Chapter 3 Adversarial Search

COMP 6721 Introduction of AI

Russell & Norvig – Section 5.1 & 5.4

Some slides from: robotics.stanford.edu/~latombe/cs121/2003/home.htm

Adversarial Search

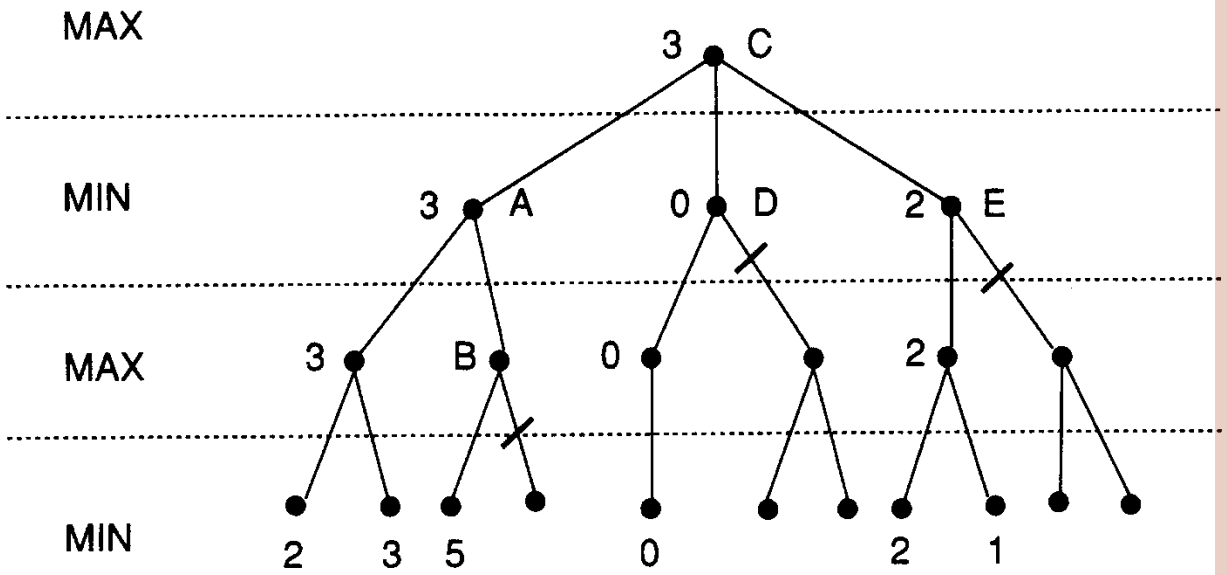
- ***MiniMax Search***
- Alpha – beta pruning

Alpha-Beta Pruning

- Optimization over minimax, that:
  - ignores (cuts off, prunes) branches of the tree that cannot possibly lead to a better solution
  - reduces branching factor
  - allows deeper search with same effort

Alpha-Beta Pruning Example 1

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



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

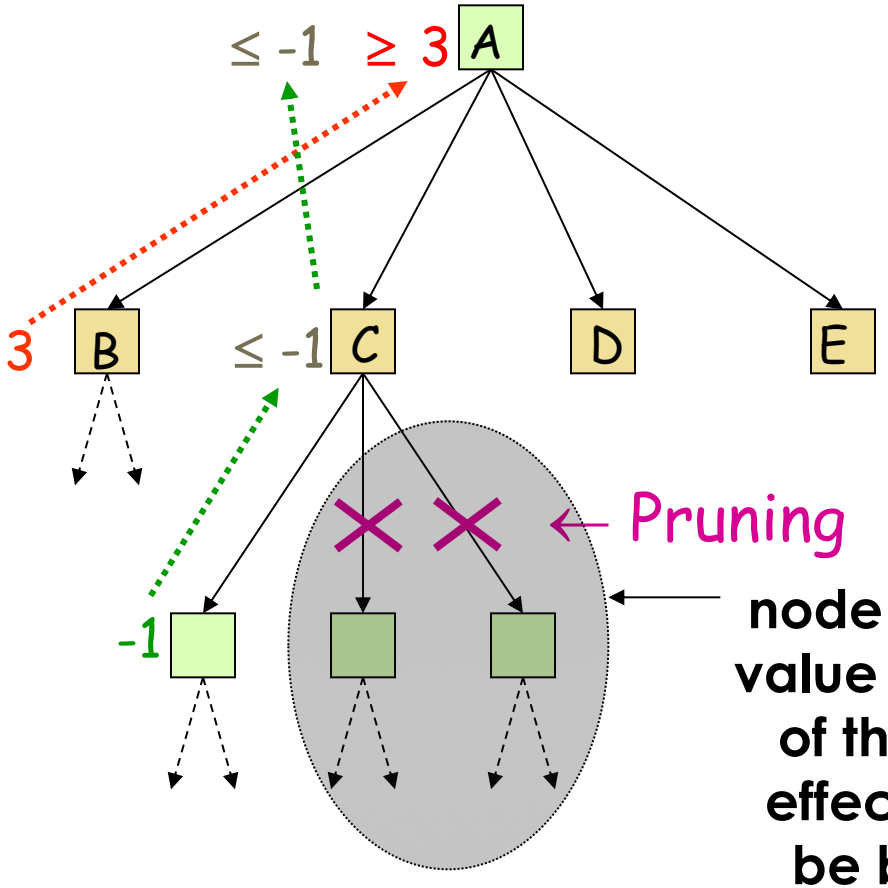
So we can ignore D's right branch

E will be <= 2.

So we can ignore E's right branch

Because C will be 3.

A Closer Look



max level

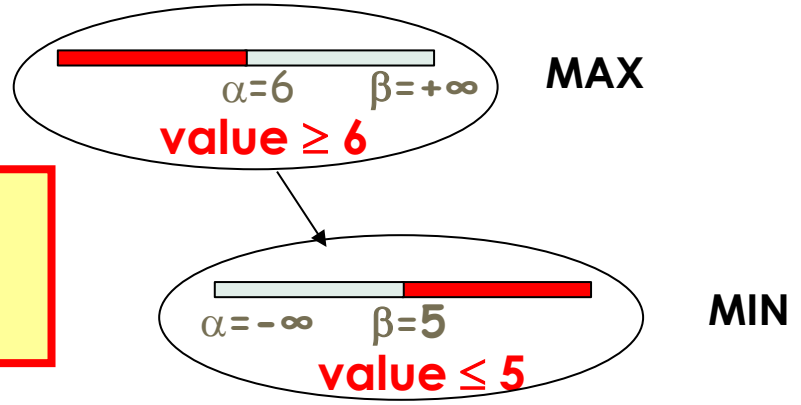
min level

node C will not contribute its value to node A… so this part of the tree can't have any effect on the value that will be backed up to node A

Alpha-Beta Pruning Algorithm

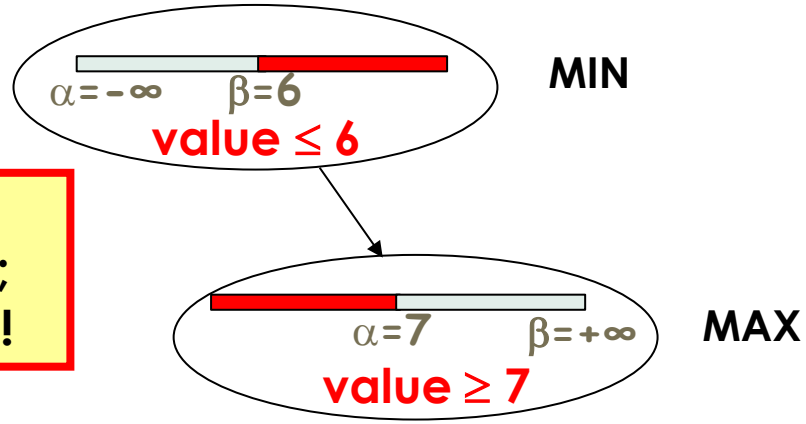
- α : lower bound on the final backed-up value.
- β : upper bound on the final backed-up value.
- **Alpha pruning**:
  - eg. if MAX node's $\alpha = 6$, then the search can prune branches from a MIN descendant that has a $\beta <= 6$.
  - if child β <= ancestor α → prune

incompatible…
so stop searching the right branch;
the value cannot come from there!



Alpha-Beta Pruning Algorithm

- α : lower bound on the final backed-up value.
- β : upper bound on the final backed-up value.
- **Beta pruning**:
  - eg. if a MIN node's $\beta = 6$, then the search can prune branches from a MAX descendant that has an $\alpha >= 6$.
  - if ancestor β <= child α → prune



incompatible…
so stop searching the right branch;
the value cannot come from there!

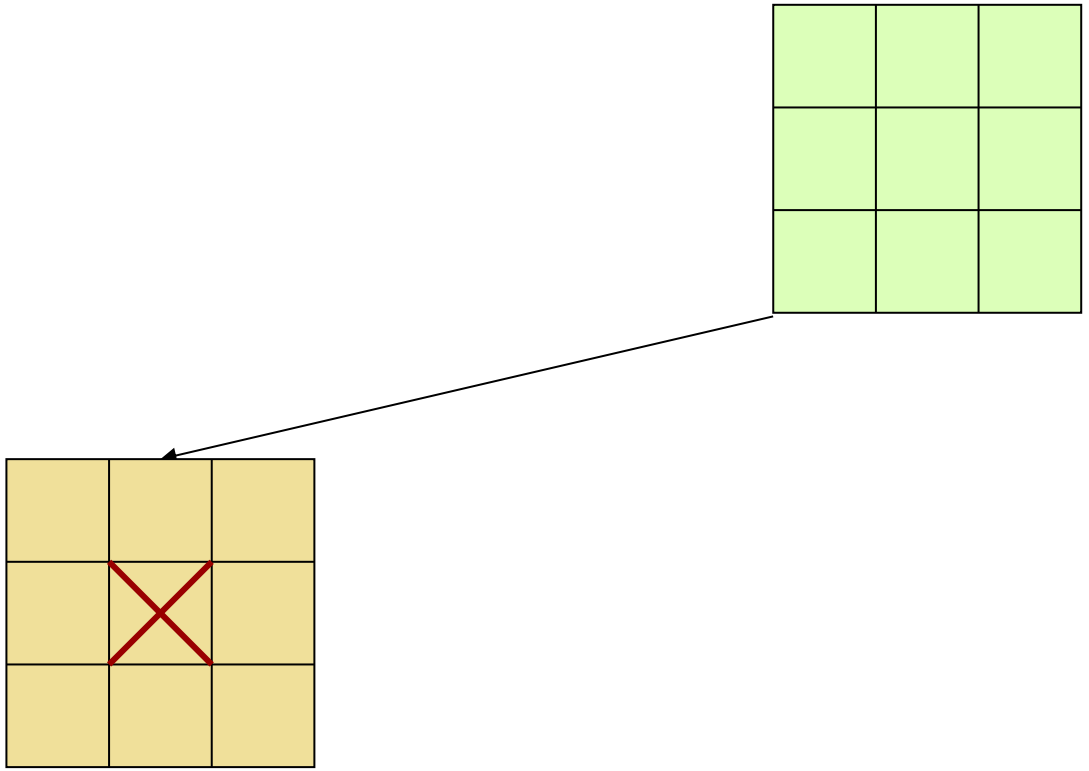
Alpha-Beta Pruning Algorithm

```
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

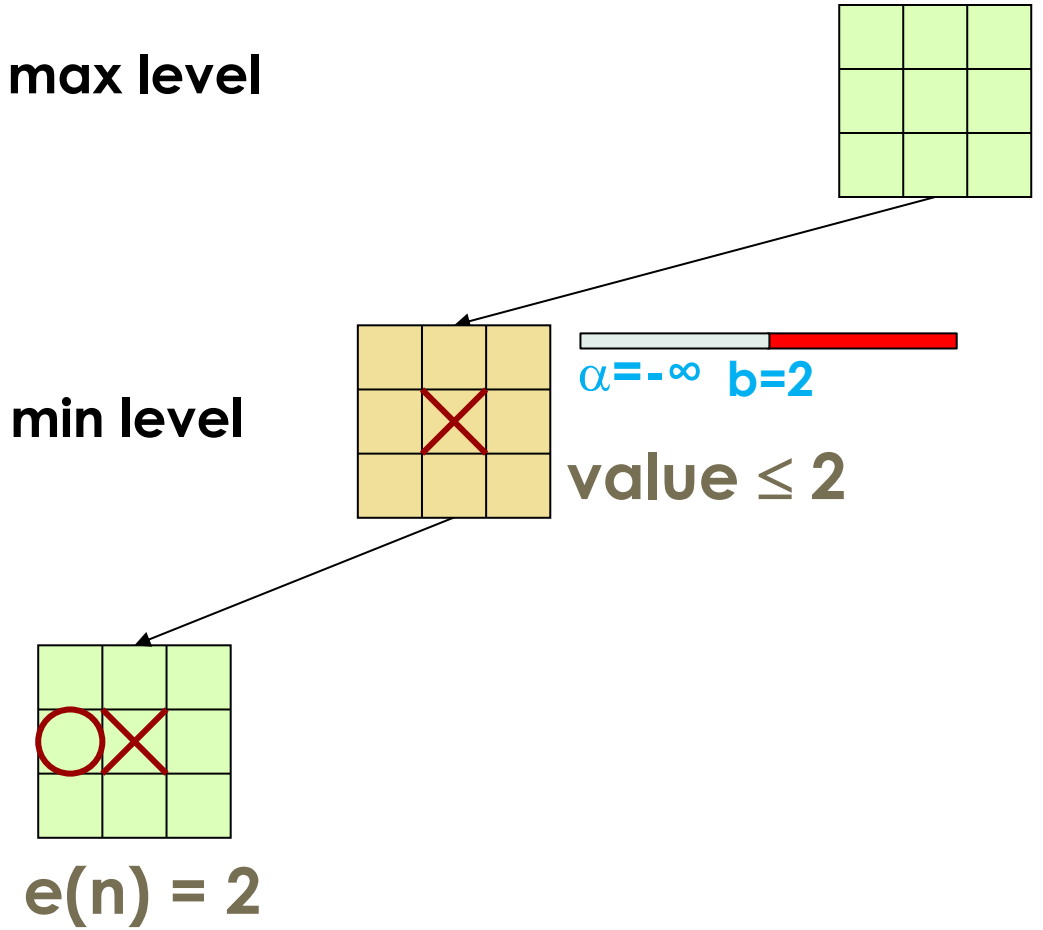
Example: tic-tac-toe



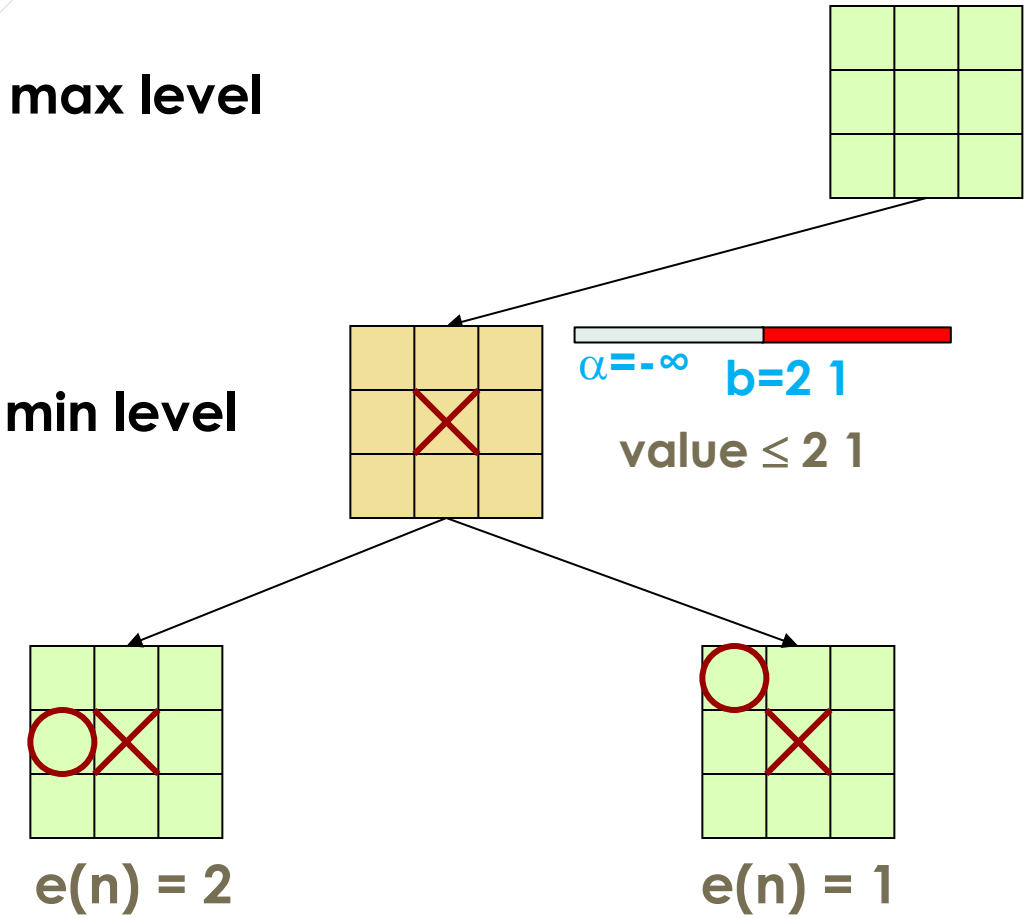
max level

min level

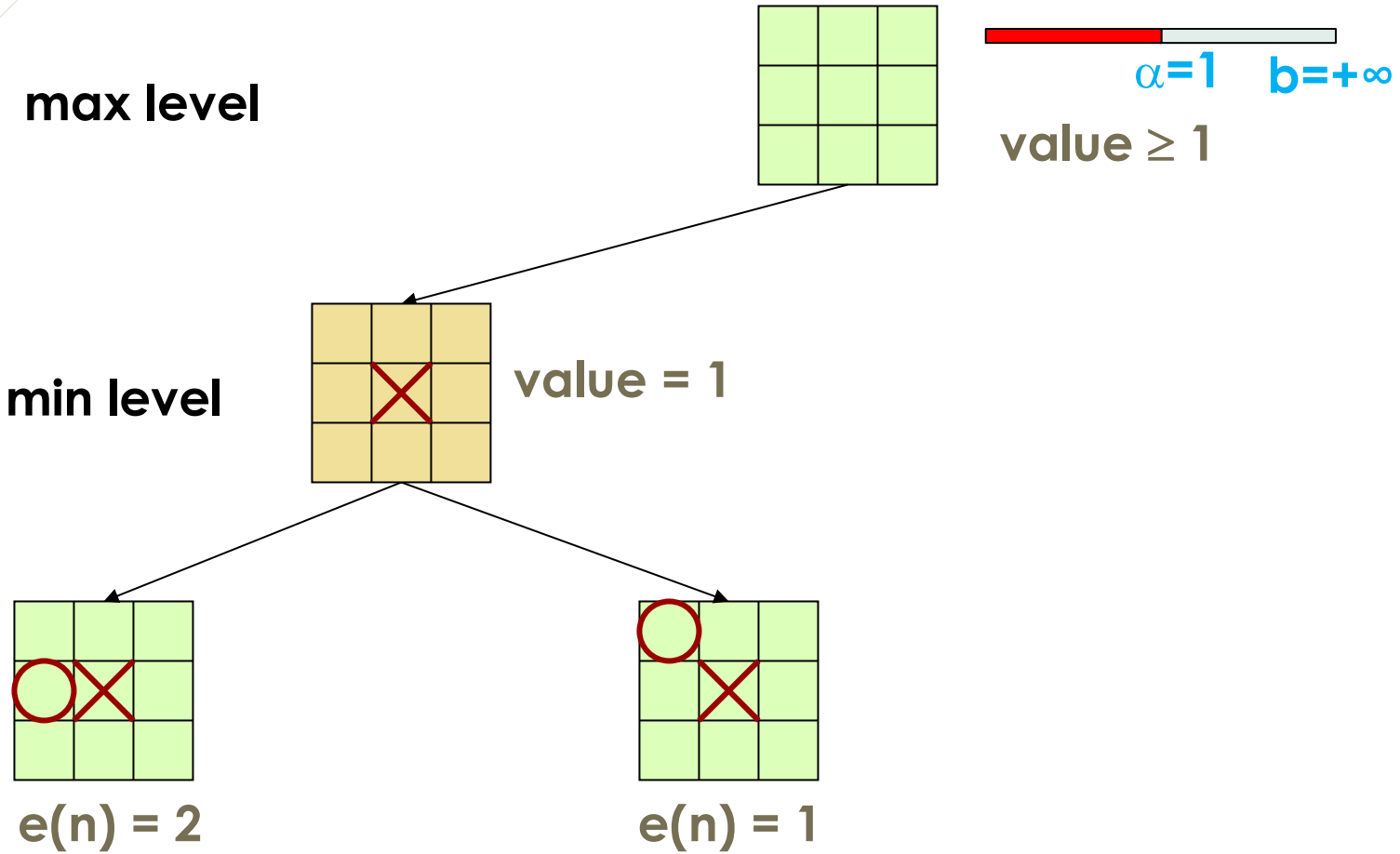
Example: tic-tac-toe



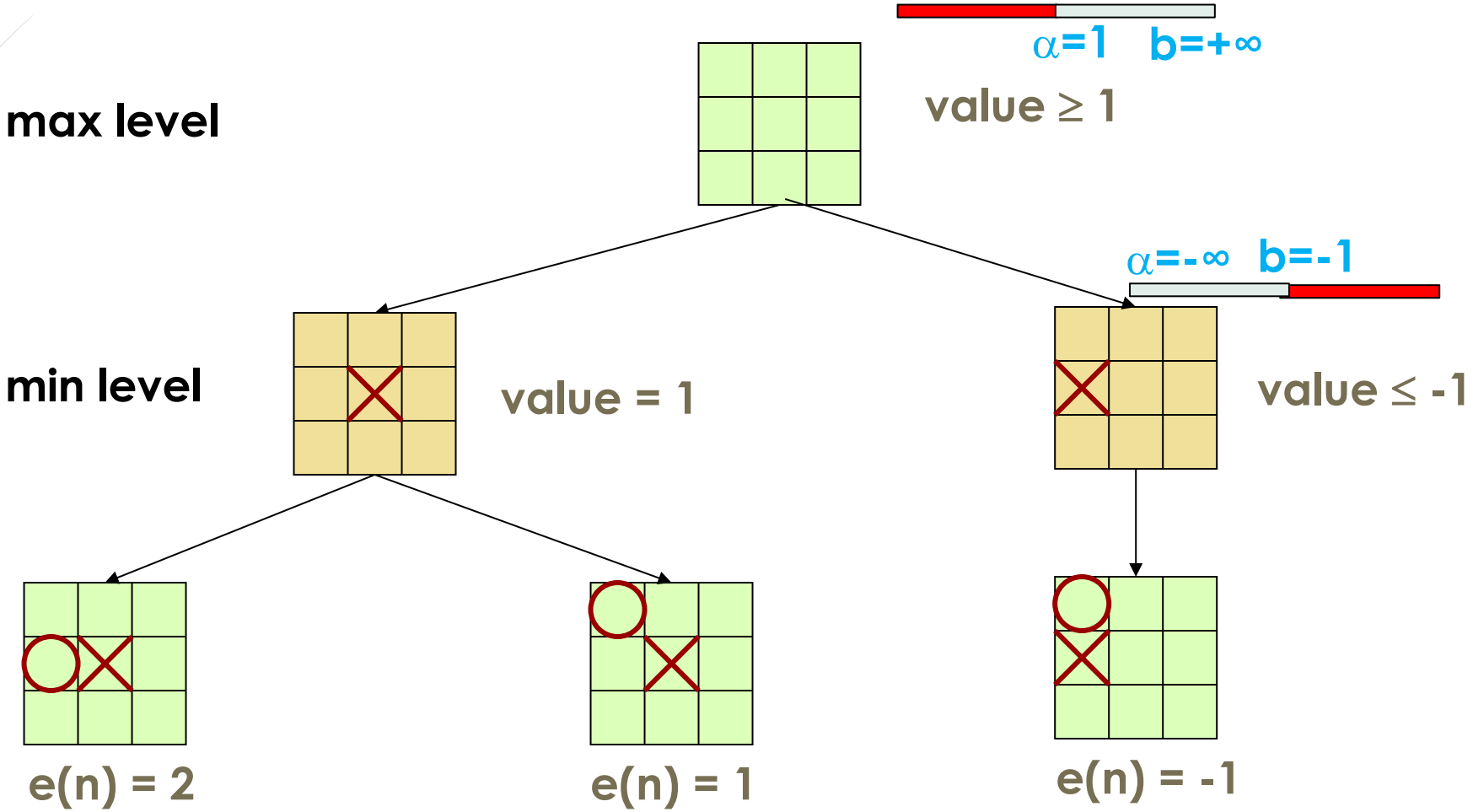
Example: tic-tac-toe



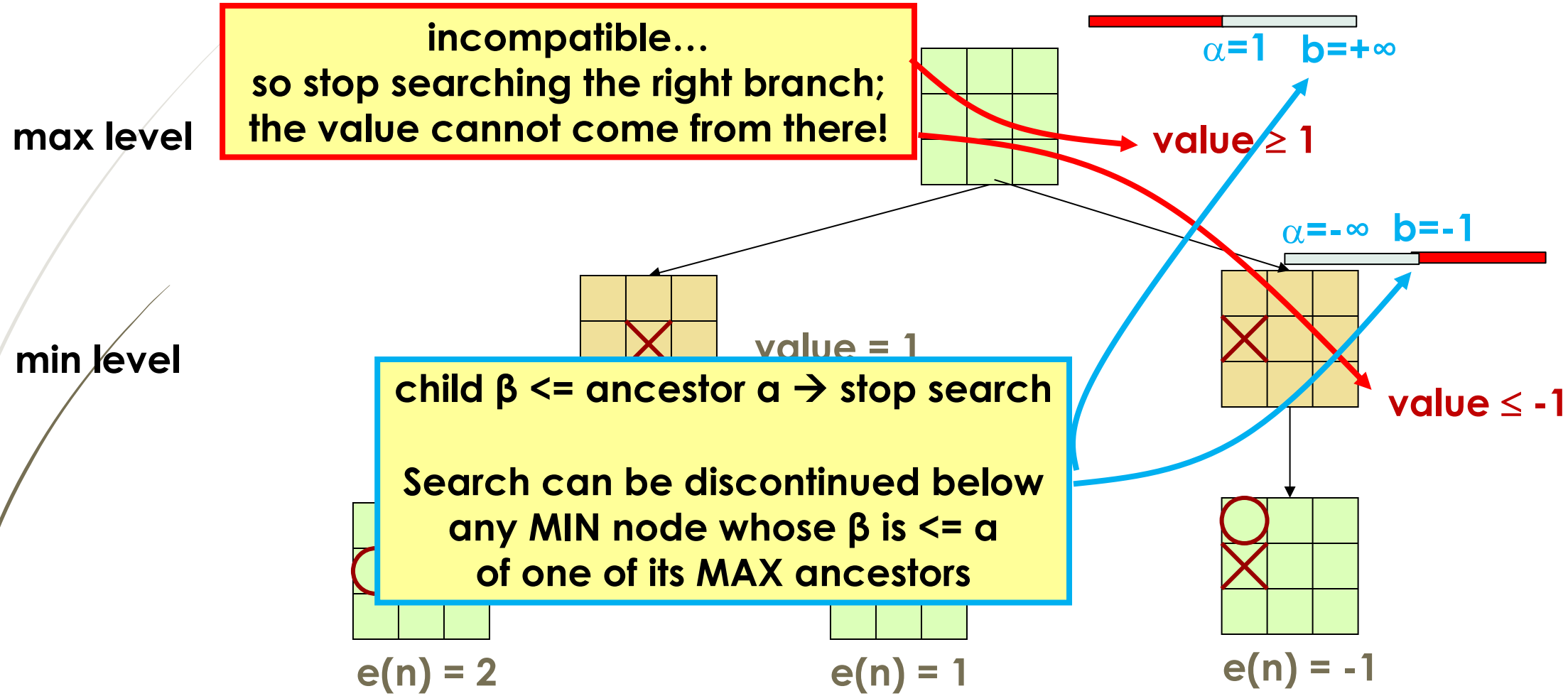
Example: tic-tac-toe



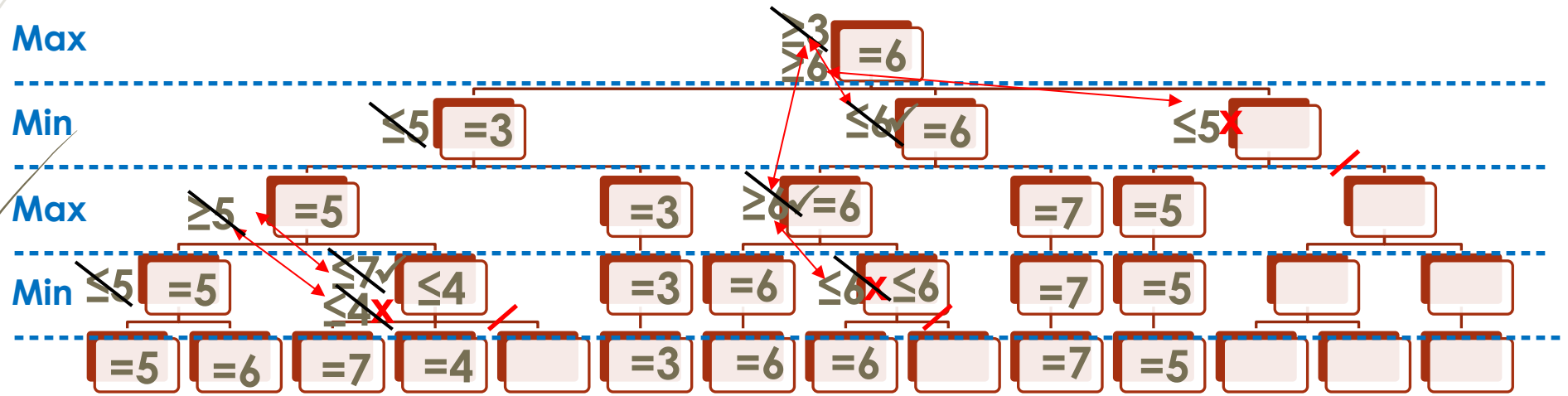
Example: tic-tac-toe



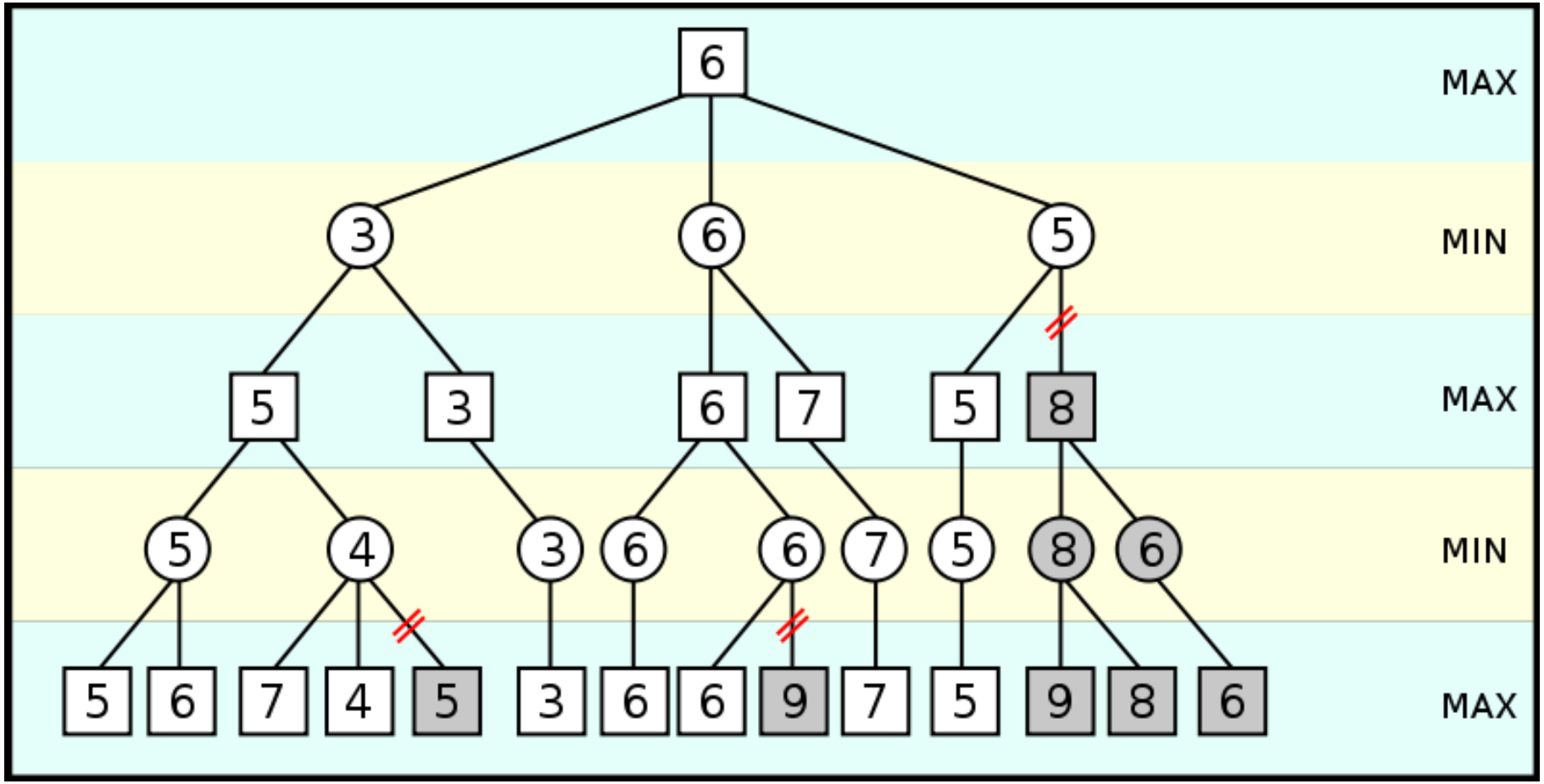
Example: tic-tac-toe



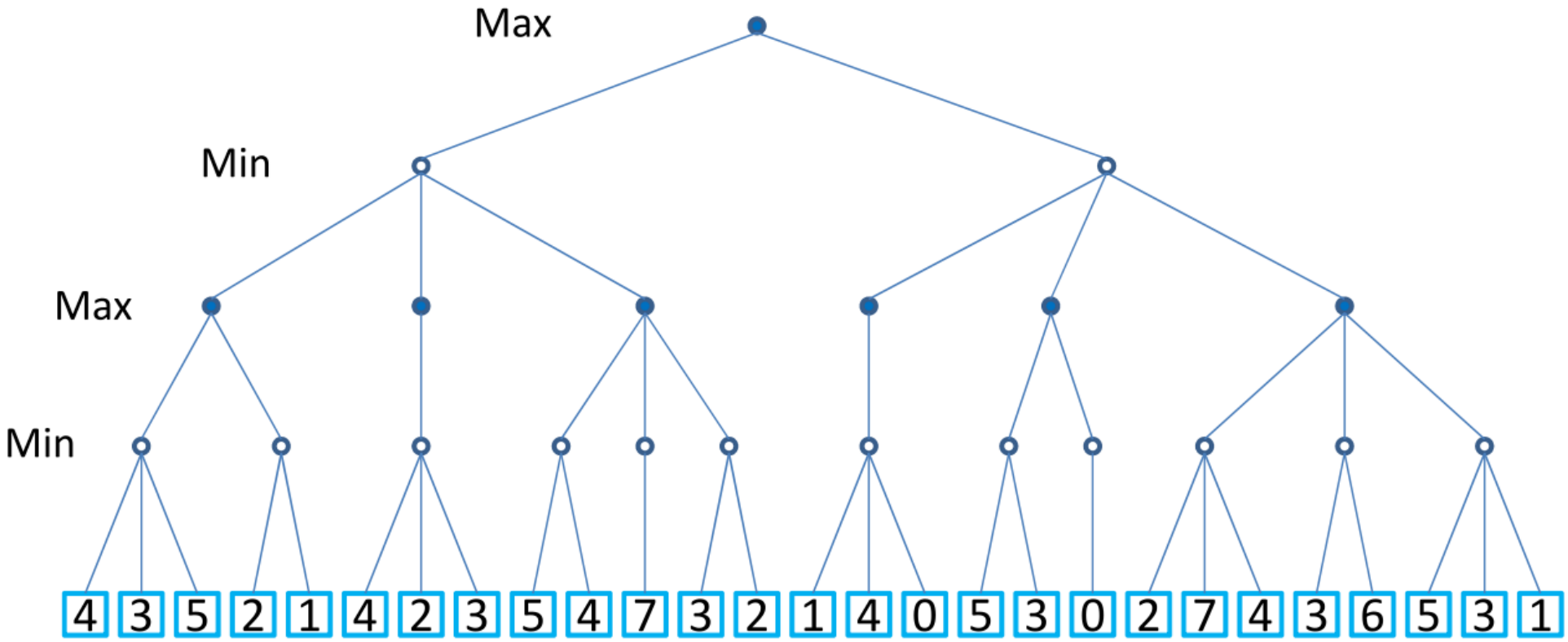
Alpha – Beta Pruning Example 2



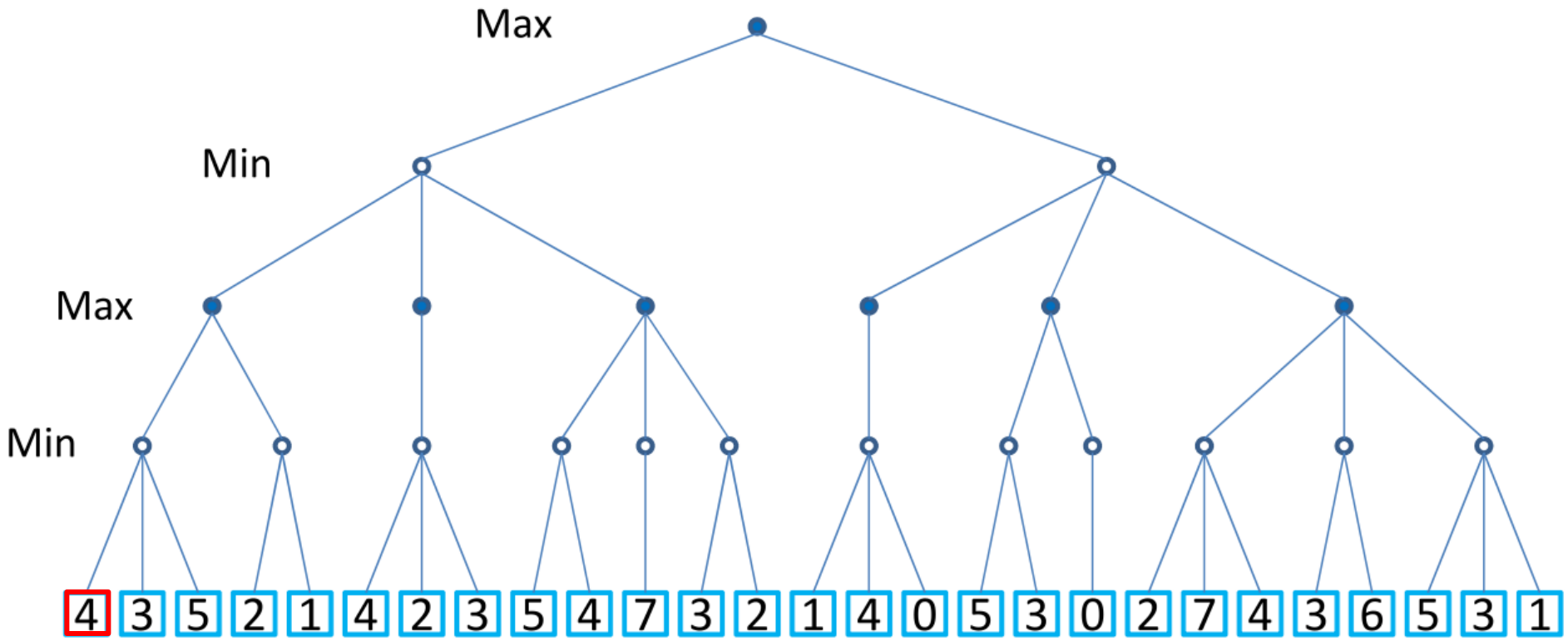
Alpha – Beta Pruning Example 2



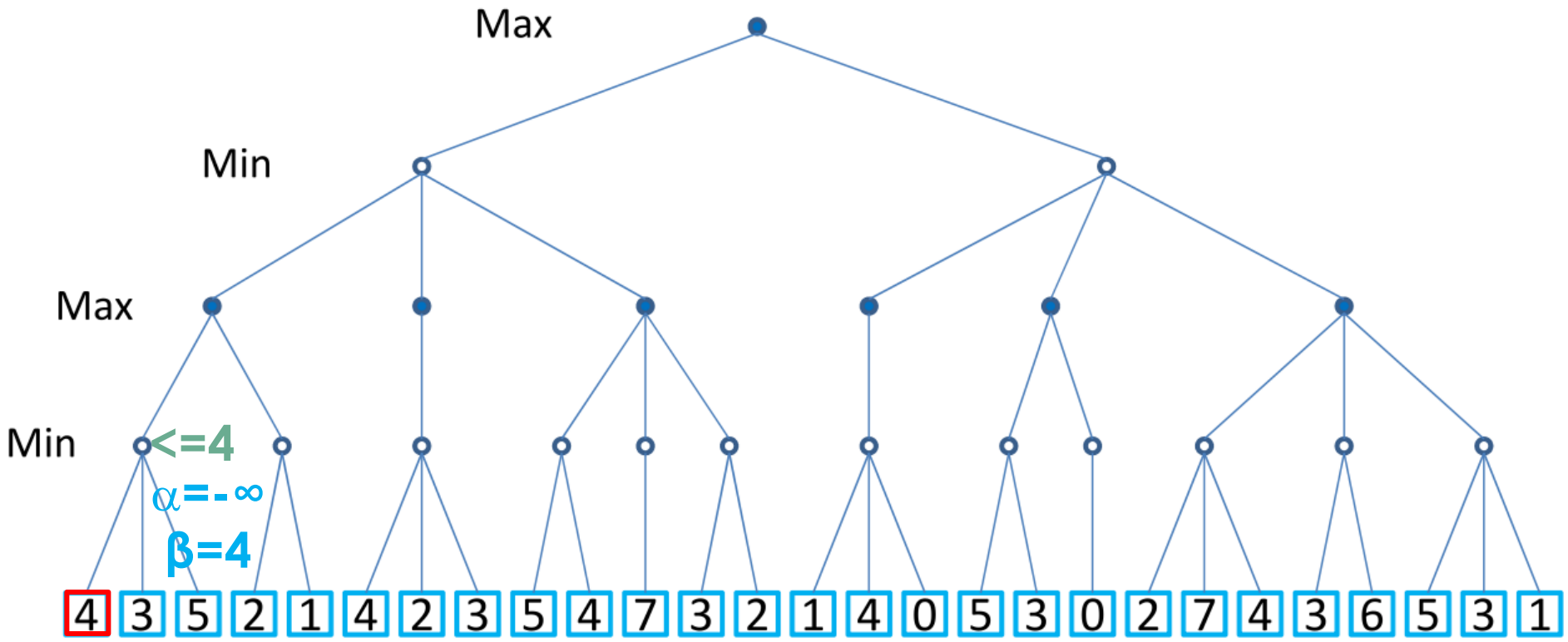
Alpha – Beta Pruning Example 3



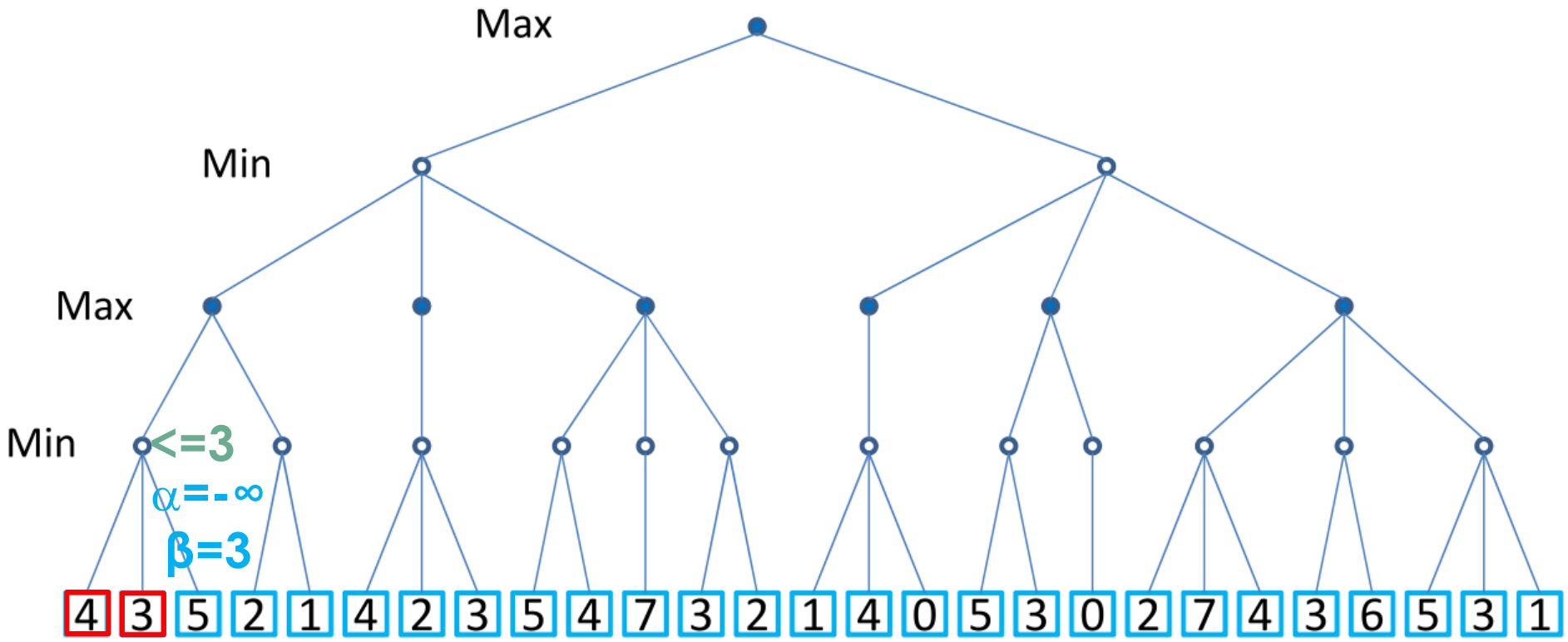
Alpha – Beta Pruning Example 3



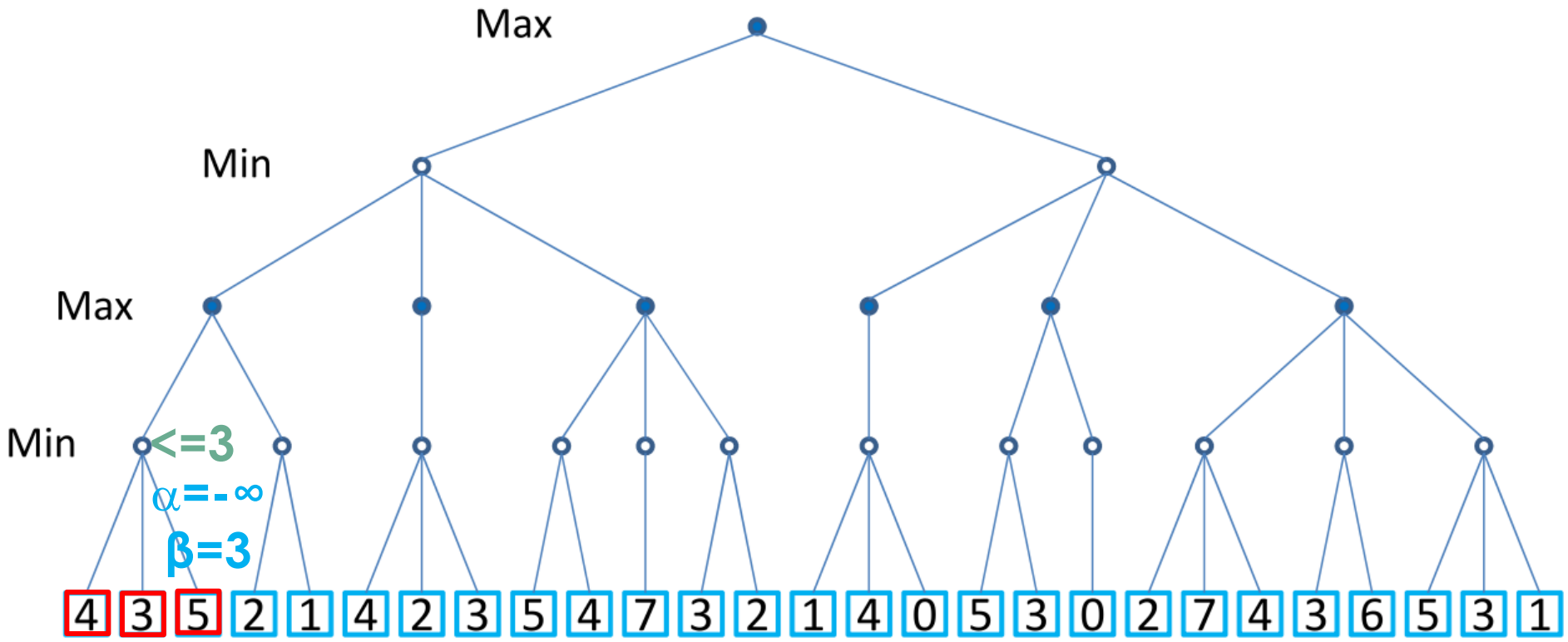
Alpha – Beta Pruning Example 3



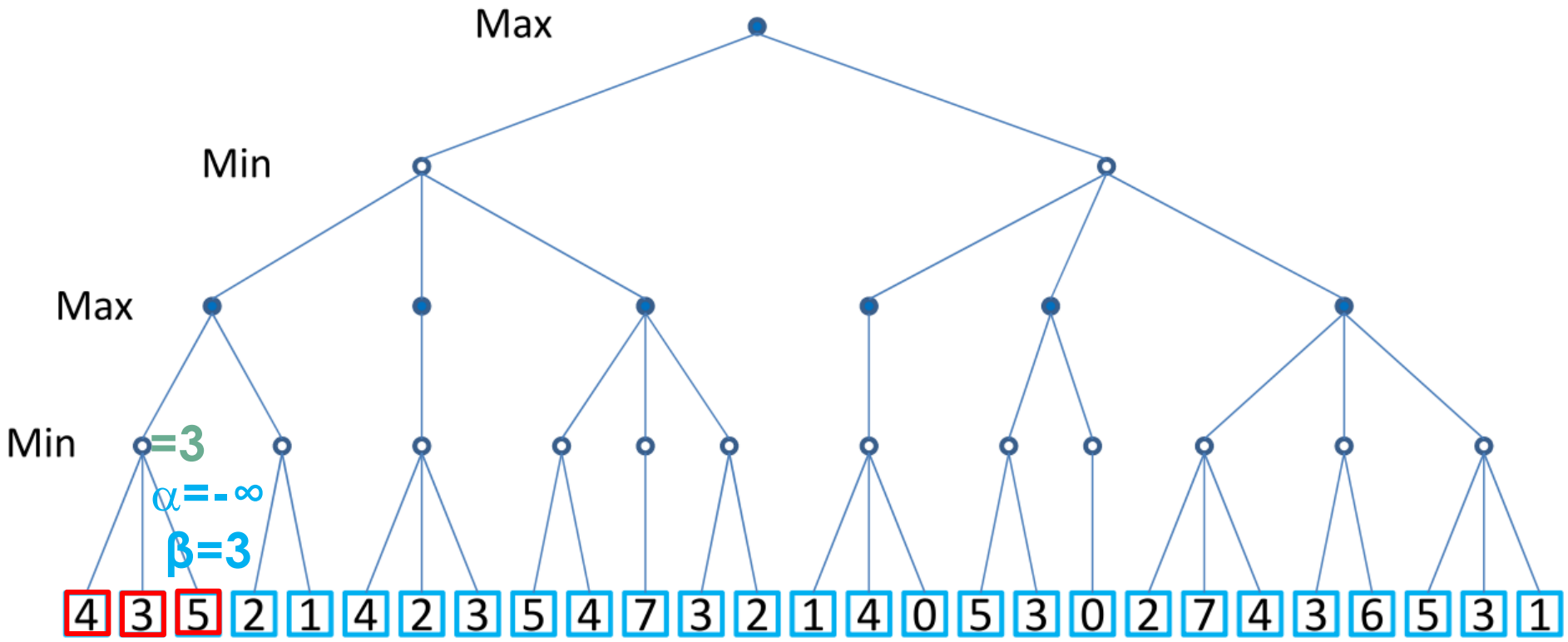
Alpha – Beta Pruning Example 3



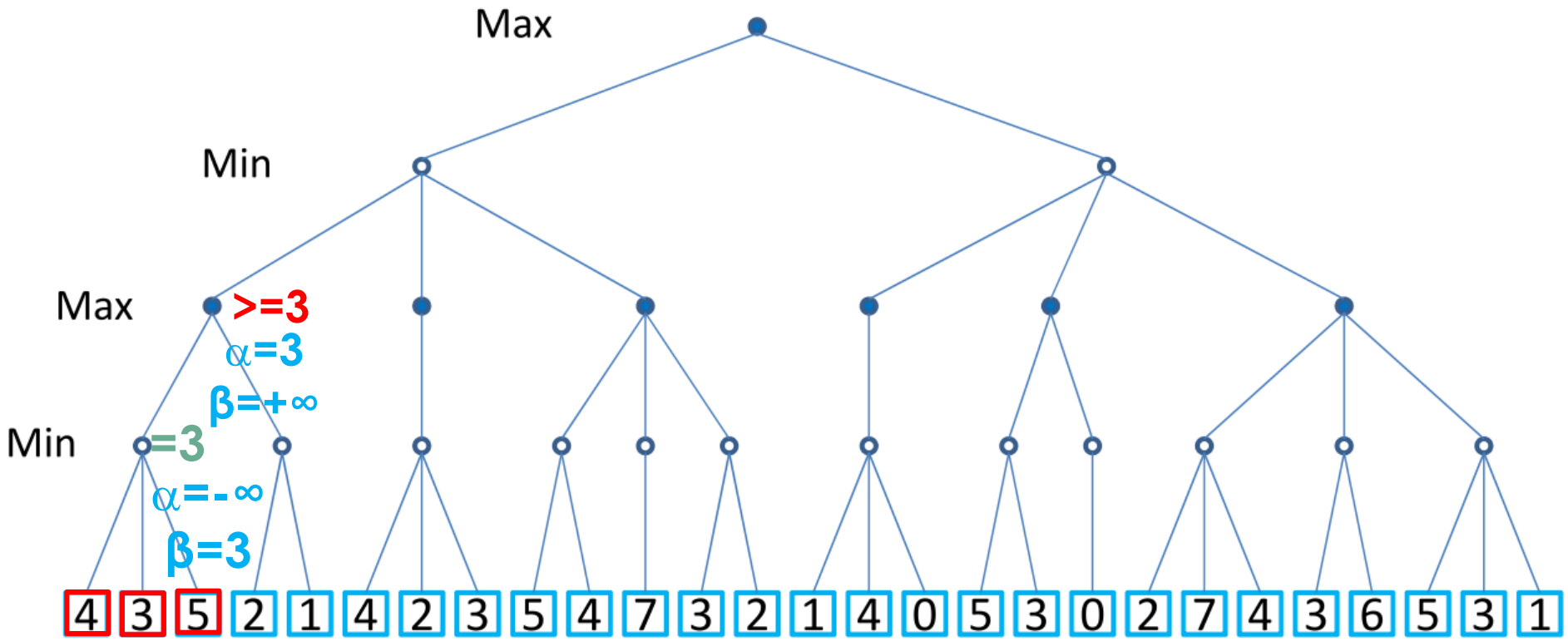
Alpha – Beta Pruning Example 3



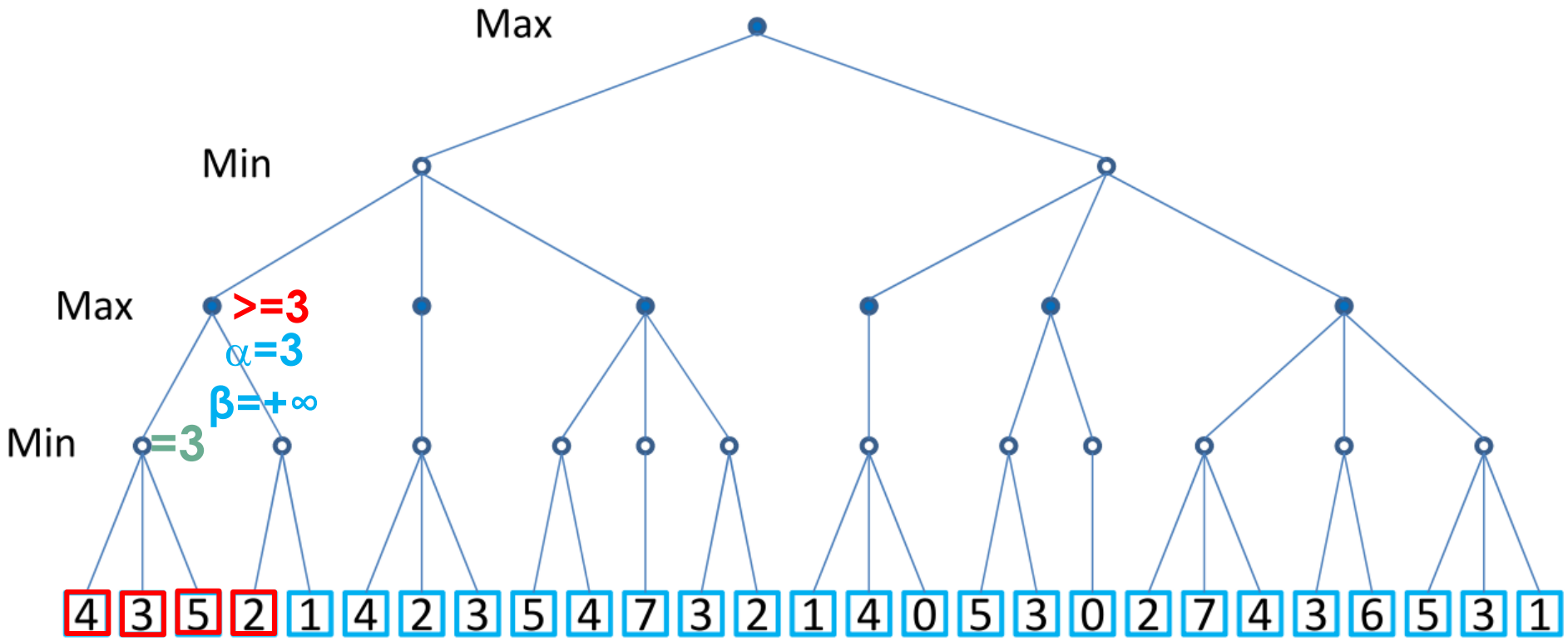
Alpha – Beta Pruning Example 3



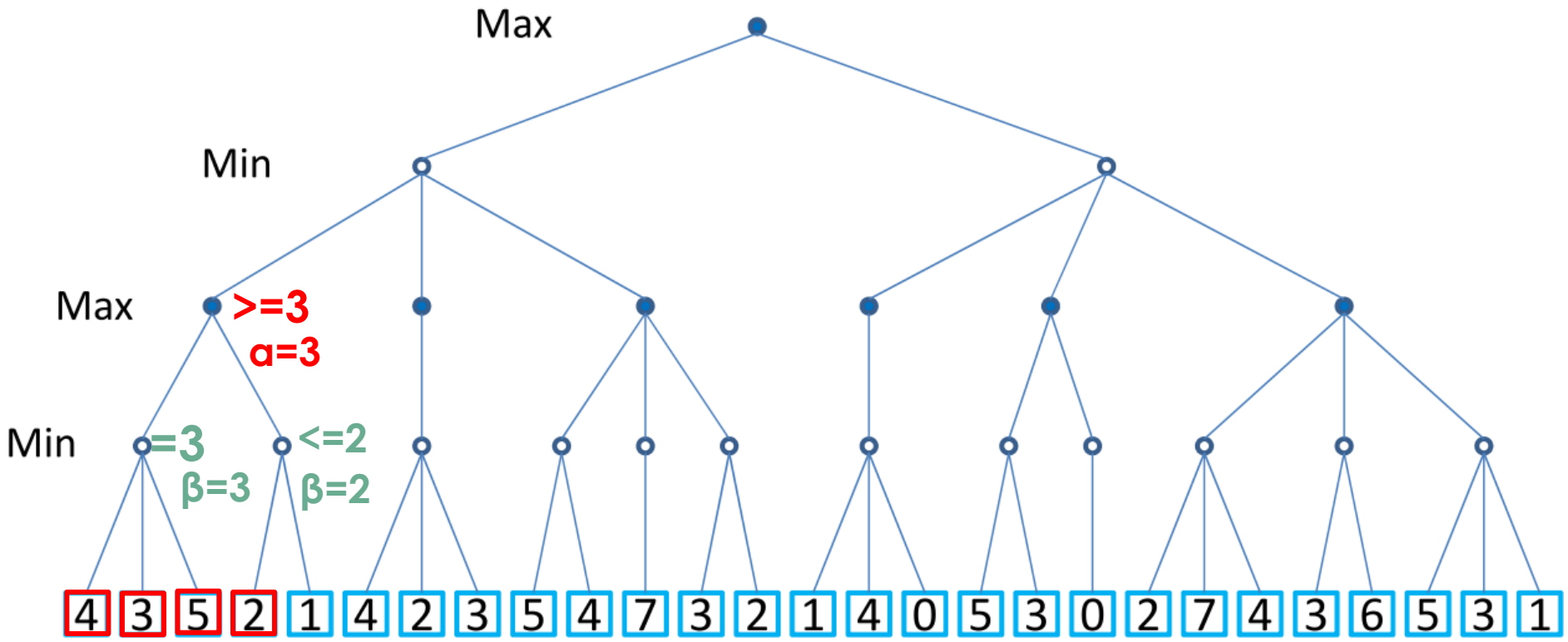
Alpha – Beta Pruning Example 3



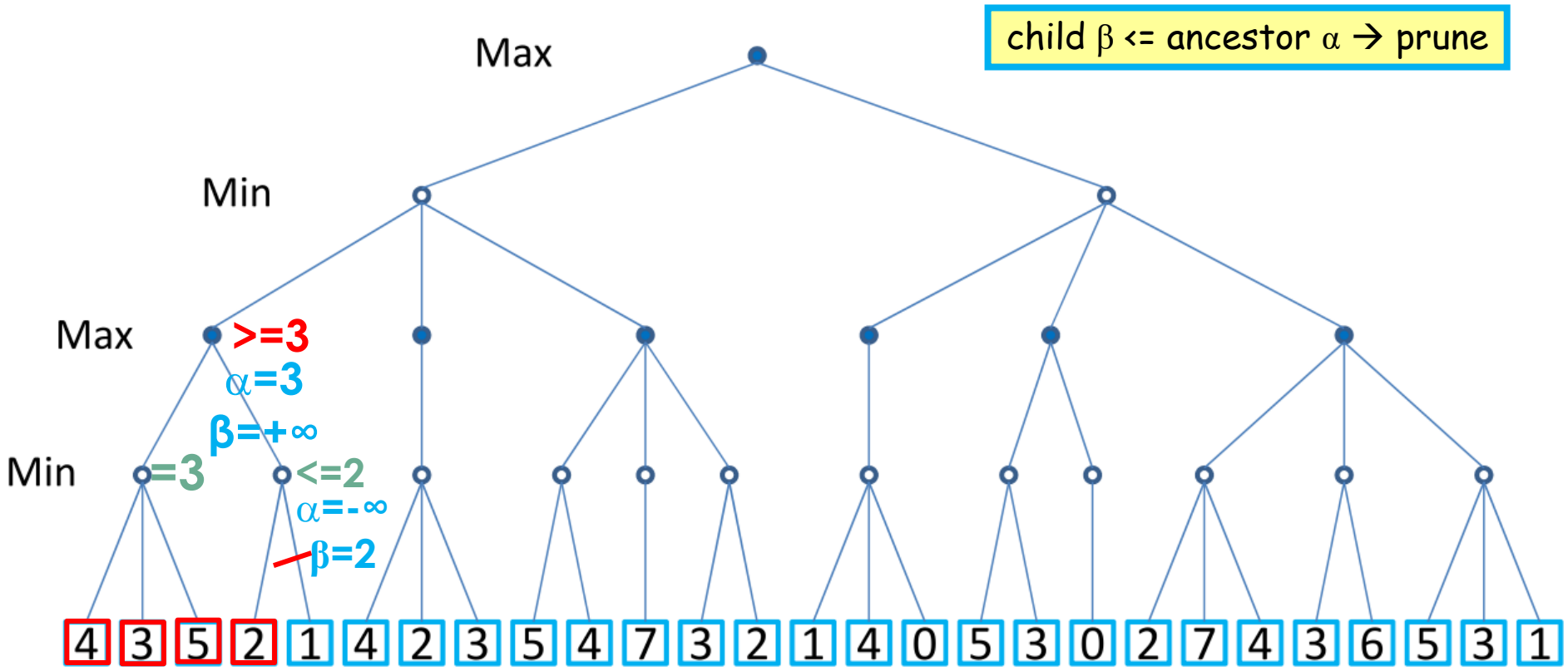
Alpha – Beta Pruning Example 3



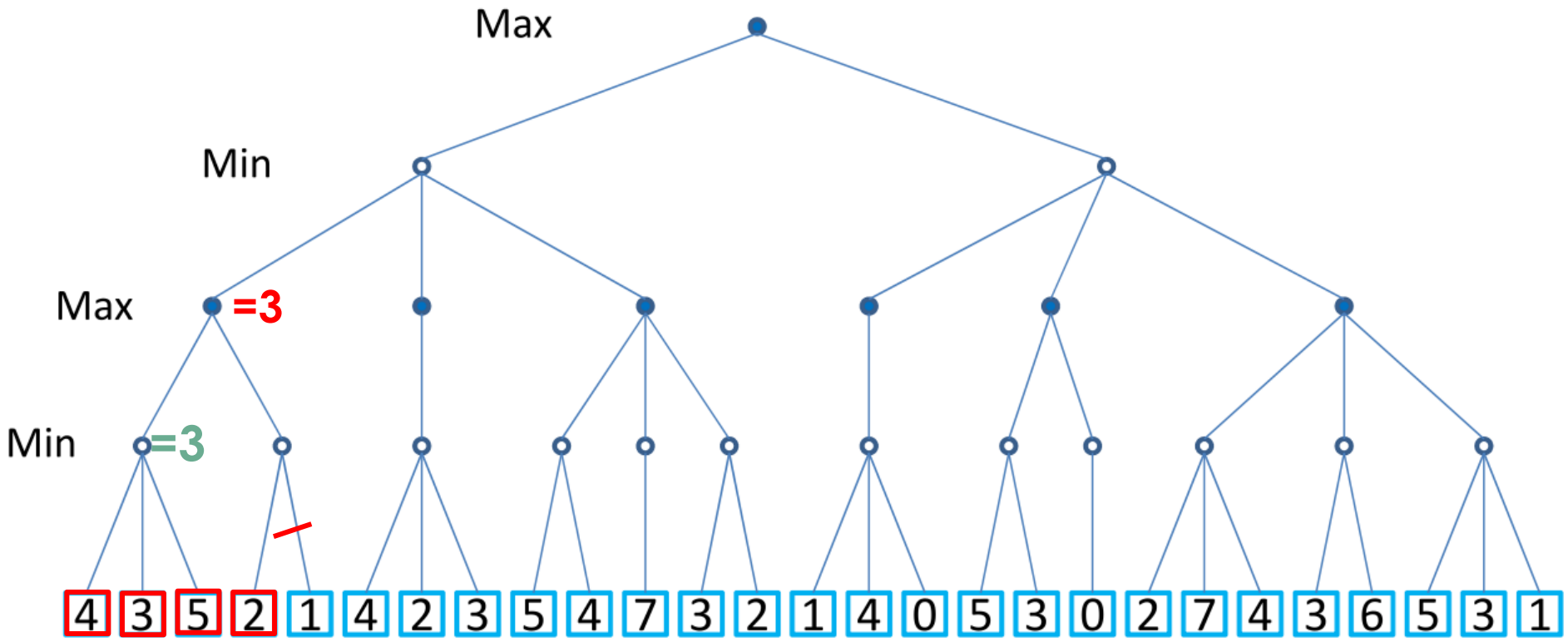
Alpha – Beta Pruning Example 3



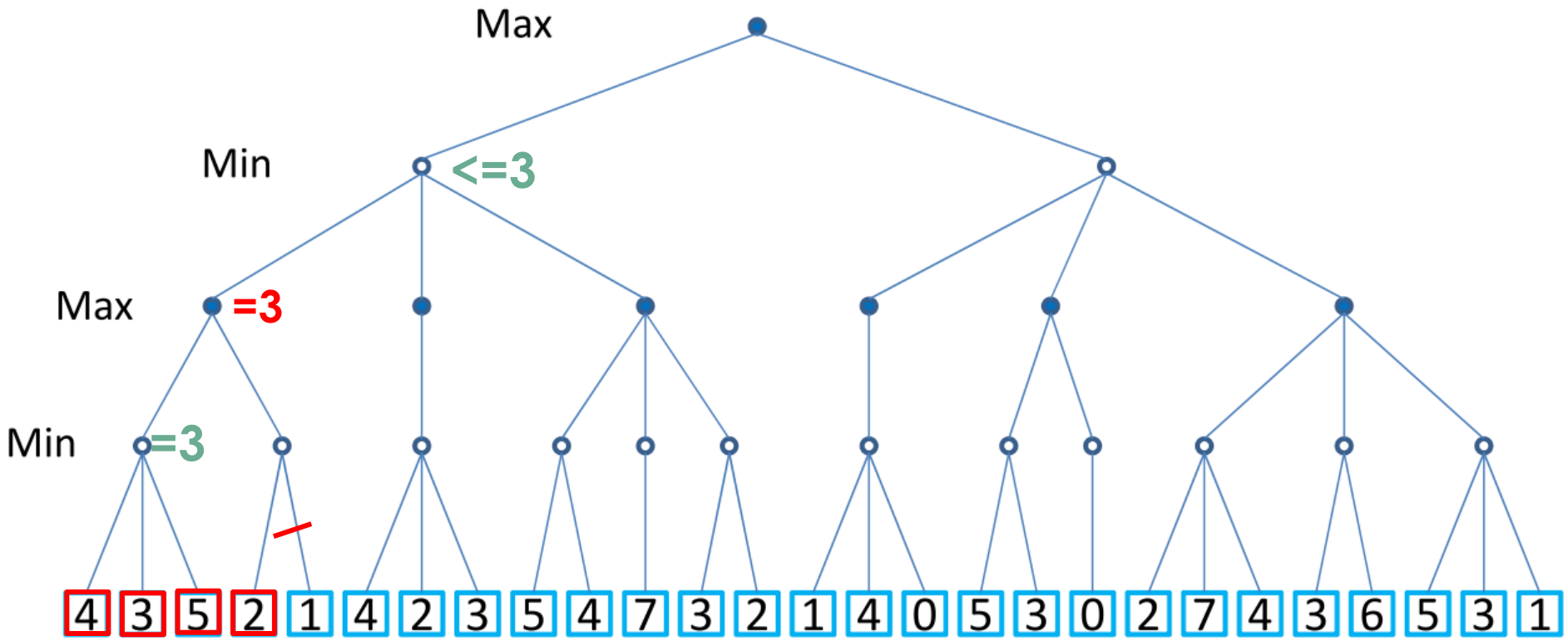
Alpha – Beta Pruning Example 3



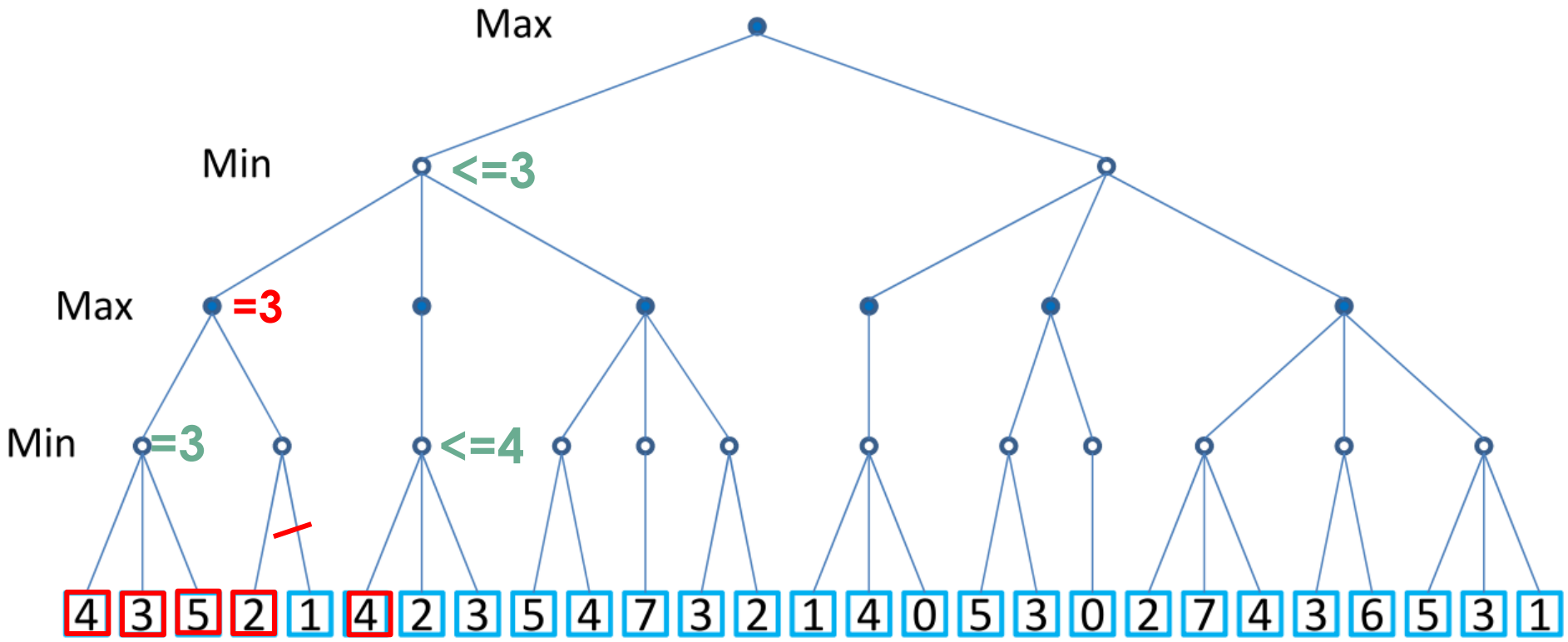
Alpha – Beta Pruning Example 3



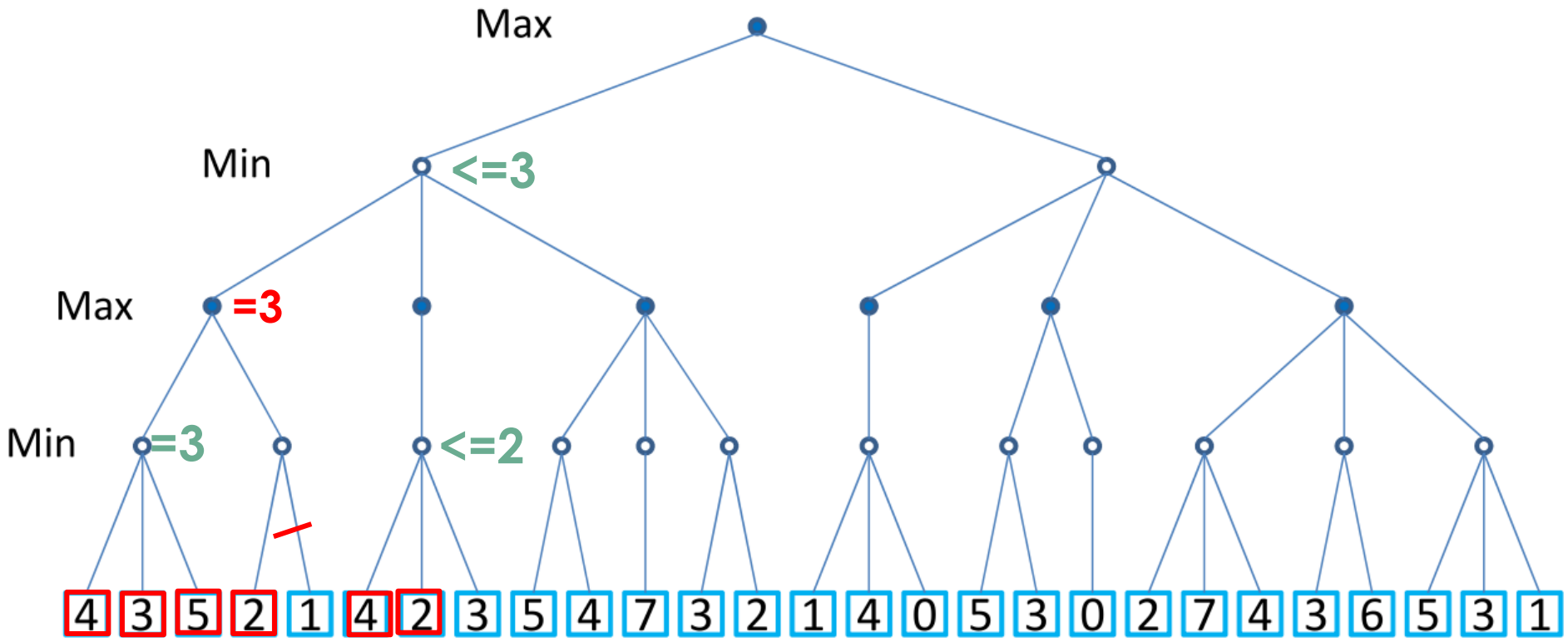
Alpha – Beta Pruning Example 3



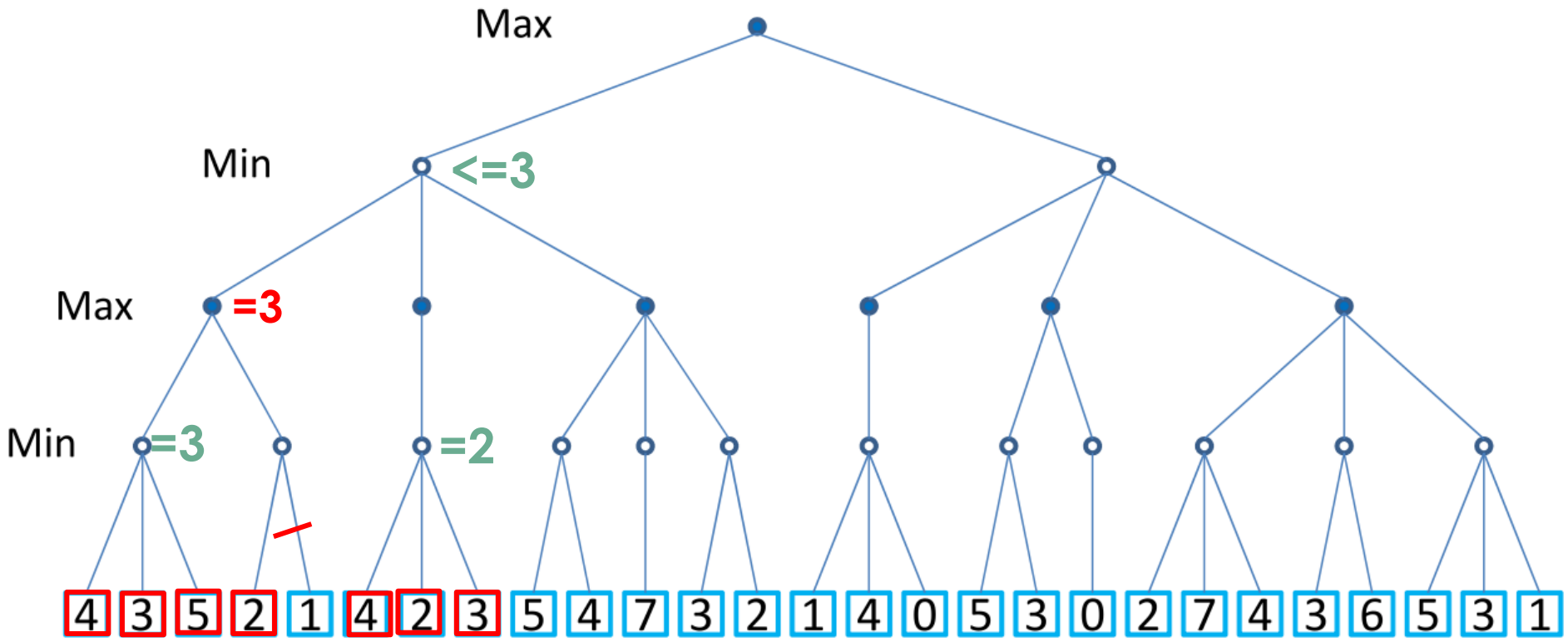
Alpha – Beta Pruning Example 3



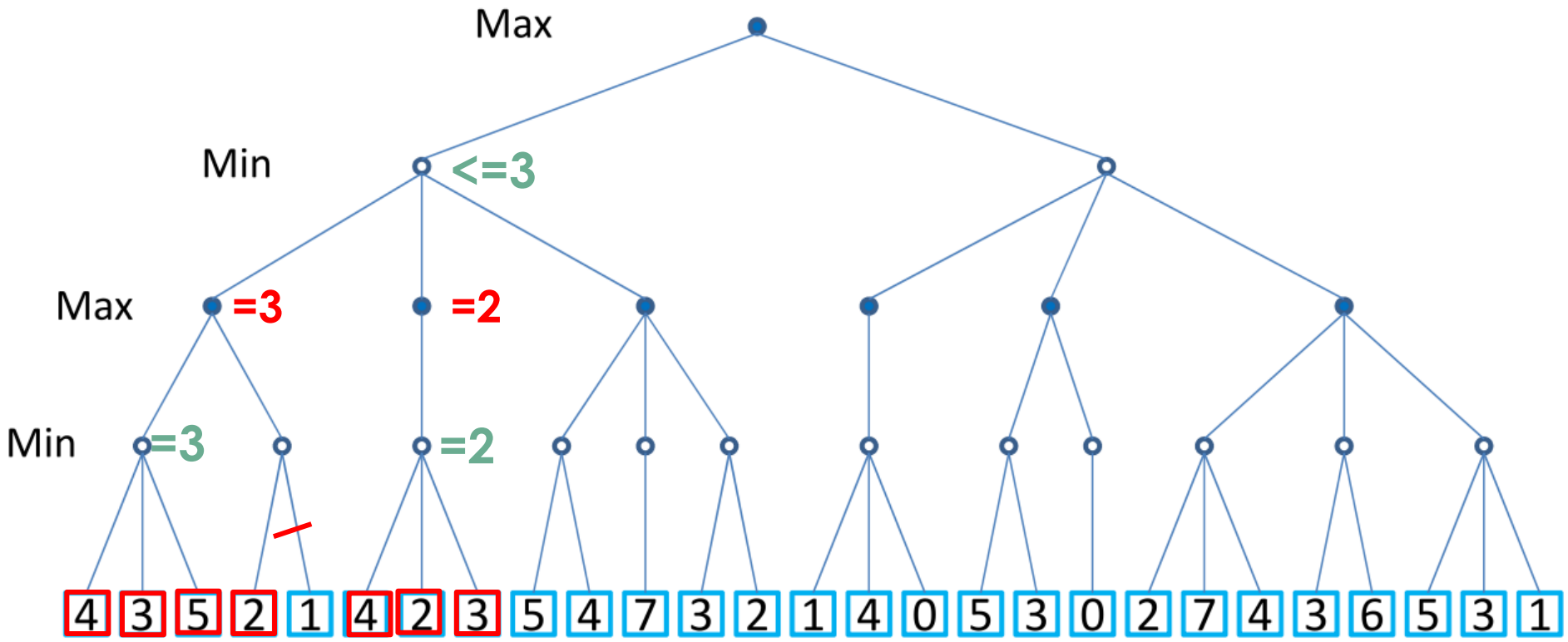
Alpha – Beta Pruning Example 3



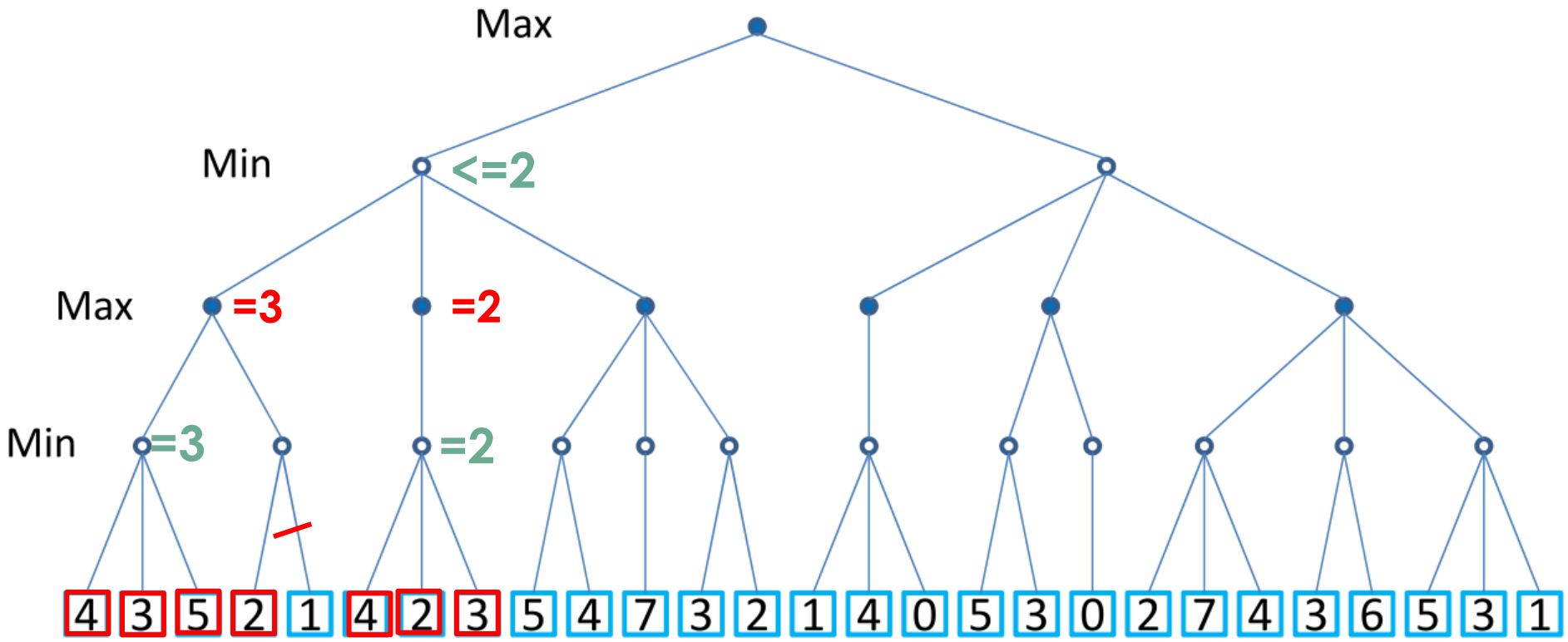
Alpha – Beta Pruning Example 3



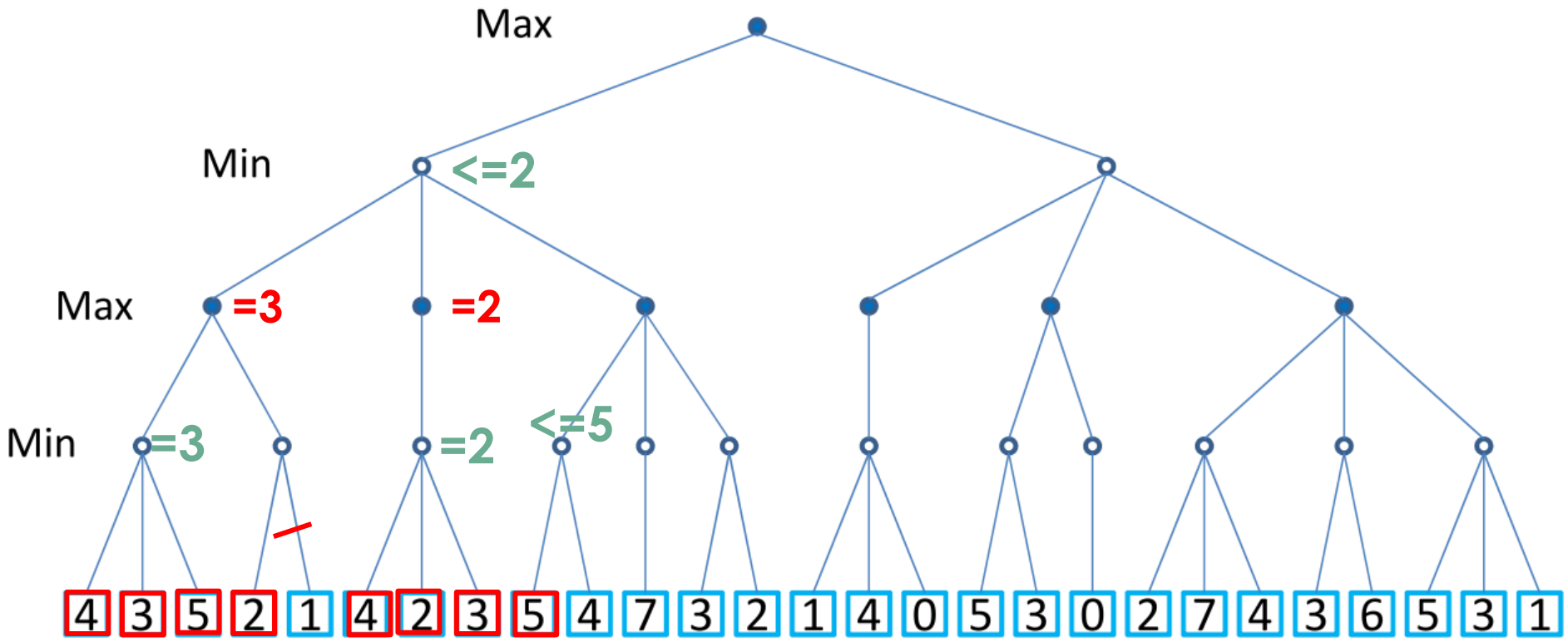
Alpha – Beta Pruning Example 3



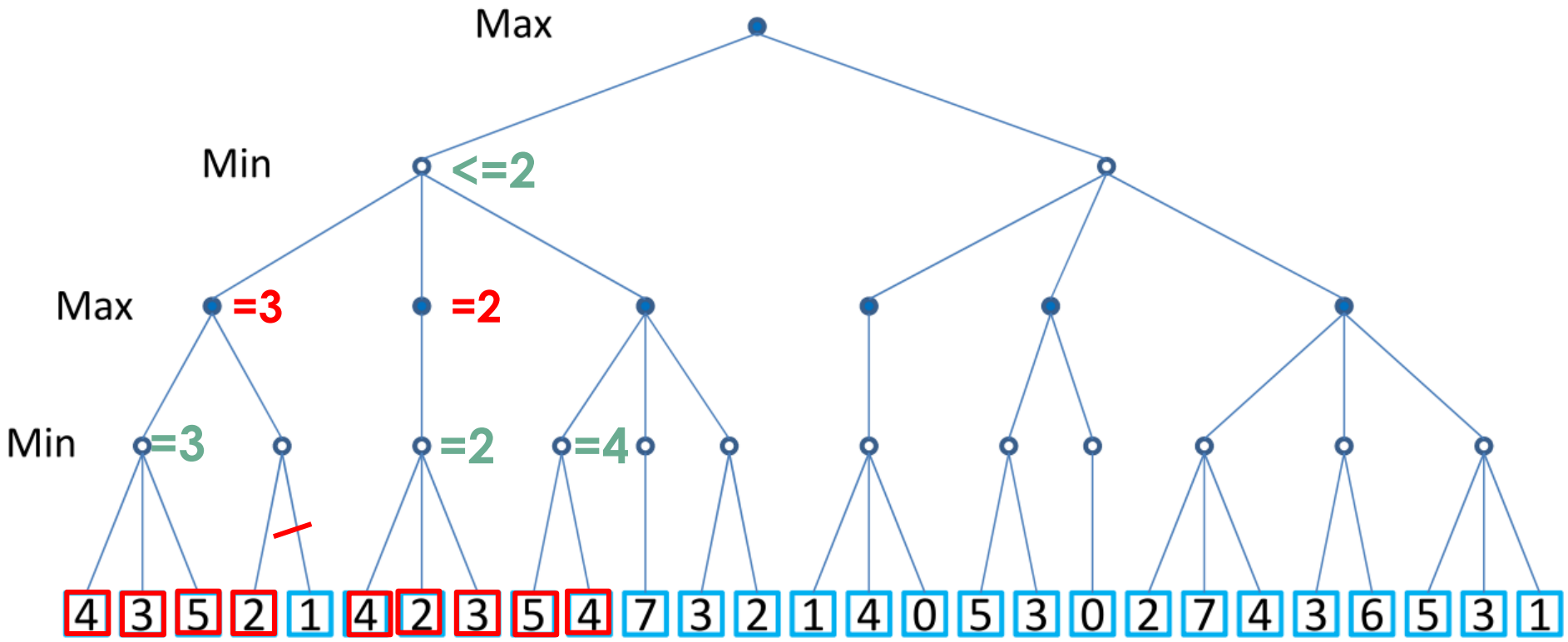
Alpha – Beta Pruning Example 3



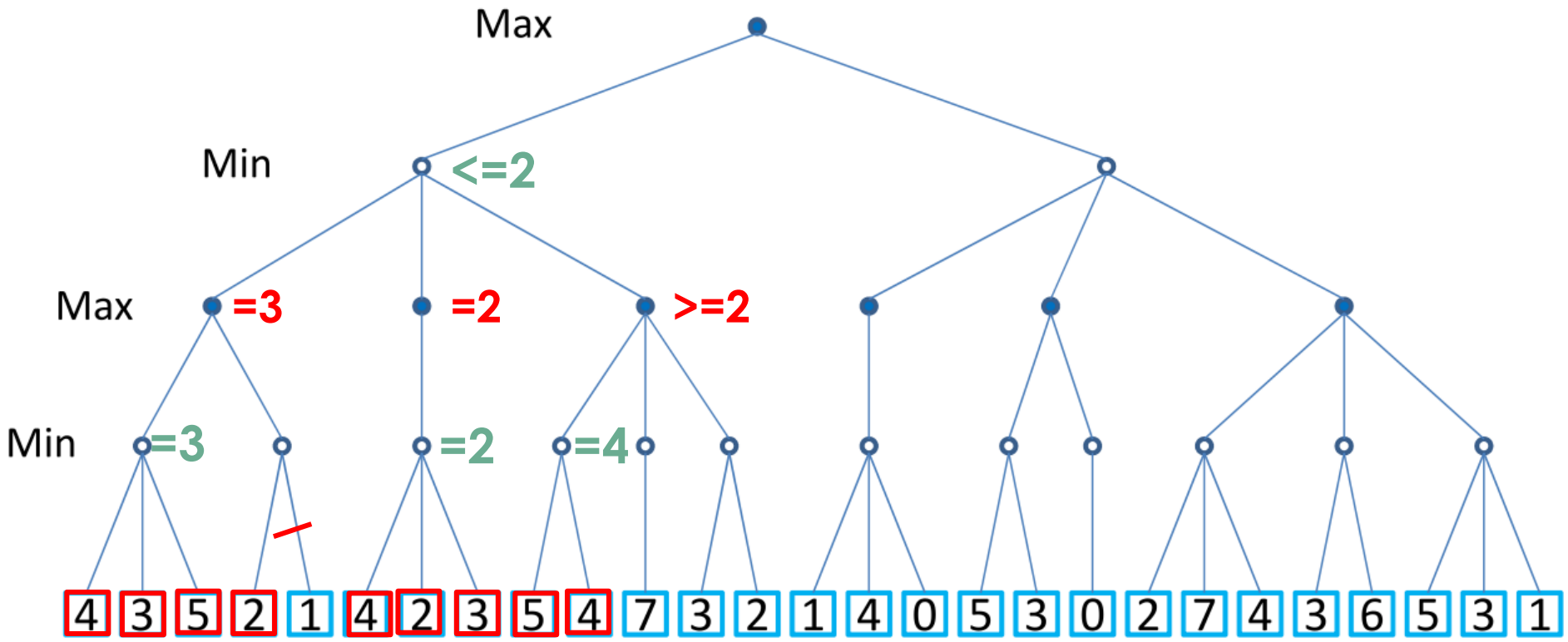
Alpha – Beta Pruning Example 3



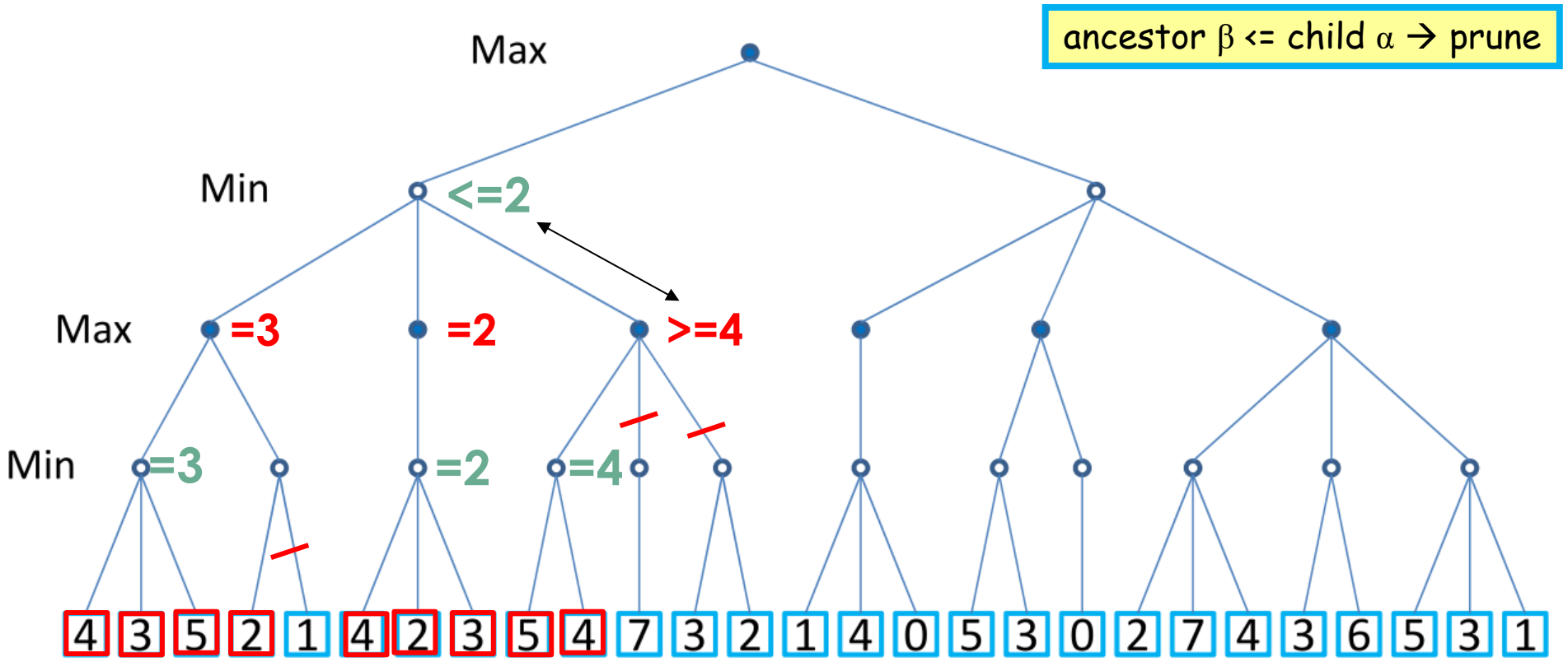
Alpha – Beta Pruning Example 3



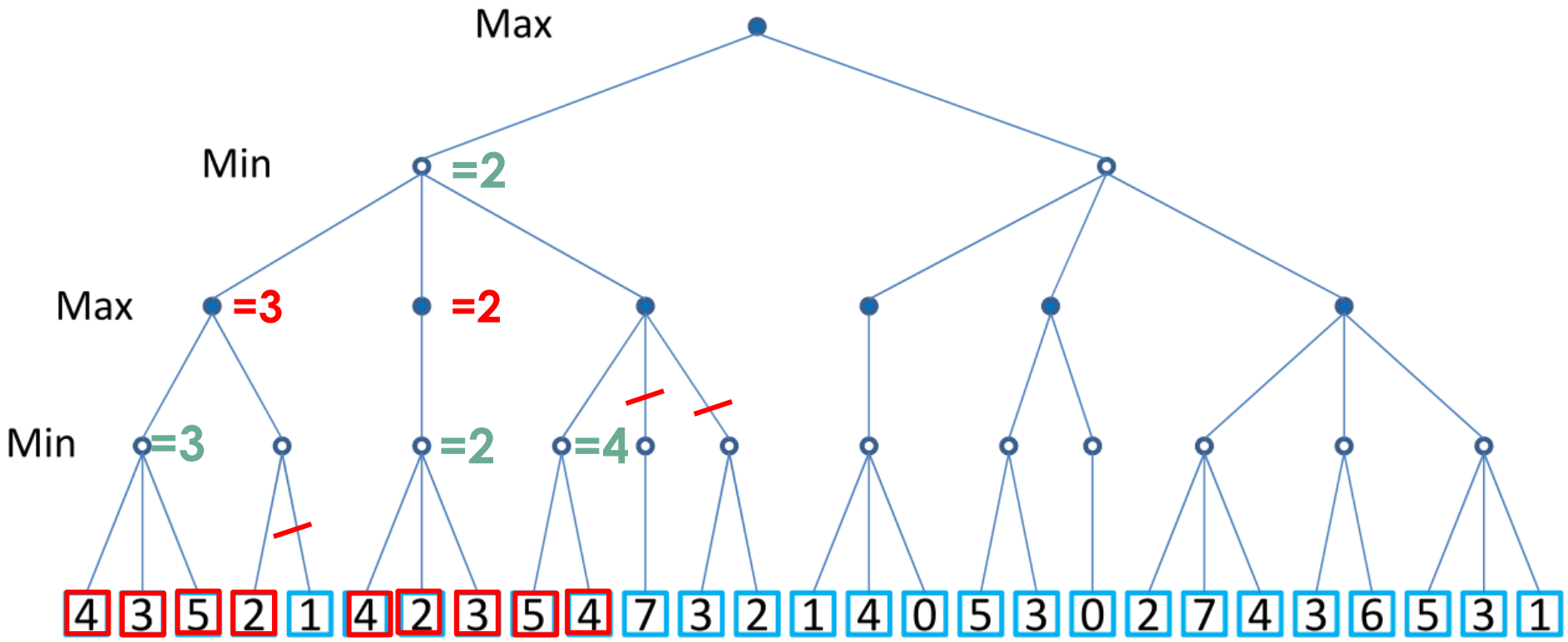
Alpha – Beta Pruning Example 3



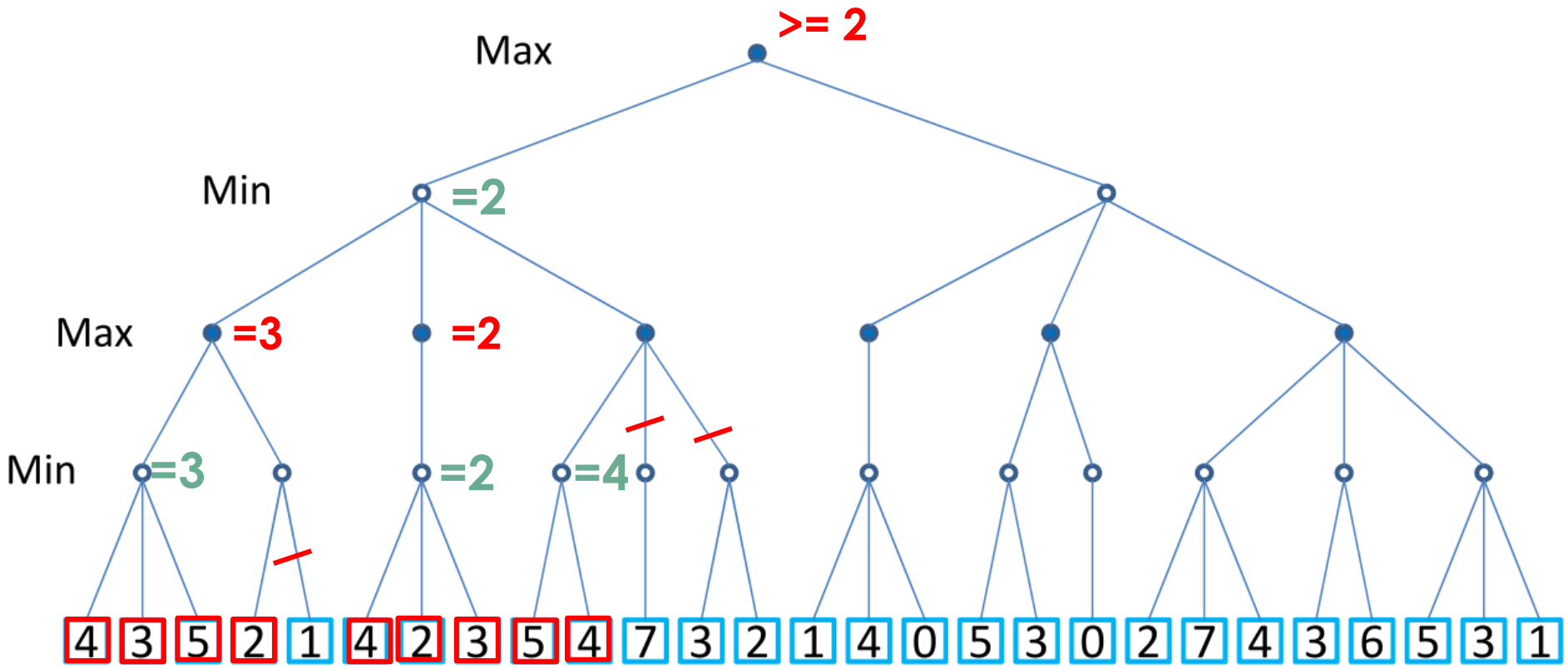
Alpha – Beta Pruning Example 3



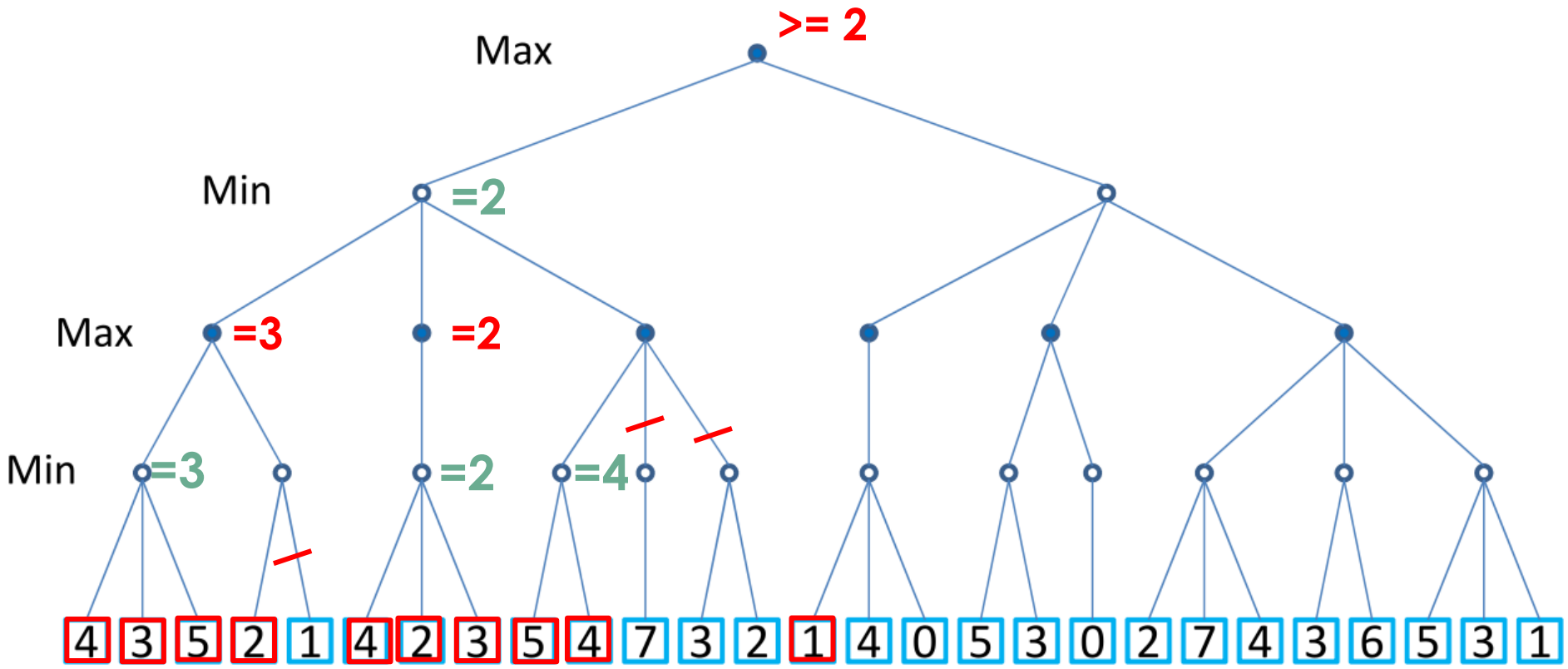
Alpha – Beta Pruning Example 3



Alpha – Beta Pruning Example 3

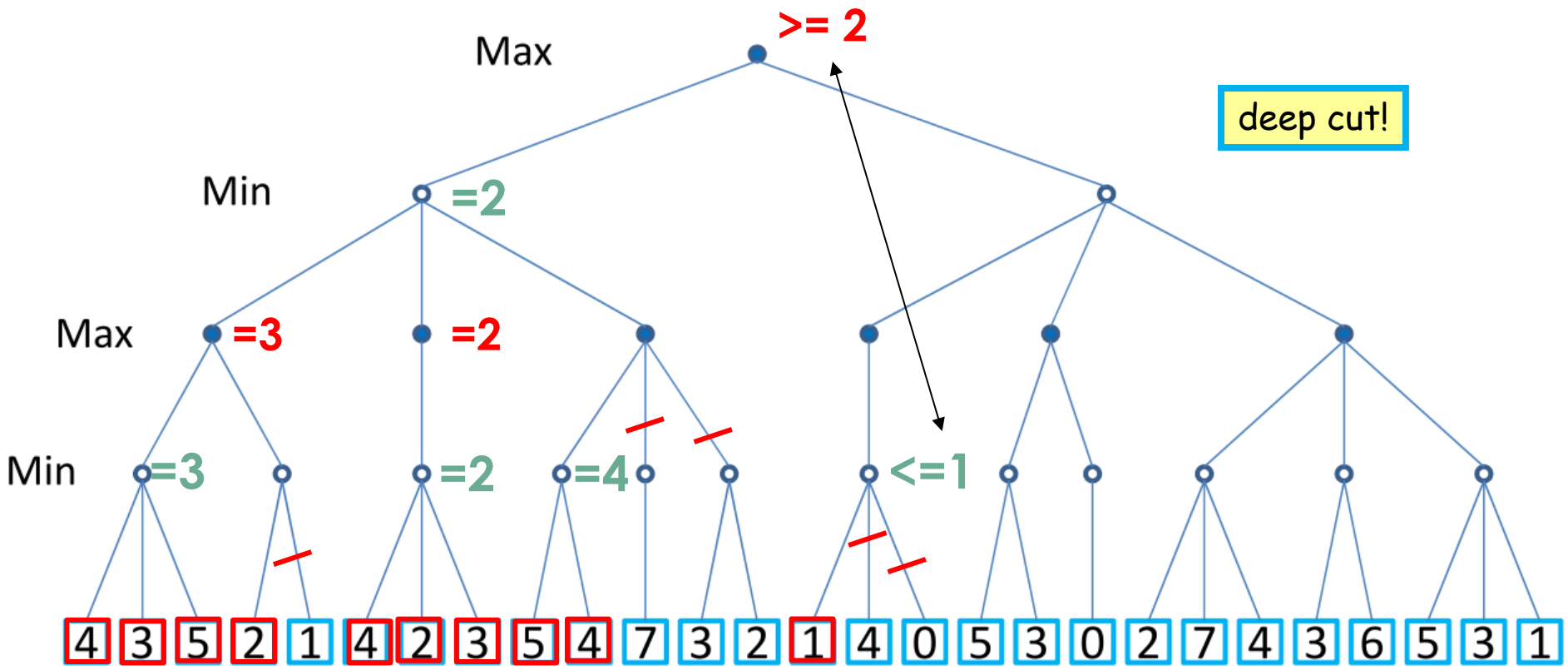


Alpha – Beta Pruning Example 3

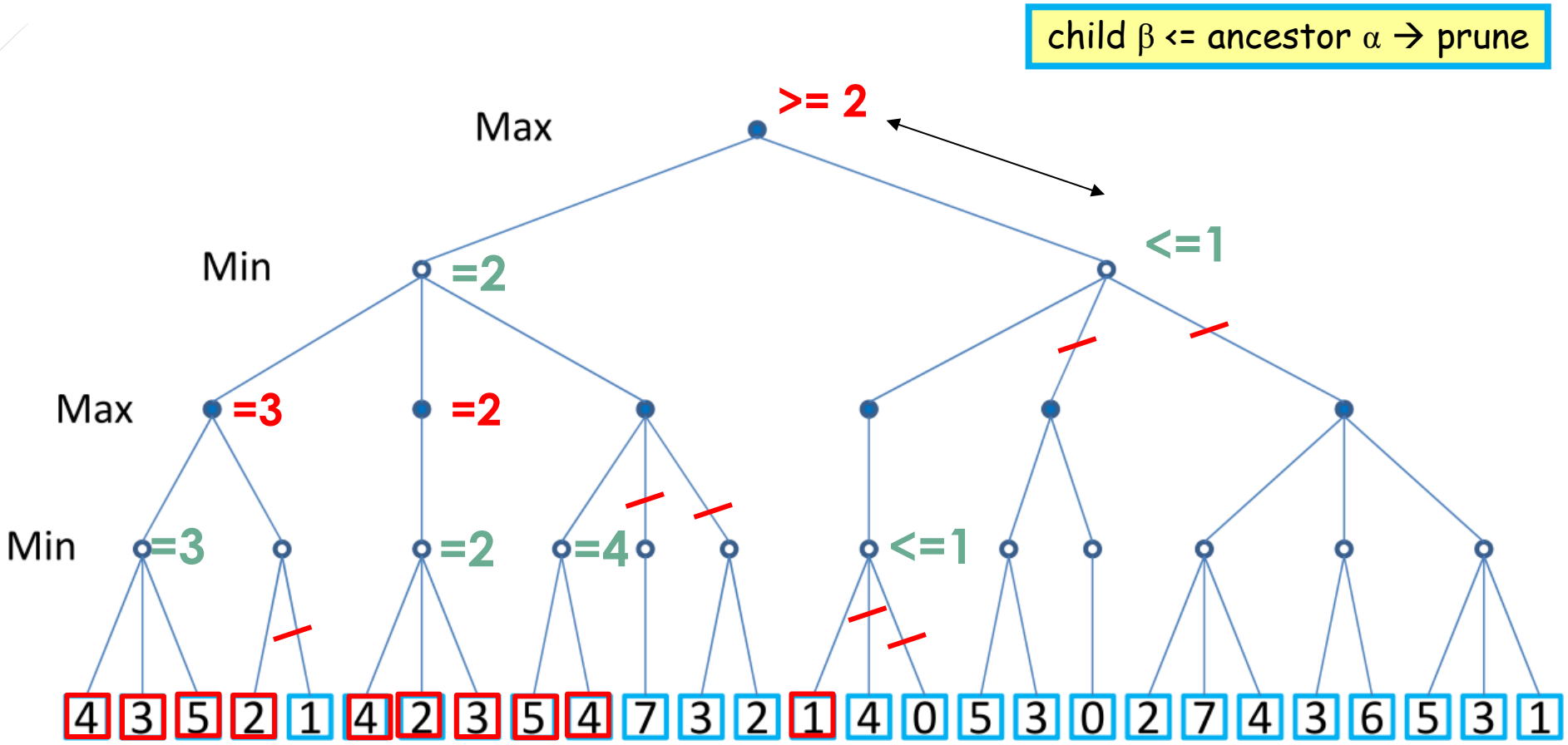


Alpha – Beta Pruning Example 3

child β <= ancestor α → prune

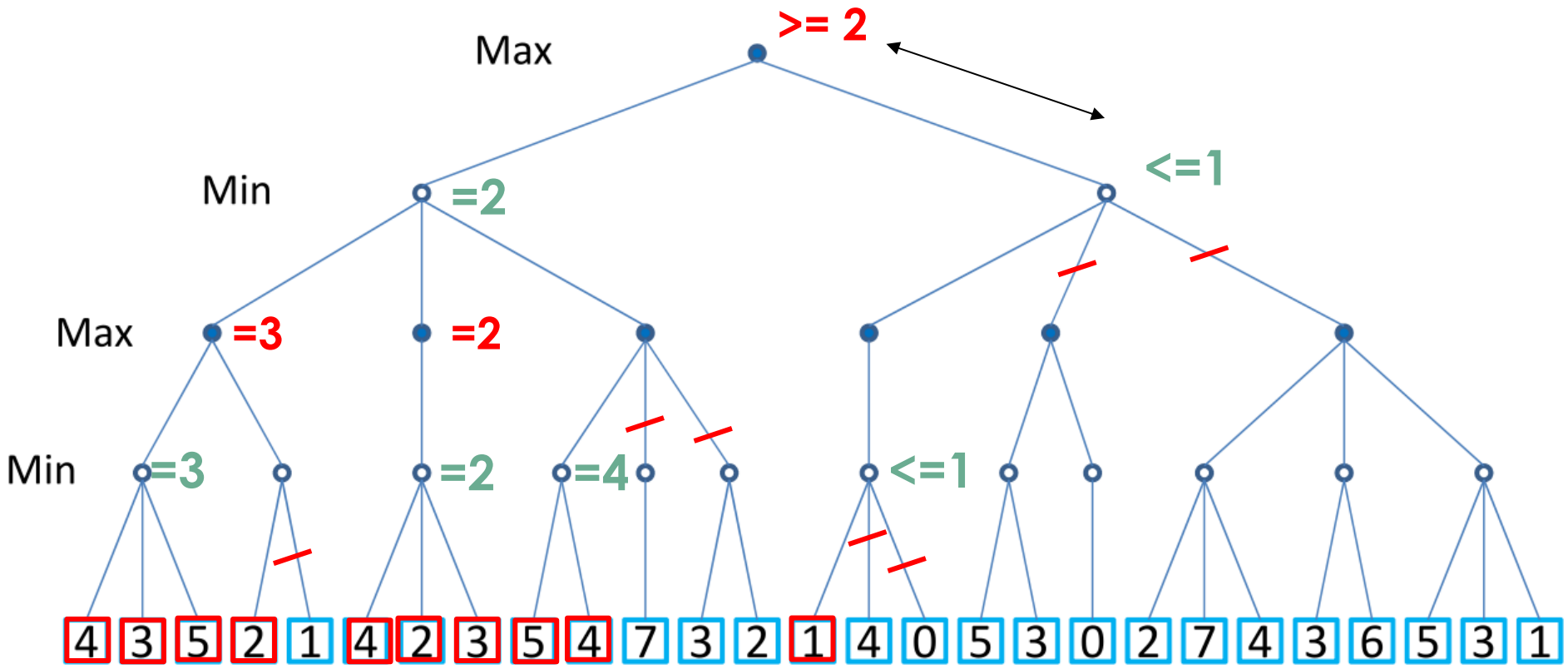


Alpha – Beta Pruning Example 3



Alpha – Beta Pruning Example 3

child β <= ancestor α → prune

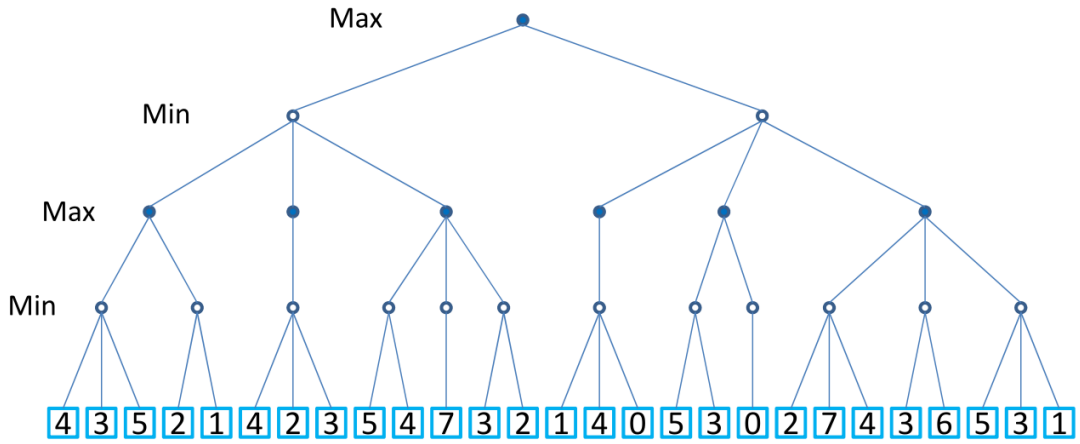


11 nodes explored out of 27

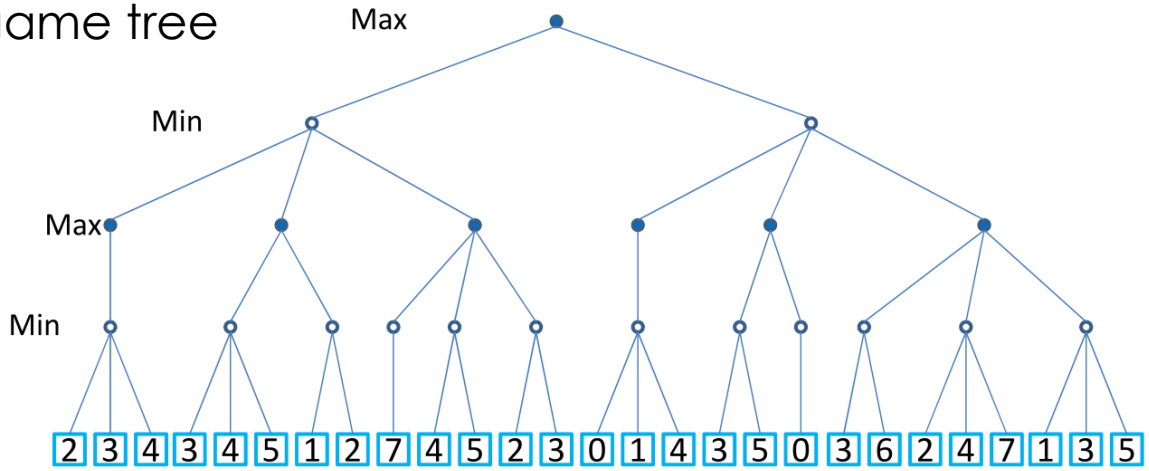
Efficiency

- Depends on the order the siblings
- In worst case:
  - alpha-beta provides no pruning
- In best case:
  - branching factor is reduced to its square root
  - so can search can go twice as deep with the same amount of computation

Alpha-Beta: Best ordering



Original (arbitrary) game tree

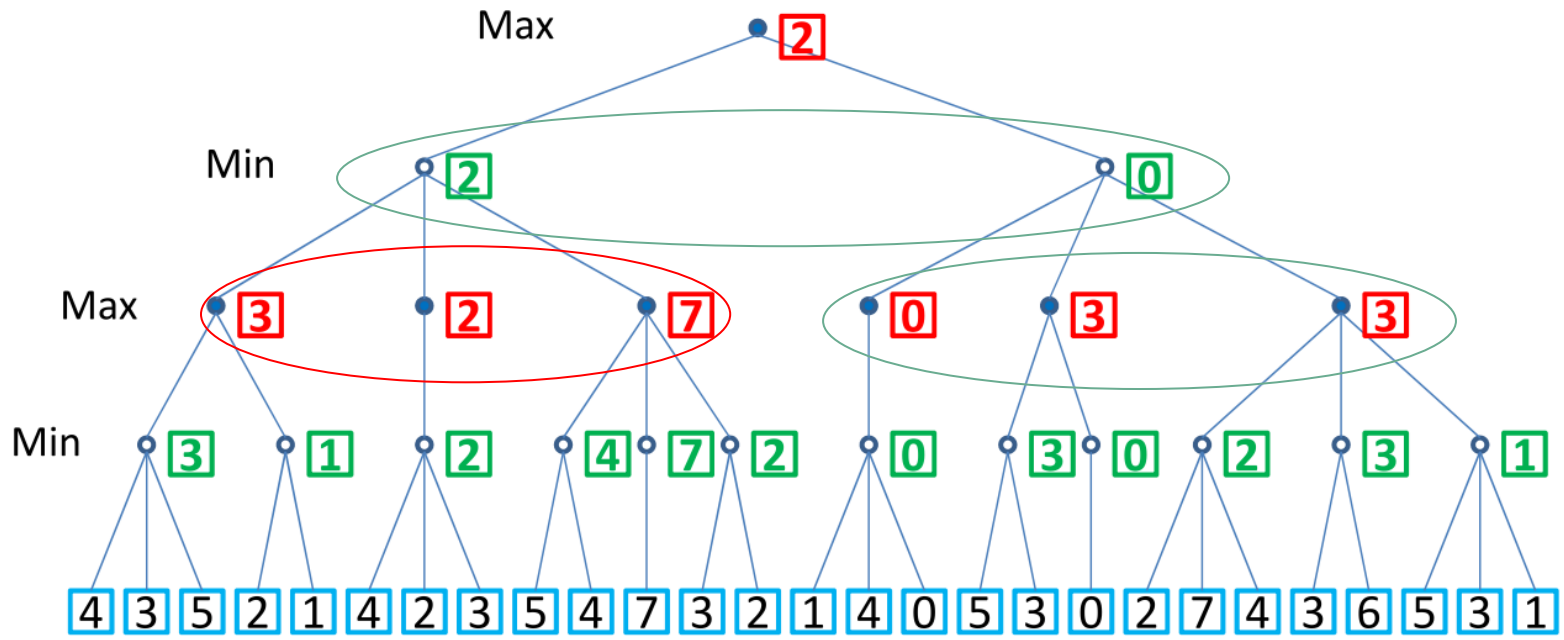


Best ordering for alpha-beta

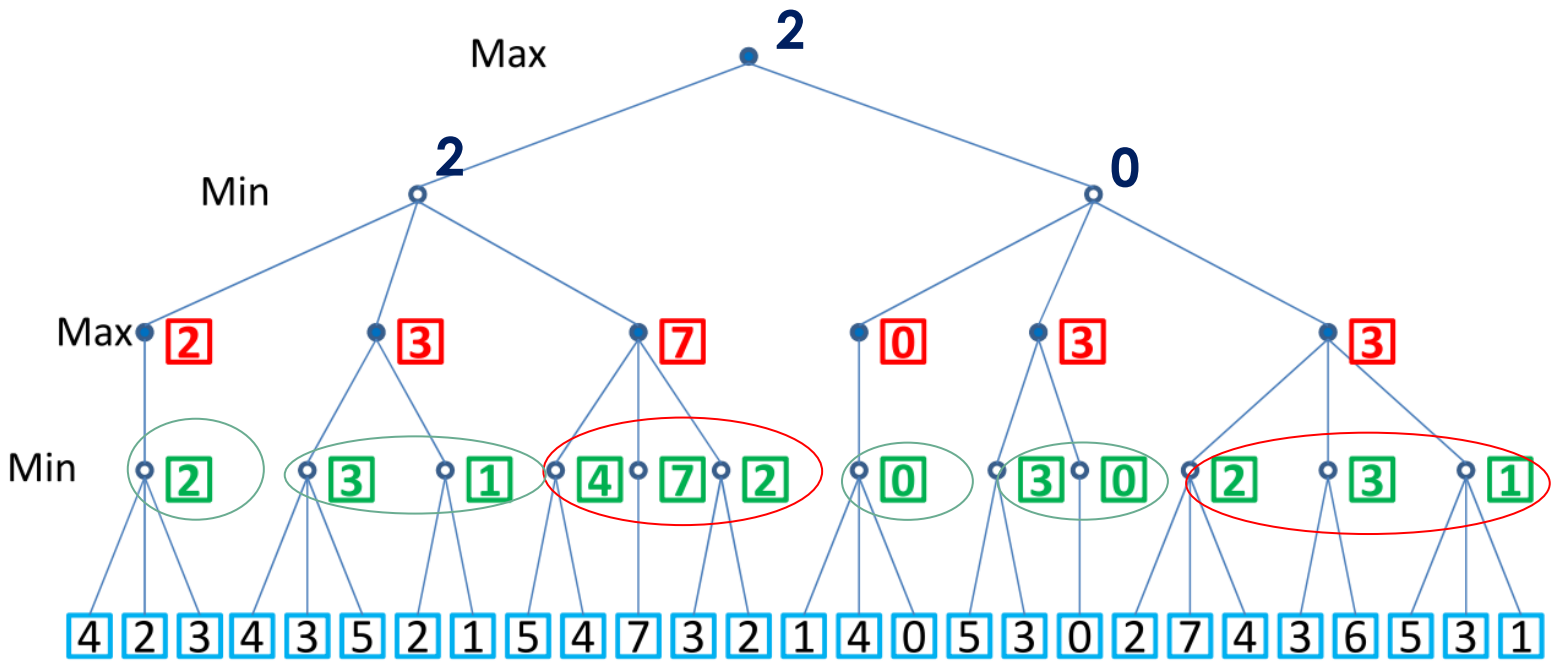
source: G. Luger (2005)

Alpha-Beta: Best ordering

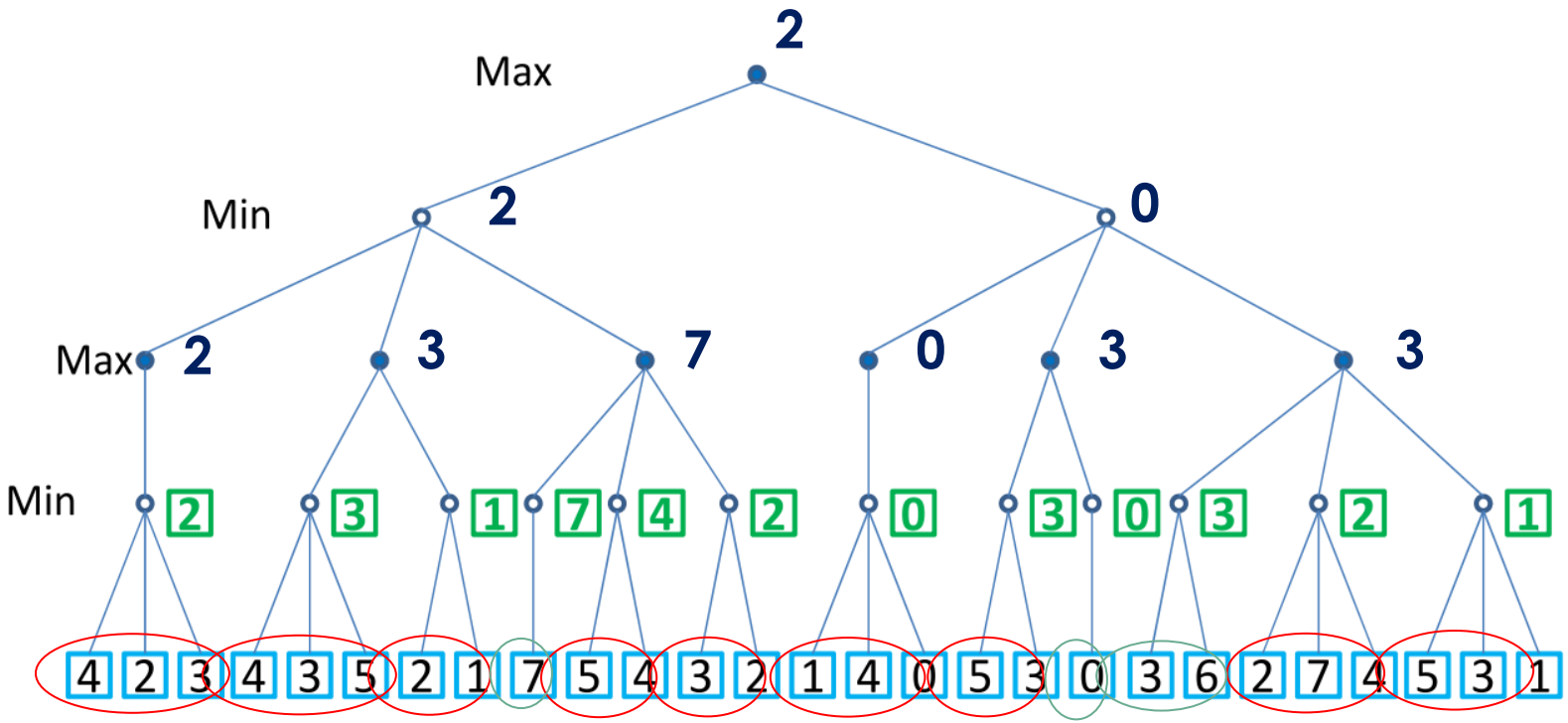
- best ordering:
  1. children of MIN : smallest node first
  2. children of MAX: largest node first



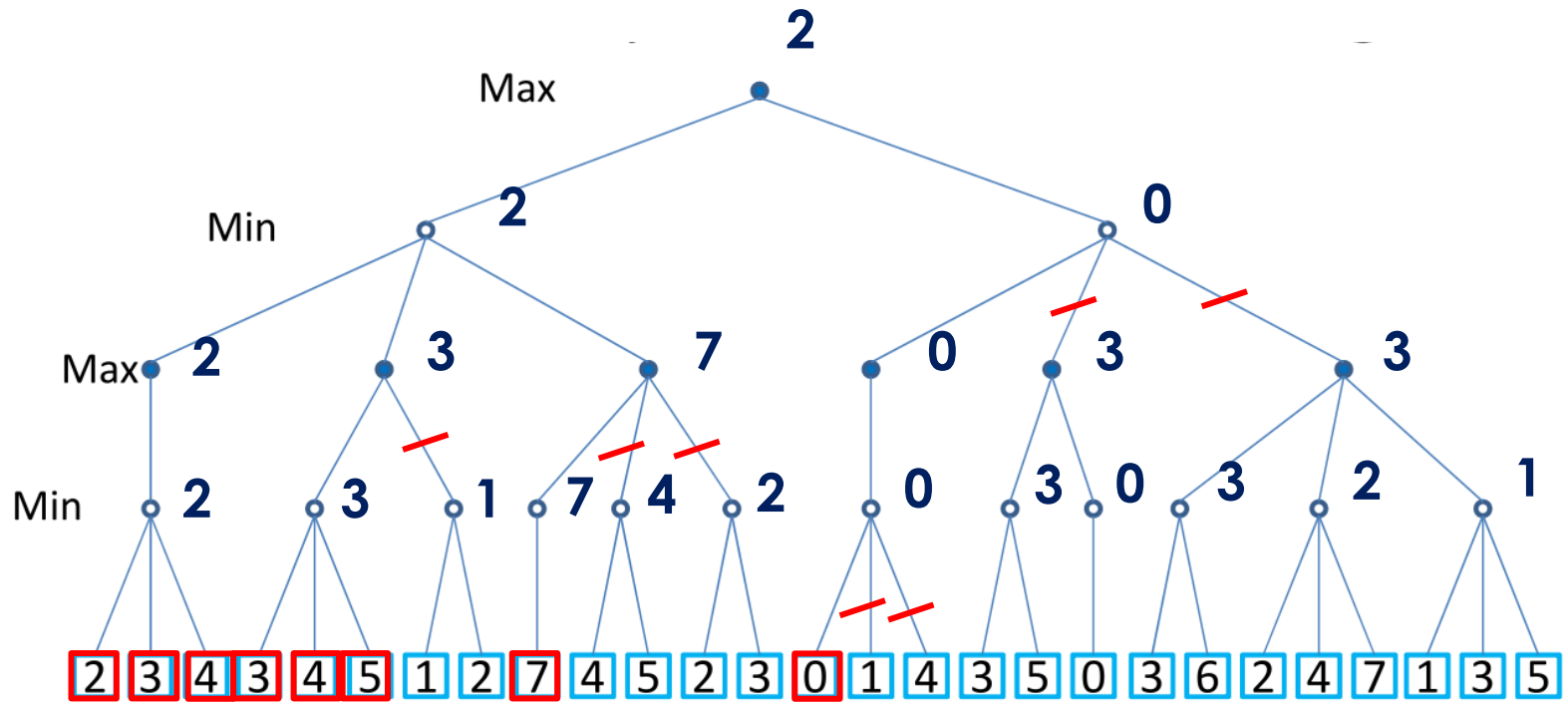
Alpha-Beta: Best ordering



Alpha-Beta: Best ordering



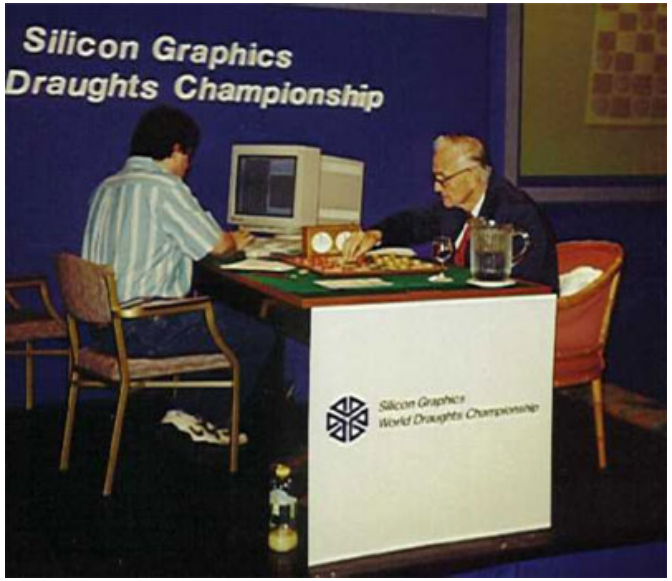
Alpha-Beta: Best ordering



8 nodes explored out of 27

Applications

- **1992 – 1994 Checkers: Tinsley VS Chinook**
- Marion Tinsley – World Champion for over 40 years
- Chinook – Developed by Jonathan Schaeffer, professor at University of Alberta
- In 1992, Tinsley beat Chinook in 4 games to 2, with 33 draws.
- In 1996: 6 draws
- In 2007, Schaeffer announced that checkers was solved, and anyone playing against Chinook would only be able to draw, never win.



https://webdocs.cs.ualberta.ca/~chinook/play/

Applications

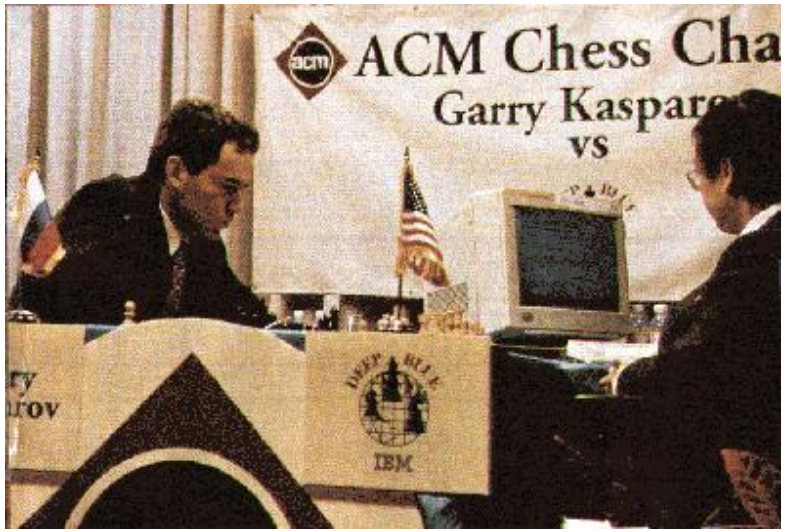
- **1997 Othello: Murakami VS Logistello**
- Takeshi Murakami – World Othello Champion
- Logistello – Developed by Michael Buro runs on a standard PC
- Logistello beat Murakami by 6 games to 0

https://skatgame.net/mburo/log.html

(including source code)

Applications

- **1997 Chess: Kasparov VS Deep Blue**
- Garry Kasparov – 50 billion neurons 2 positions/sec
- Deep Blue – 32 RISC (Reduced Instruction-Set Computer) processors + 256 VLSI (Very Large-Scale Integration) chess engines 200,000,000 pos/sec
- Deep Blue wins by 3 wins, 1 loss, and 2 draws



Applications

- **1997 Chess: Kasparov VS Deep Junior**
- Garry Kasparov – still 50 billion neurons 2 positions/sec
- Deep Junior – 8 CPU, 8 GB RAM, Win 2000 2,000,000 pos/sec Available at $100
- Match ends in a 3/3 tie!

Applications

- **2016 Go: AlphaGo VS Lee Se-dol**
- GO was always considered a much harder game to automate than chess because of its very high branching factor (35 for chess vs 250 for Go!)
- In 2016, AlphaGo beat Lee Sedol in a five-game match of GO.
- In 2017 AlphaGo beat Ke Jie, the world No.1 ranked player at the time

- uses a Monte Carlo tree search algorithm to find its moves based on knowledge previously "learned" by deep learning

The End