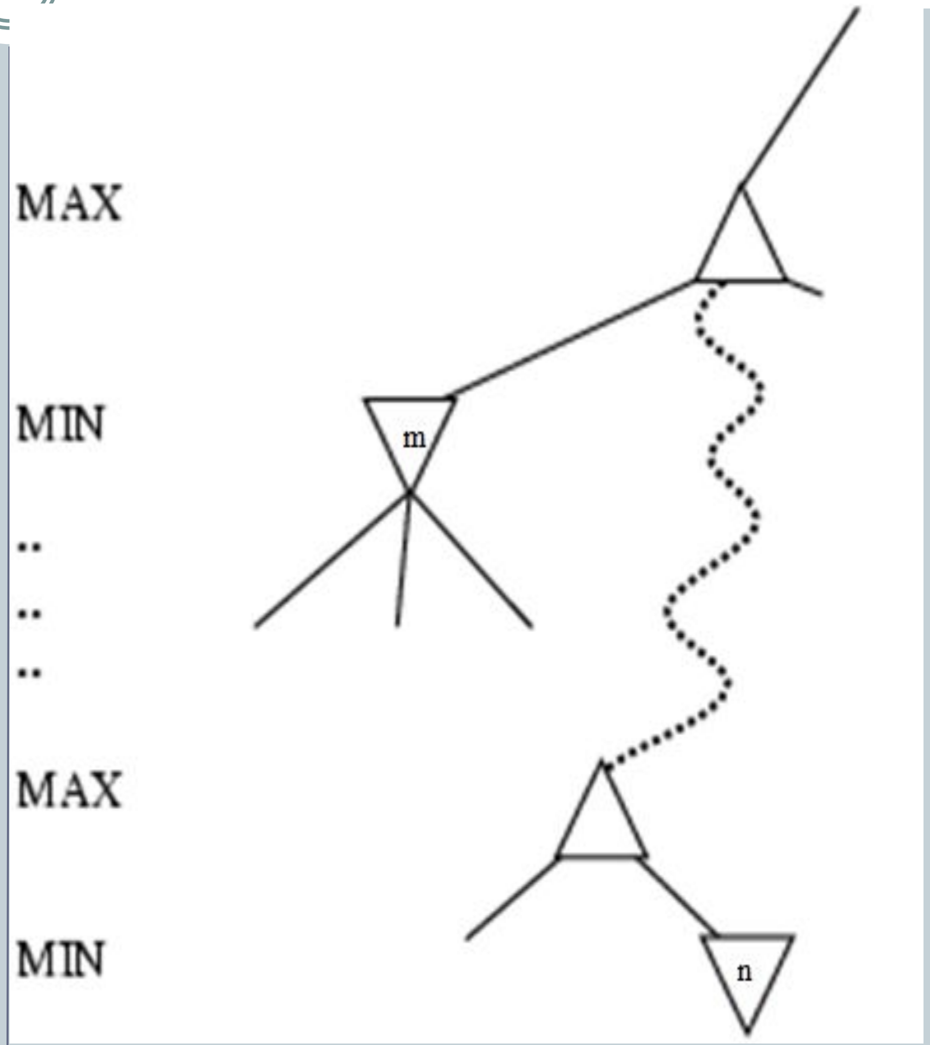


Pruning example



General case of alpha-beta pruning

- If m is better than n for MAX, we will never get to n in the play



Alpha-beta Algorithm

- Depth first search – only considers nodes along a single path at any time
- It gets its name from two parameters that describe the bounds on the backed-up values that appear anywhere along the path

α = the value of the best (highest-value) choice that we have found so far at any choice point along the path of MAX

β = the value of the best (lowest-value) choice that we have found so far at any choice point along the path of MIN

- update values of α and β during search and prunes remaining branches as soon as the value is known to be worse than the current α or β value for MAX or MIN respectively.

The α-β algorithm

function ALPHA-BETA-SEARCH(*state*) **returns** *an action*
  inputs: *state*, current state in game

  $v \leftarrow$ MAX-VALUE(*state*, $-\infty, +\infty$)
  return the *action* in SUCCESSORS(*state*) with value v

---

function MAX-VALUE(*state*, α, β) **returns** *a utility value*
  inputs: *state*, current state in game
    α, the value of the best alternative for MAX along the path to *state*
    β, the value of the best alternative for MIN along the path to *state*

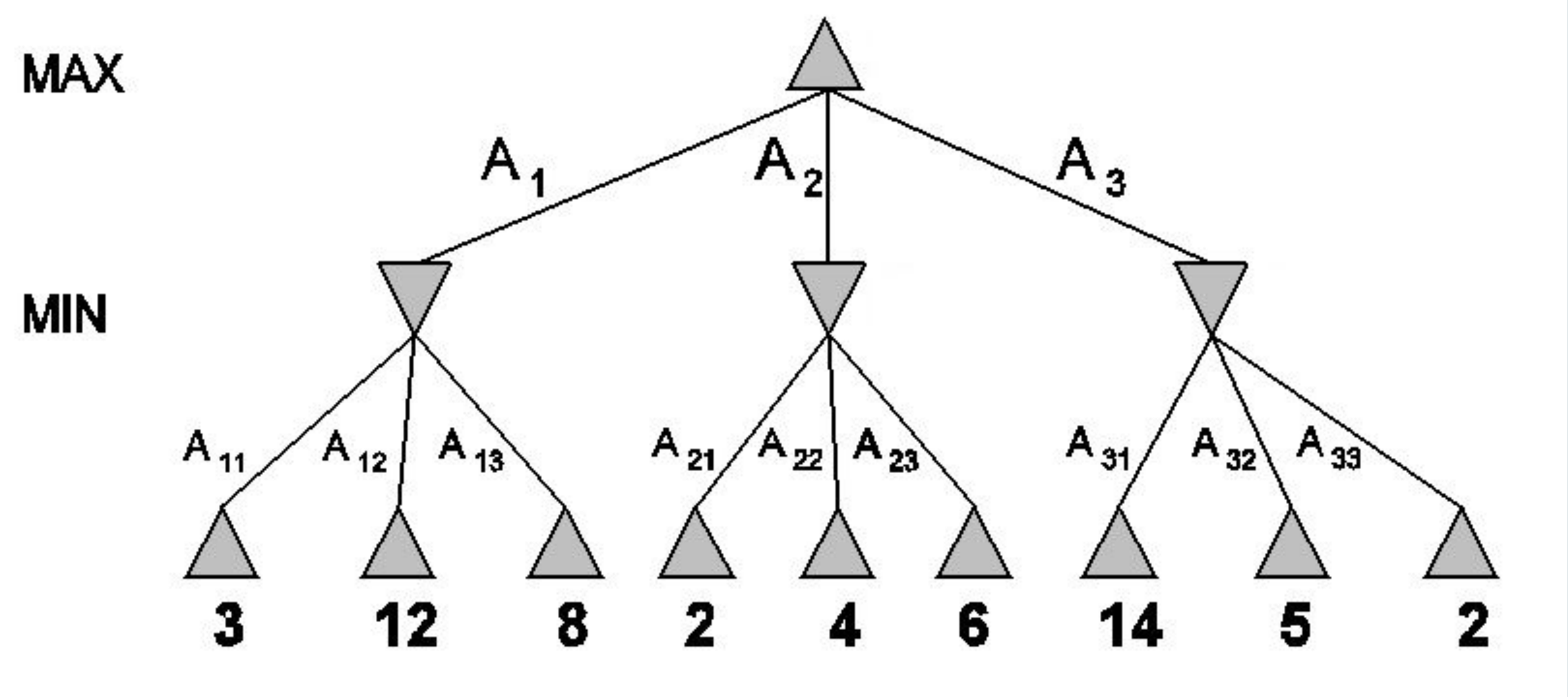
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s, α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

The α-β algorithm

function MIN-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  inputs: *state*, current state in game
    α, the value of the best alternative for MAX along the path to *state*
    β, the value of the best alternative for MIN along the path to *state*

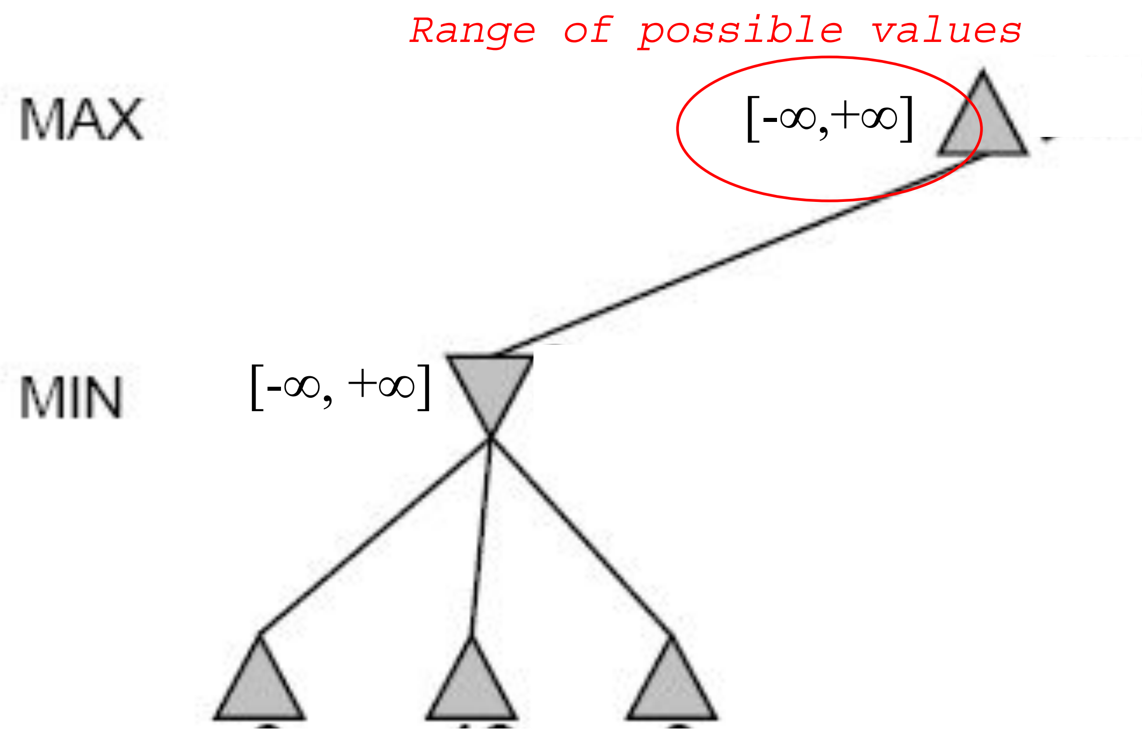
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow +\infty$
  for a, s in SUCCESSORS($state$) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

Alpha-Beta Pruning example

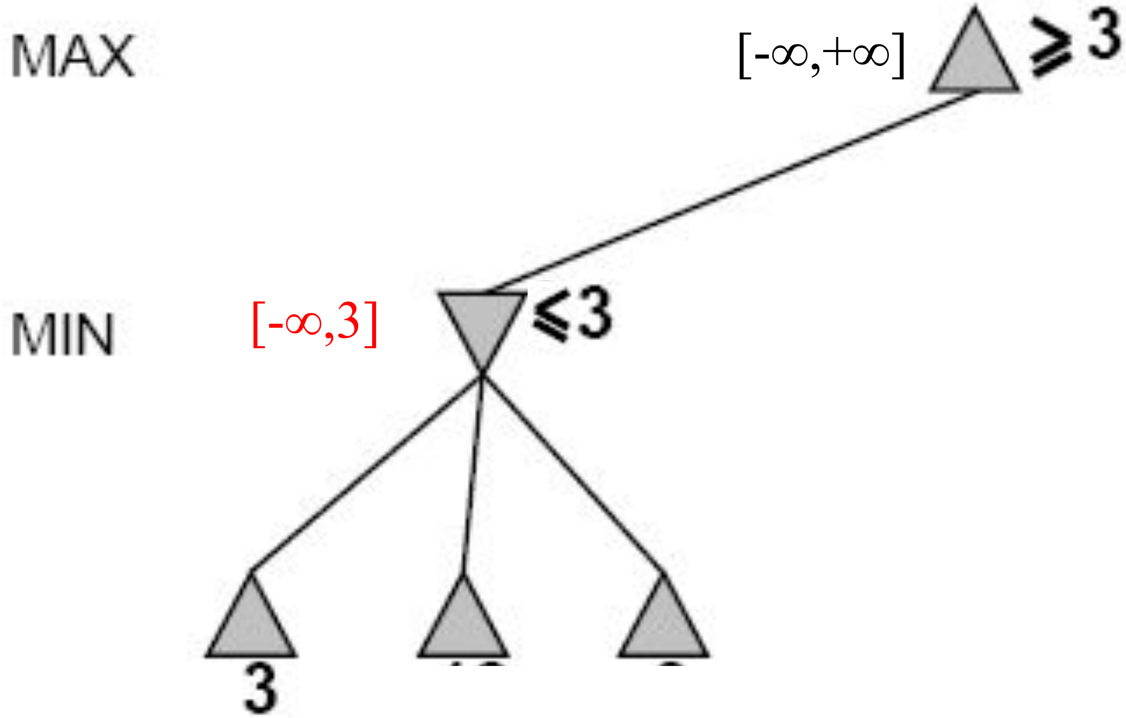


Alpha-Beta Example

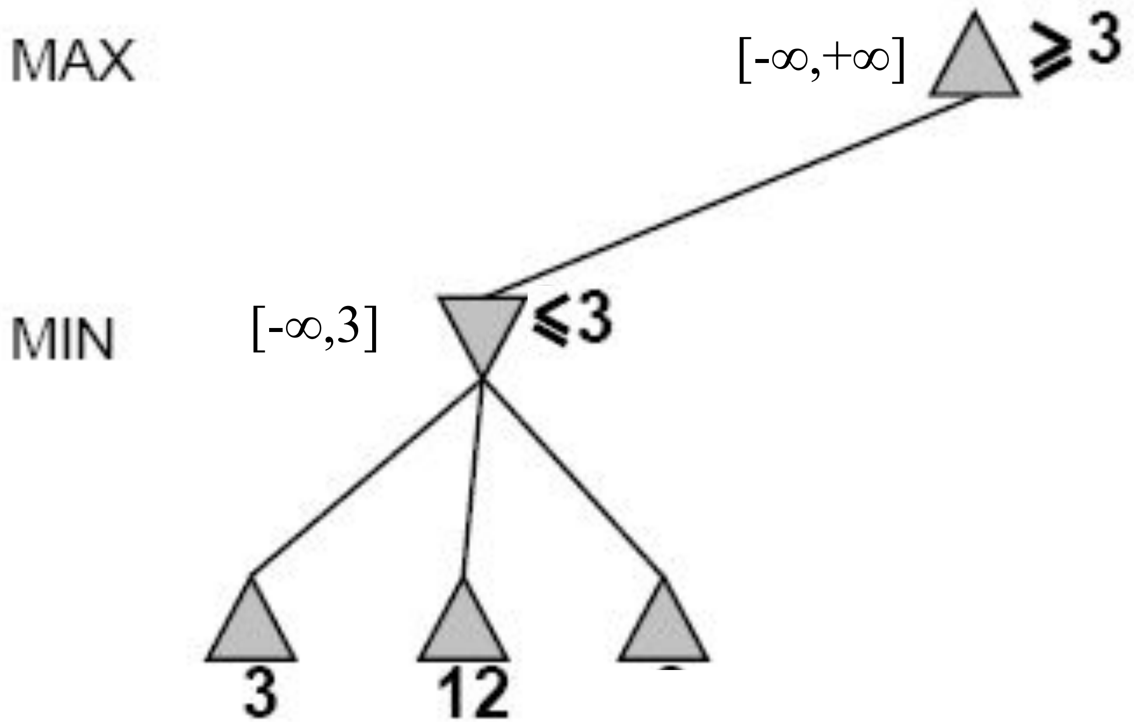
Do DF-search until first leaf



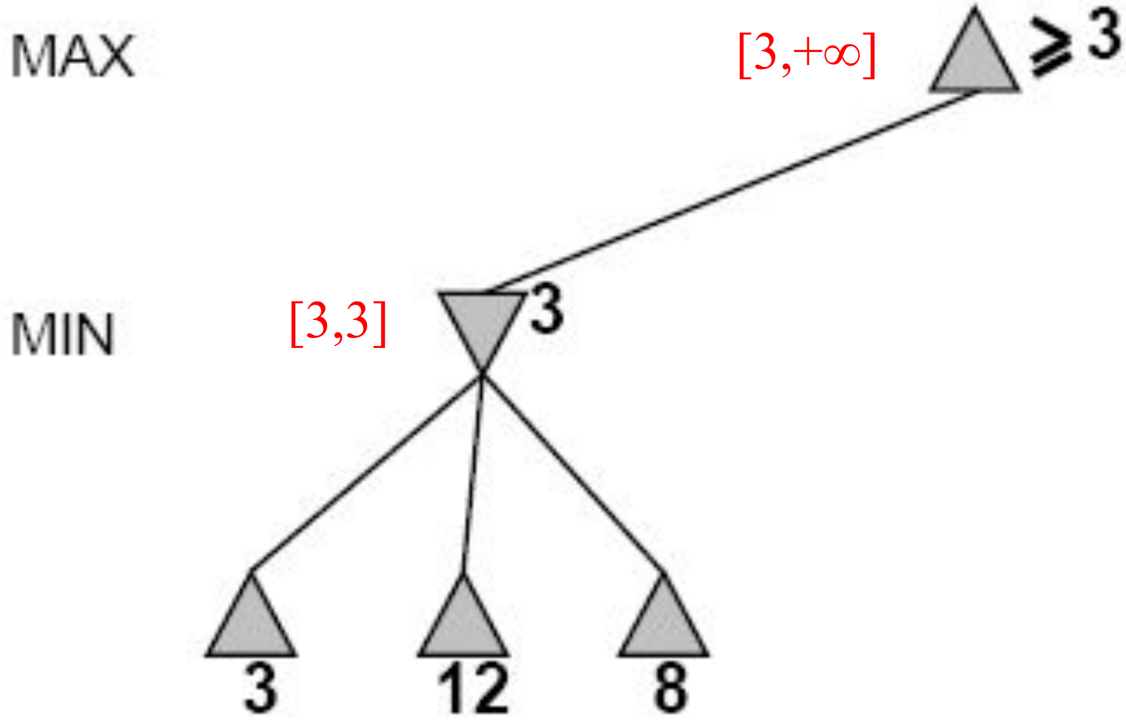
Alpha-Beta Example (continued)



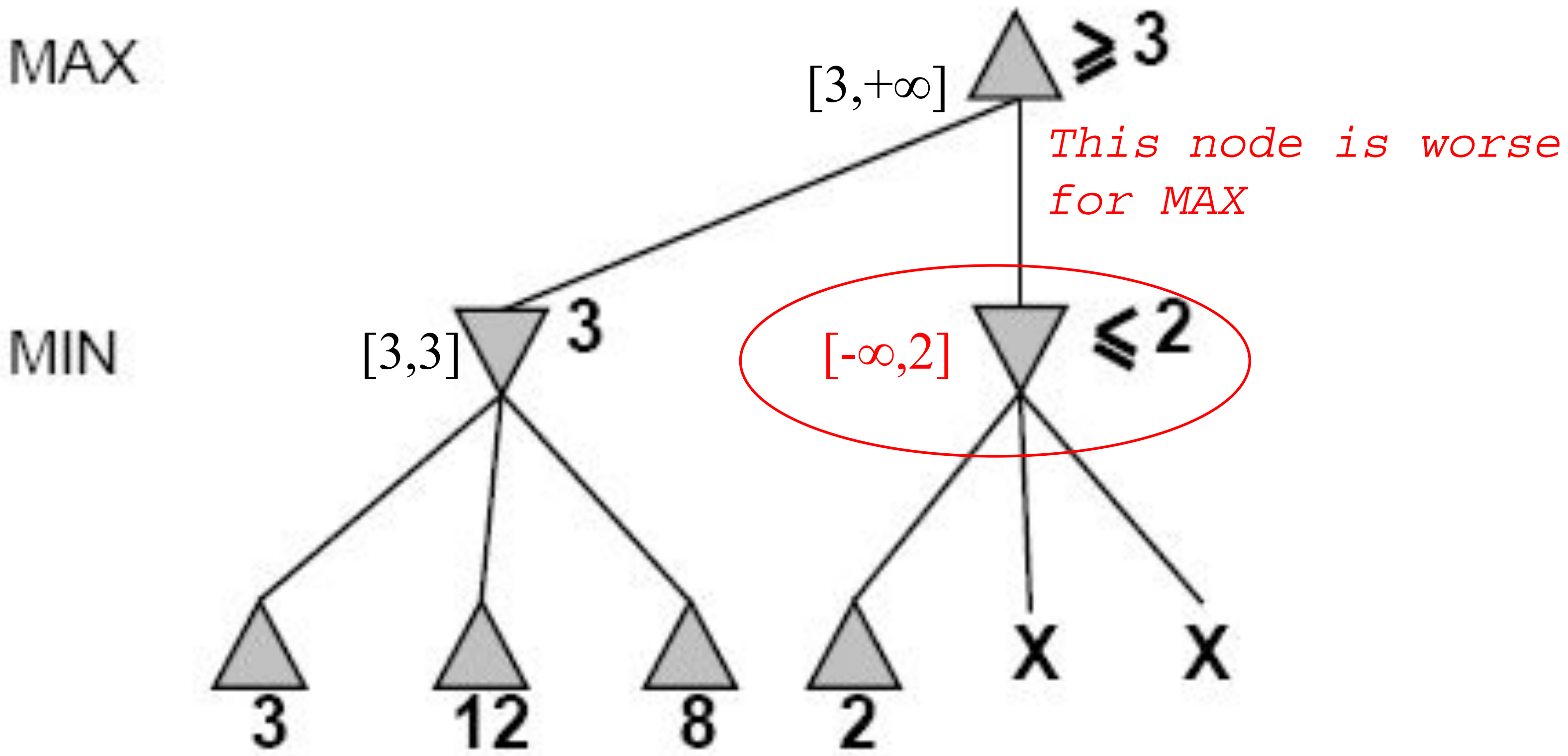
Alpha-Beta Example (continued)



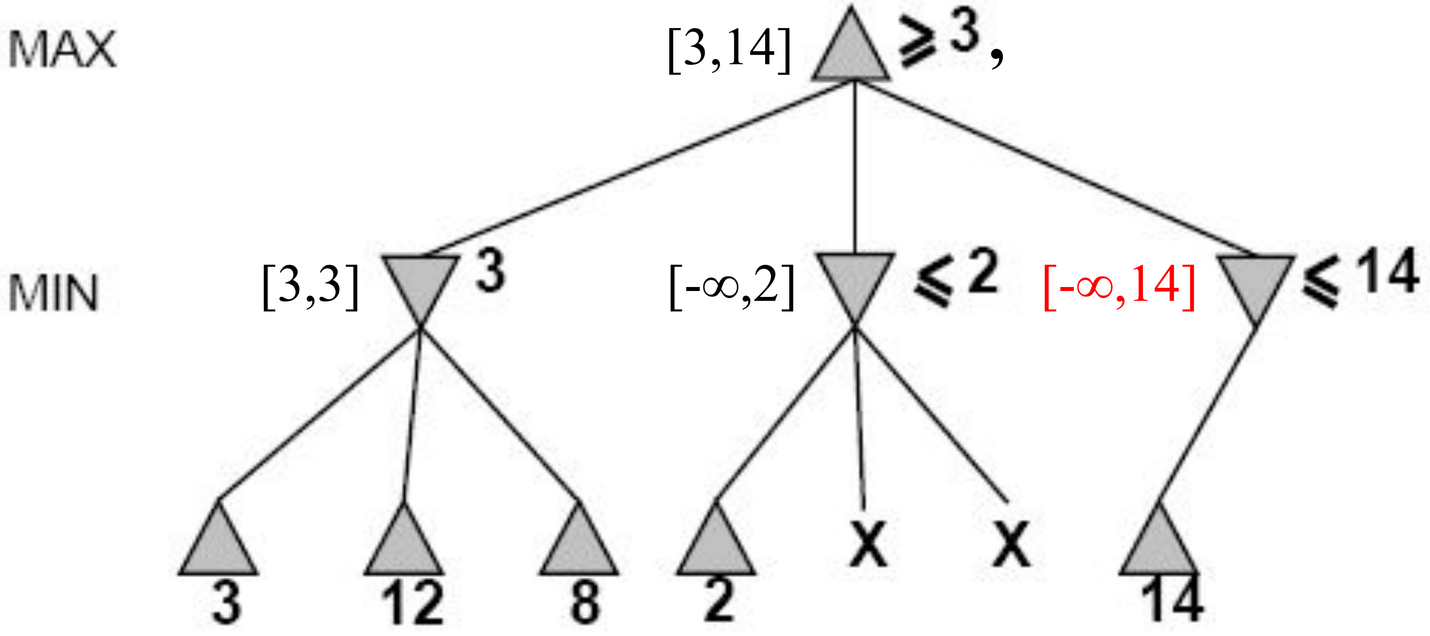
Alpha-Beta Example (continued)



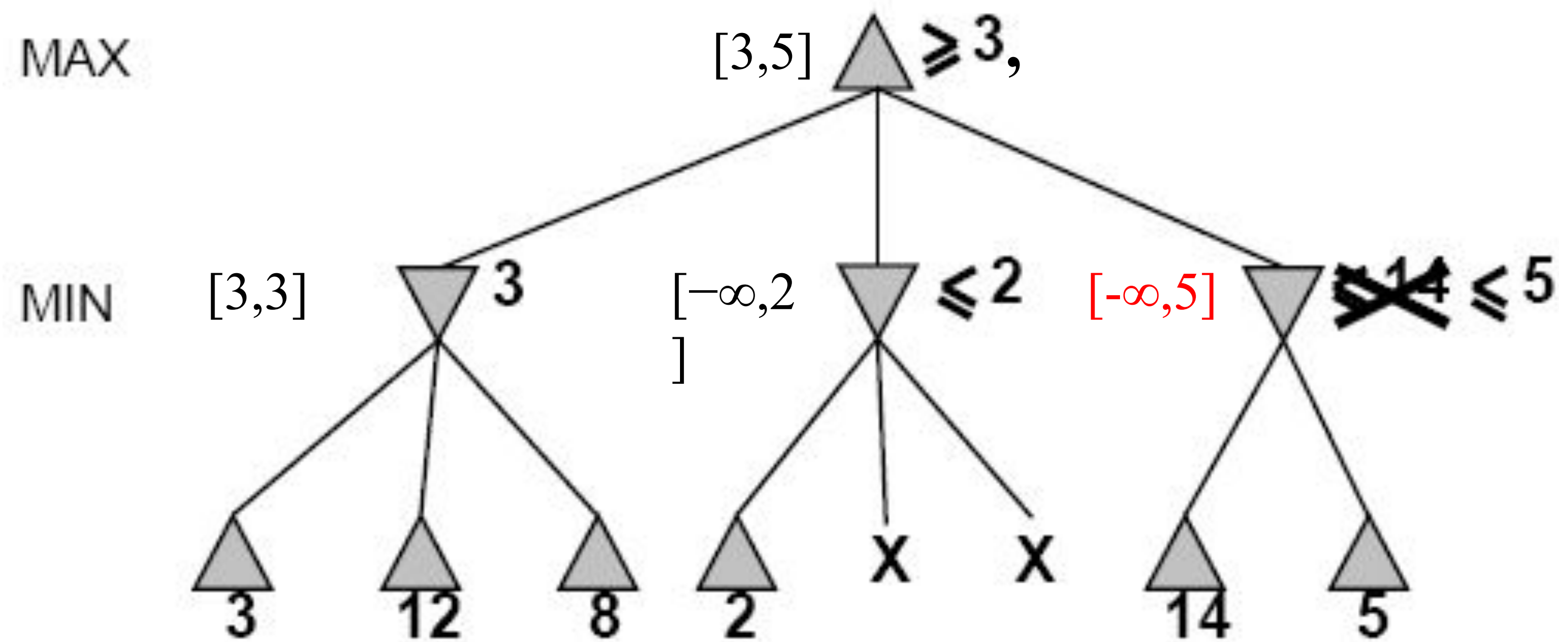
Alpha-Beta Example (continued)



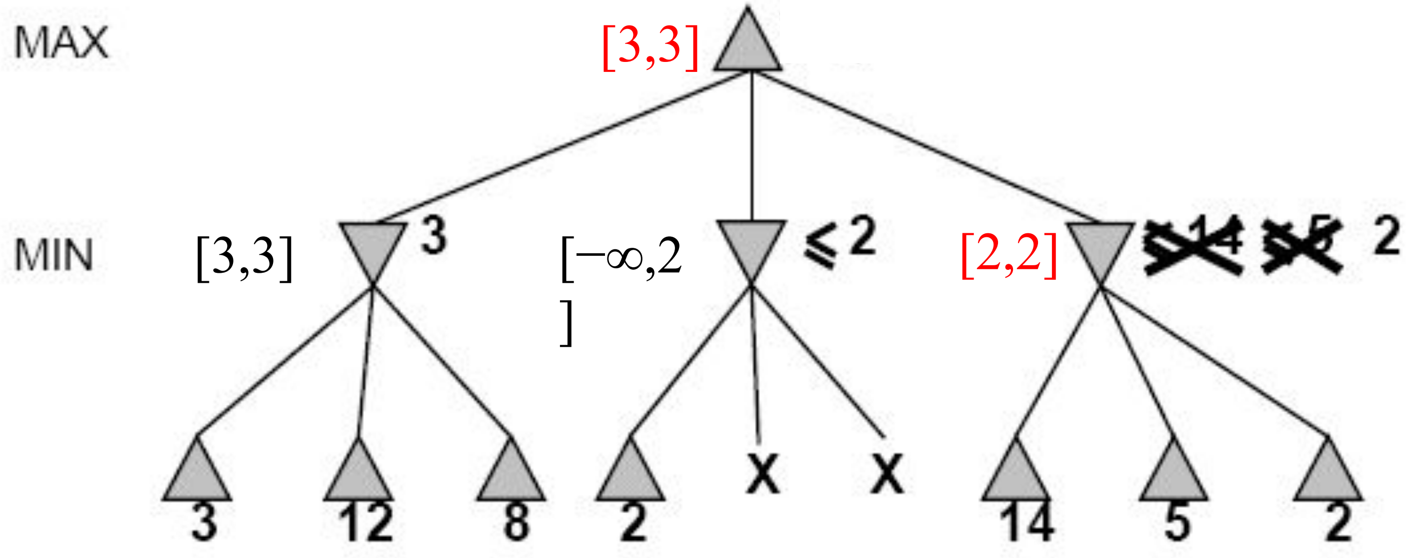
Alpha-Beta Example (continued)



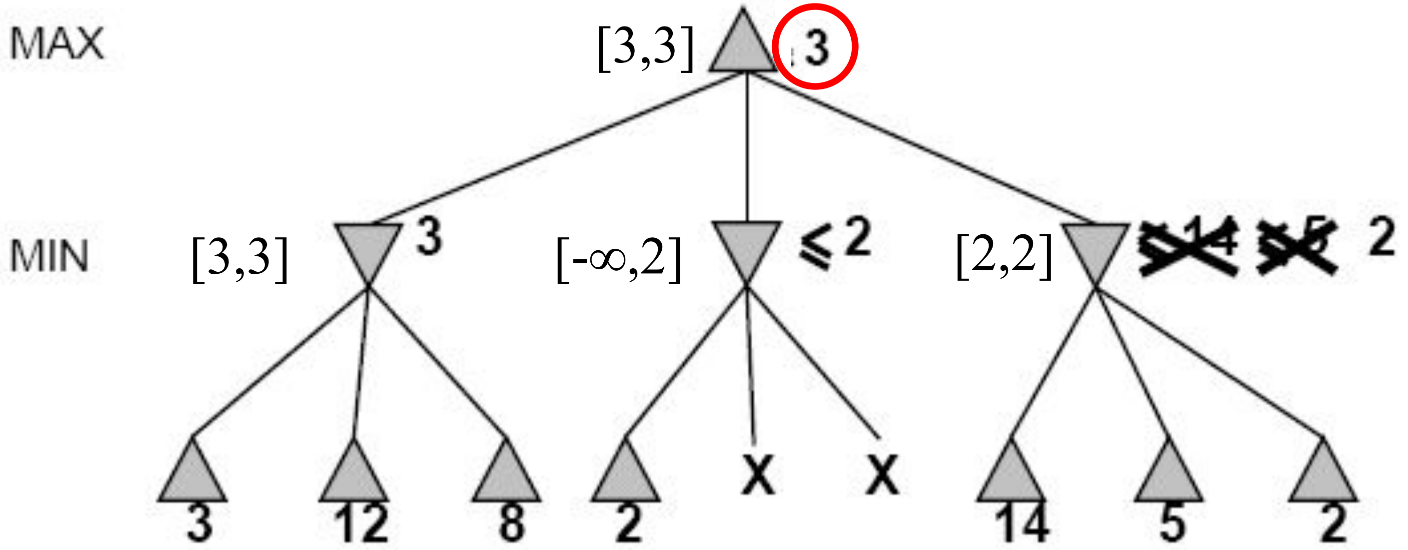
Alpha-Beta Example (continued)



Alpha-Beta Example (continued)



Alpha-Beta Example (continued)



Properties of α-β

- Pruning does not affect final result
  Effectiveness highly depends on the order in which the states are examined (in prev ex we could not prune any successor of min node in right branch at all as the worst successor from the MIN viewpoint were generated first)
- Good move ordering improves effectiveness of pruning
  With "perfect ordering," time complexity = $O(b^{m/2})$
  - doubles depth of search i.e. it can solve a tree roughly twice as deep as minimax in the same amount of time
- If successors are examined in random order than best-first, the total number of nodes examined are roughly $O(b^{3m/4})$ for moderate b
- Killer moves, transpositions, transposition table- Self learn

Alpha-Beta Example 2