COMP 472 Artificial Intelligence: Adversarial Search part 4
Alpha-Beta Pruning video 2

- Russell & Norvig: Chapter 5

Today

- Adversarial Search
  1. Minimax
  2. Alpha-beta pruning 

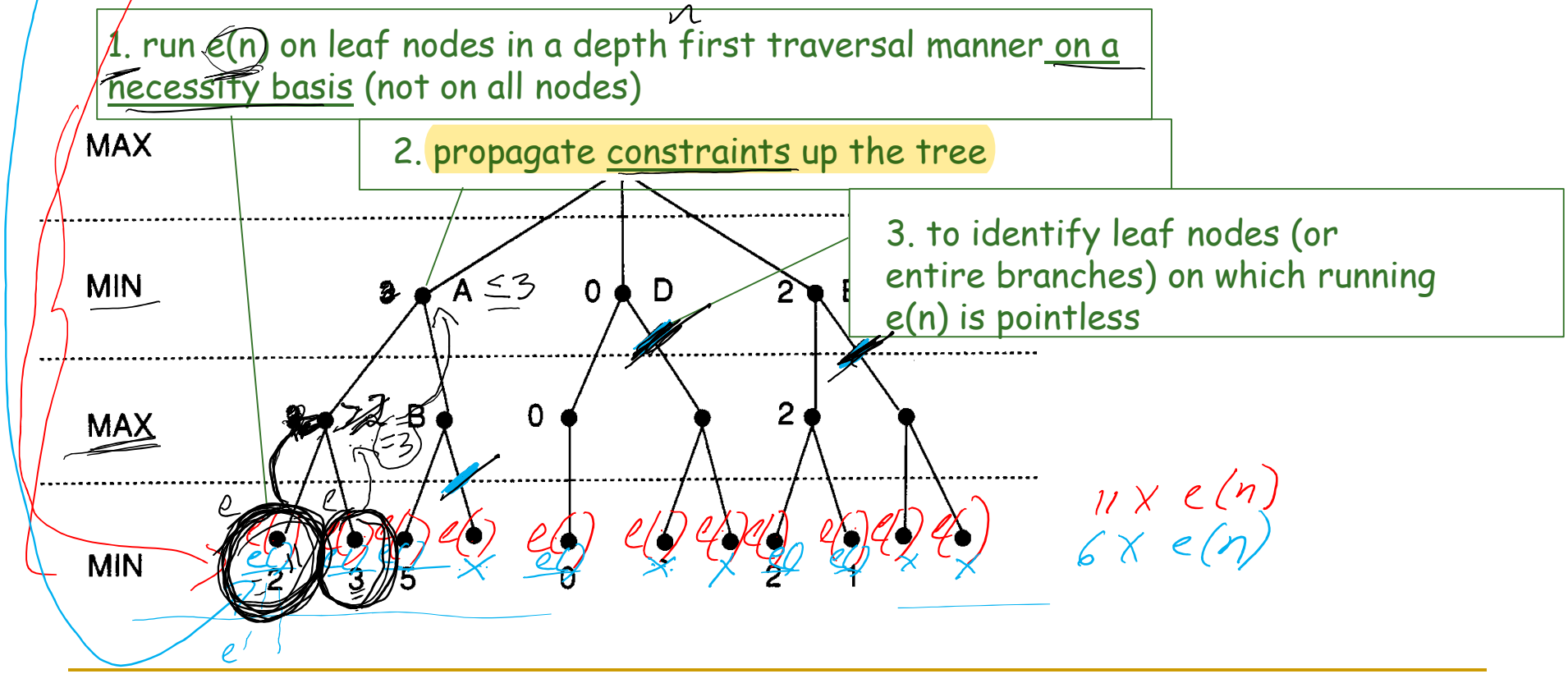
  3. Other Adversarial Searc.
     1. Multiplayer Games
     2. Stochastic Games
     3. Monte Carlo Tree Search

Alpha-Beta Pruning

- Optimization over Minimax, that:
  - ignores (cuts off/prunes) branches of the tree that cannot contribute to the solution
  - reduces branching factor
  - allows deeper search with same effort

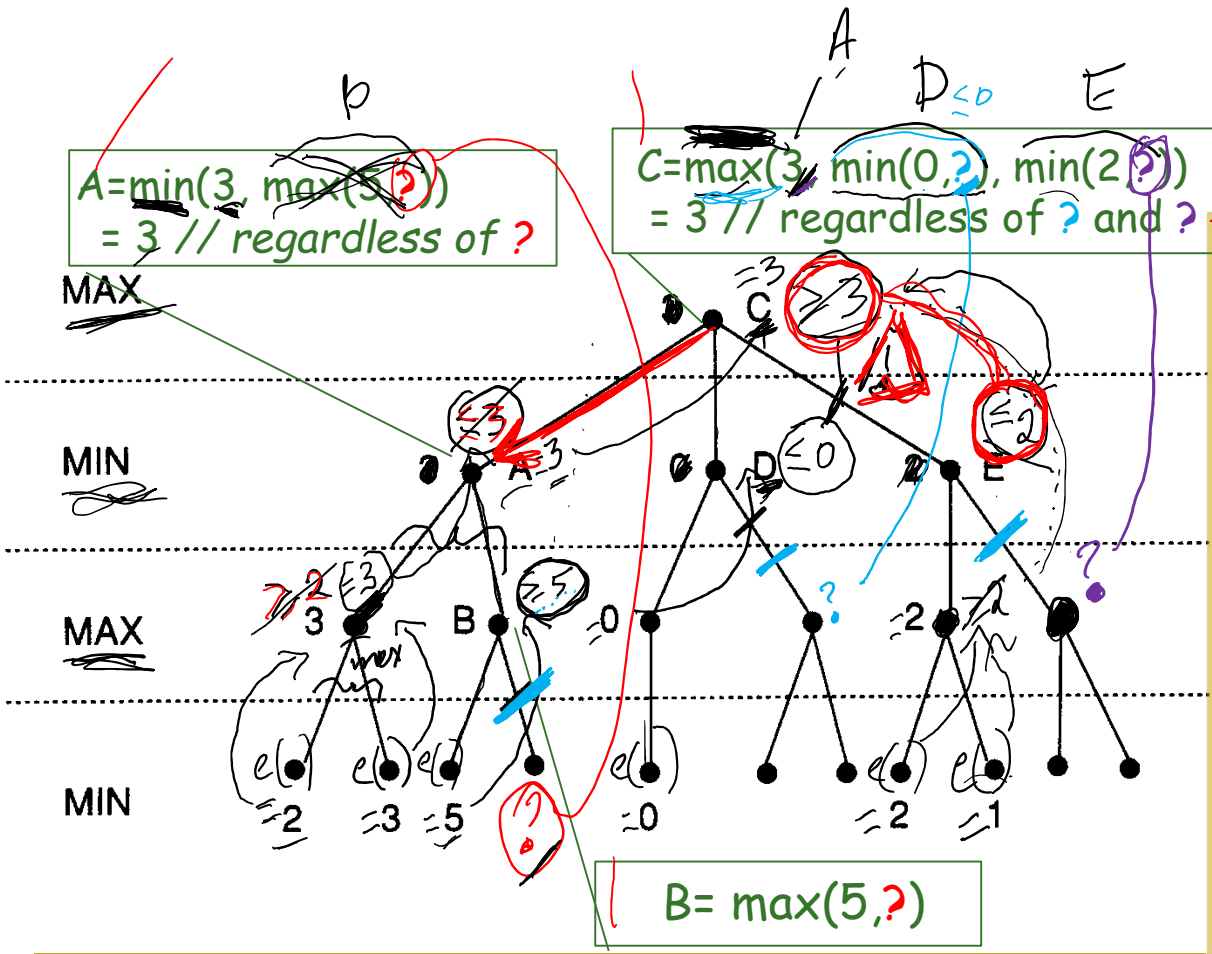
Alpha-Beta Pruning: Example 1

- With Minimax, we look at all nodes at depth *n*
- With α-β pruning, we ignore branches that could not possibly contribute to the final decision



source: G. Luger (2005)

Alpha-Beta Pruning: Example 1



B will be >= 5

So we can ignore B's right branch, because A must be 3

D will be <= 0

But C will be >= 3

So we can ignore D's right branch

E will be <= 2.

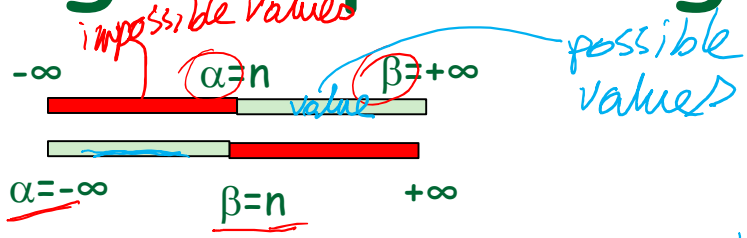
So we can ignore E's right branch

Because C will be 3.

source: G. Luger (2005)

Alpha-Beta Pruning – α pruning

- α = minimum possible value value ≥ n
- β = maximum possible value value ≤ n

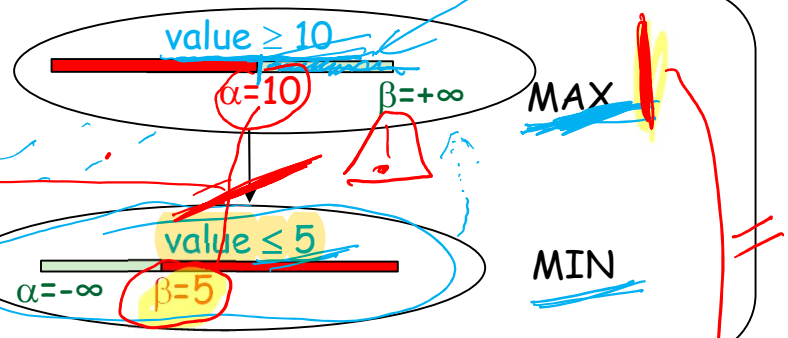


- Alpha pruning – parent is a MAX node

1. if MAX node's α = n, then we can prune branches from a MIN descendant that has a β ≤ n.

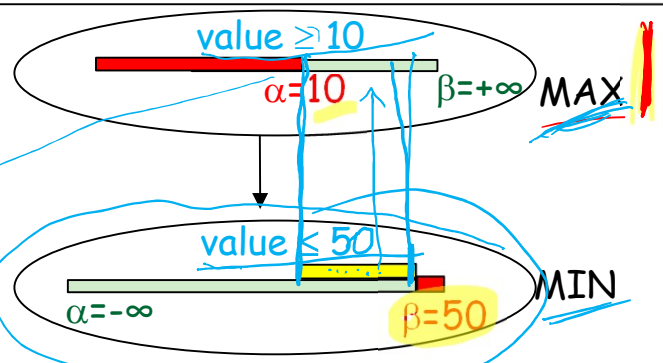
   if β_child ≤ α_ancestor → prune

   incompatible...
   so stop searching this branch;
   the value cannot come from there!



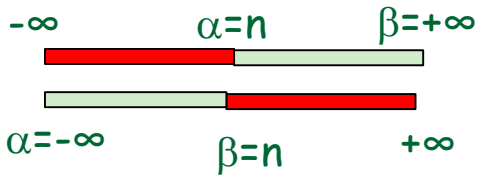
2. if β_child > α_ancestor → cannot prune

   compatible...
   we need to search this branch;
   the value could come from there!



Alpha-Beta Pruning - β pruning

- α = minimum possible value — value ≥ n
- β = maximum possible value — value ≤ n

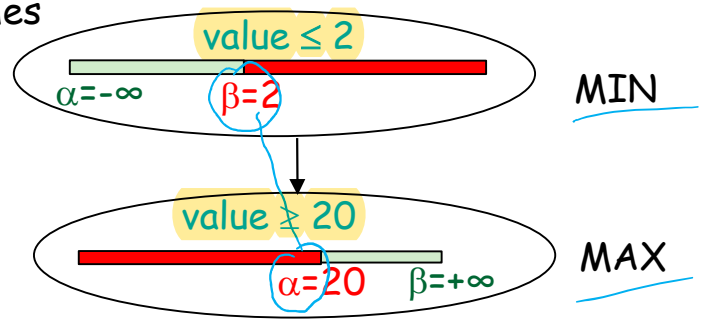


same

- Beta pruning – parent is a MIN node

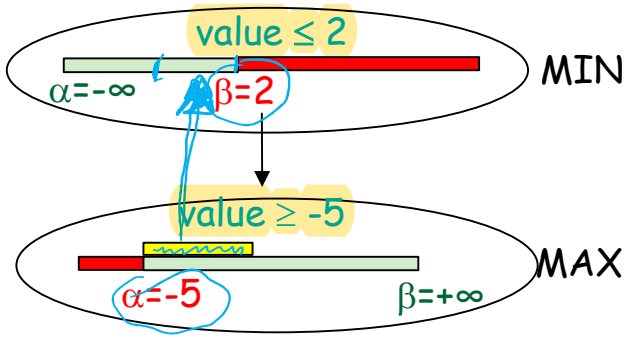
1. if a MIN node's β = n, then we can prune branches from a MAX descendant that has an α ≥ n.
   if α_child ≥ β _ancestor → prune

incompatible...
so stop searching this branch;
the value cannot come from there!



2. if α_child < β _ancestor → cannot prune

compatible...
we need to search this branch;
the value could come from there!



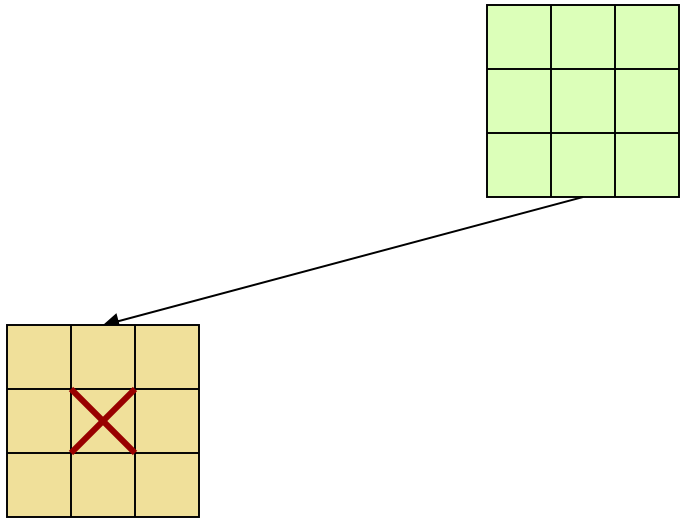
Alpha-Beta Pruning Algorithm

```
01 function alphabeta(node, depth, α, β, maximizingPlayer)
02      if depth = 0 or node is a terminal node
03          return the heuristic value of node
04      if maximizingPlayer
05          v := -∞
06          for each child of node
07              v := max(v, alphabeta(child, depth - 1, α, β, FALSE))
08              α := max(α, v)
09              if β ≤ α
10                  break (* β cut-off *)
11          return v
12      else
13          v := ∞
14          for each child of node
15              v := min(v, alphabeta(child, depth - 1, α, β, TRUE))
16              β := min(β, v)
17              if β ≤ α
18                  break (* α cut-off *)
19          return v
```

Initial call:
`alphabeta(origin, depth, -∞, +∞, TRUE)`

source: http://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning

Example with tic-tac-toe

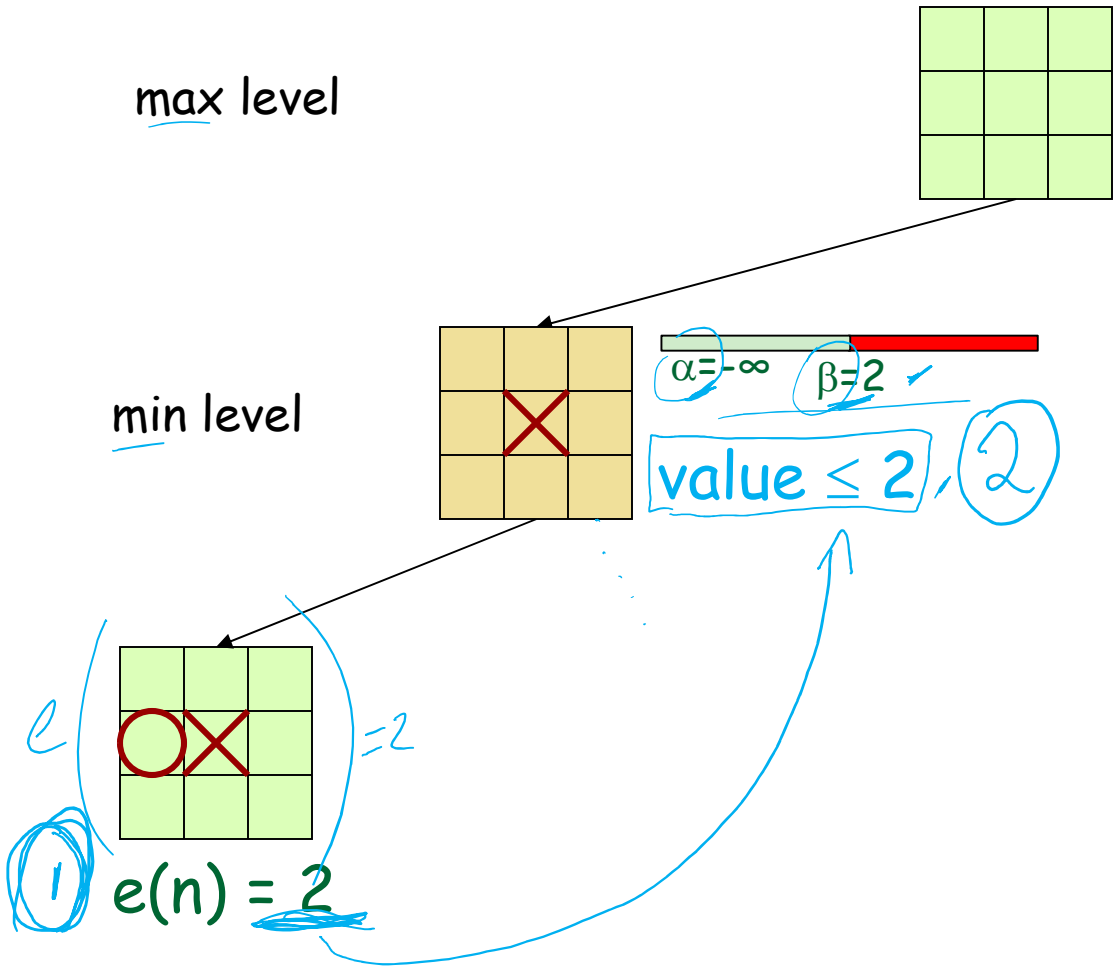


max level

min level

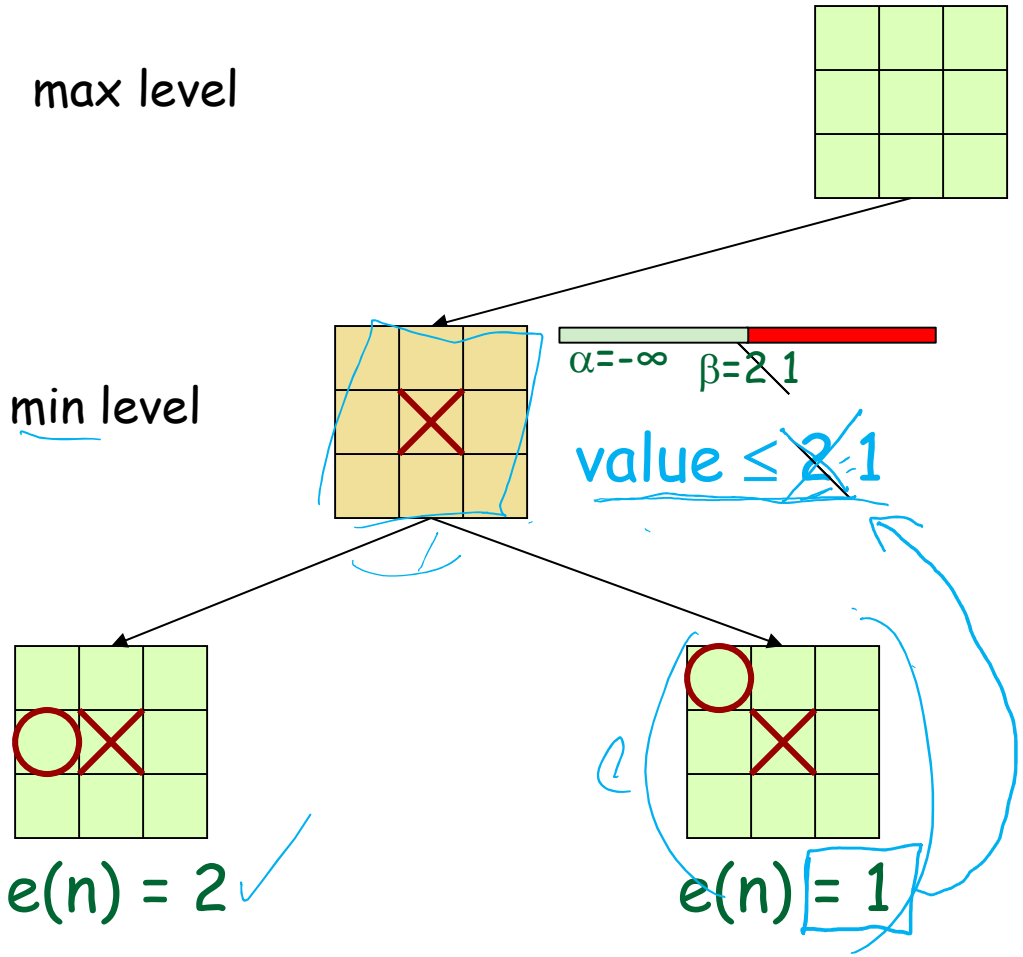
source: robotics.stanford.edu/~latombe/cs121/2003/home.htm

Example with tic-tac-toe



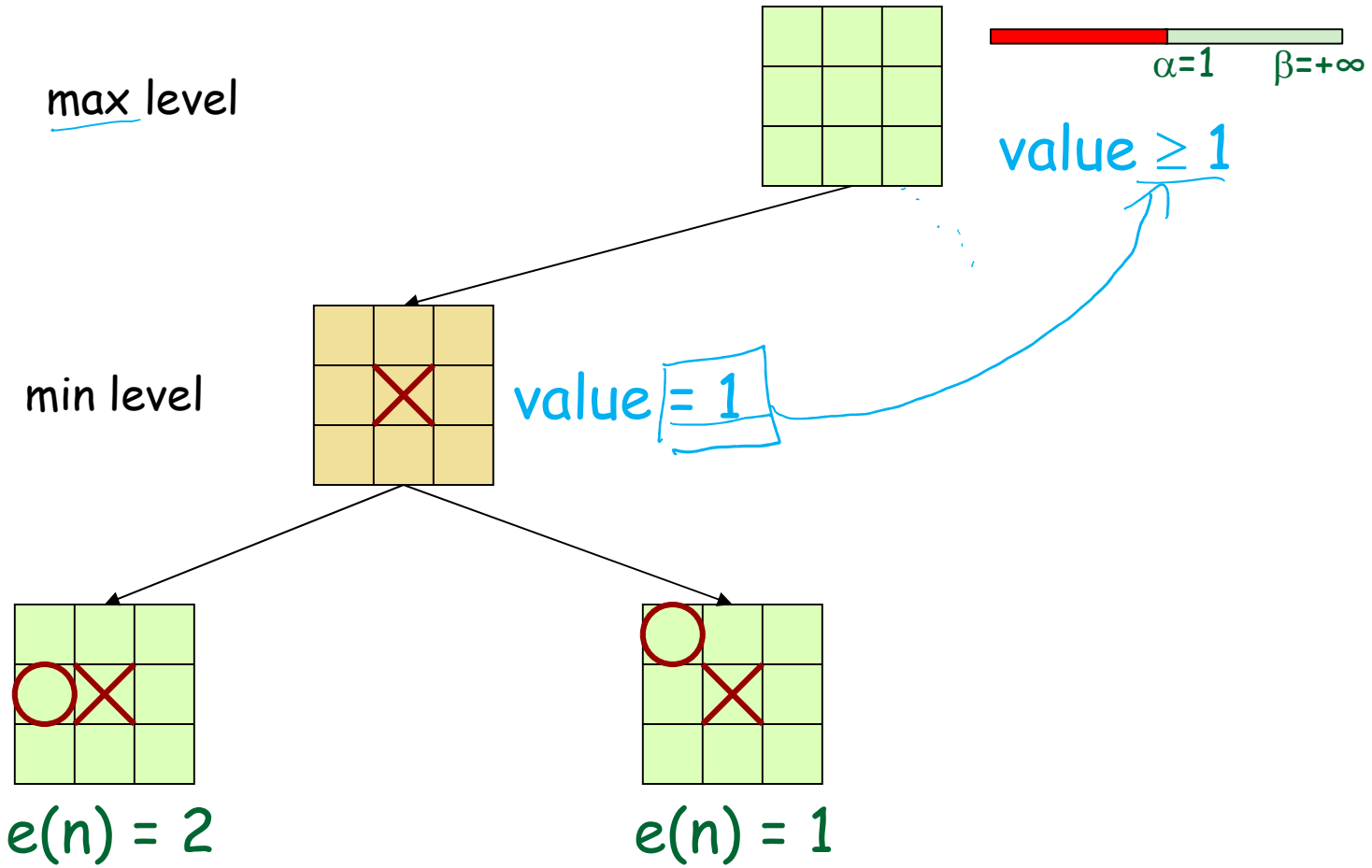
source: robotics.stanford.edu/~latombe/cs121/2003/home.htm

Example with tic-tac-toe



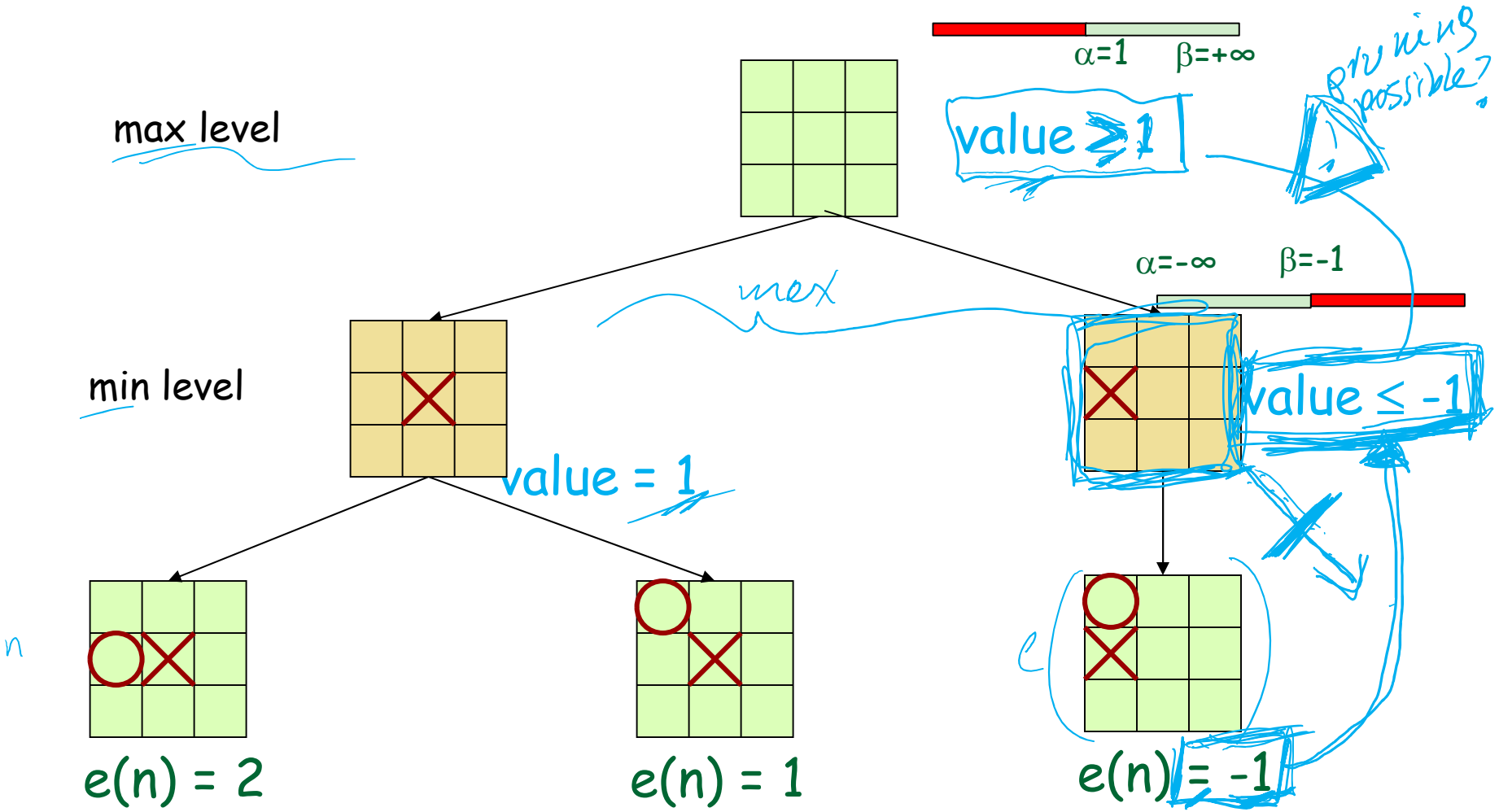
source: robotics.stanford.edu/~latombe/cs121/2003/home.htm

Example with tic-tac-toe



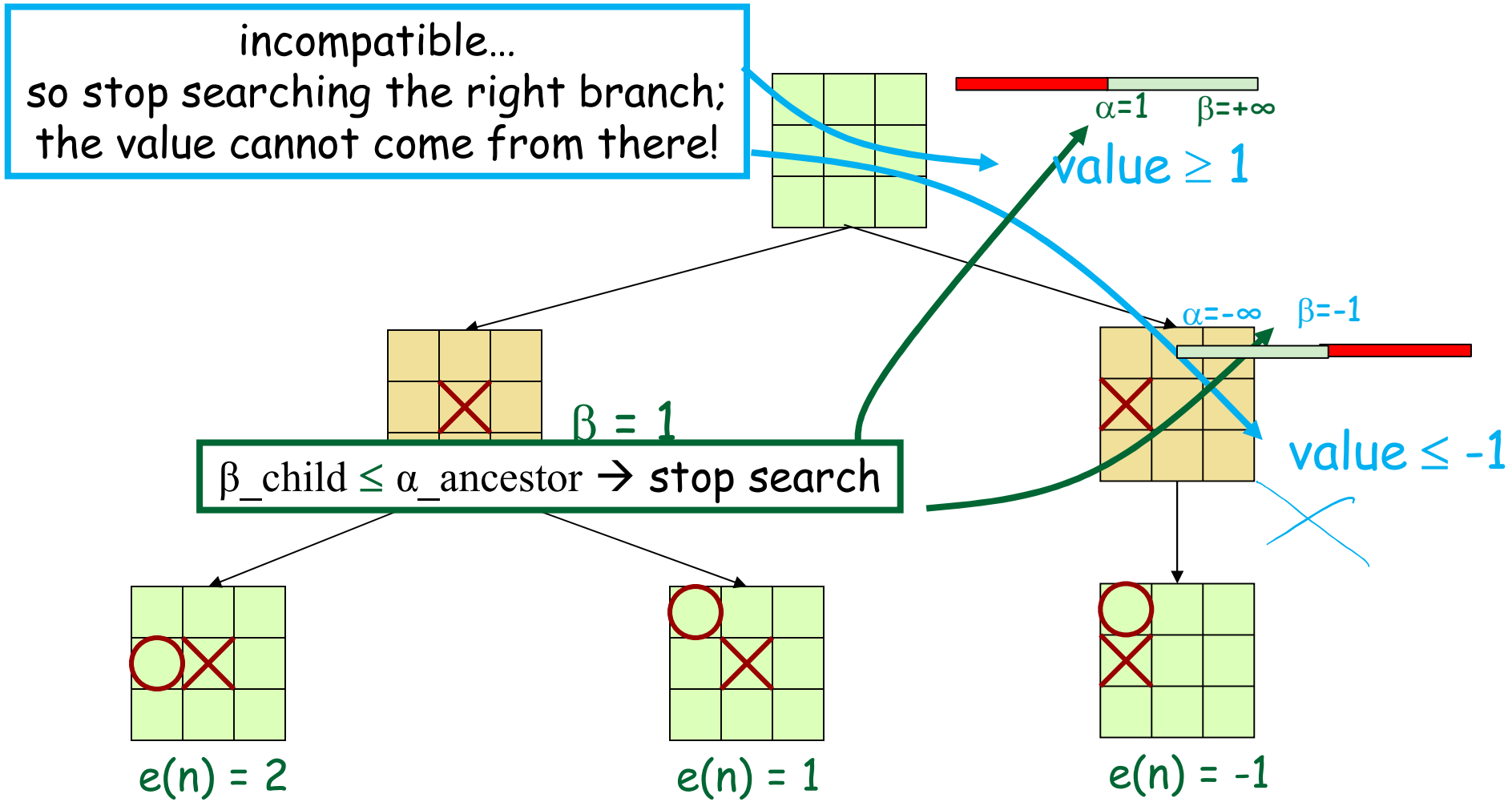
source: robotics.stanford.edu/~latombe/cs121/2003/home.htm

Example with tic-tac-toe



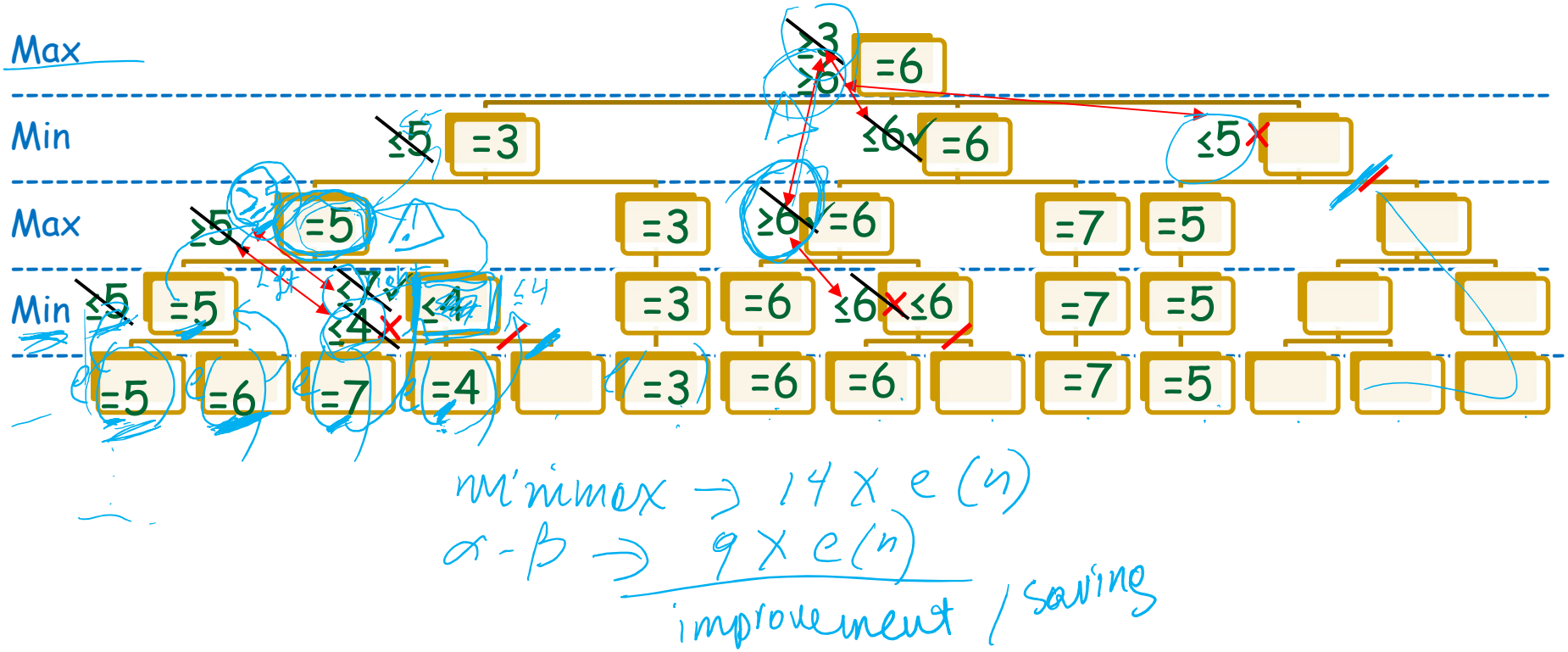
source: robotics.stanford.edu/~latombe/cs121/2003/home.htm

Example with tic-tac-toe



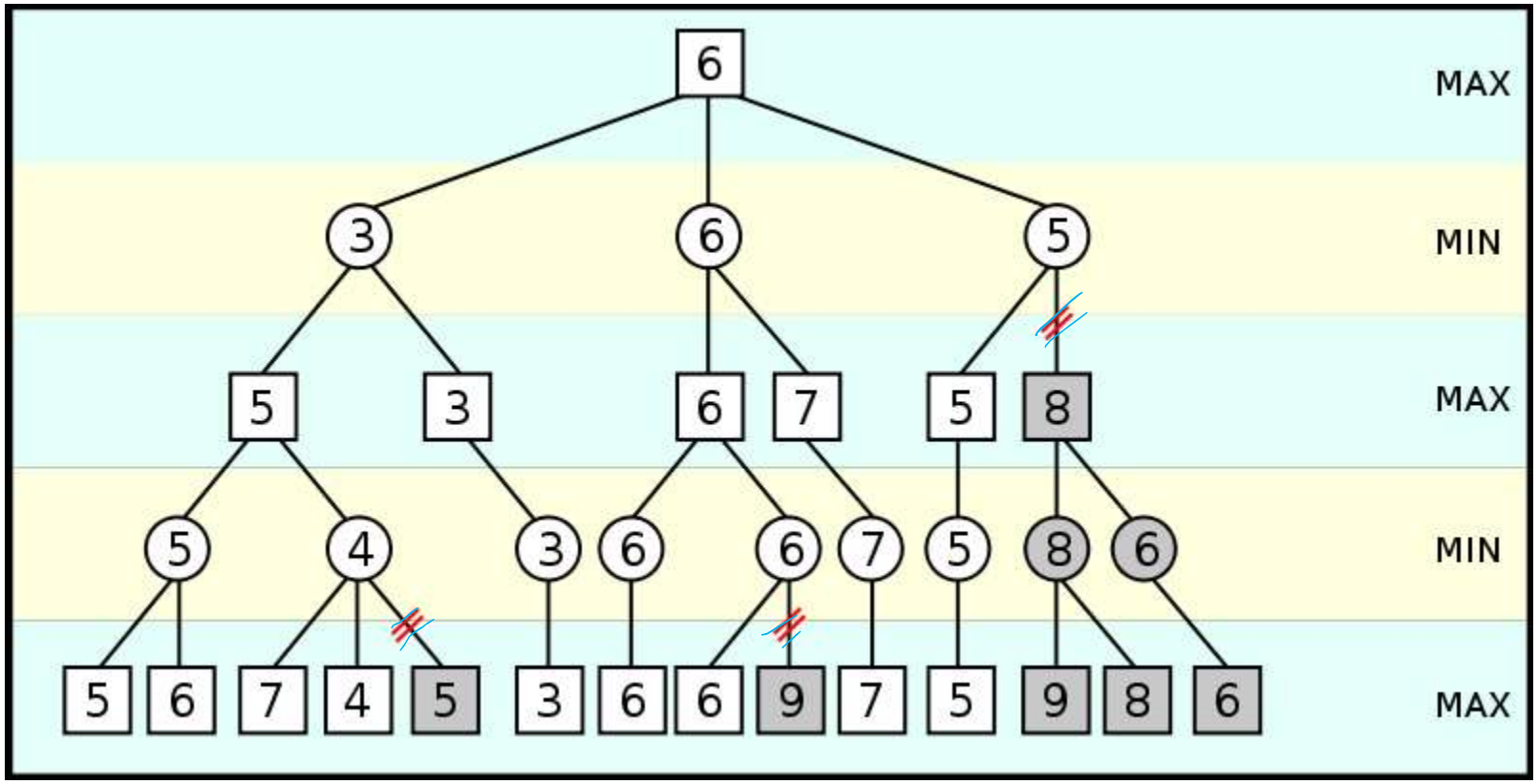
source: robotics.stanford.edu/~latombe/cs121/2003/home.htm

Alpha-Beta Pruning: Example 2



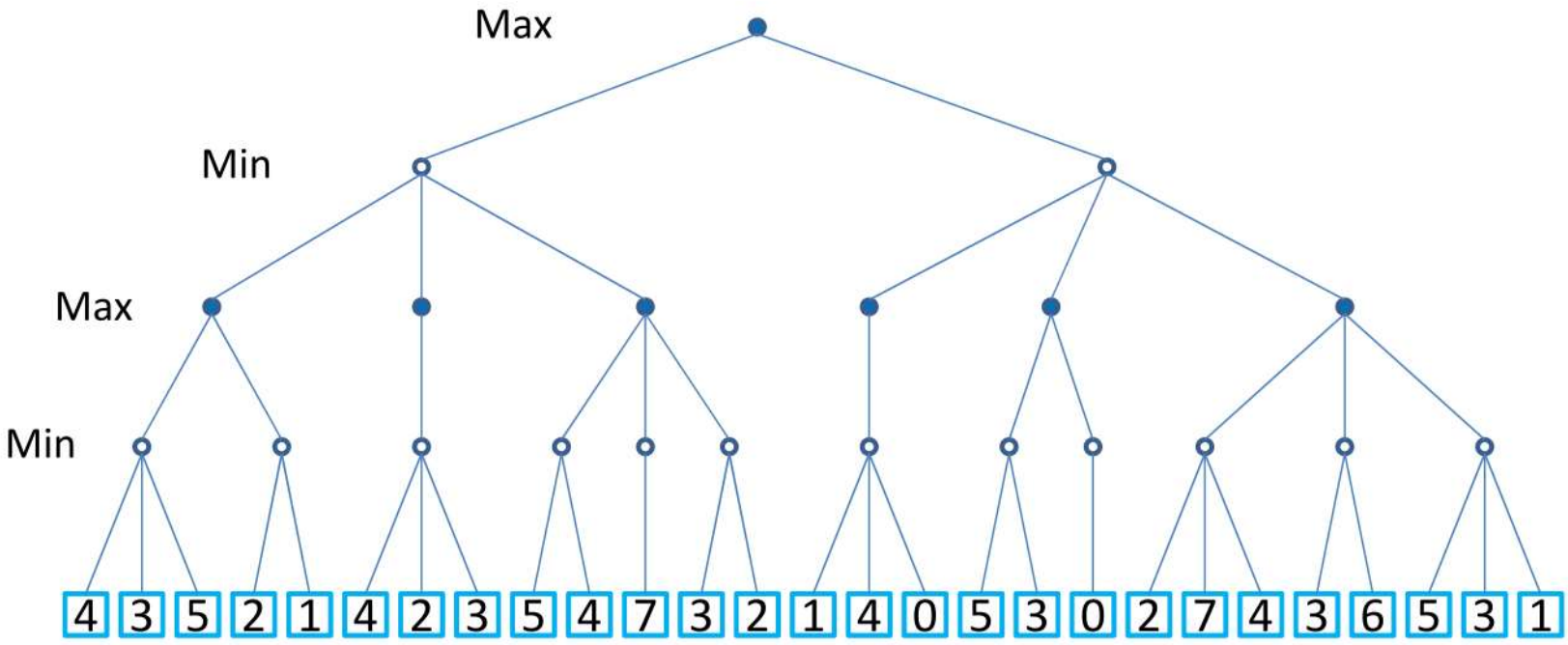
source: http://en.wikipedia.org/wiki/File:AB_pruning.svg

Alpha-Beta Pruning: Example 2

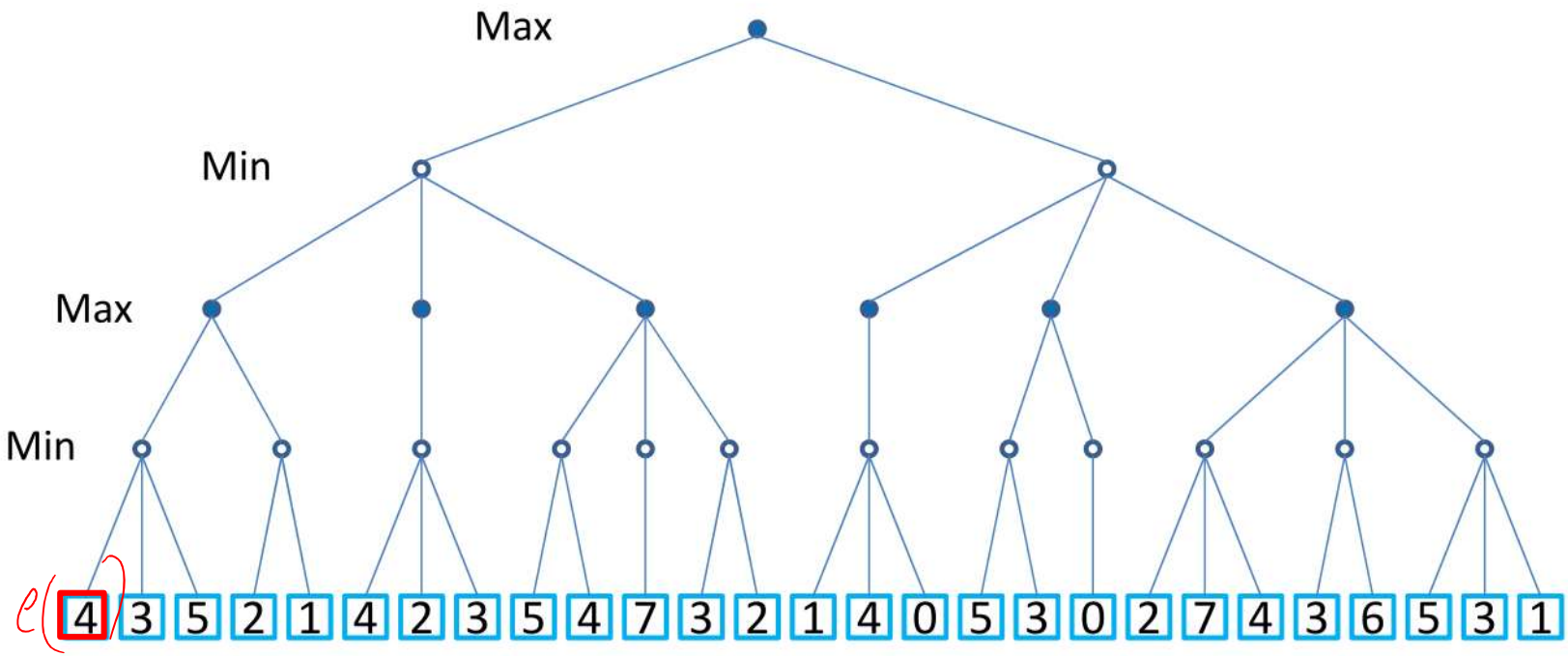


source: http://en.wikipedia.org/wiki/File:AB_pruning.svg

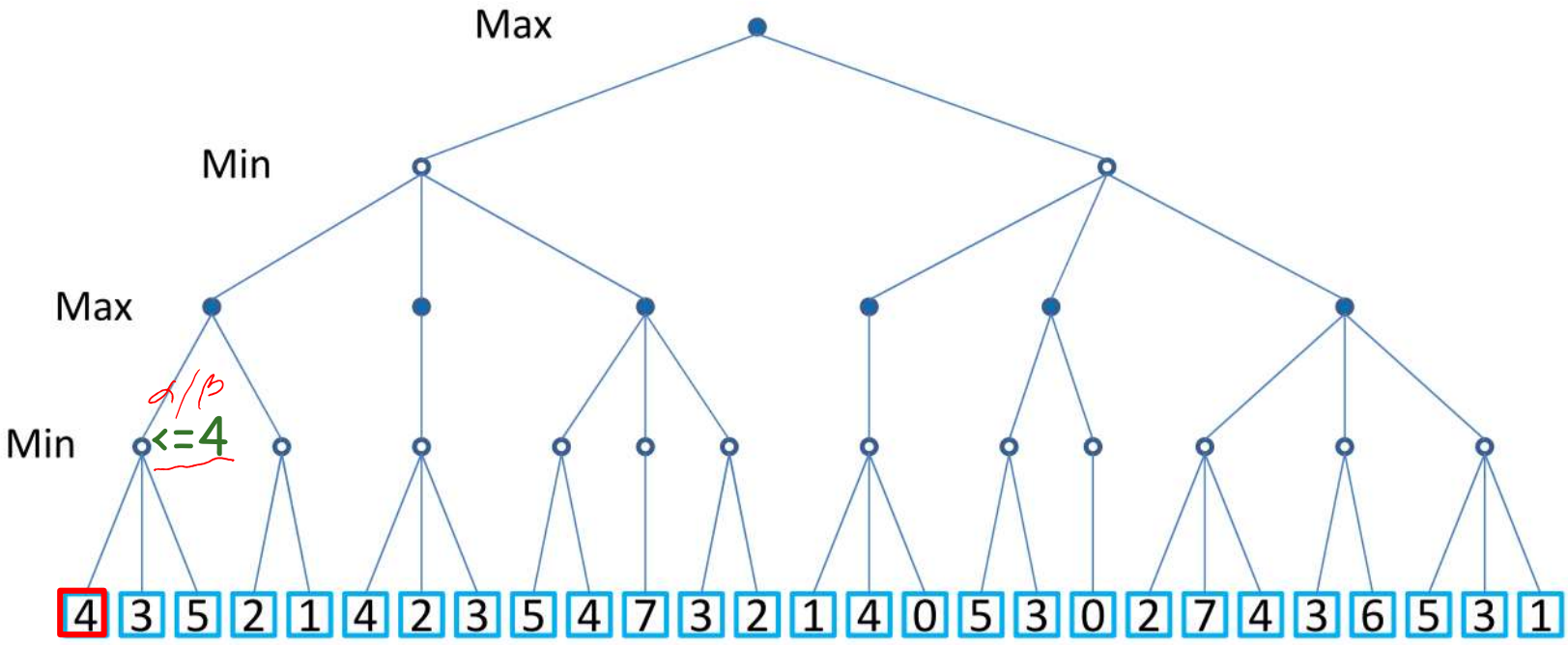
Alpha-Beta Pruning: Example 3



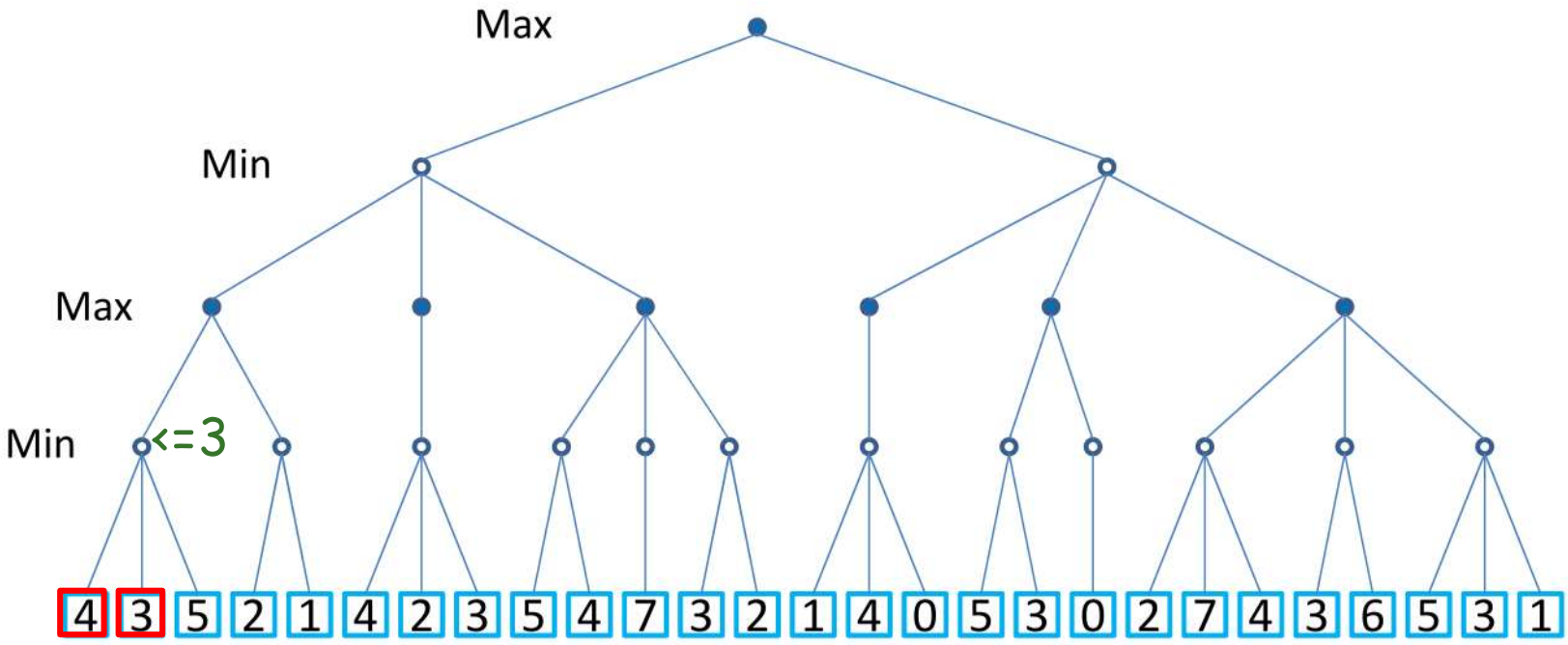
Alpha-Beta Pruning: Example 3



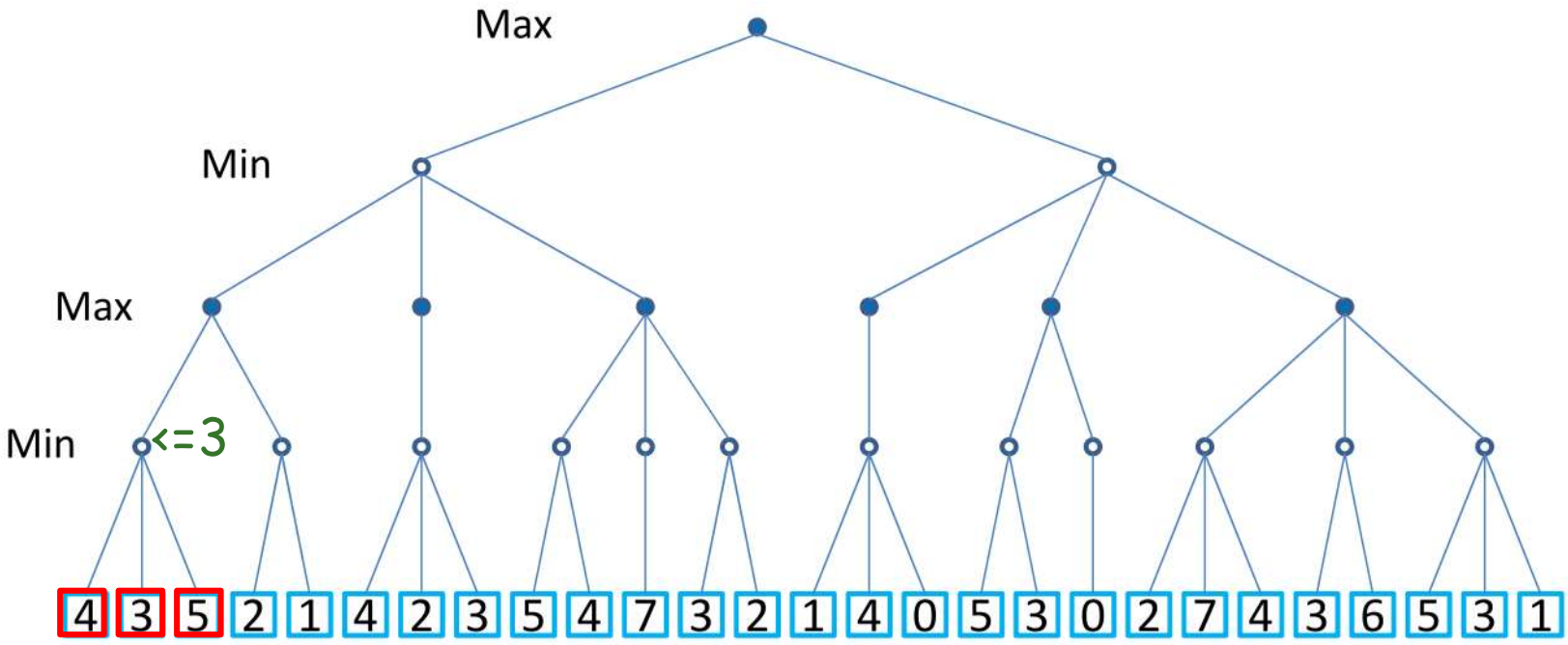
Alpha-Beta Pruning: Example 3



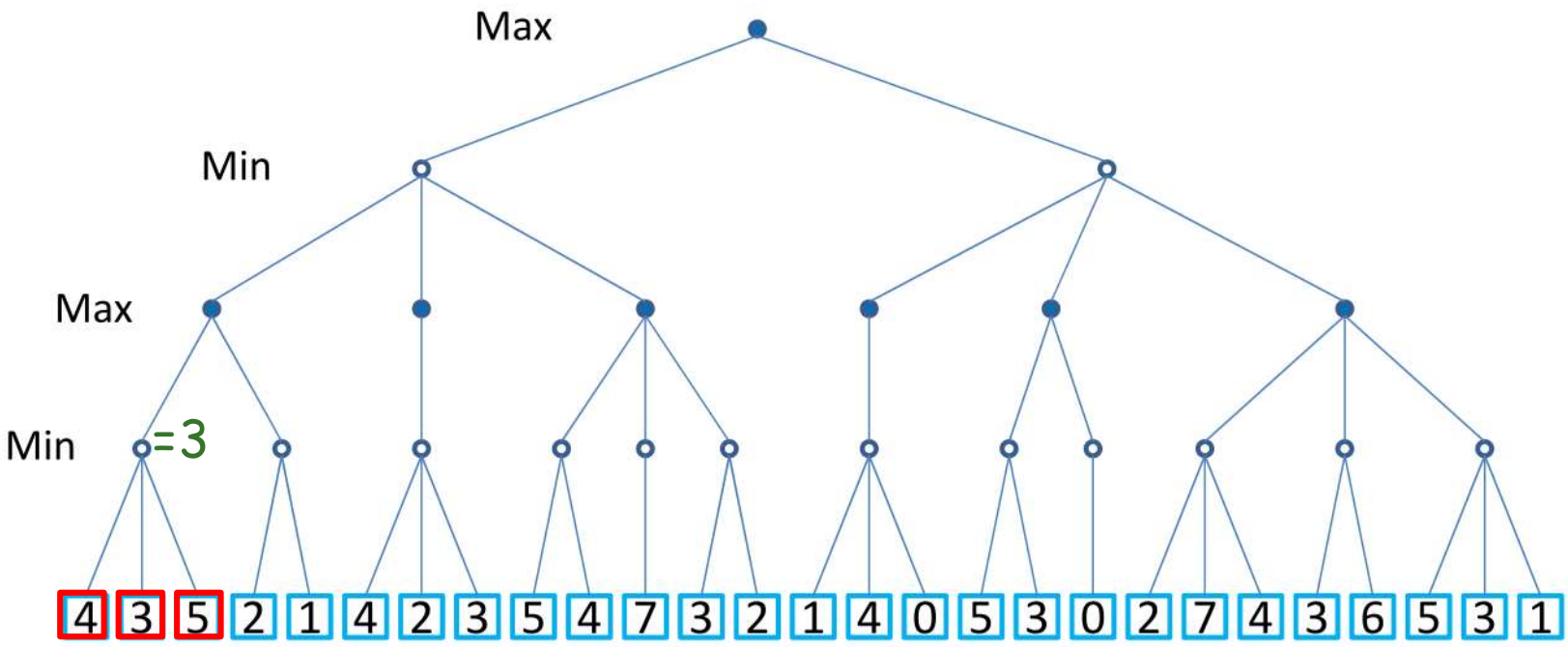
Alpha-Beta Pruning: Example 3



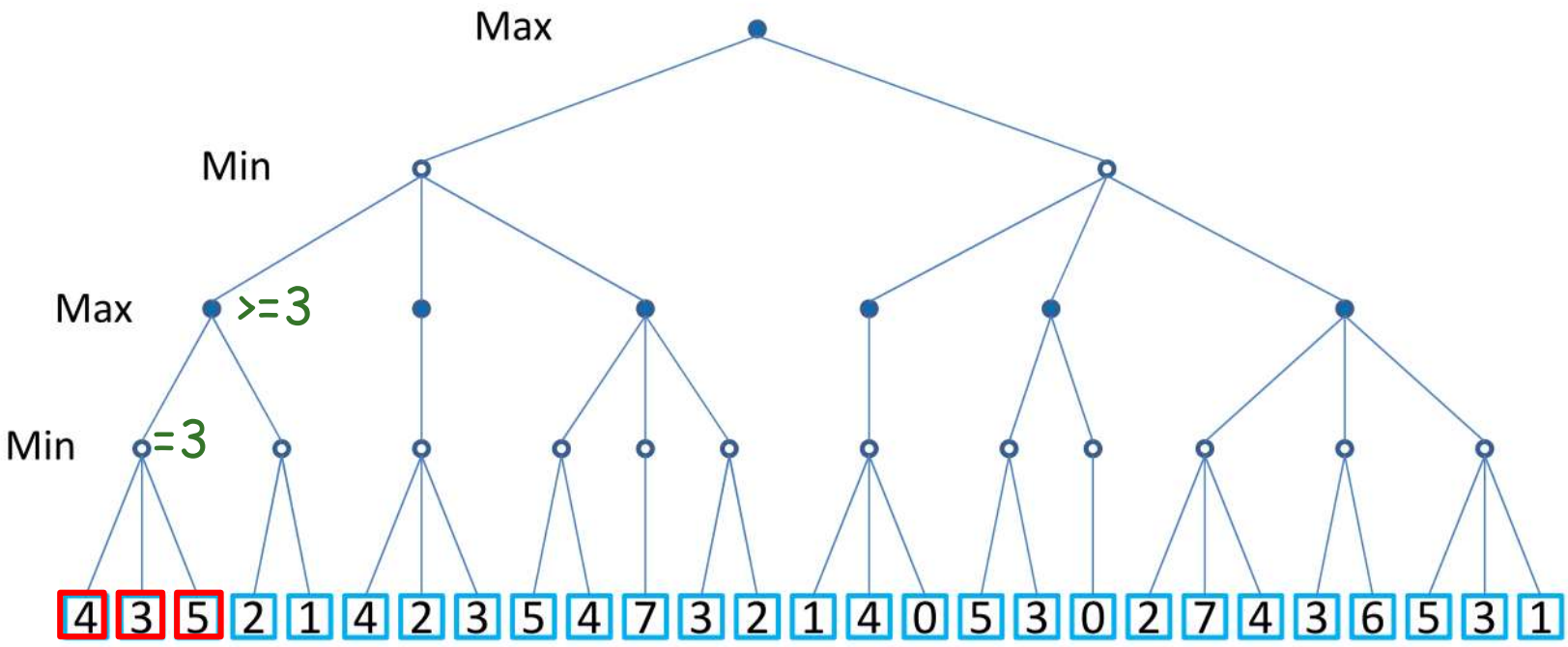
Alpha-Beta Pruning: Example 3



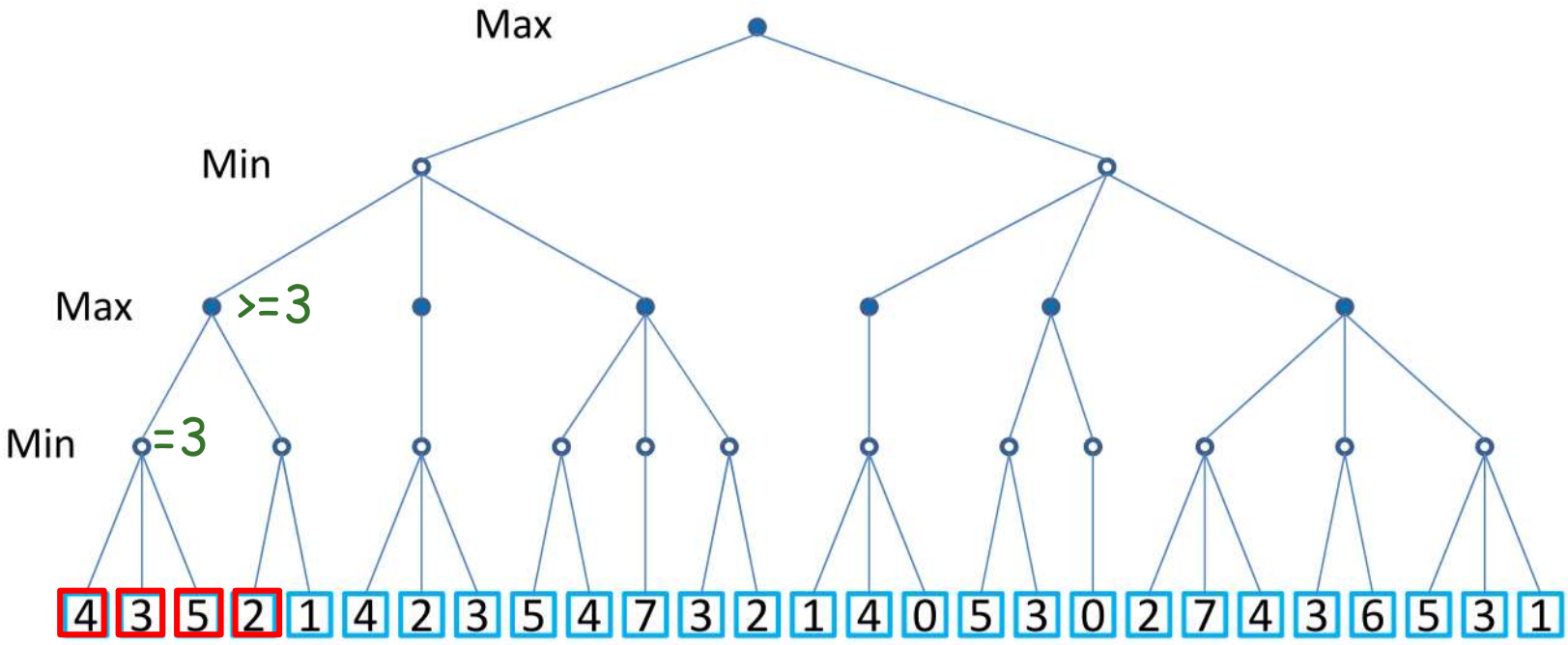
Alpha-Beta Pruning: Example 3



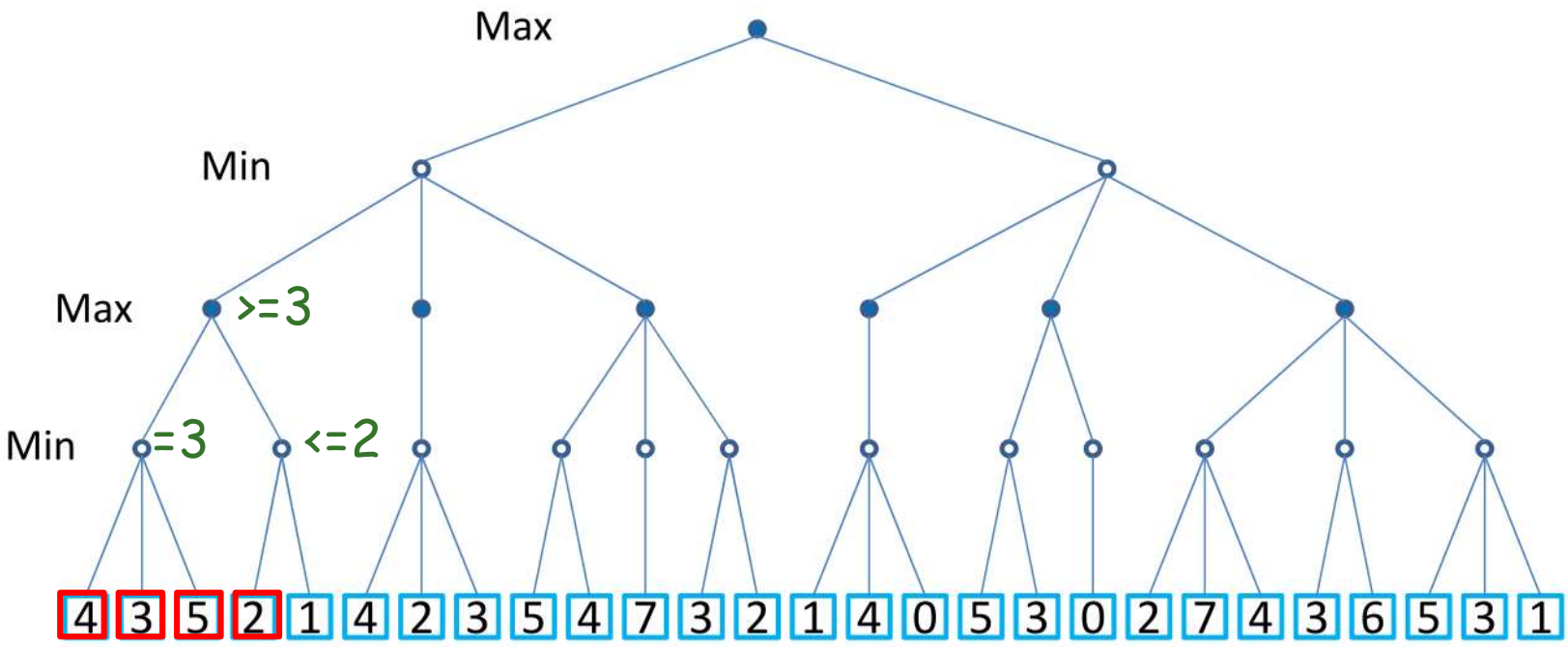
Alpha-Beta Pruning: Example 3



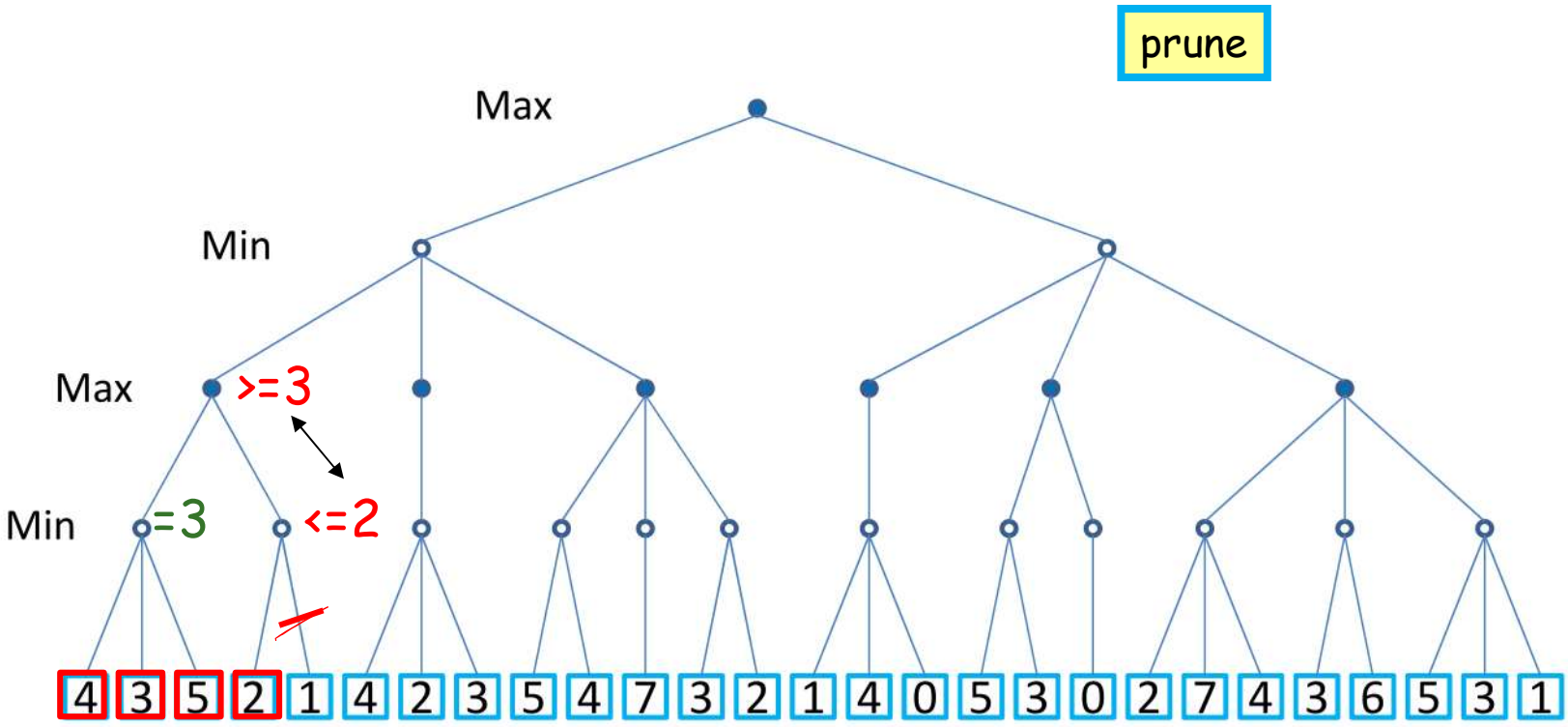
Alpha-Beta Pruning: Example 3



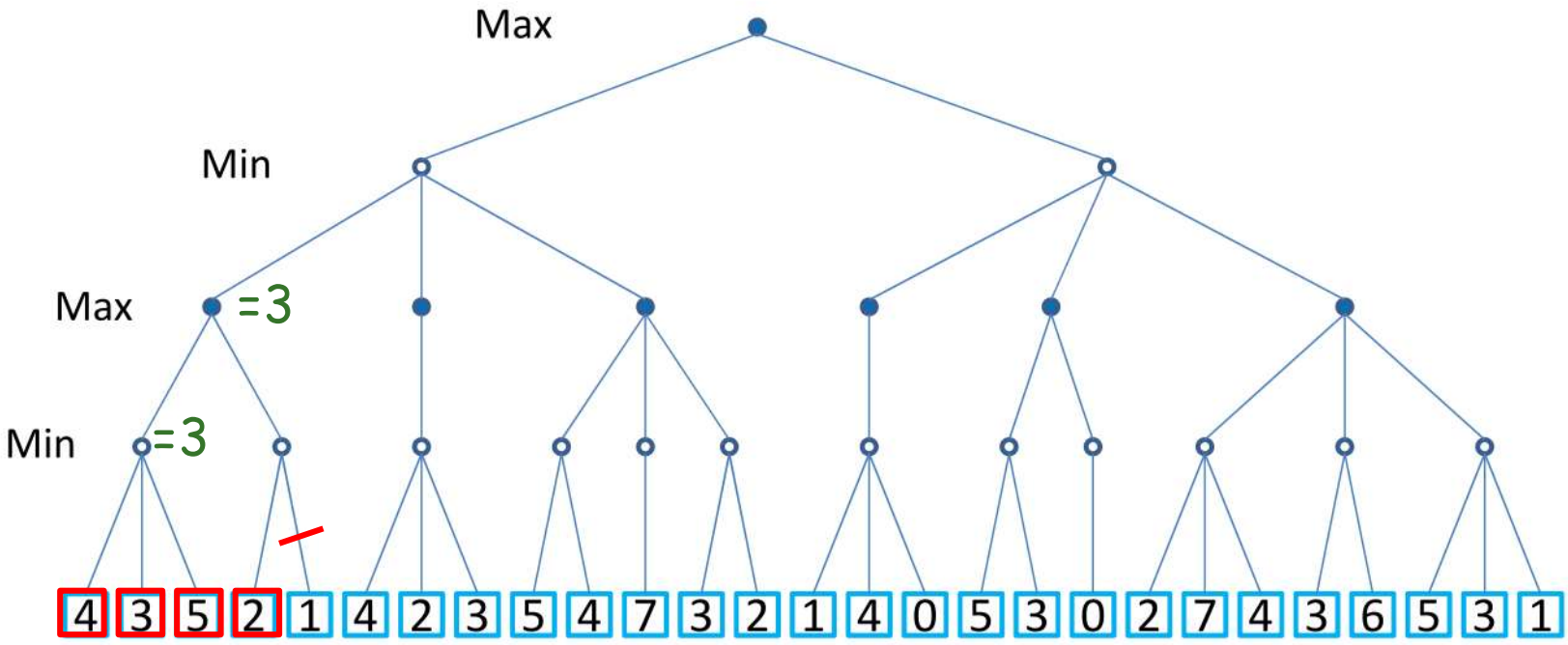
Alpha-Beta Pruning: Example 3



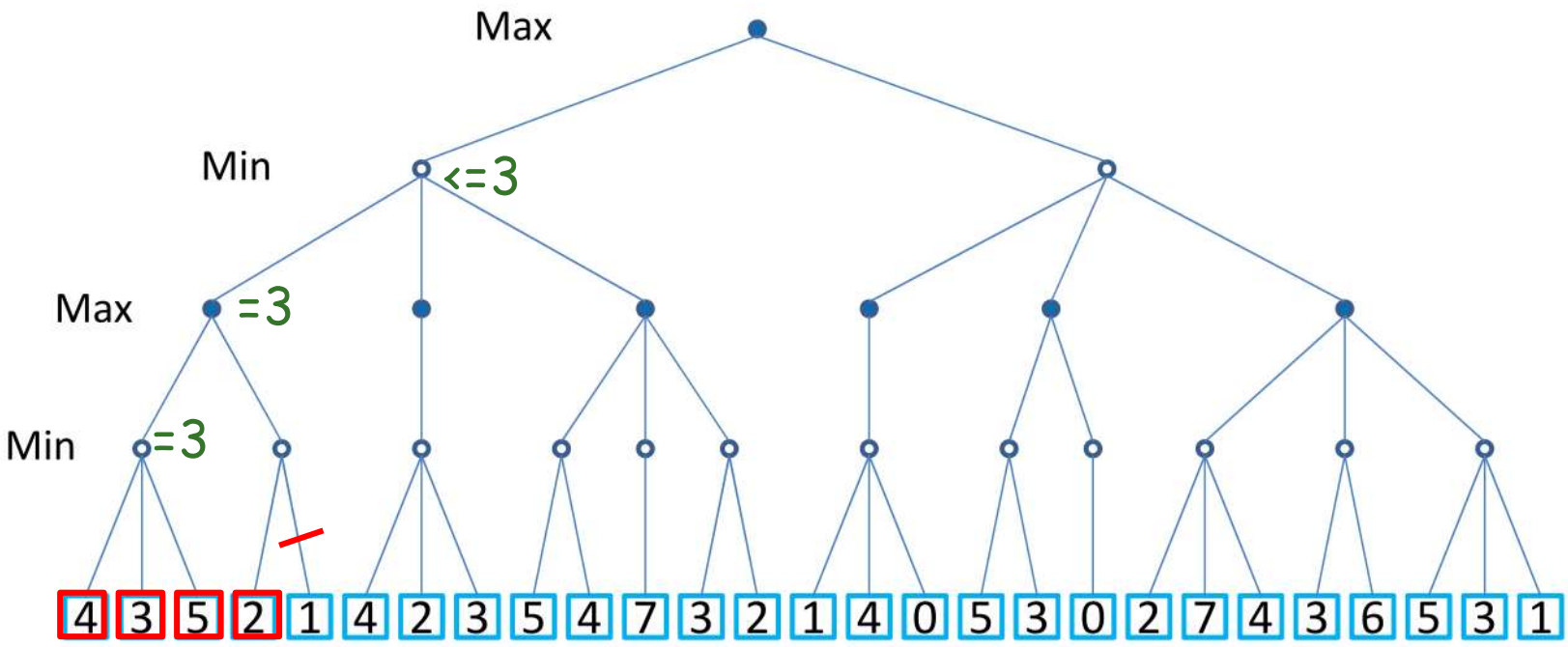
Alpha-Beta Pruning: Example 3



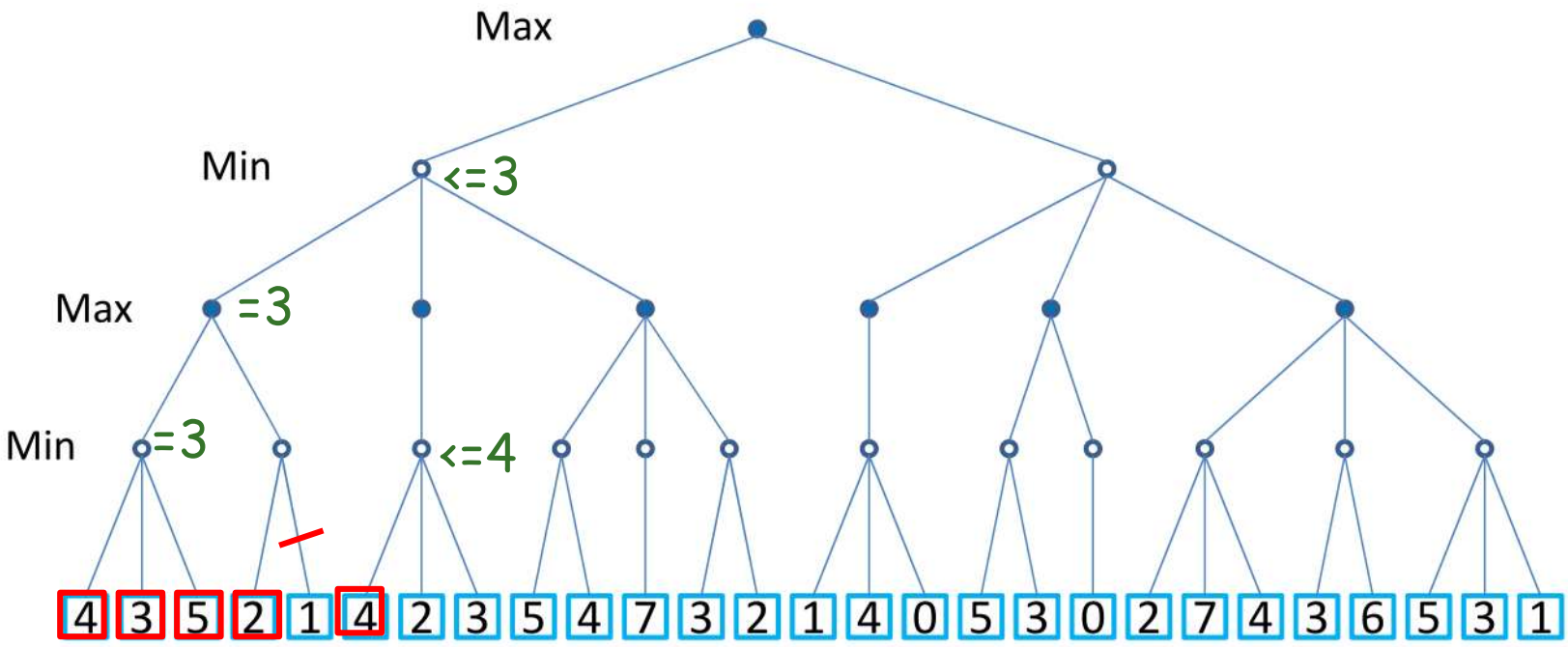
Alpha-Beta Pruning: Example 3



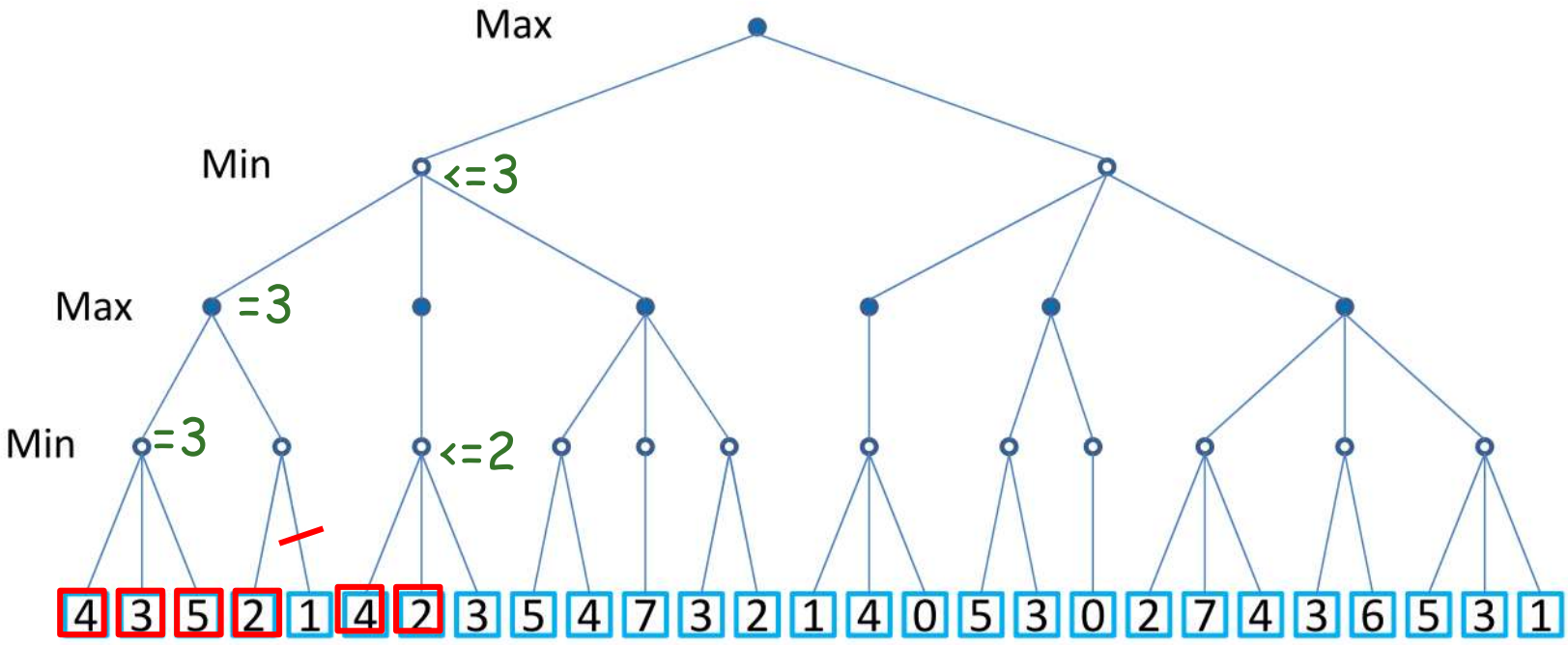
Alpha-Beta Pruning: Example 3



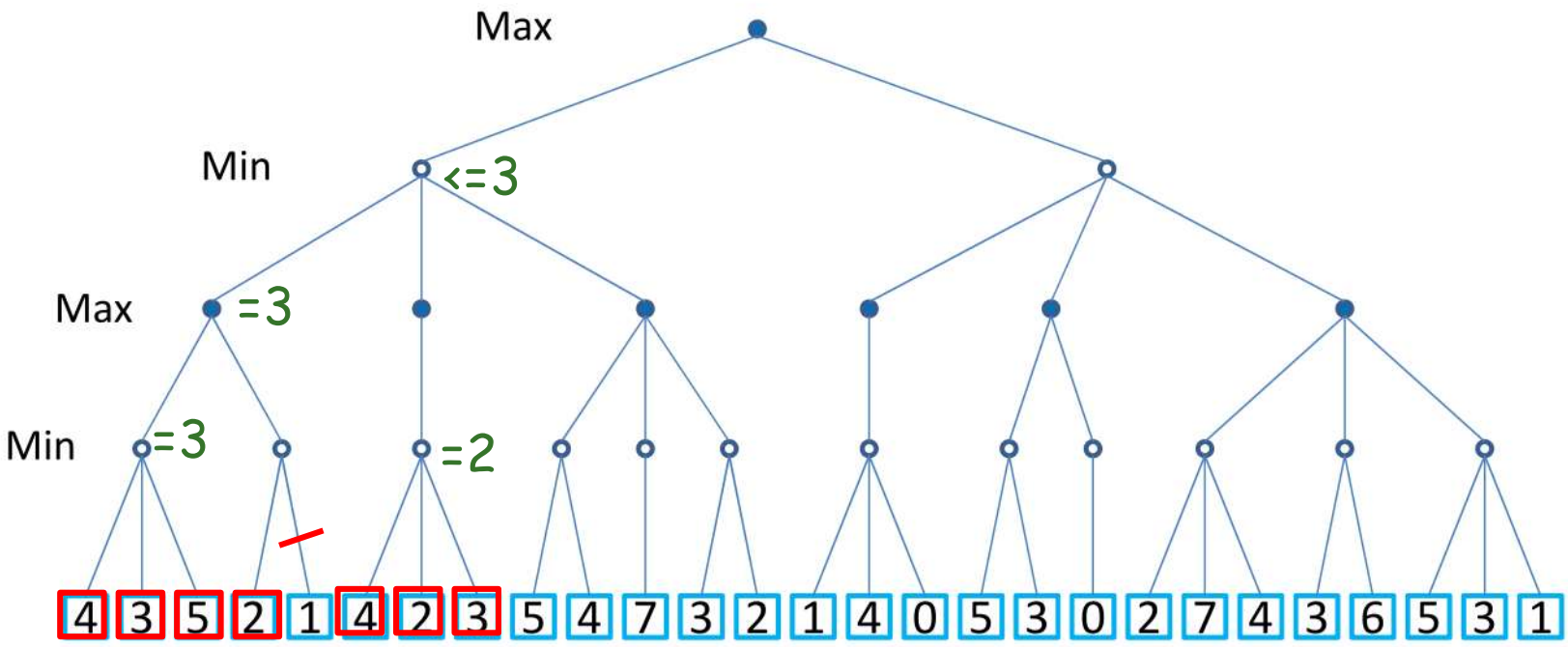
Alpha-Beta Pruning: Example 3



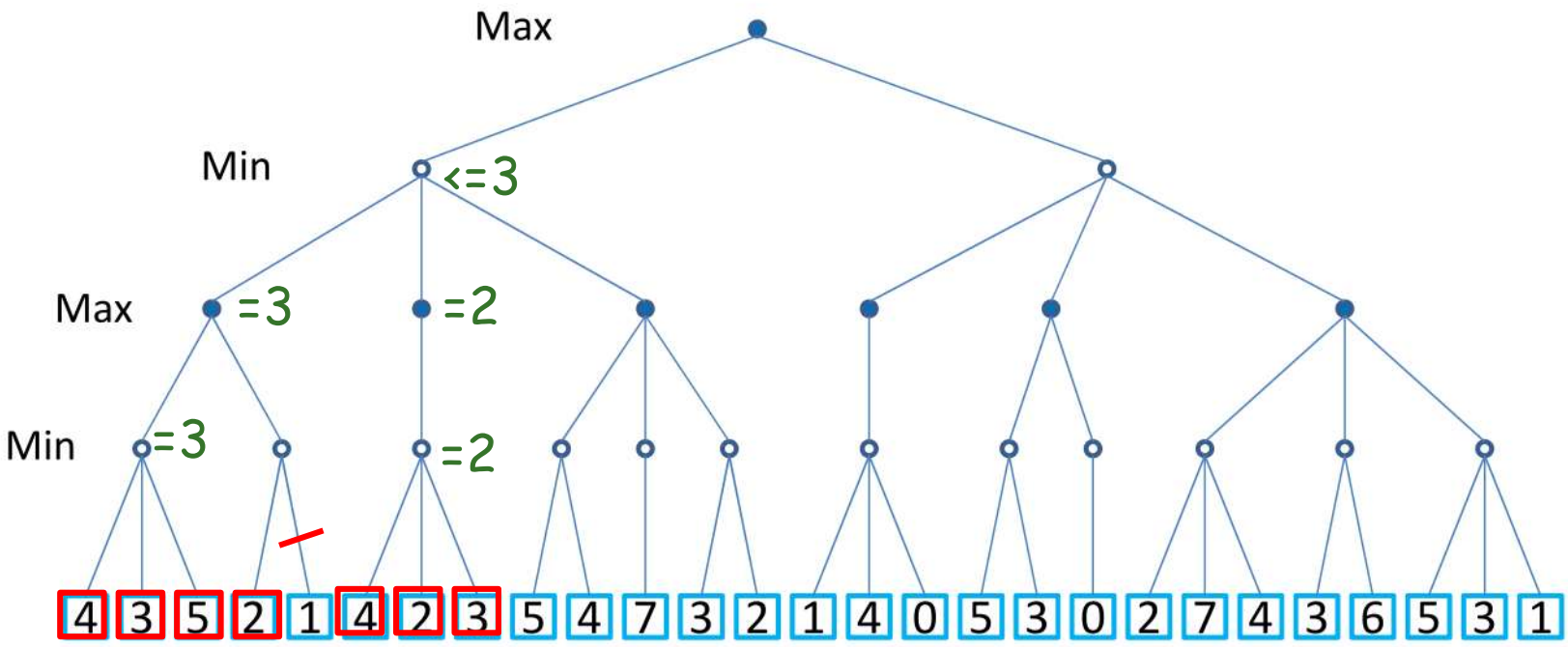
Alpha-Beta Pruning: Example 3



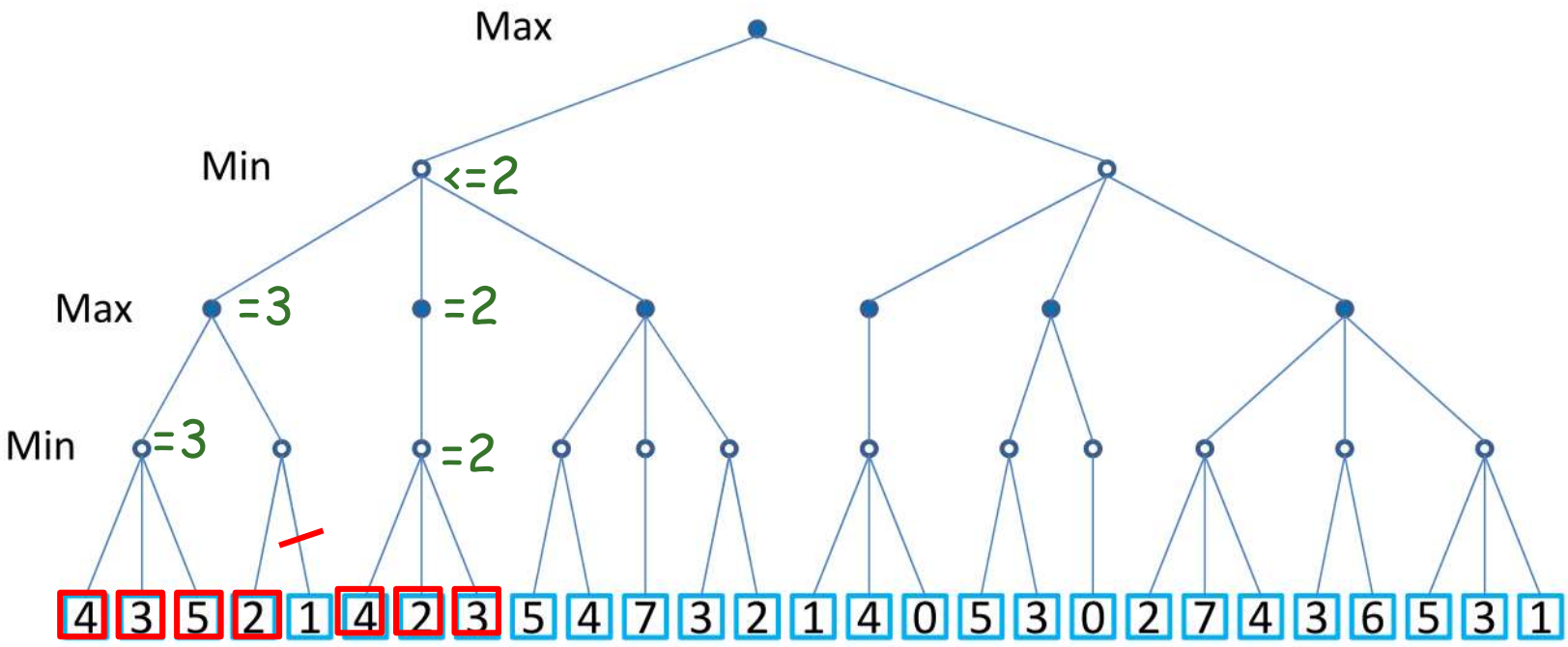
Alpha-Beta Pruning: Example 3



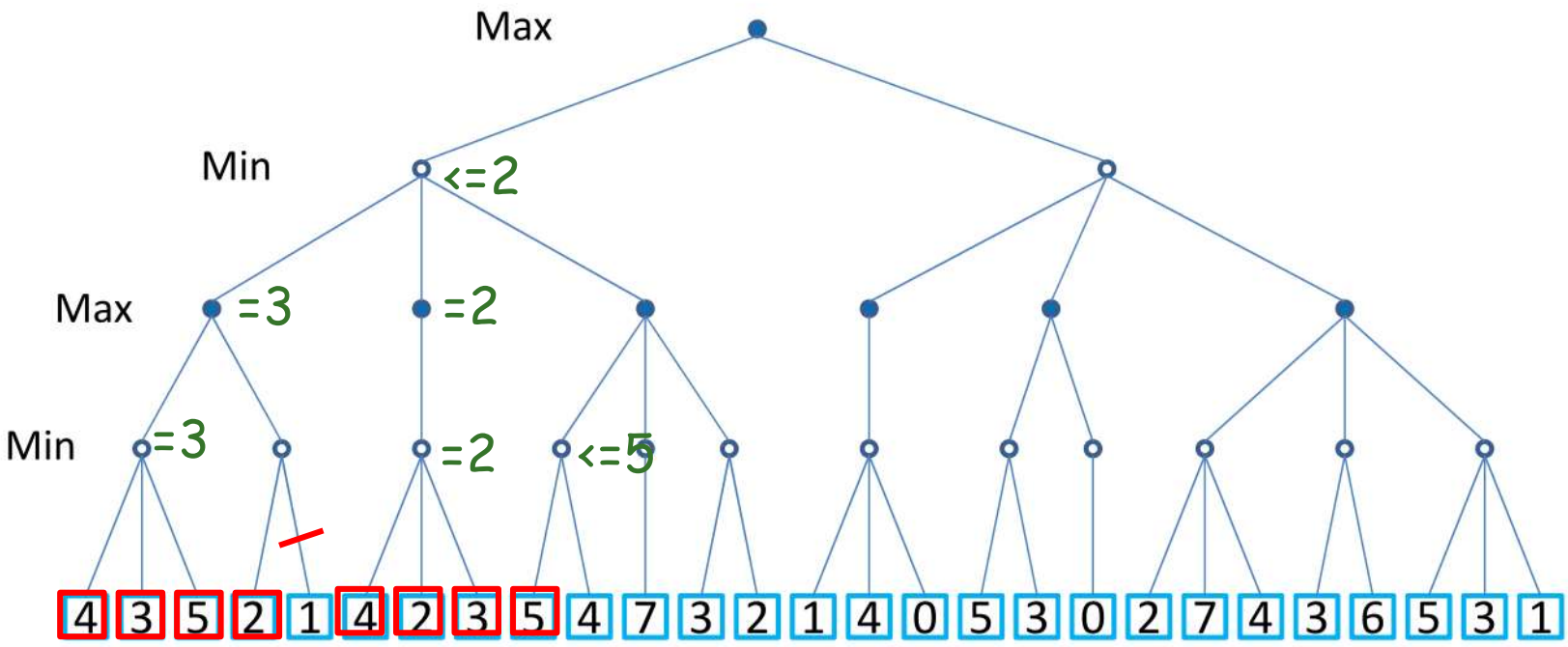
Alpha-Beta Pruning: Example 3



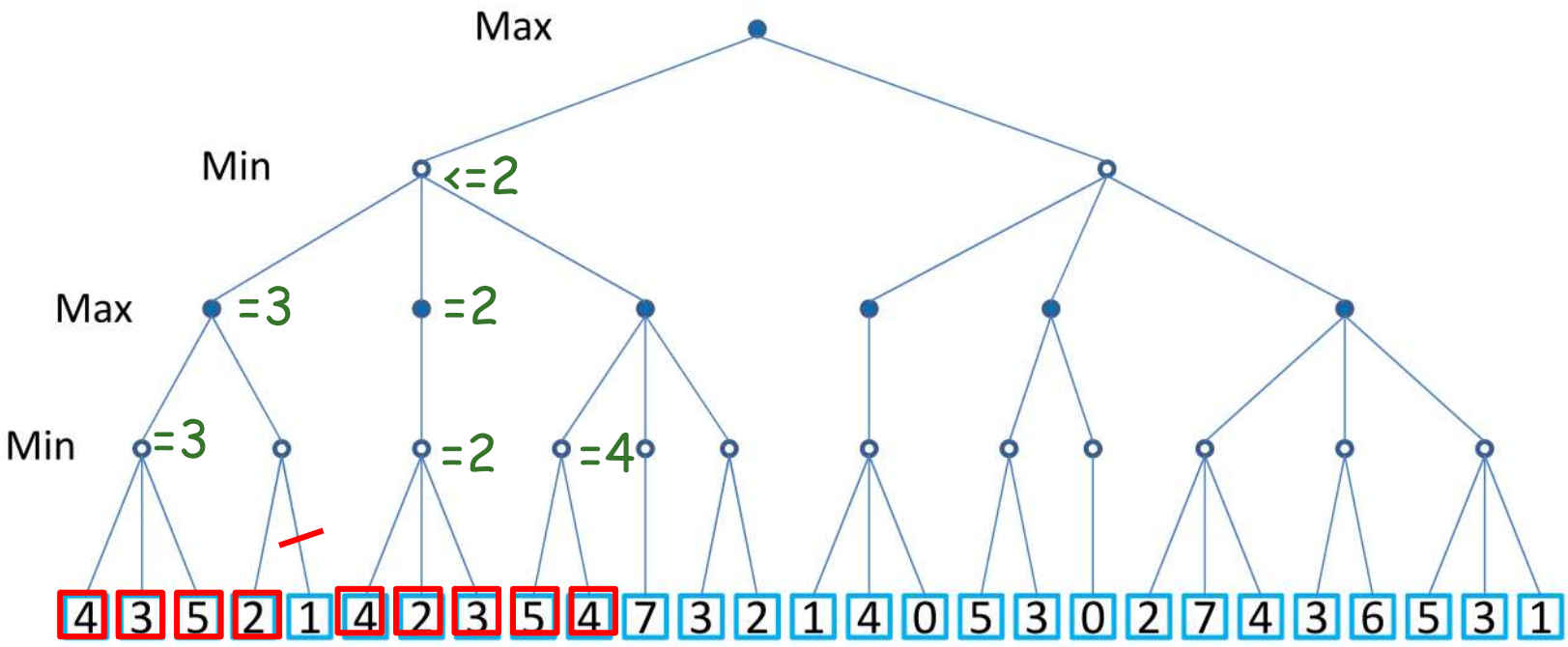
Alpha-Beta Pruning: Example 3



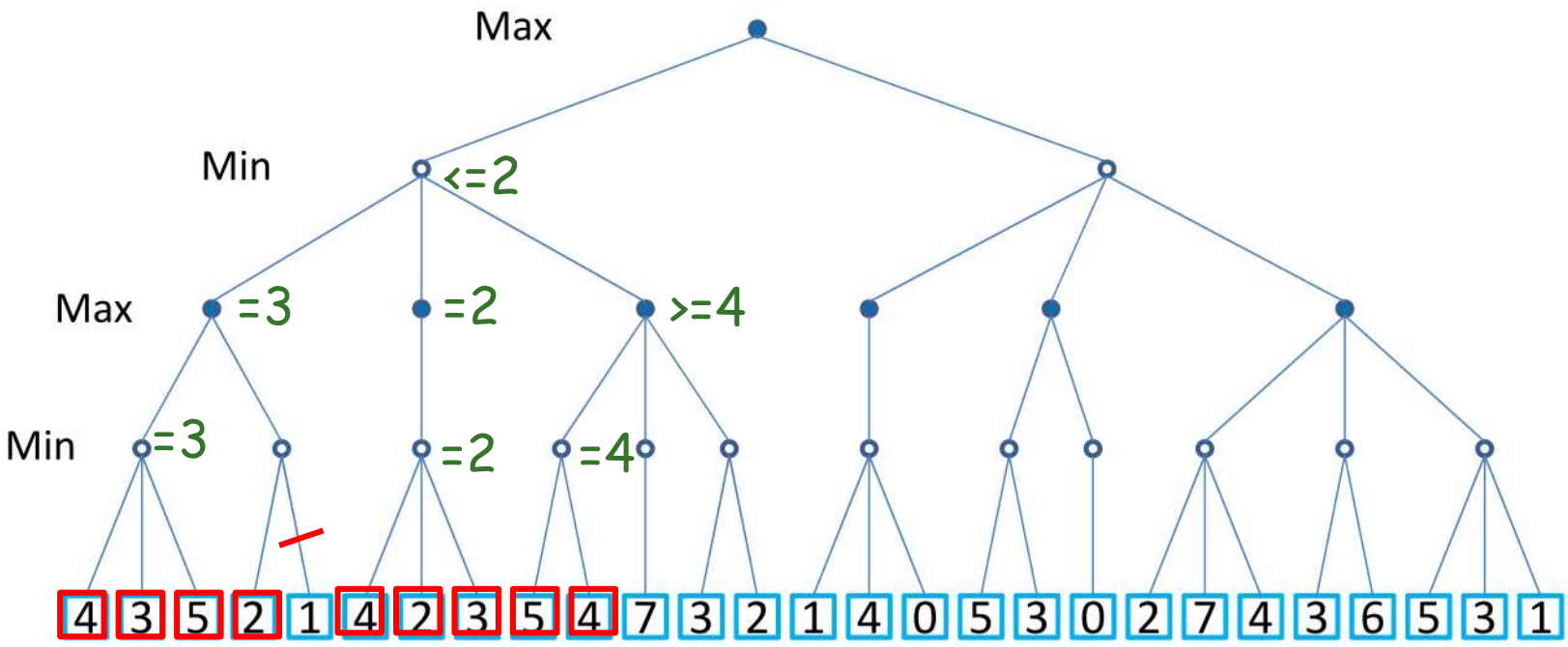
Alpha-Beta Pruning: Example 3



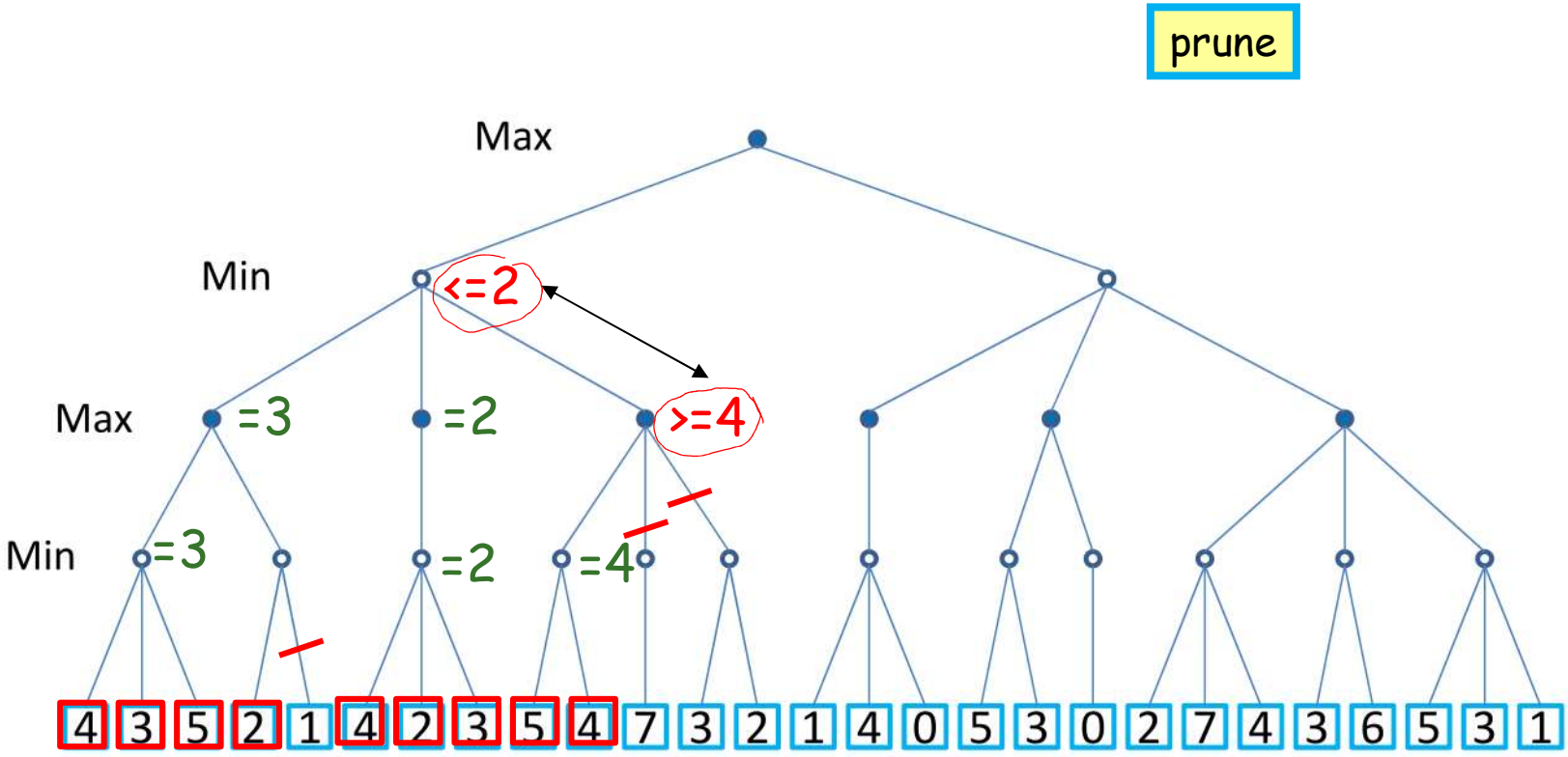
Alpha-Beta Pruning: Example 3



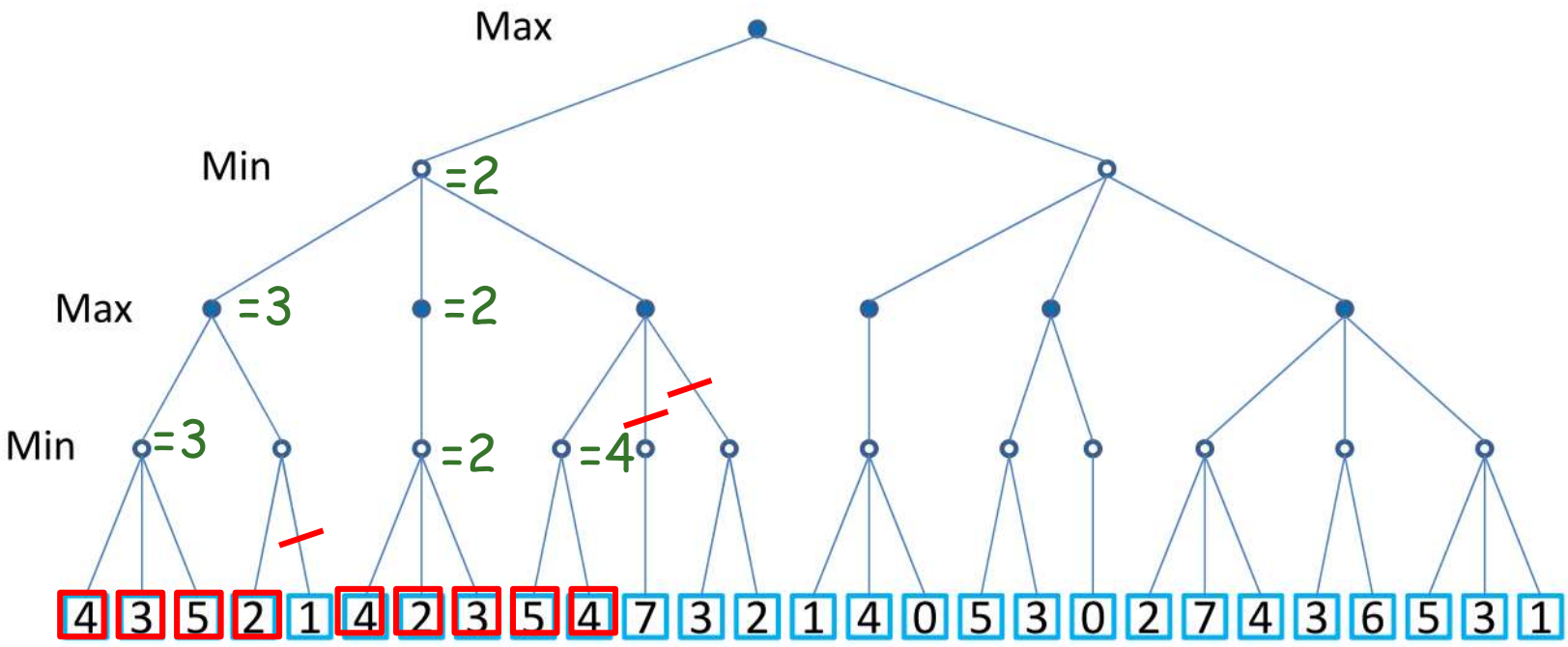
Alpha-Beta Pruning: Example 3



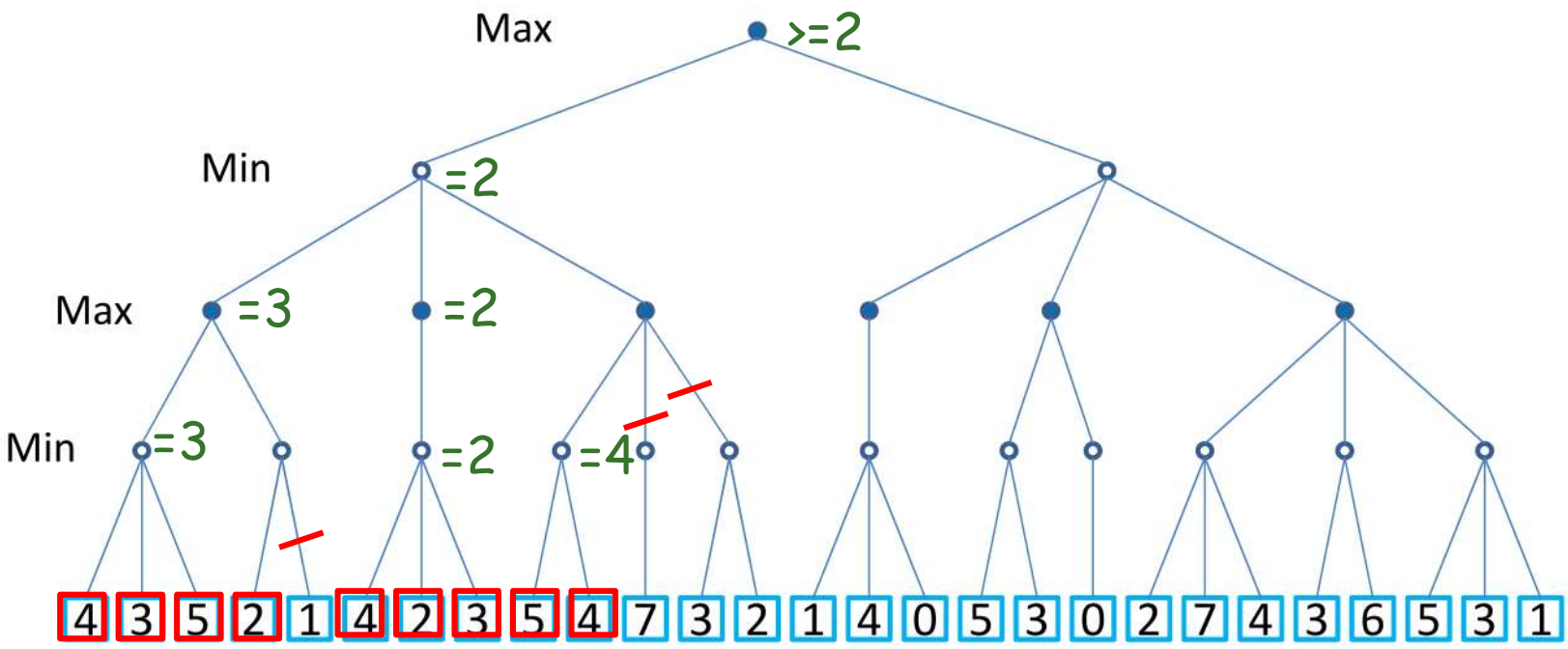
Alpha-Beta Pruning: Example 3



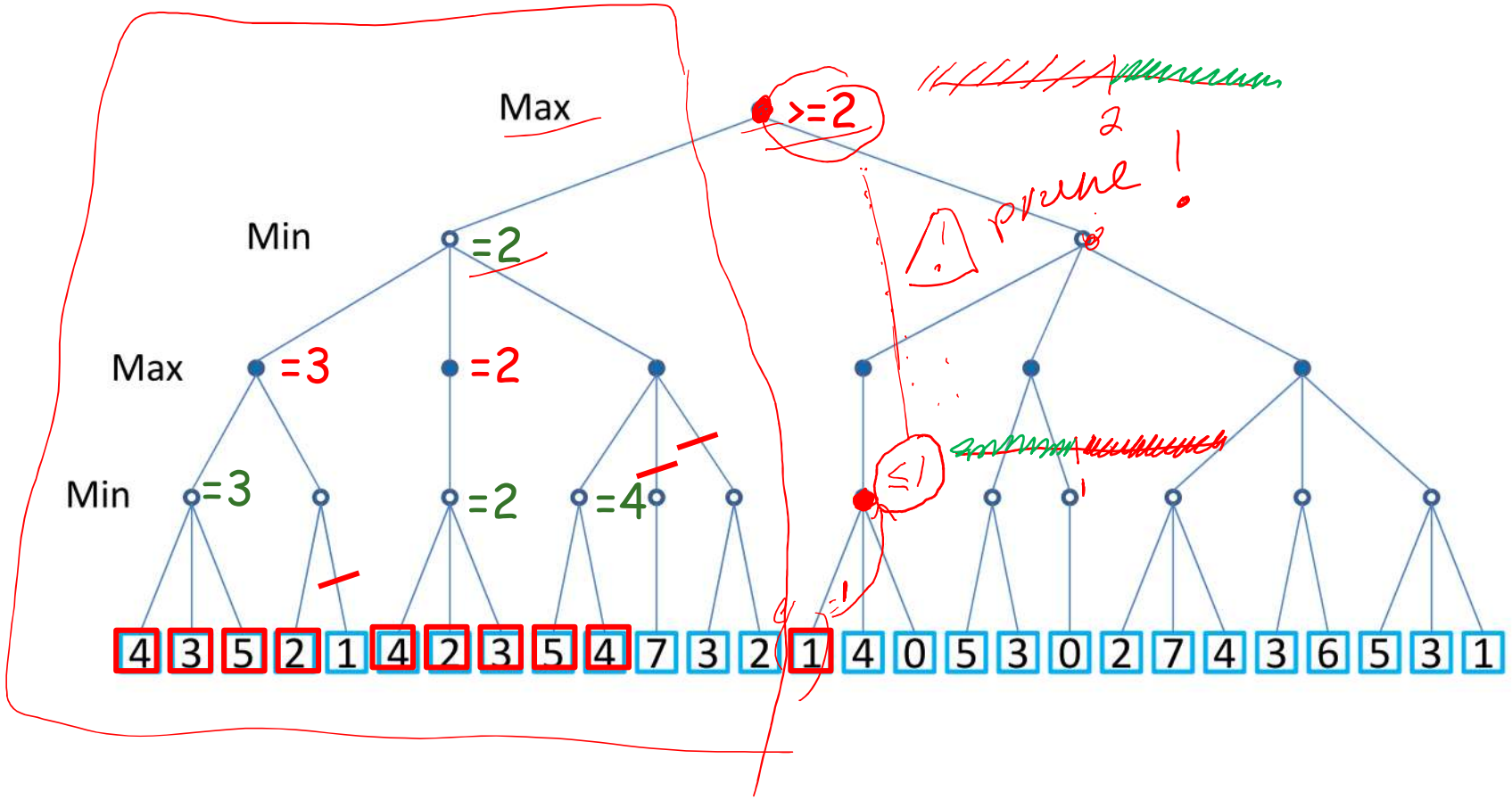
Alpha-Beta Pruning: Example 3



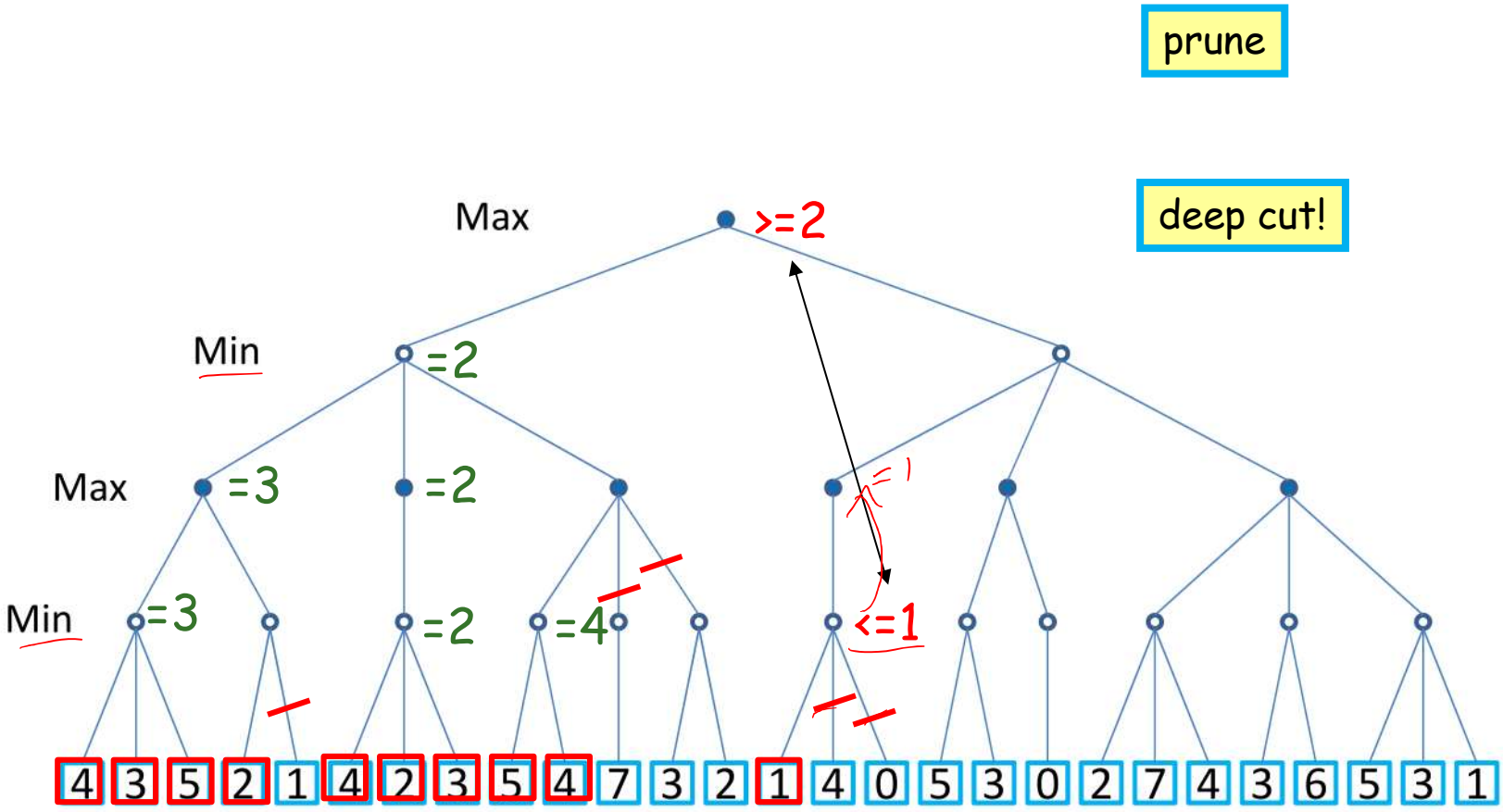
Alpha-Beta Pruning: Example 3



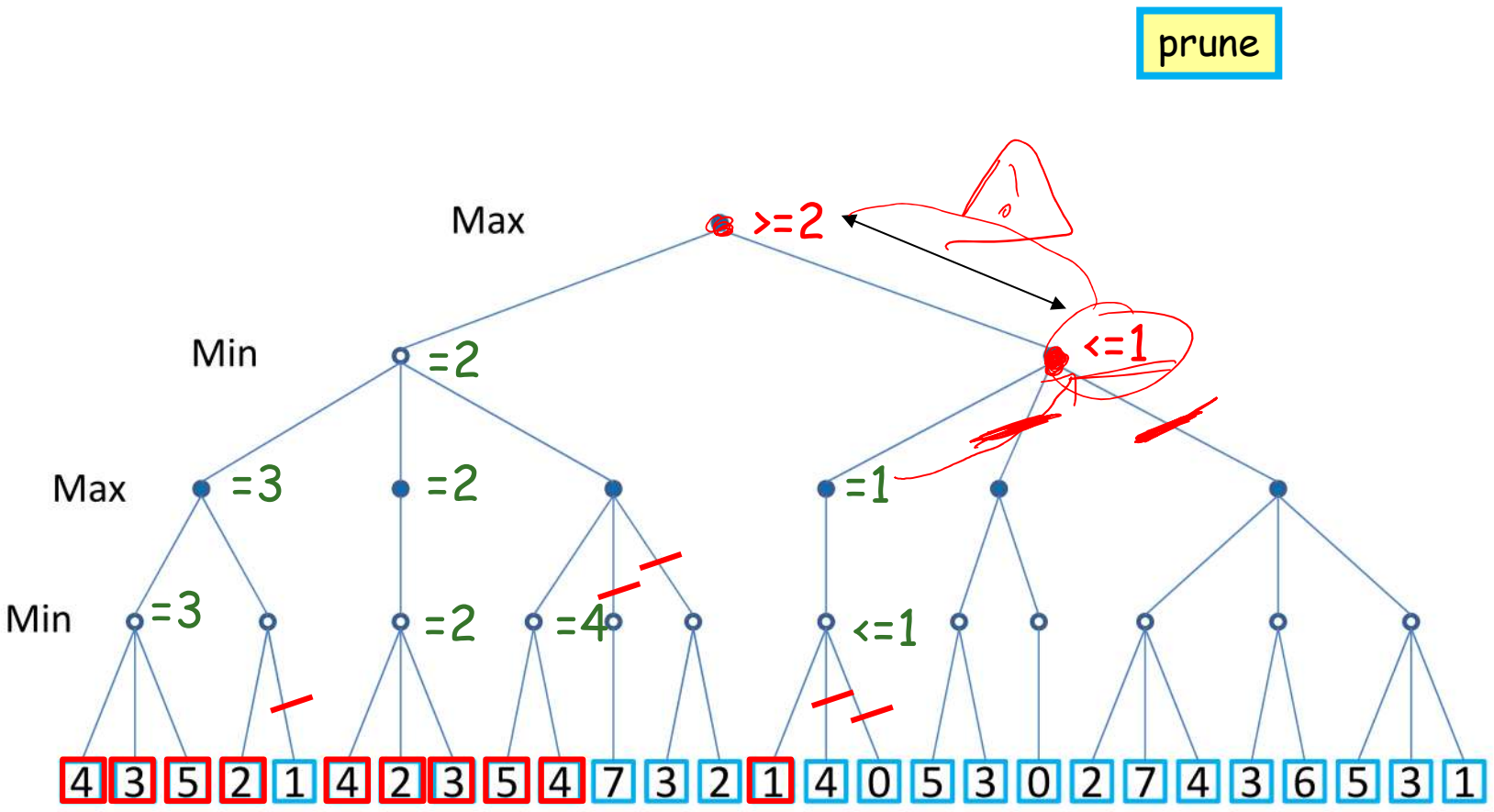
Alpha-Beta Pruning: Example 3



Alpha-Beta Pruning: Example 3



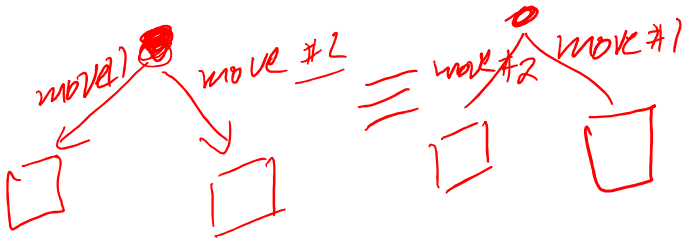
Alpha-Beta Pruning: Example 3



10 nodes explored out of 27

Efficiency of Alpha-Beta Pruning

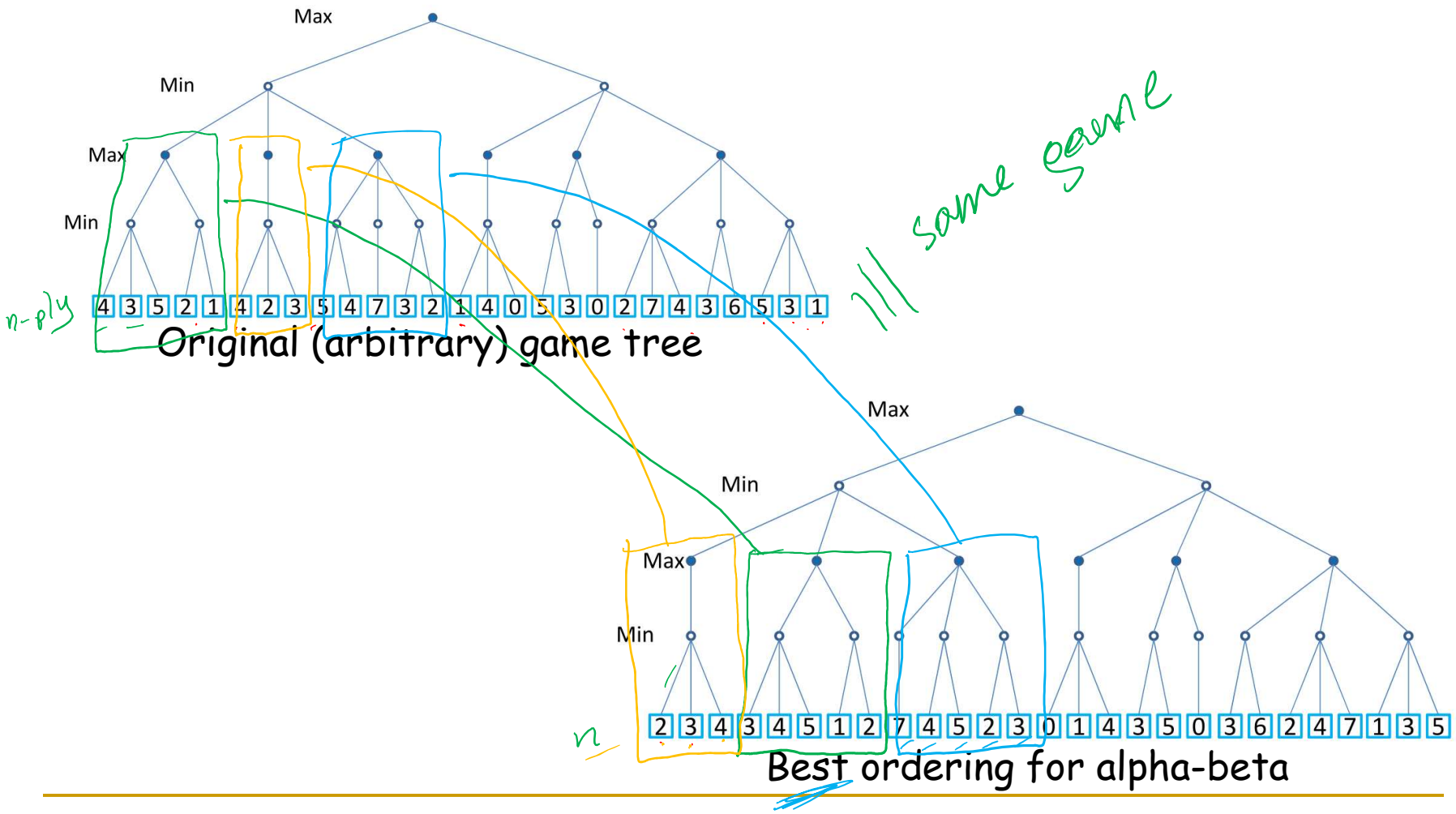
- Depends on the order the siblings
  - which is an arbitrary choice ;-(
- In worst case:
  - alpha-beta provides no pruning
  - plus extra overhead cost ;-(
- In best case:
  - branching factor is reduced to its square root



$36 \approx (n)$

$\Rightarrow 6 \approx (n)$

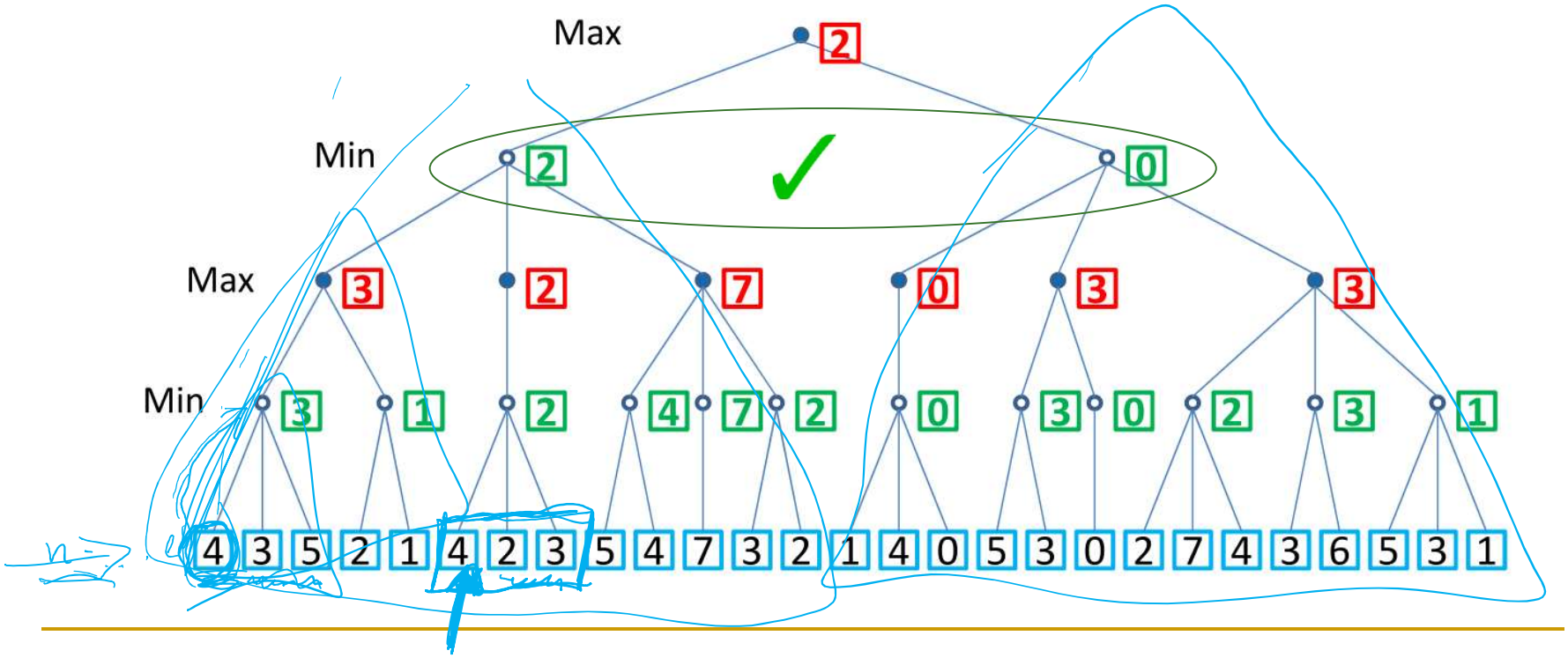
Alpha-Beta: Best ordering



Alpha-Beta: Best ordering

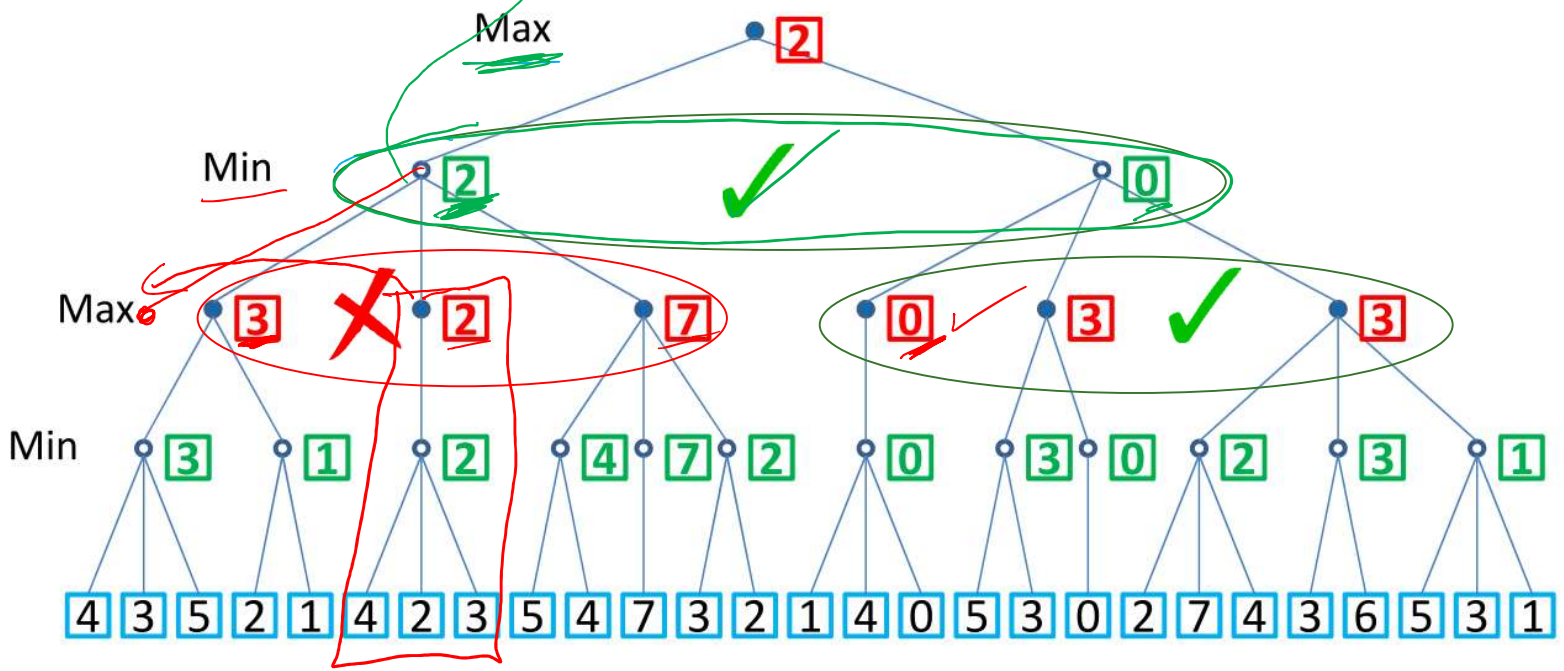
- best ordering:
  - strongest constraint placed first, ie:

1. children of MIN : smallest node first
2. children of MAX: largest node first



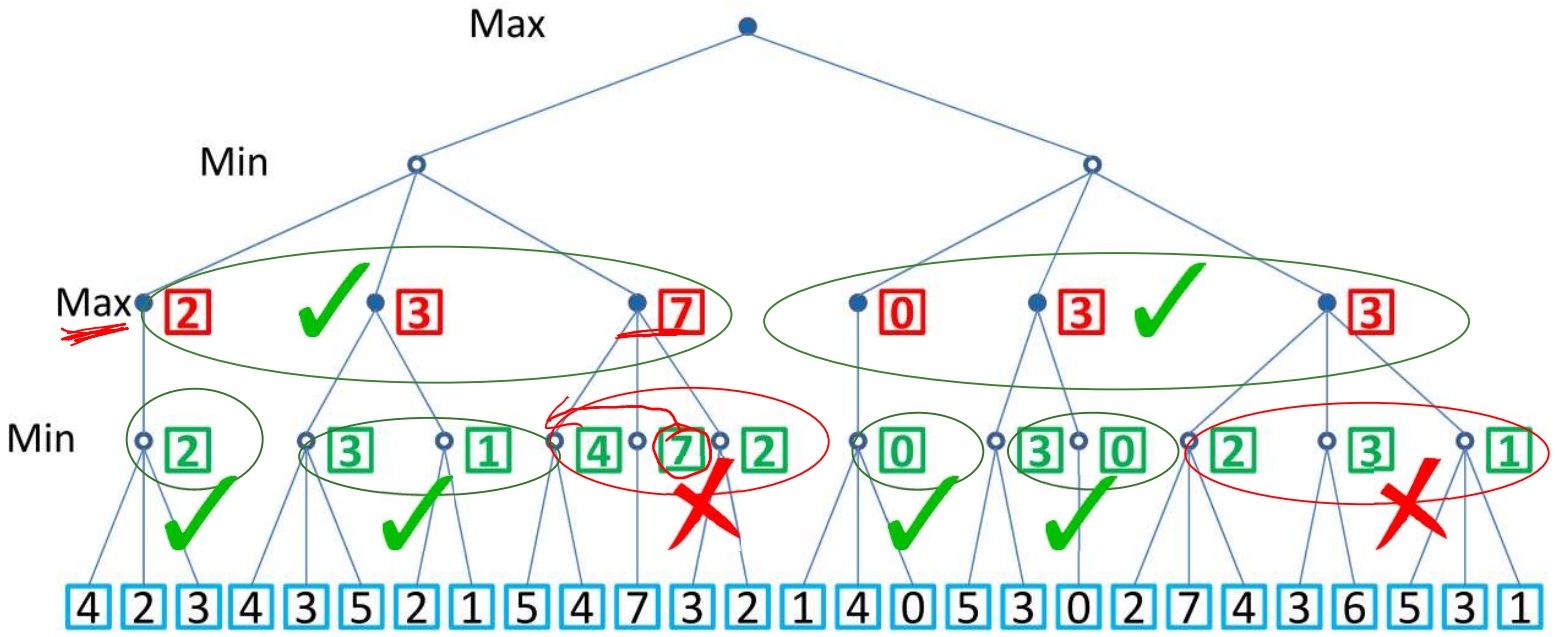
Alpha-Beta: Best ordering

- best ordering:
  1. children of MIN : smallest node first
  2. children of MAX: largest node first

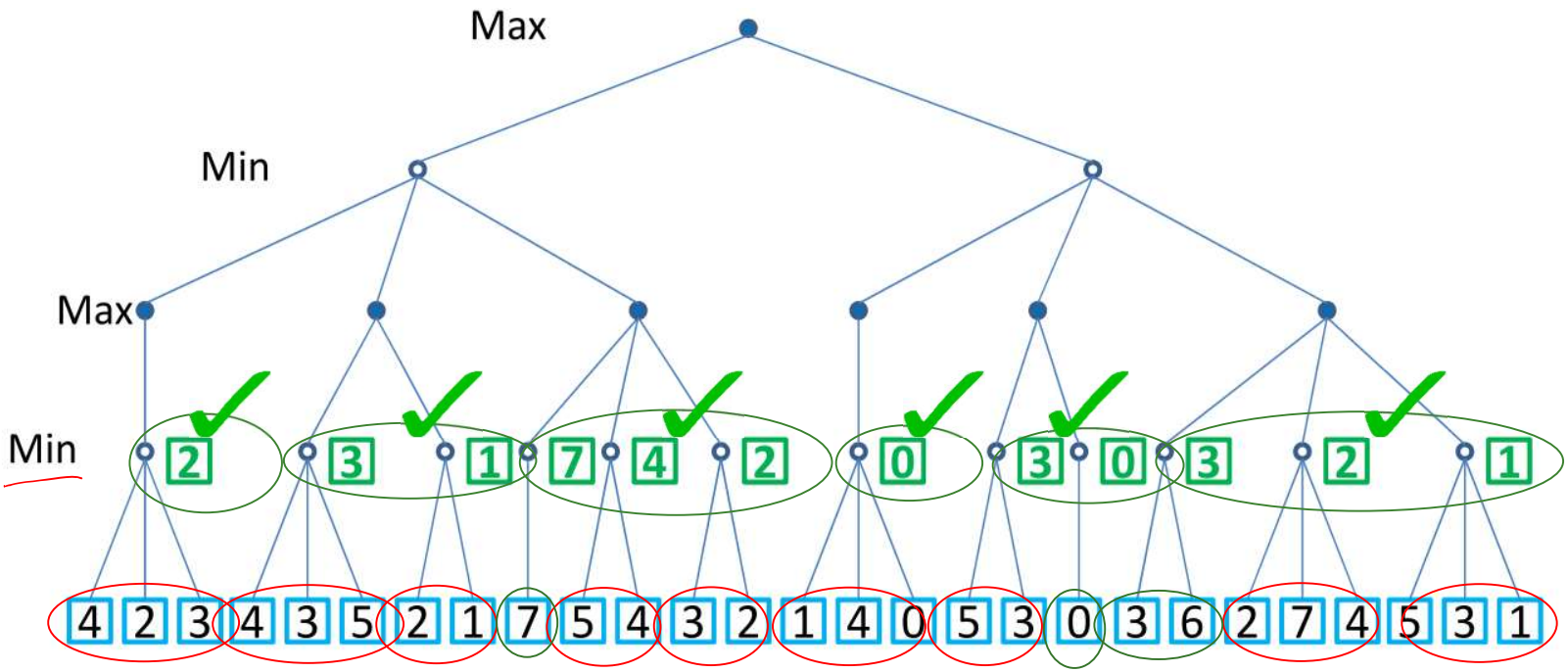


Alpha-Beta: Best ordering

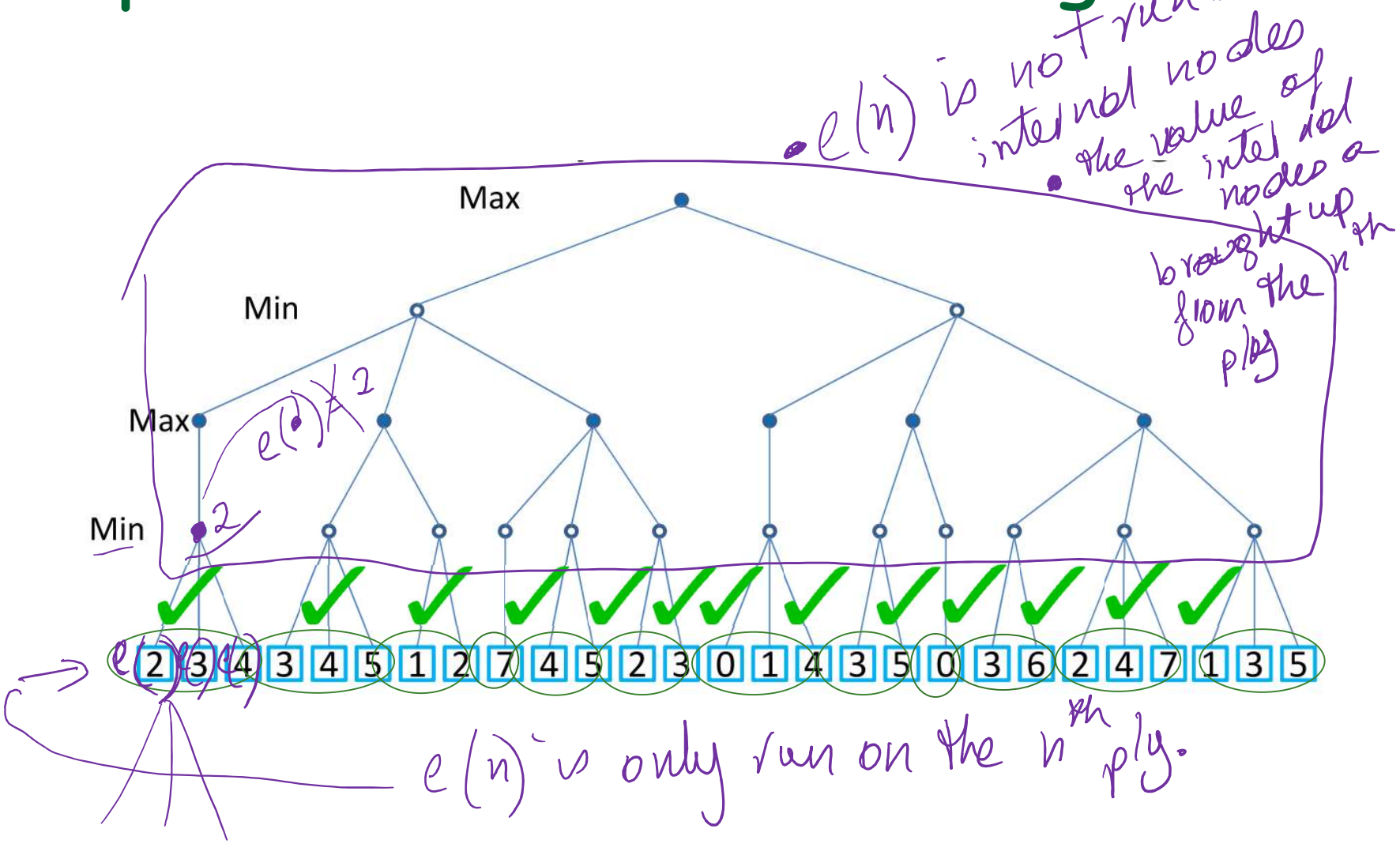
- best ordering:
  1. children of MIN : smallest node first
  2. children of MAX: largest node first



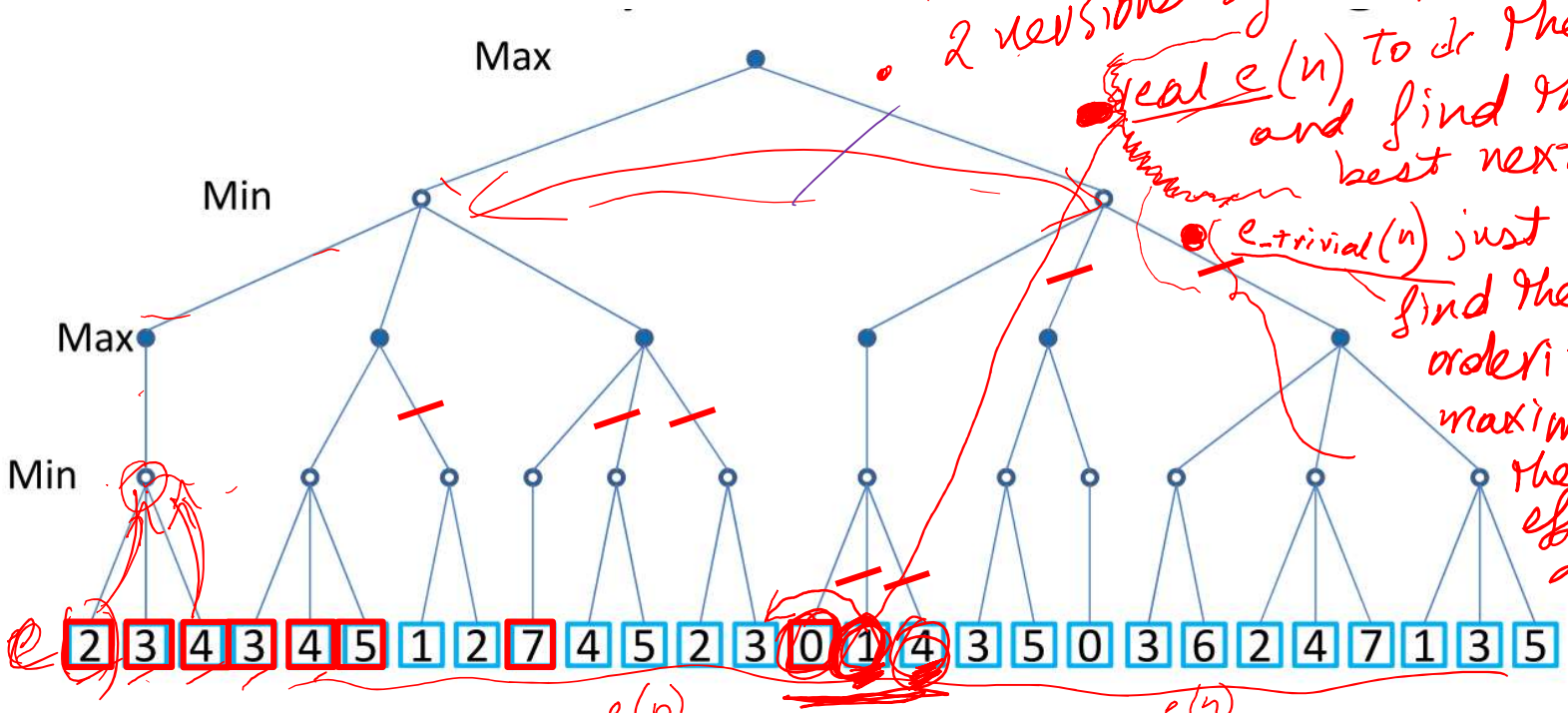
Alpha-Beta: Best ordering



Alpha-Beta: Best ordering



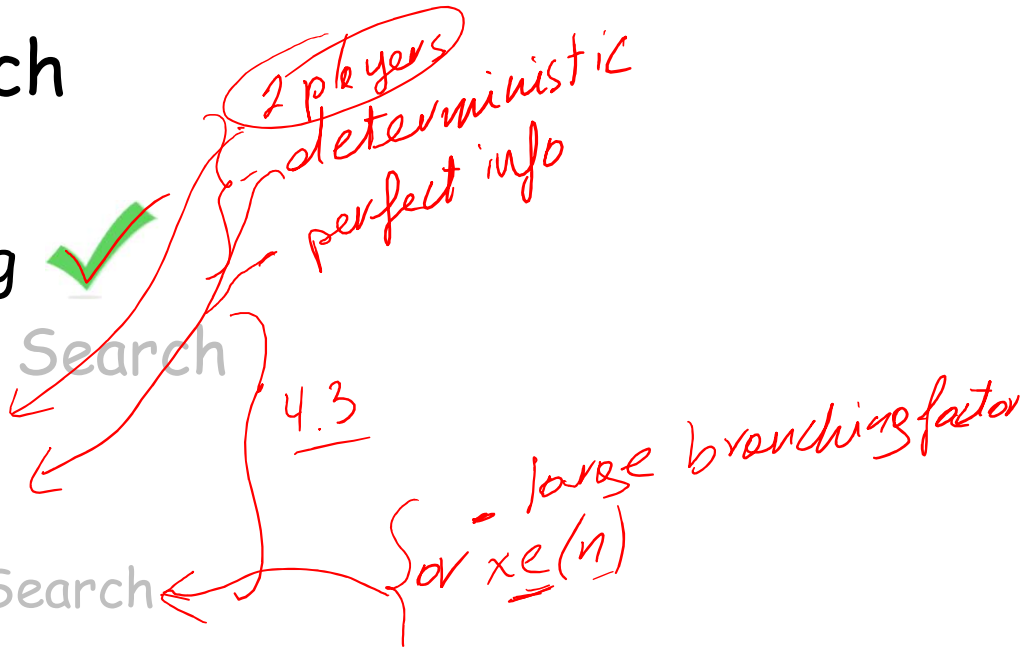
Alpha-Beta: Best ordering



8 nodes explored out of 27

Today

- Adversarial Search
  1. Minimax ✓
  2. Alpha-beta pruning ✓
  3. Other Adversarial Search
     1. Multiplayer Games
     2. Stochastic Games
     3. Monte Carlo Tree Search



Up Next

- Adversarial Search
  1. Minimax
  2. Alpha-beta pruning
  3. Other Adversarial Search
     1. Multiplayer Games
     2. Stochastic Games
     3. Monte Carlo Tree Search