

Adversarial Search (Heuristics in Two-Person Games)

Dr. Mohamed Mostafa

AI 511 Foundations of Artificial Intelligence

Lecture 6



- Revision on uninformed Search

- Adversarial Search
(Heuristics in Two-Person Games)

- Minimax Procedure

- Exhaustive Minimax for Nim
(Splitting Stacks)
- Minimax to a Fixed Ply Depth

- Alpha-Beta Procedure

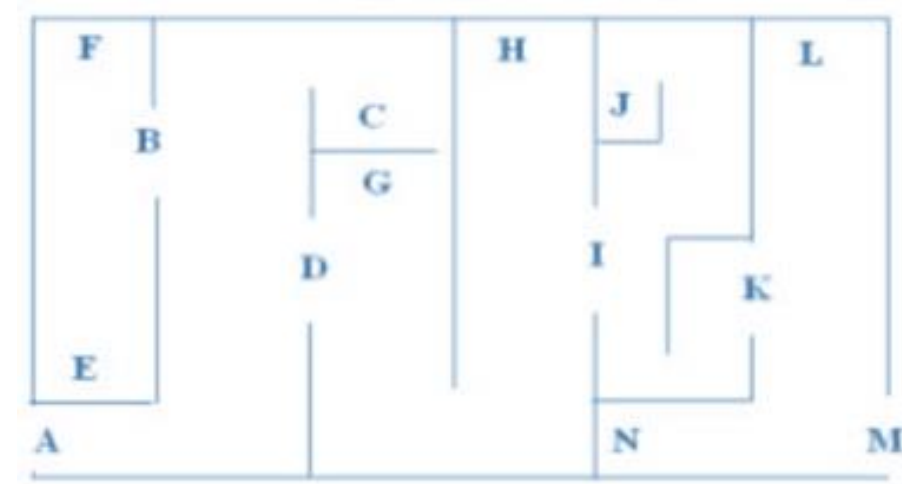
- Alpha-Beta Example

Resources for this
lecture

- **Chapter 3 (Structures & Strategies for State Space Search; sections 3.2, and 3.3) and Chapter 4 (Heuristic Search) from George F. Luger, "Artificial Intelligence: Structures and strategies for complex problem solving", " Sixth edition (2009), Pearson Education Limited..**

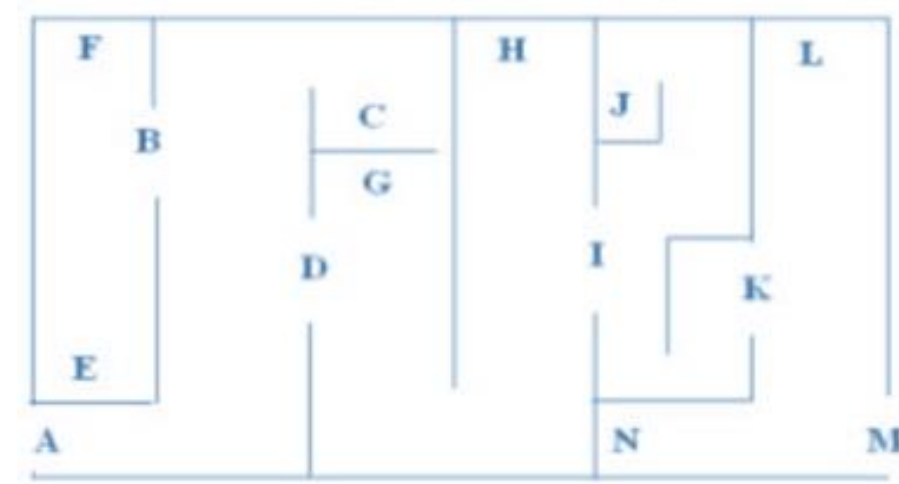
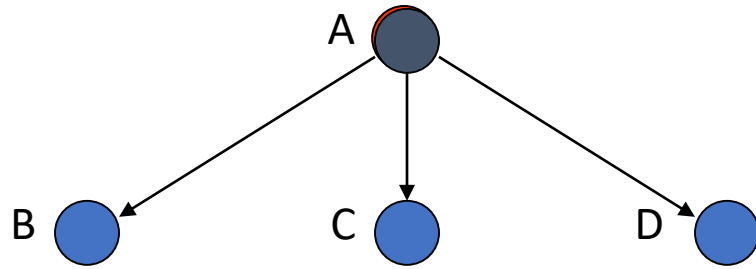
Breadth First Search for a Maze Problem

- Step 1: Create the graphical representation for the problem.



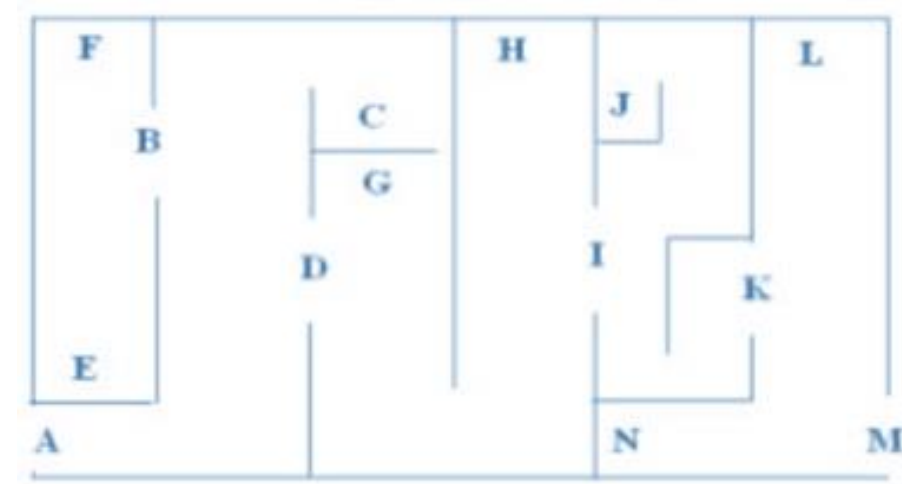
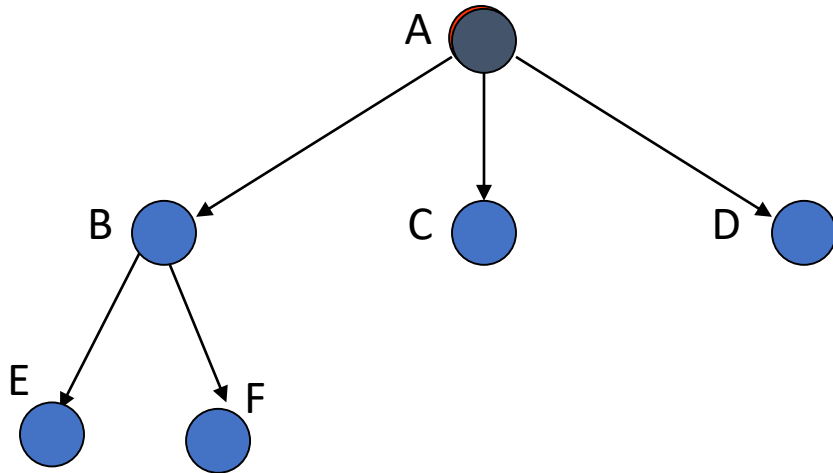
Breadth First Search for a Maze Problem

- Step 1: Create the graphical representation for the problem.



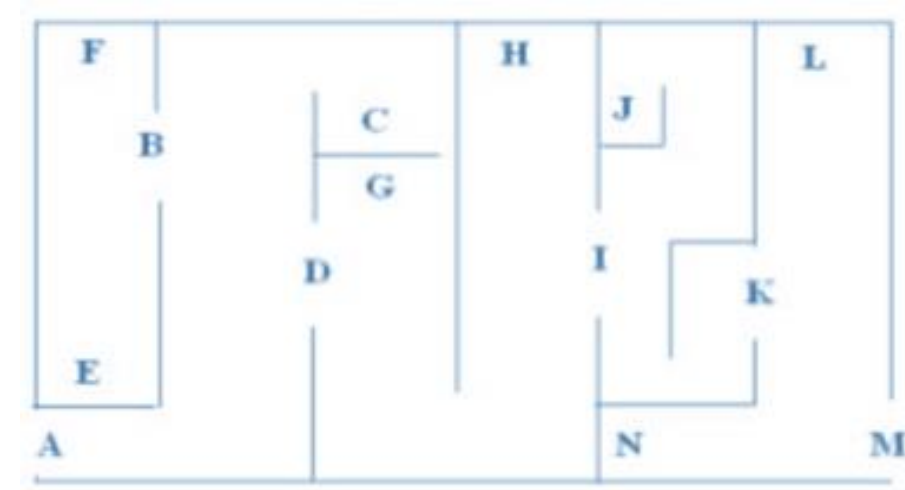
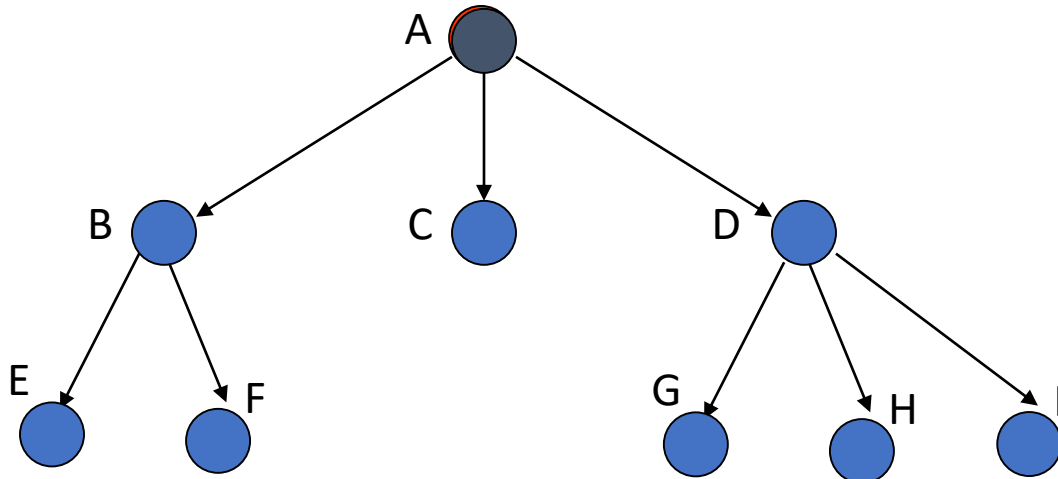
Breadth First Search for a Maze Problem

- Step 1: Create the graphical representation for the problem.



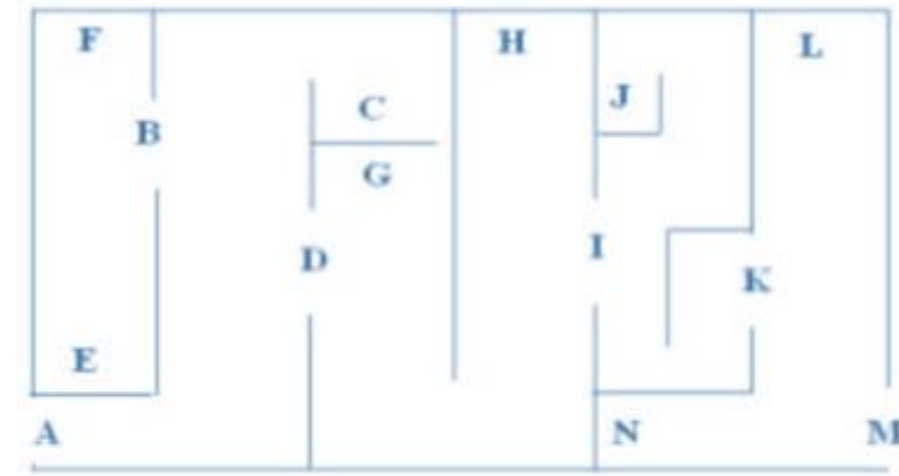
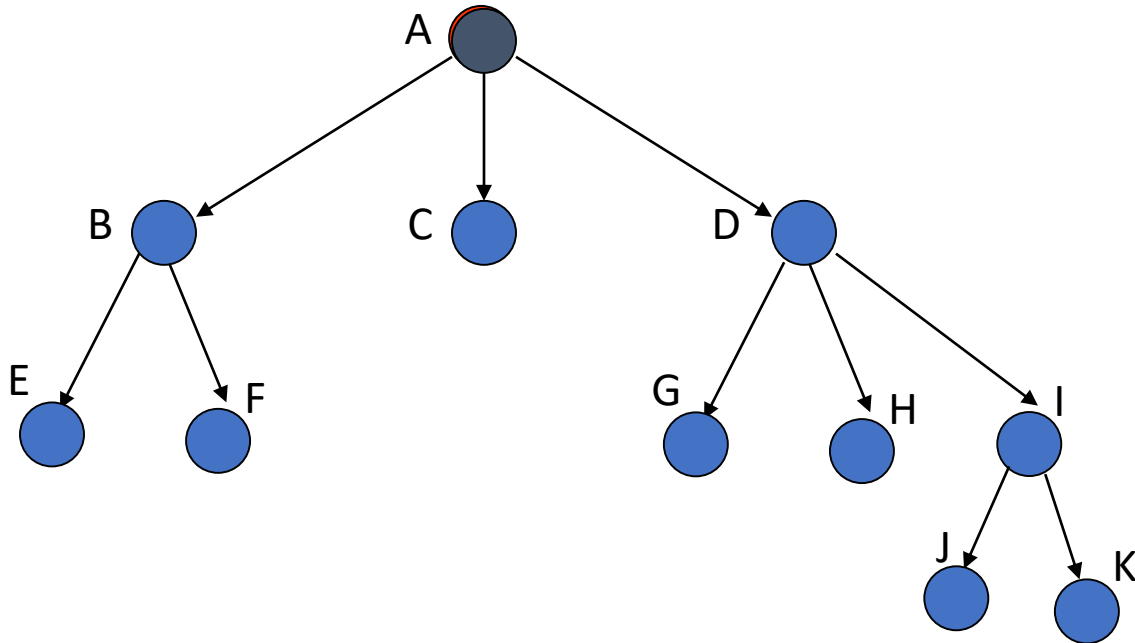
Breadth First Search for a Maze Problem

- Step 1: Create the graphical representation for the problem.



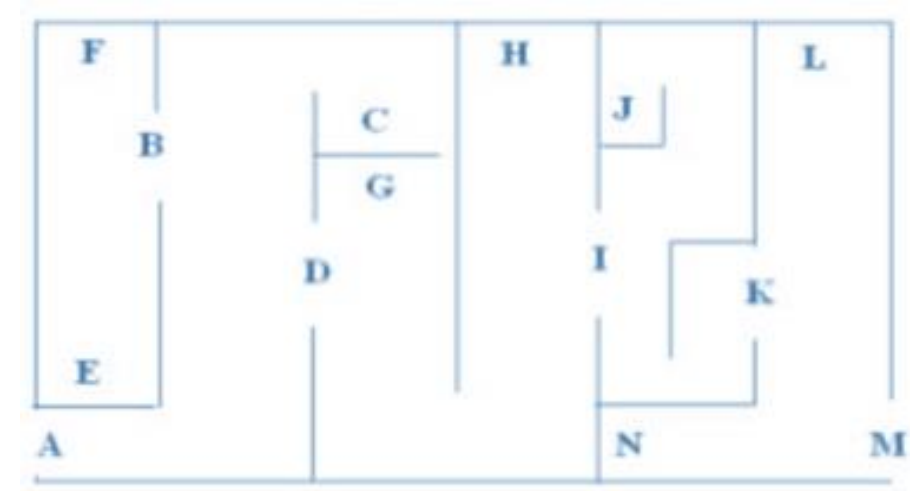
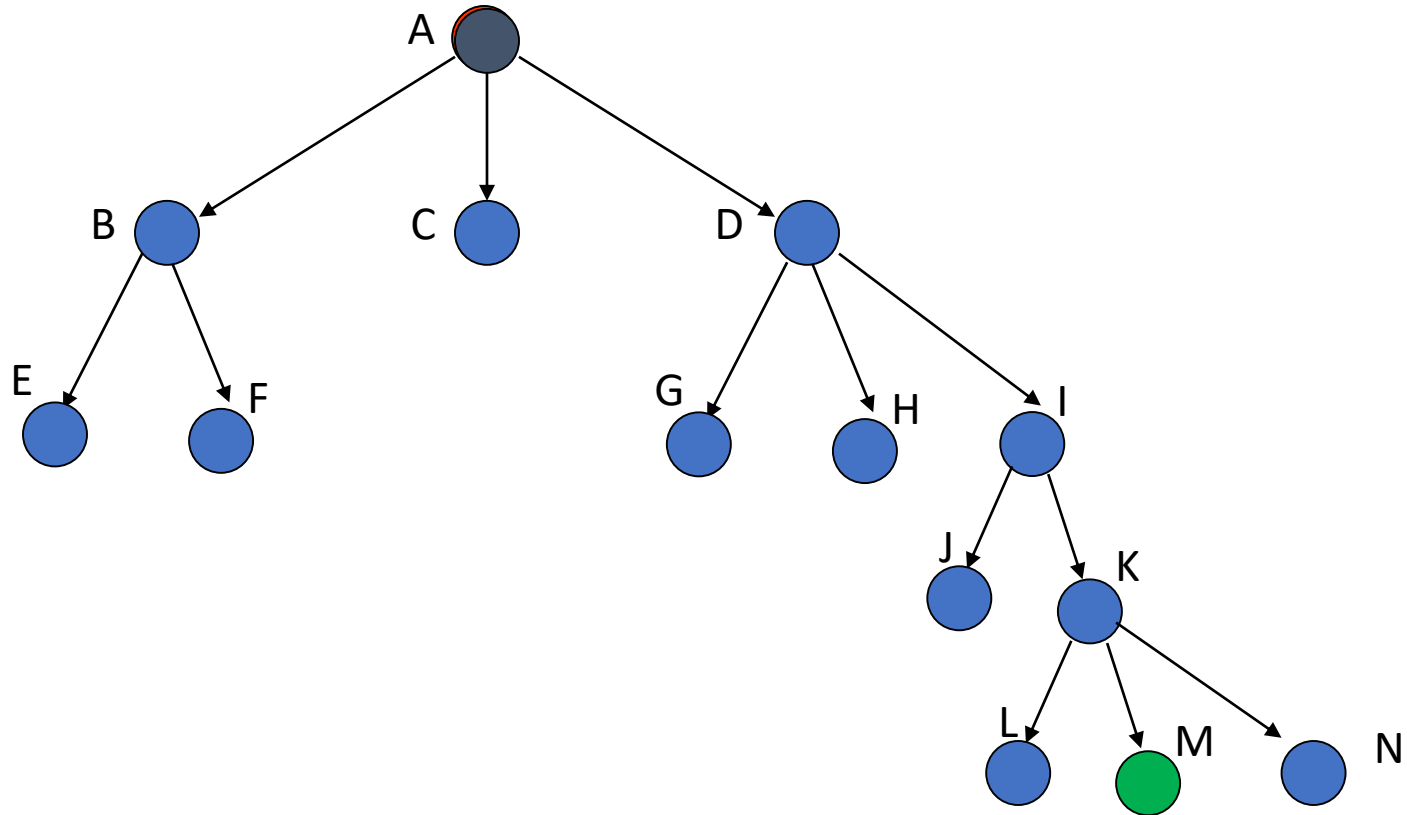
Breadth First Search for a Maze Problem

- Step 1: Create the graphical representation for the problem.



Breadth First Search for a Maze Problem

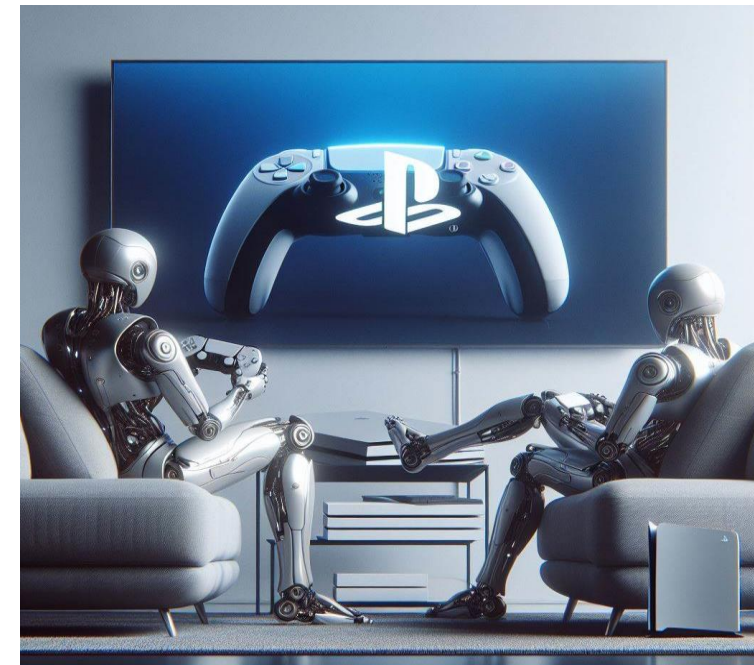
- Step 1: Create the graphical representation for the problem.



Adversarial Search Strategies

Adversarial Search Strategy

- It is particularly in two-person games like chess or tic-tac-toe, involves the development and utilization of heuristics to guide decision-making processes for players.
- In the context of two-person games, heuristics play a crucial role in determining which moves to make, especially when exhaustive search of all possible moves is not feasible due to the game's complexity; these heuristics aim to evaluate the current state of the game and estimate the likelihood of success for each possible move.
- Two **heuristic pruning** procedures:
 - Minimax Algorithm.
 - Alpha-beta Algorithm.



The Minimax Adventure

- **Meet Our Players:**

- **Max** and **Min** are our star players. They're like rival game characters, each trying to outsmart the other.
- **Max** wants to maximize their score, while **Min** aims to minimize Max's gains.

- **The Game Tree:**

- Imagine a tree with branches representing all possible moves in our game. Each leaf at the end of a branch holds a score.
- **Max** and **Min** will explore this tree together.

- **Max's Turn:**

- **Max** starts at the top of the tree. They look at all the branches and pick the one with the **highest** score. Max thinks, "I'm going big!"

- **Min's Turn:**

- **Min** follows **Max** down the chosen branch. **Min** examines all the branches and selects the one with the **lowest** score. Min grins, "I'll bring Max down!"

The Minimax Adventure

- **Back and Forth:**

- **Max** and **Min** take turns, climbing up and down the tree. **Max** always wants the biggest score, while **Min** tries to keep it small. It's like a seesaw battle!

- **Terminal Nodes:**

- Finally, they reach the leaves—the terminal nodes. These hold actual scores.
- **Max** and **Min** compare these scores and decide who wins that branch.

- **Backtrack:**

- Now, they backtrack. **Max** and **Min** remember their choices and adjust their strategies.
- They climb back up, recalculating scores as they go.

- **The Ultimate Decision:**

- At the very top, **Max** and **Min** reveal their final scores.
- **Max** celebrates if it's high, while **Min** smirks if it's low.
- And that's how the **Minimax Algorithm** works! 🎮♣️

The Minimax Adventure

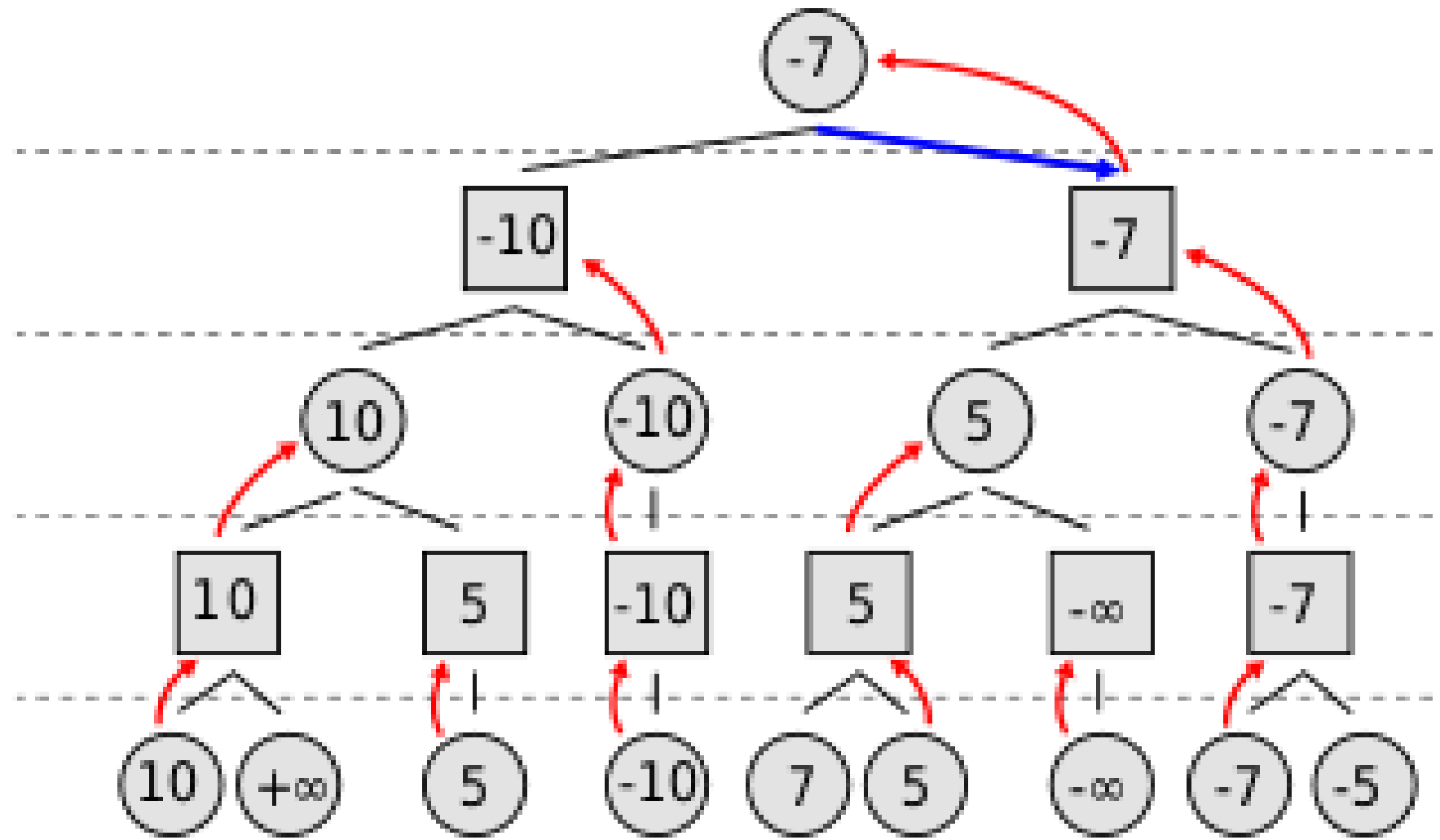
0 (max)

1 (min)

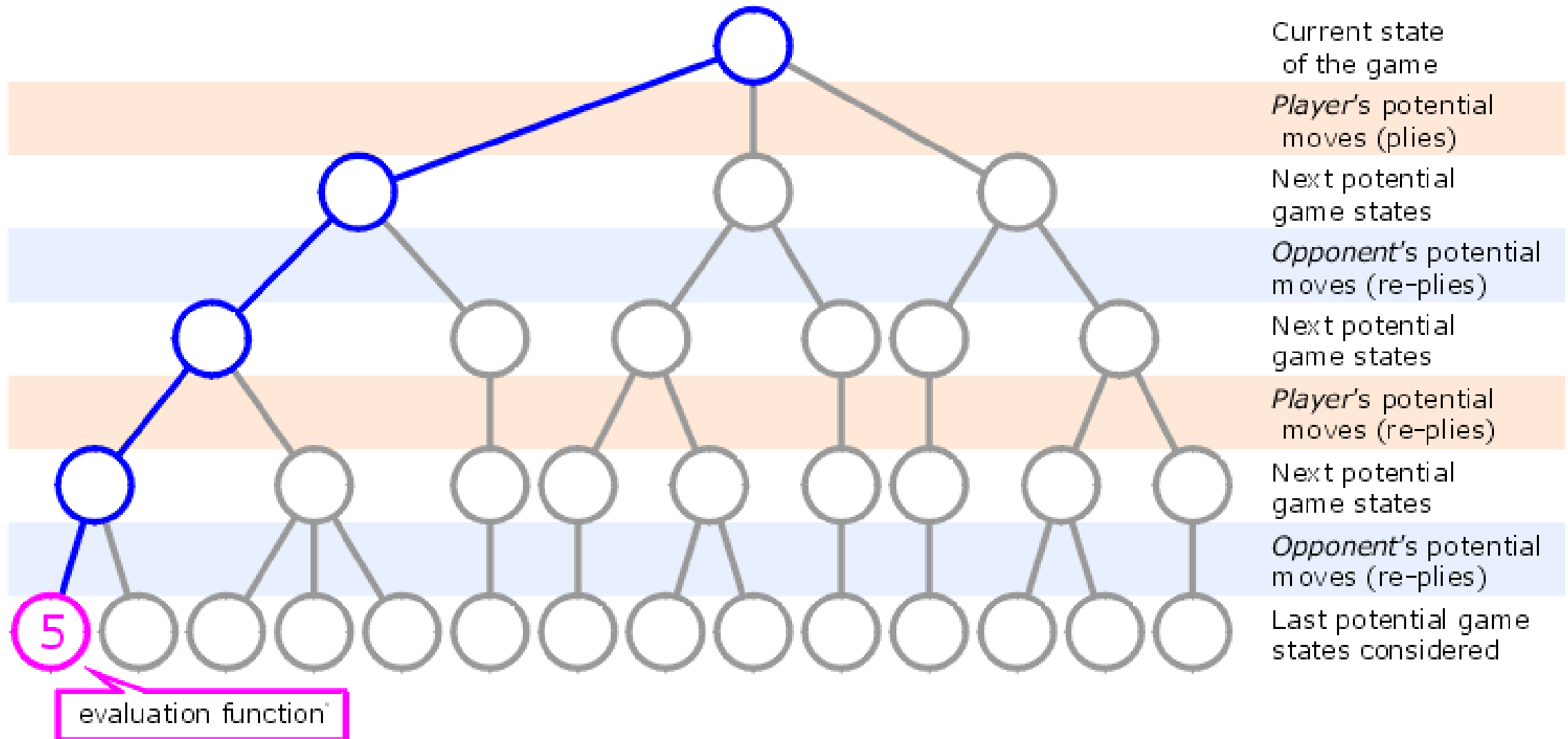
2 (max)

3 (min)

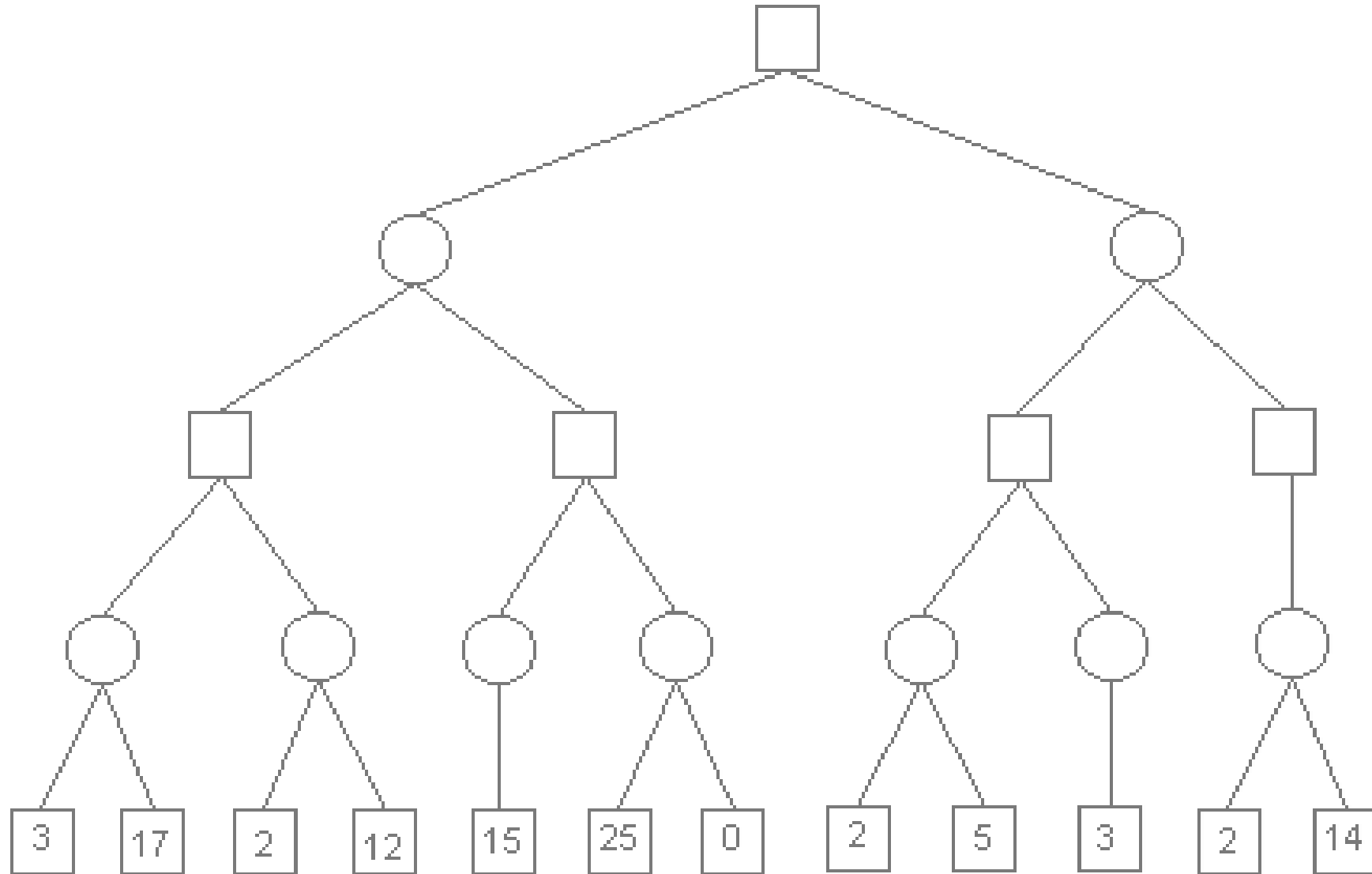
4 (max)



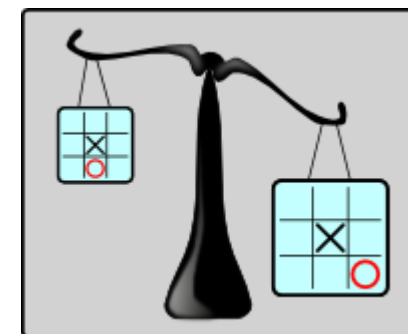
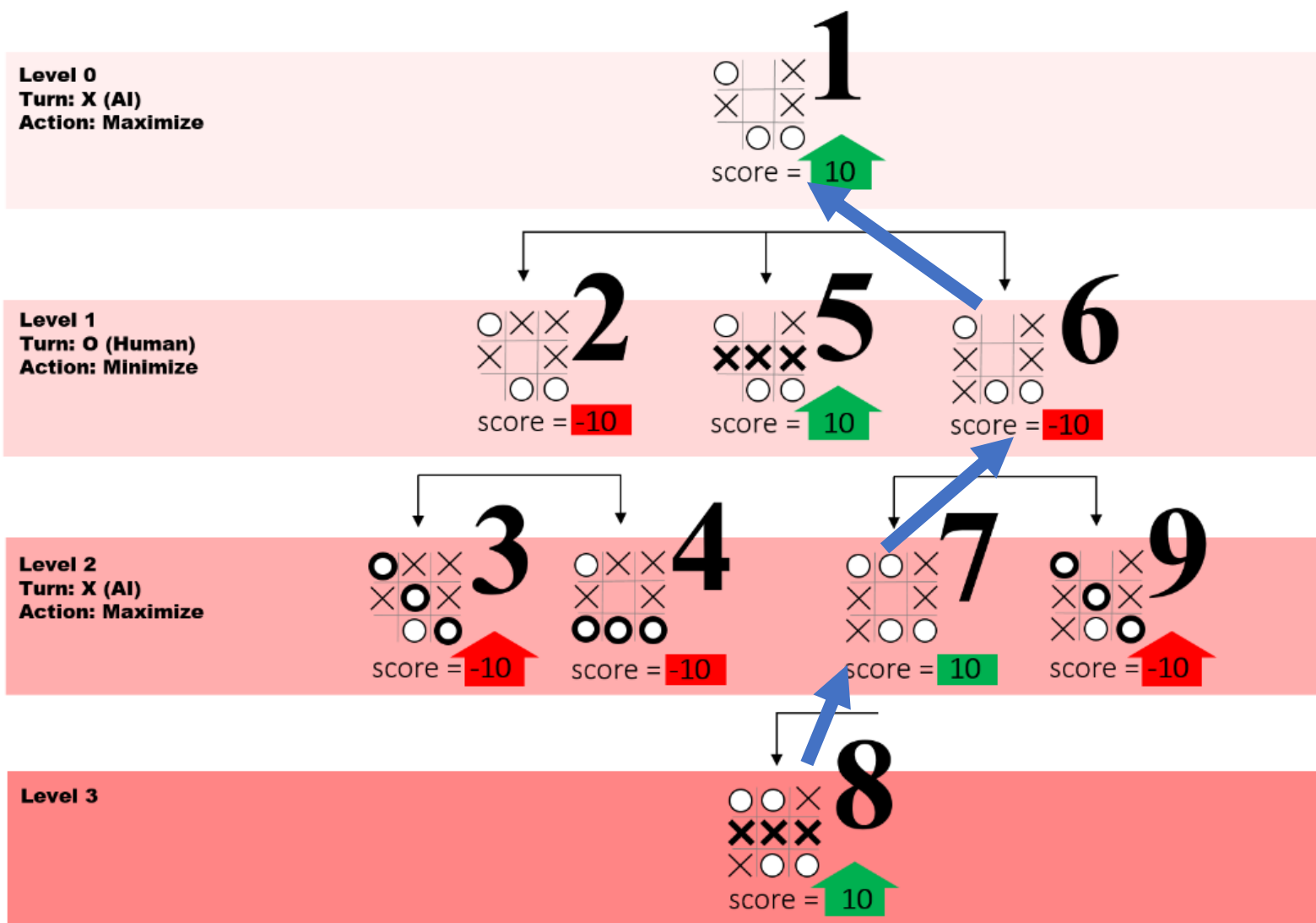
Minimax to a Fixed Depth



The Minimax Adventure



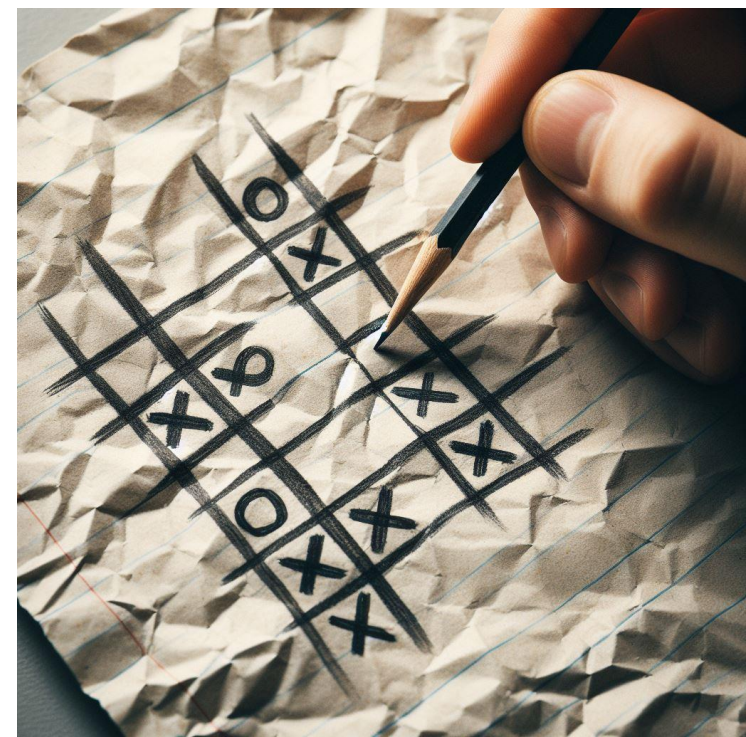
Tic Tac Toe Example



Some moves are better than others.

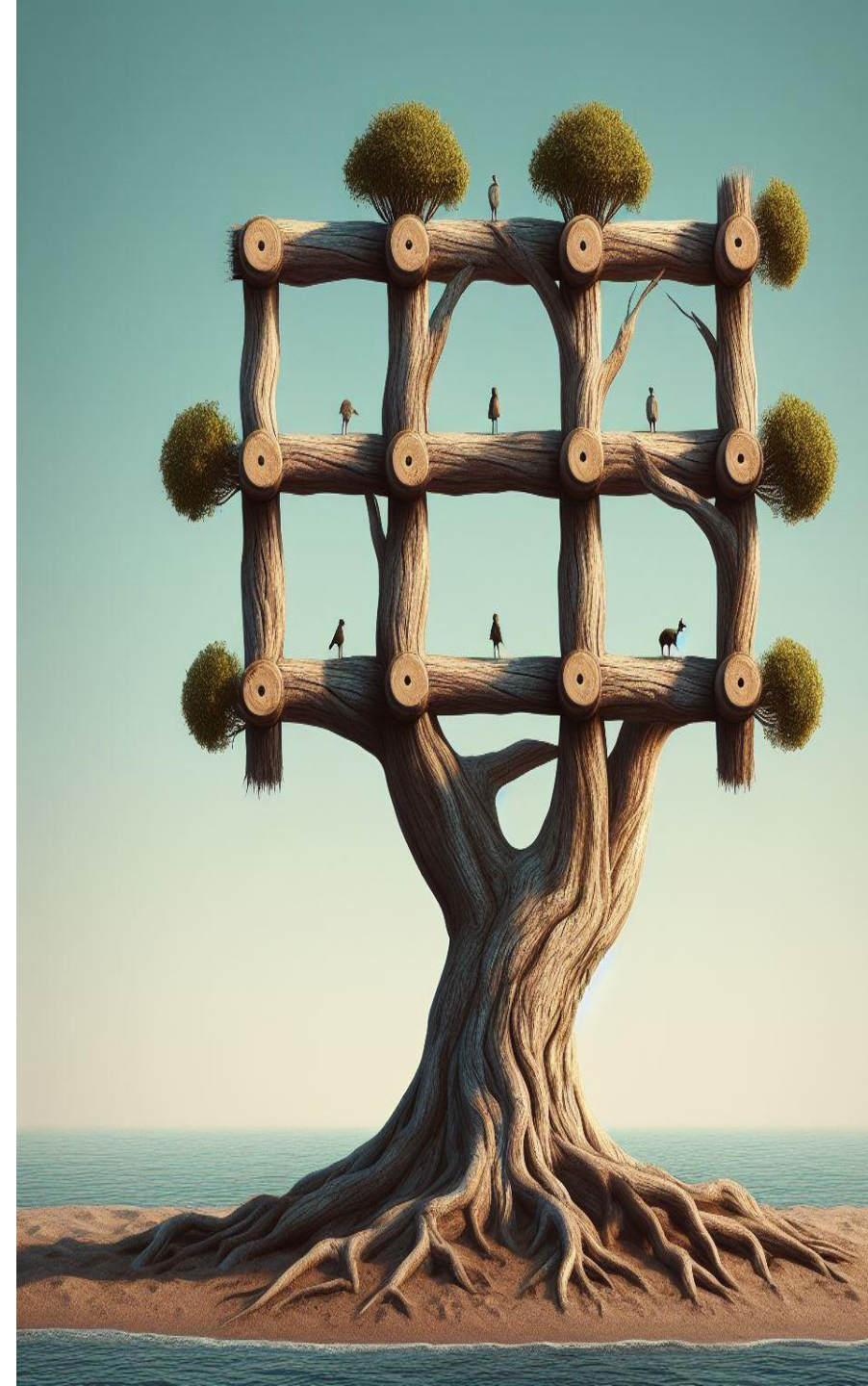
Alpha–Beta Pruning Algorithm

- **Alpha–beta pruning** is an optimization technique applied to the **minimax algorithm**. It aims to reduce the number of nodes evaluated during the search process in a game tree.
- This algorithm is commonly used for machine playing of **two-player combinatorial games** such as **Tic-tac-toe, Chess, Connect 4**, and more.



Alpha–Beta Pruning Algorithm

- **Core Idea**
- Imagine a game tree representing different possible situations in the game. Each node in the tree corresponds to a specific game state.
- At each node, the algorithm maintains two values:
 - **α** • **Alpha**: Represents the maximum score that the maximizing player is assured of.
 - **β** • **Beta**: Represents the minimum score that the minimizing player (usually the opponent) is assured of.
- Initially, **alpha** is set to negative infinity, and **beta** is set to positive infinity—both players start with their worst possible scores.
- As the search progresses, the algorithm prunes away branches that cannot significantly influence the final decision.



Alpha–Beta Pruning Algorithm

- **Initial Setup:**

- The Alpha-beta pruning algorithm begins with an initial call to the search function, typically starting from the root node of the game tree.
- It initializes two variables, alpha and beta, representing the best scores found so far for the maximizing player and the minimizing player, respectively. Initially, alpha is set to negative infinity, and beta is set to positive infinity.

- **Recursive Tree Traversal:**

- The algorithm proceeds with a depth-first search through the game tree, evaluating nodes recursively.
- At each level of the tree, the algorithm alternates between two types of nodes: maximizing nodes (belonging to the maximizing player) and minimizing nodes (belonging to the minimizing player).

- **Alpha Update (Maximizing Node): α**

- During the evaluation of a maximizing node, the algorithm updates the alpha value if it finds a score greater than the current alpha. This new alpha represents the best score found so far for the maximizing player on the path from the root to the current node.

- **Beta Update (Minimizing Node): β**

- Conversely, during the evaluation of a minimizing node, the algorithm updates the beta value if it finds a score lower than the current beta. This new beta represents the best score found so far for the minimizing player on the path from the root to the current node.

Alpha–Beta Pruning Algorithm

- **Pruning Condition:**

- At each level of the tree, if the algorithm encounters a situation where the alpha value of a maximizing node becomes greater than or equal to the beta value of a minimizing node, it indicates that the current branch can be pruned.

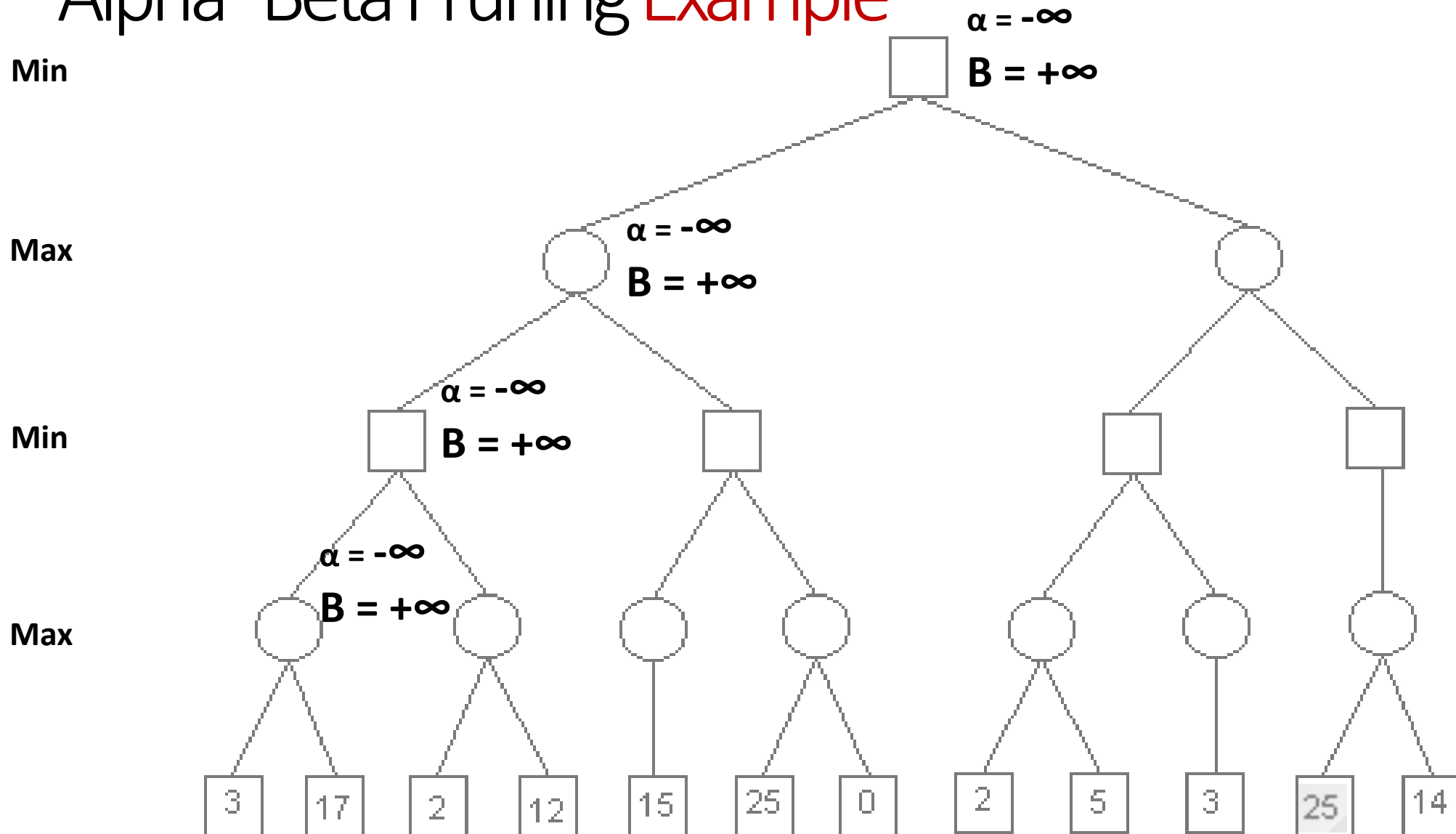
$$\alpha \geq \beta$$

- This condition implies that the minimizing player (or the opponent) will never choose this path because there exists a better alternative for them elsewhere in the tree. Hence, further exploration of this branch is unnecessary.

- **Termination and Decision-making:**

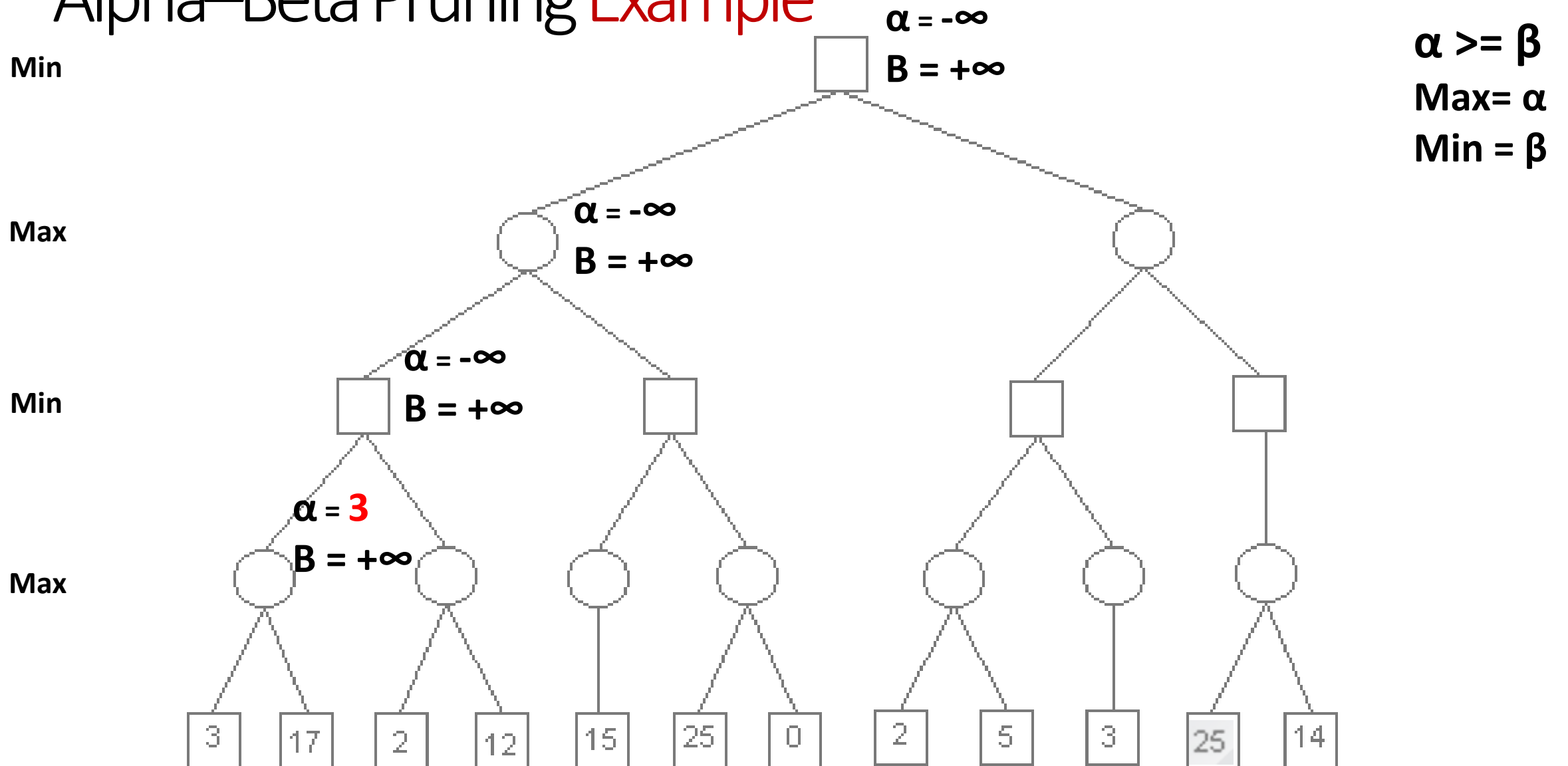
- The algorithm continues to traverse the tree recursively, updating alpha and beta values and pruning branches as necessary.
- Once the search reaches a terminal state or a specified depth limit, the algorithm returns the score associated with that state.
- Finally, the optimal move is determined based on the highest score obtained for the maximizing player at the root level.

Alpha-Beta Pruning Example

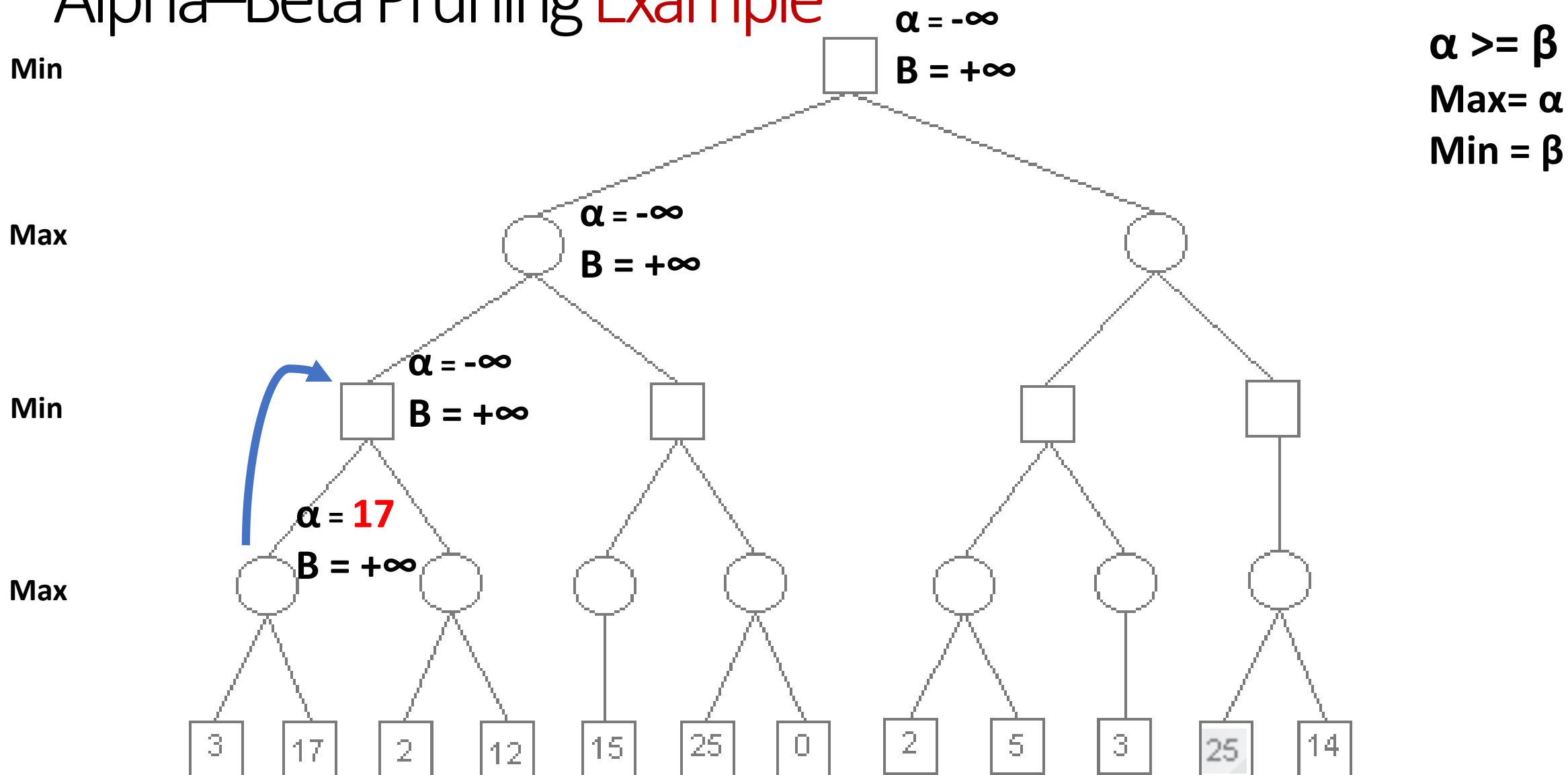


$\alpha \geq \beta$
Max = α
Min = β

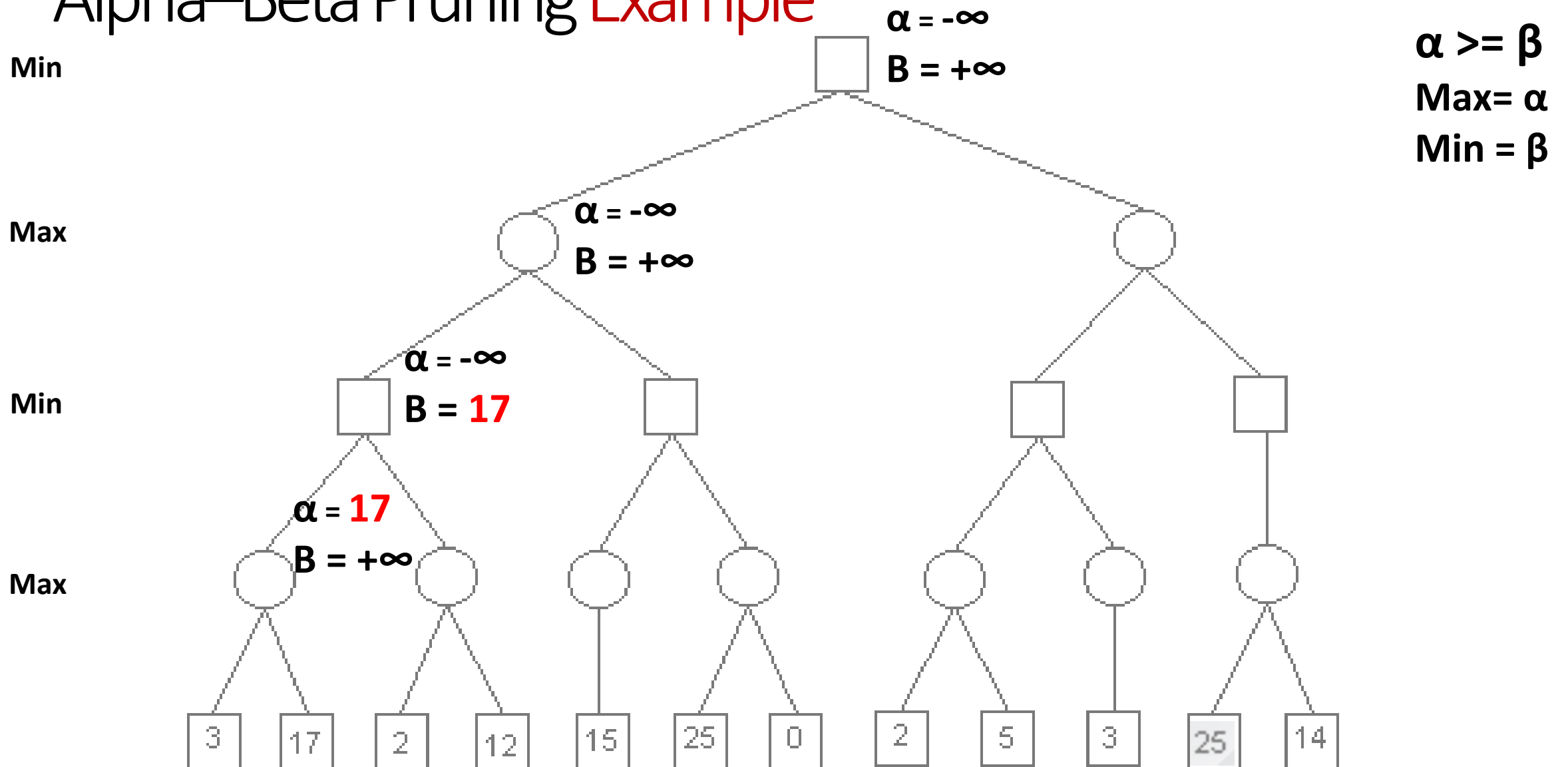
Alpha-Beta Pruning Example



Alpha-Beta Pruning Example



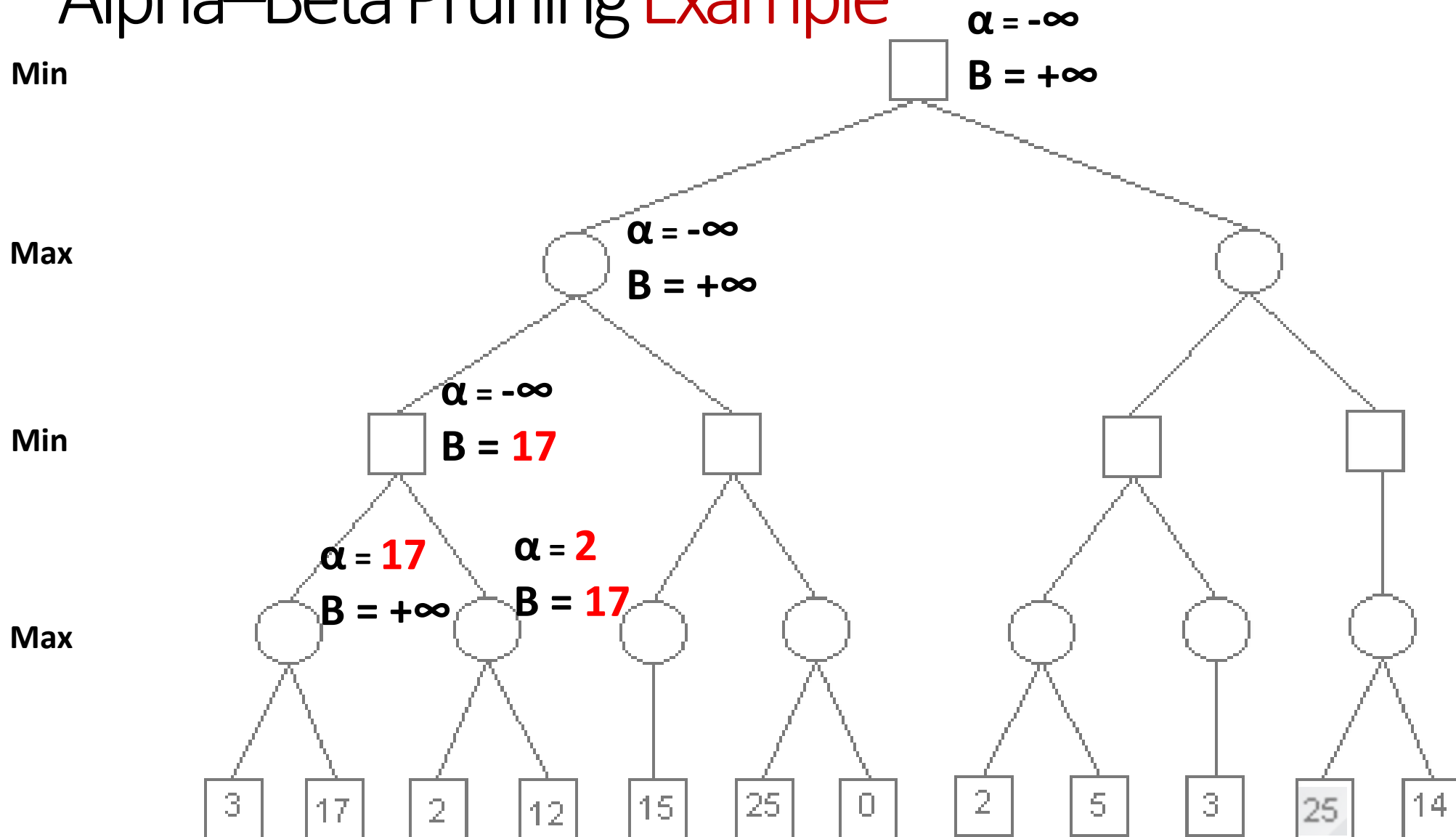
Alpha-Beta Pruning Example



$$\alpha = -\infty$$
$$\beta = +\infty$$
$$\alpha = -\infty$$
$$B = +\infty$$
$$\alpha = -\infty$$
$$B = 17$$
$$\alpha = 17$$
$$\beta = +\infty$$
$$\alpha = -\infty$$
$$B = 17$$

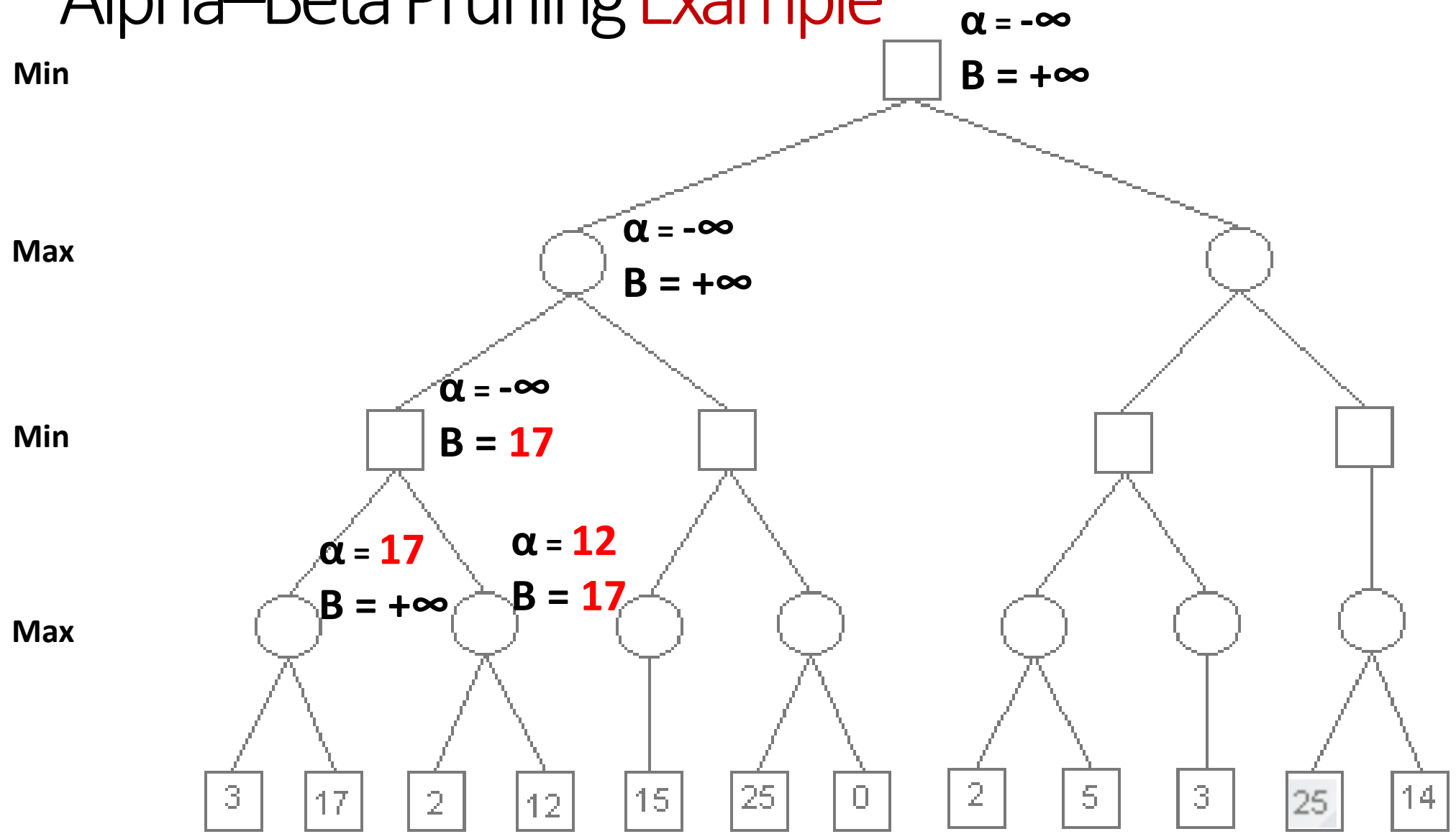
$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



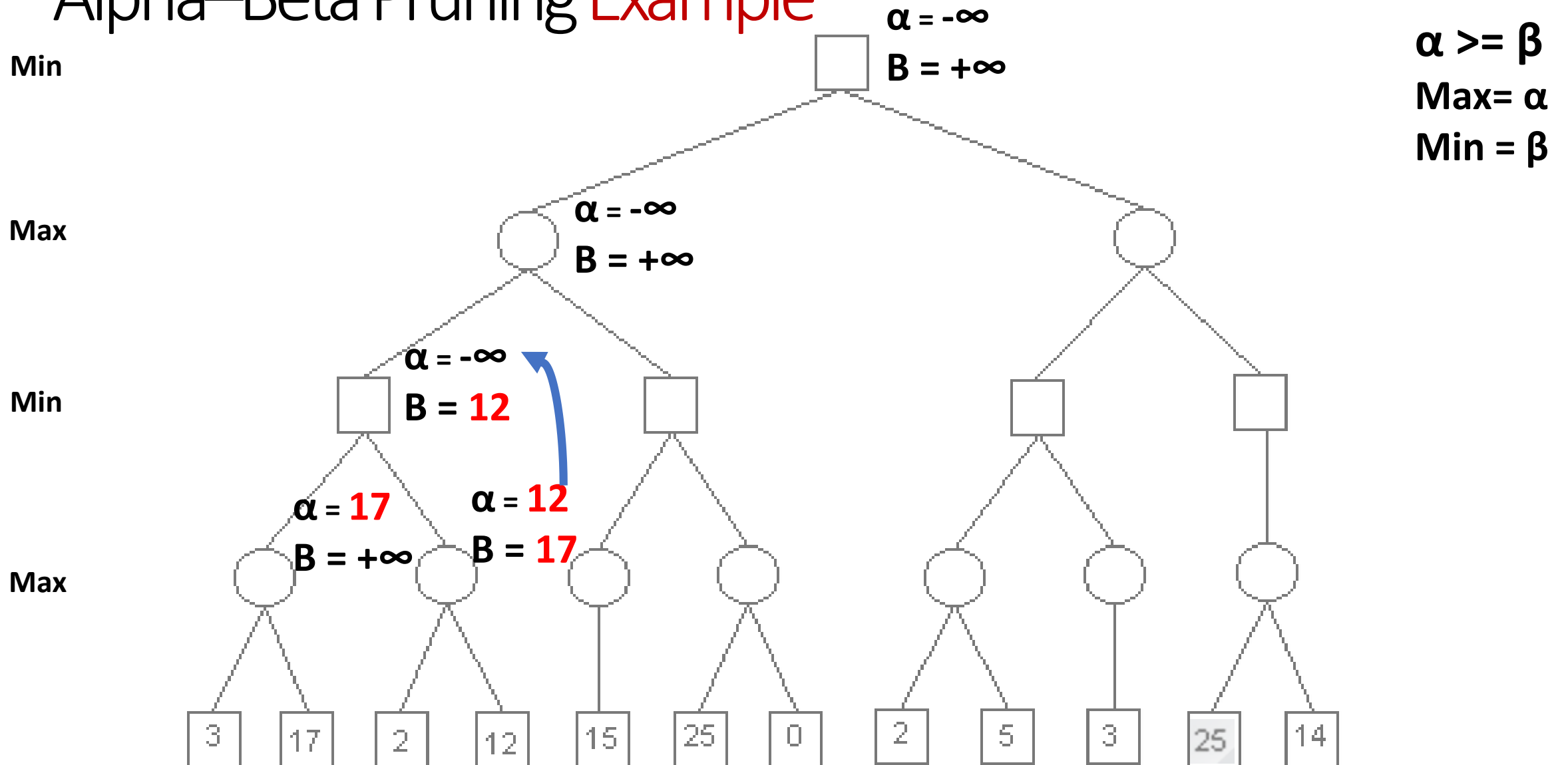
$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



$$\alpha = -\infty$$
$$\beta = +\infty$$

$\alpha = 12$
 $B = +\infty$

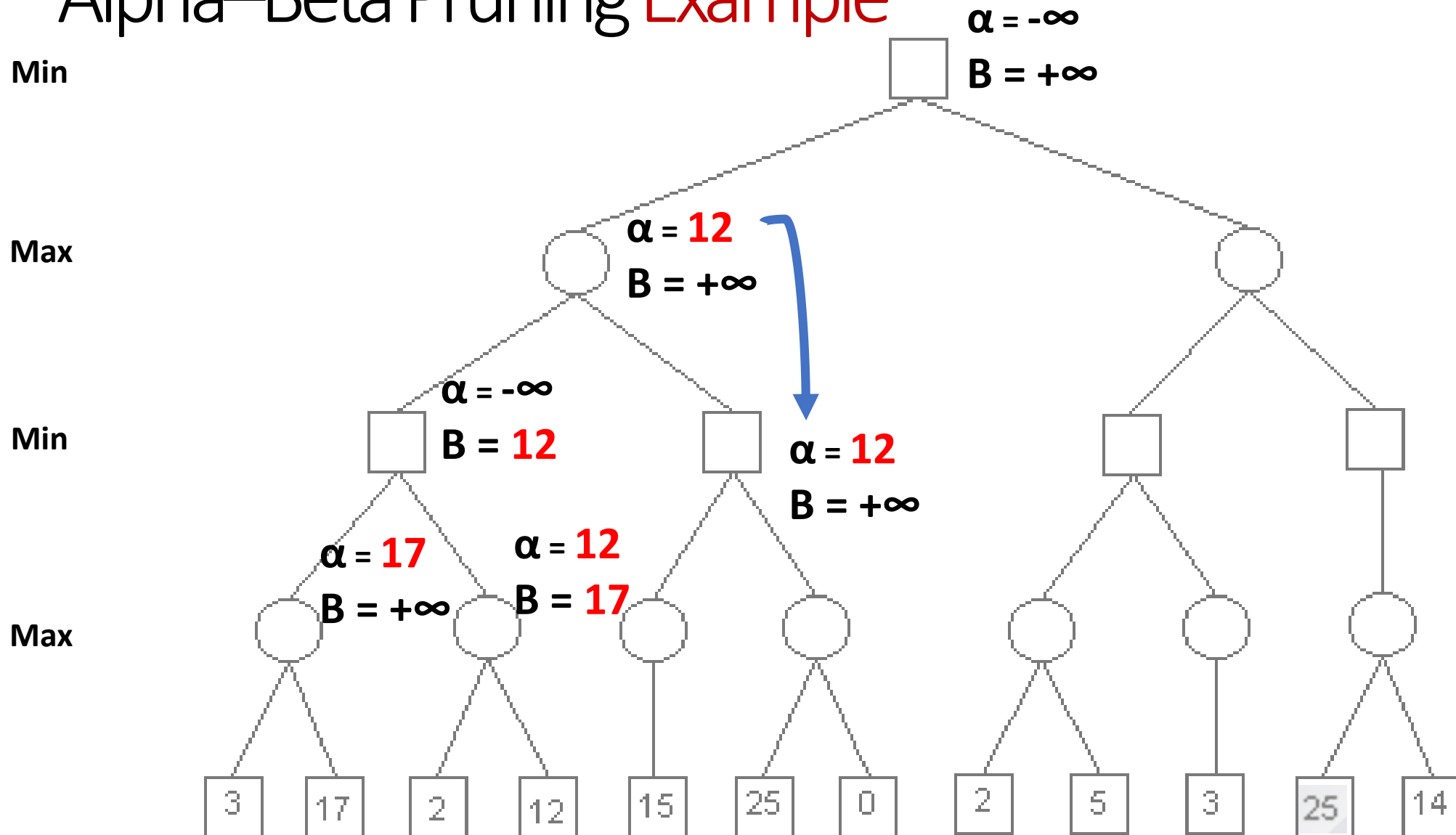
$\alpha = -\infty$
 $B = 12$

$$\alpha = 17$$
$$B = +\infty$$

$\alpha = 12$
 $B = 17$

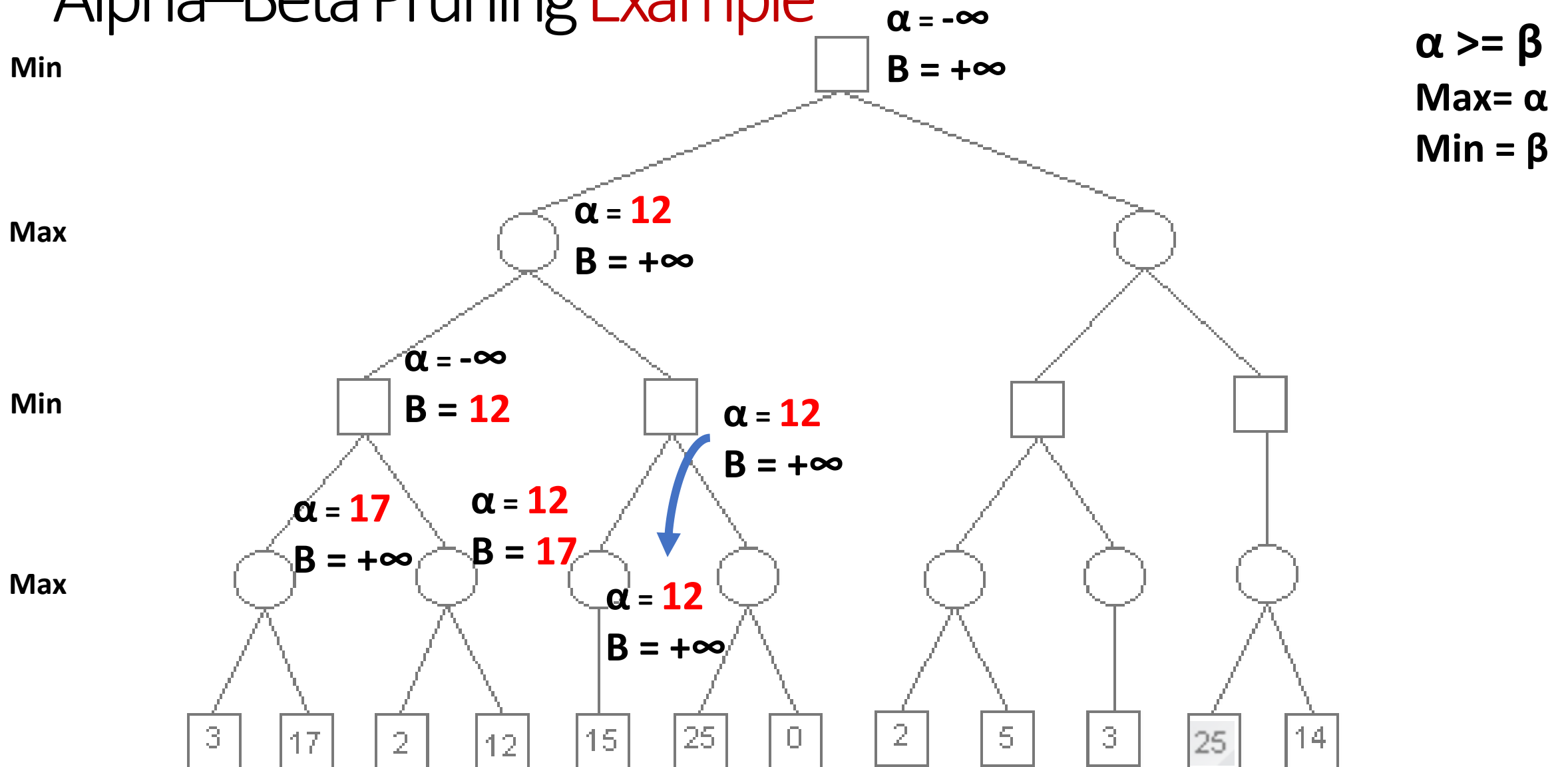
$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example

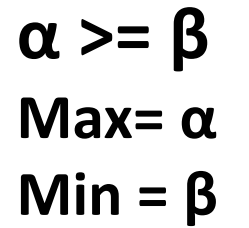


$\alpha \geq \beta$
Max = α
Min = β

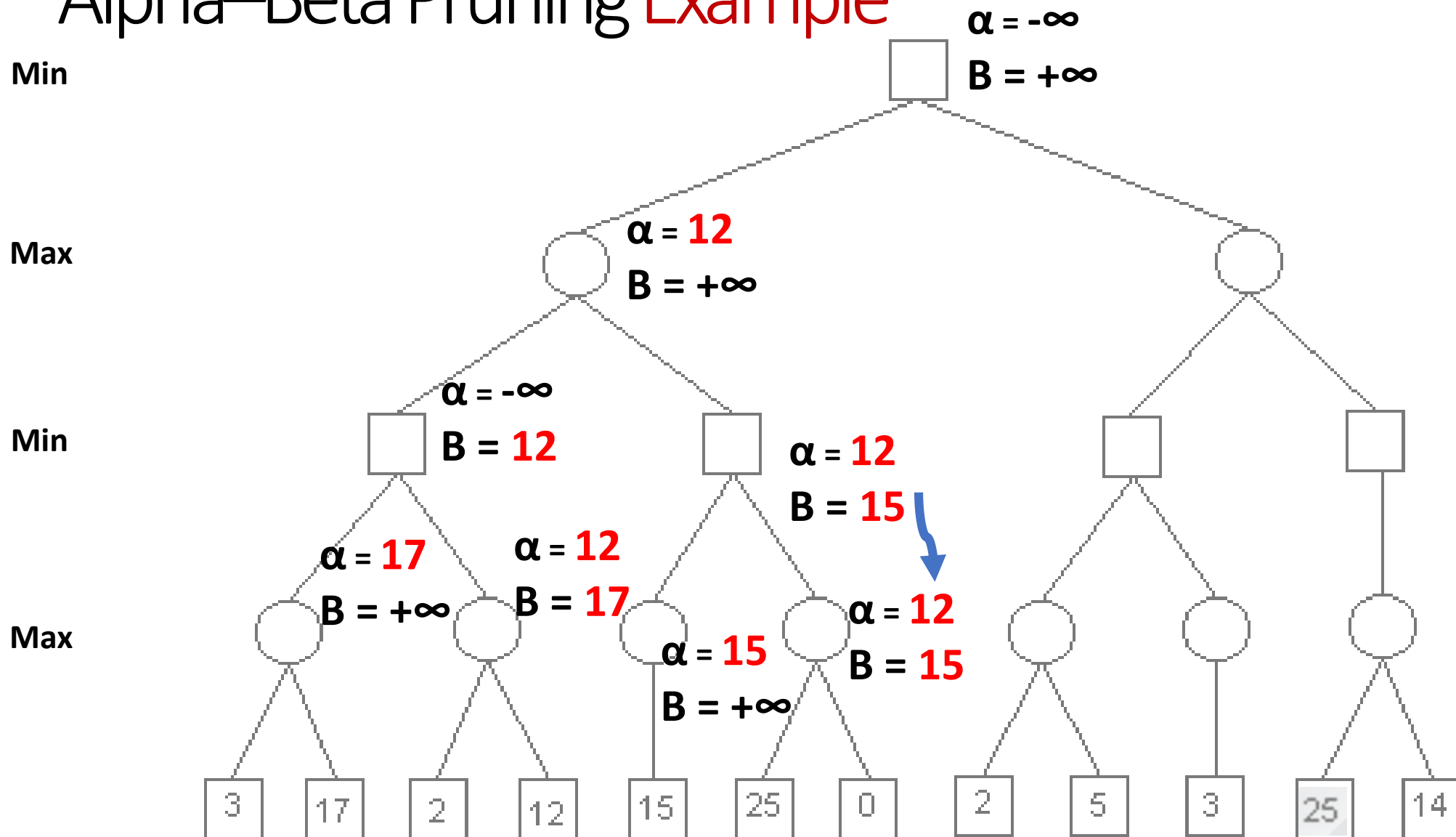
Alpha-Beta Pruning Example



Max

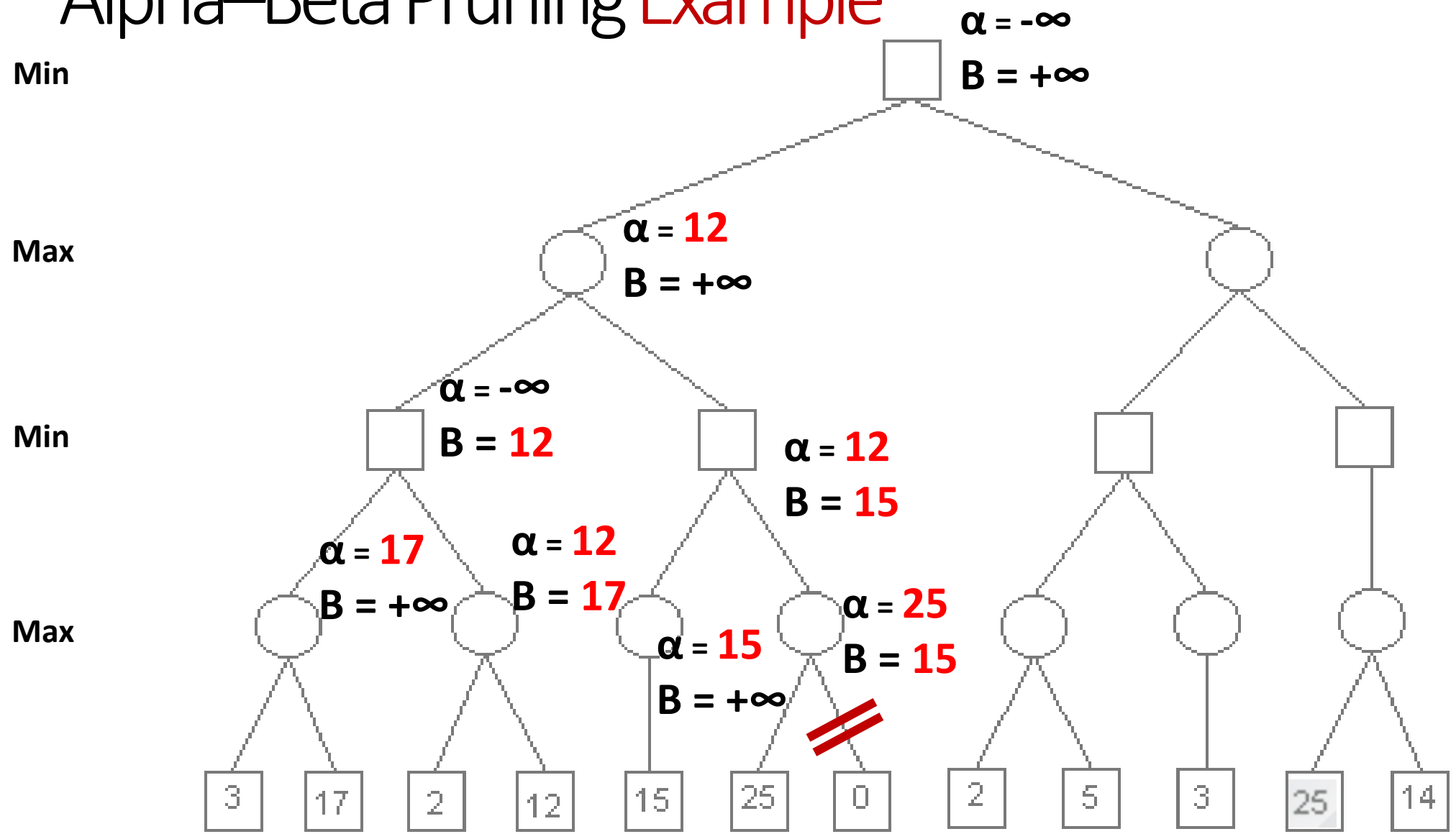


Alpha-Beta Pruning Example



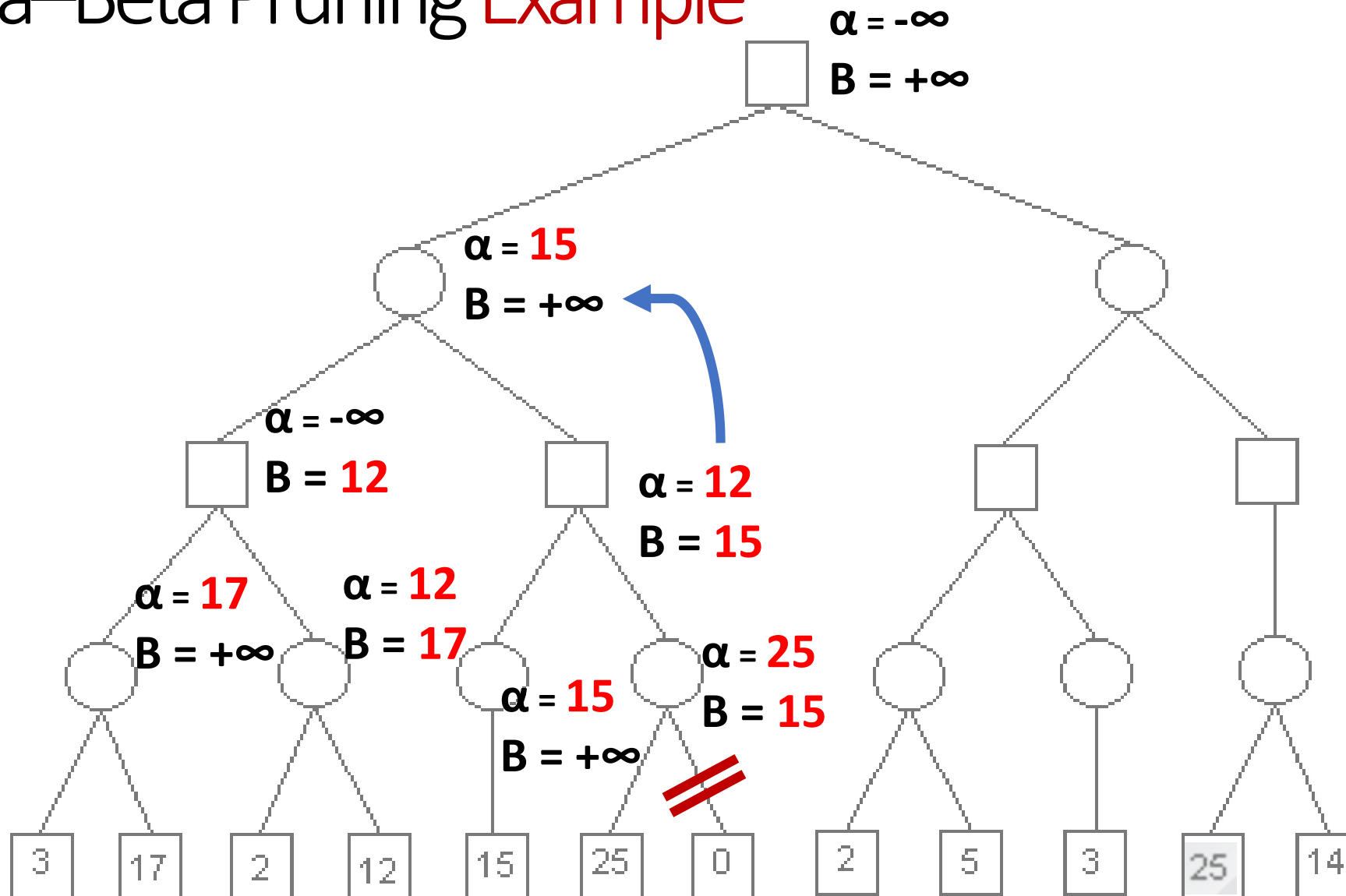
$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



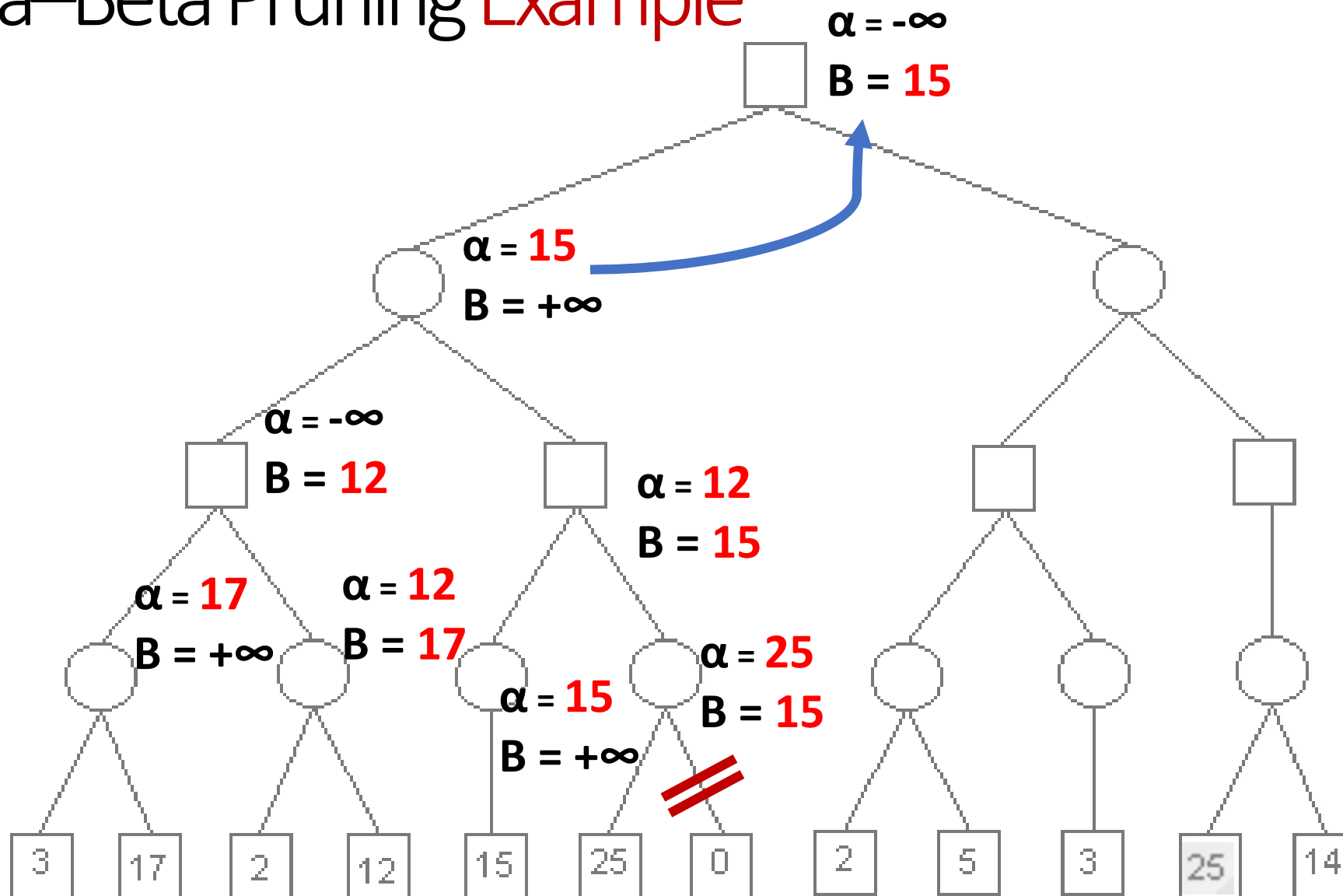
$\alpha \geq \beta$
25 > 15
Max = α
Min = β

Max



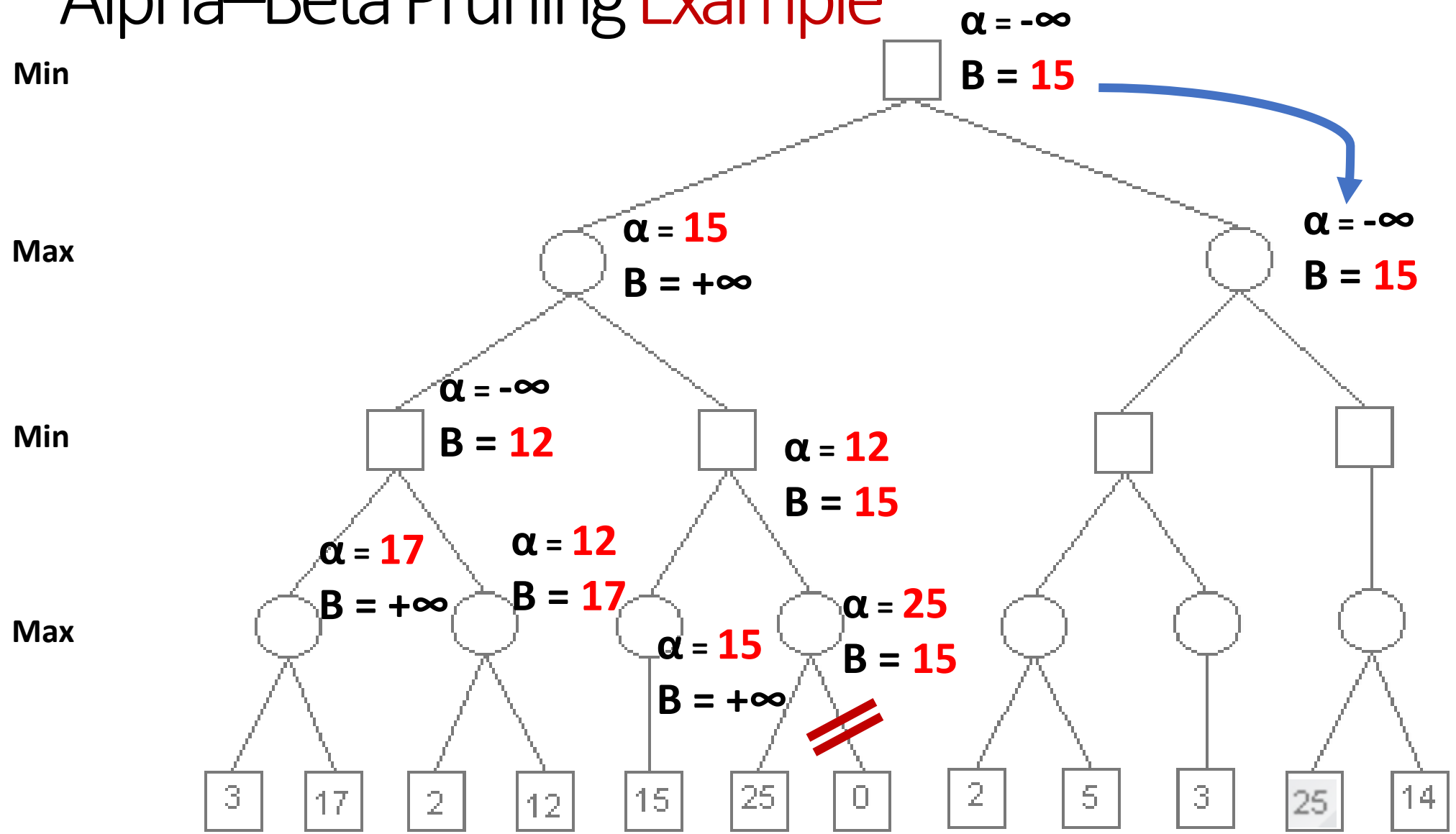
$\alpha \geq \beta$
Max = α
Min = β

Max



$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



$\alpha \geq \beta$
 Max = α
 Min = β

$$\alpha = -\infty$$
$$B = 15$$

$\alpha = 15$
 $B = +\infty$

$$\alpha = -\infty$$
$$B = 15$$
$$\alpha = -\infty$$
$$B = 12$$

$\alpha = 12$
 $B = 15$

$$\alpha = -\infty$$
$$B = 15$$

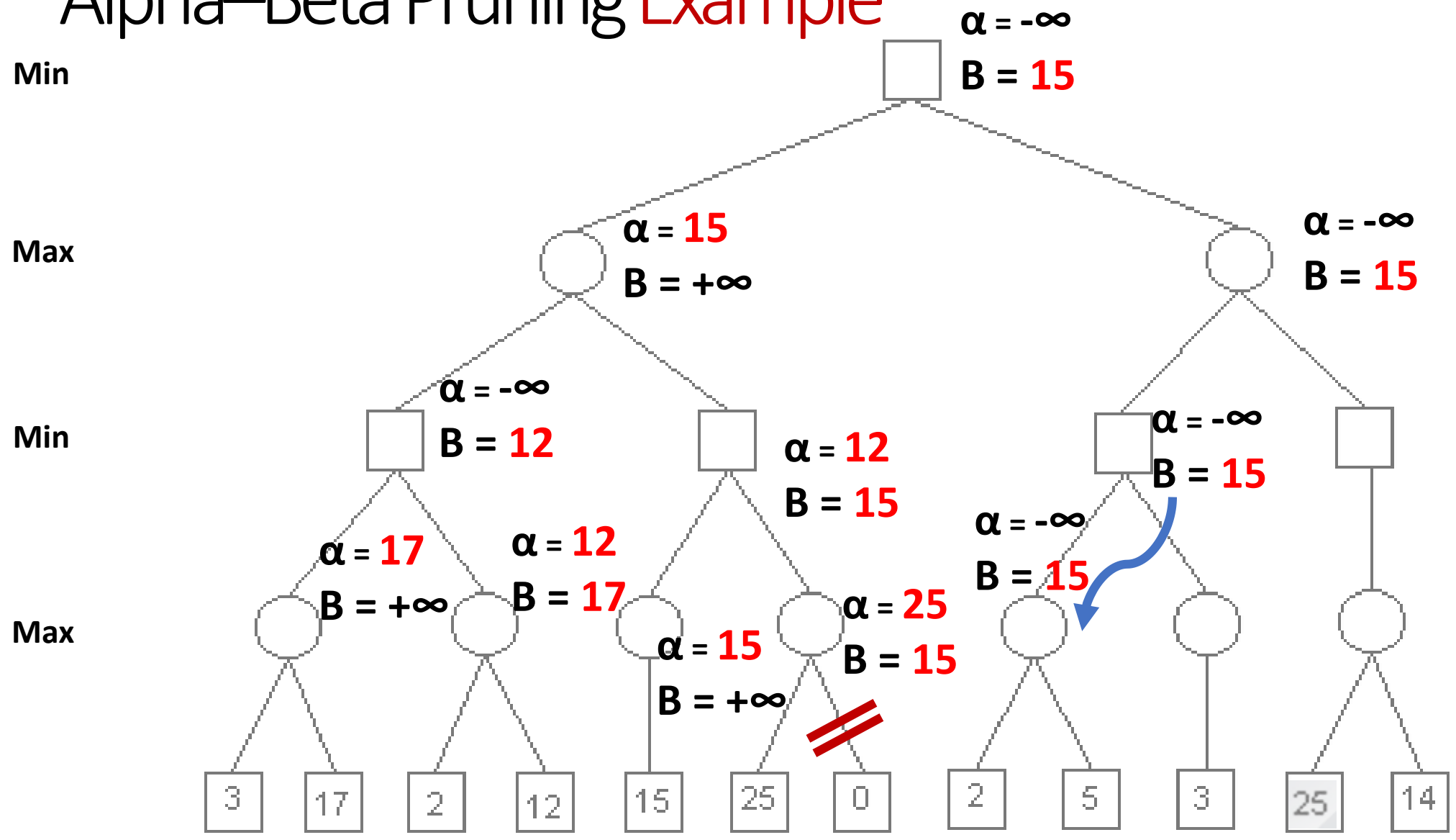
$\alpha = 17$
 $B = +\infty$

$\alpha = 12$
 $B = 17$

Diagram illustrating a node in a minimax tree. The node is a circle with two children. The left child is a square node with $\alpha = 15$ and $B = +\infty$. The right child is a circle node with $\alpha = 25$ and $B = 15$. A red arrow points to the right child.

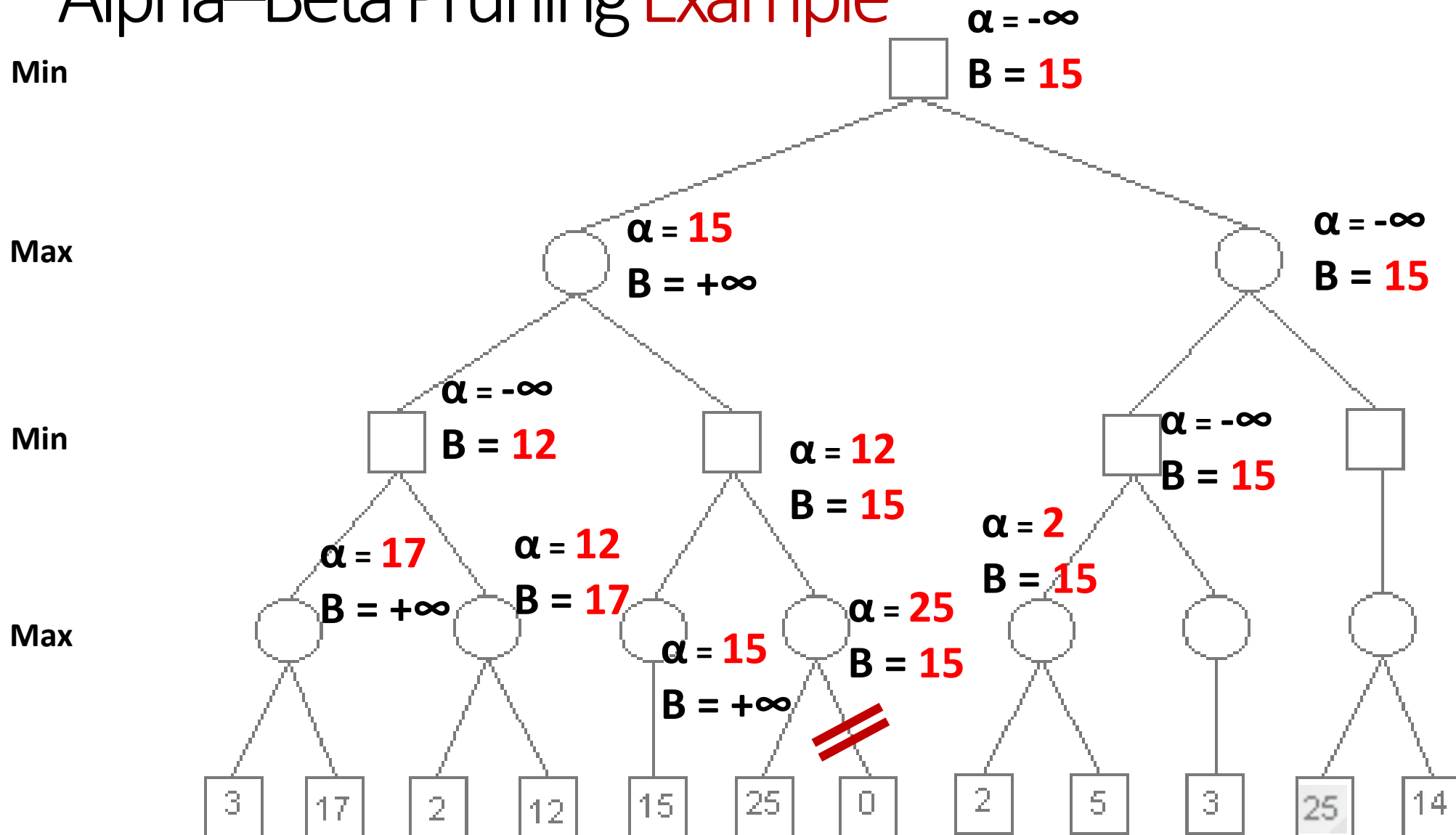
$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



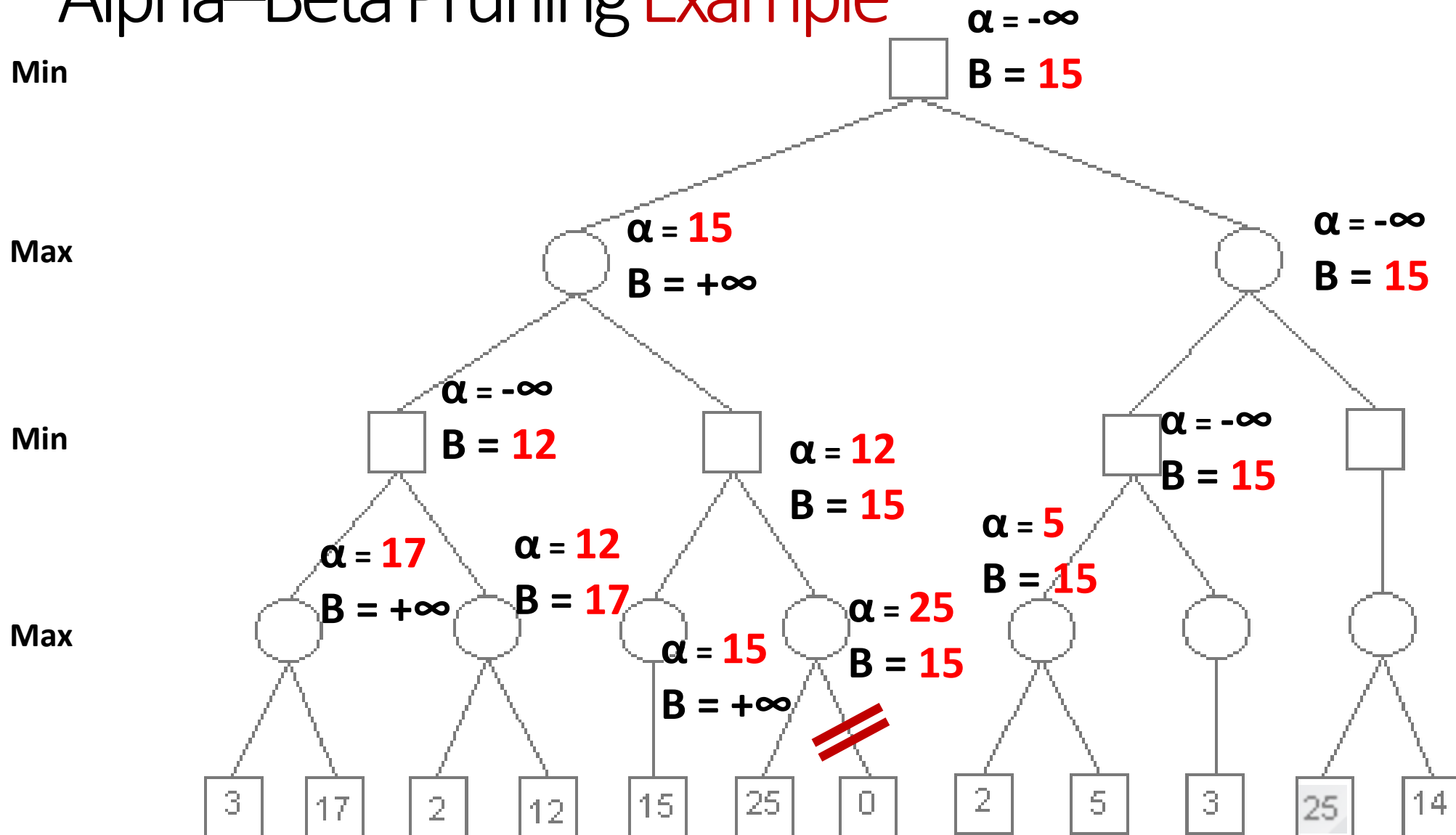
$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



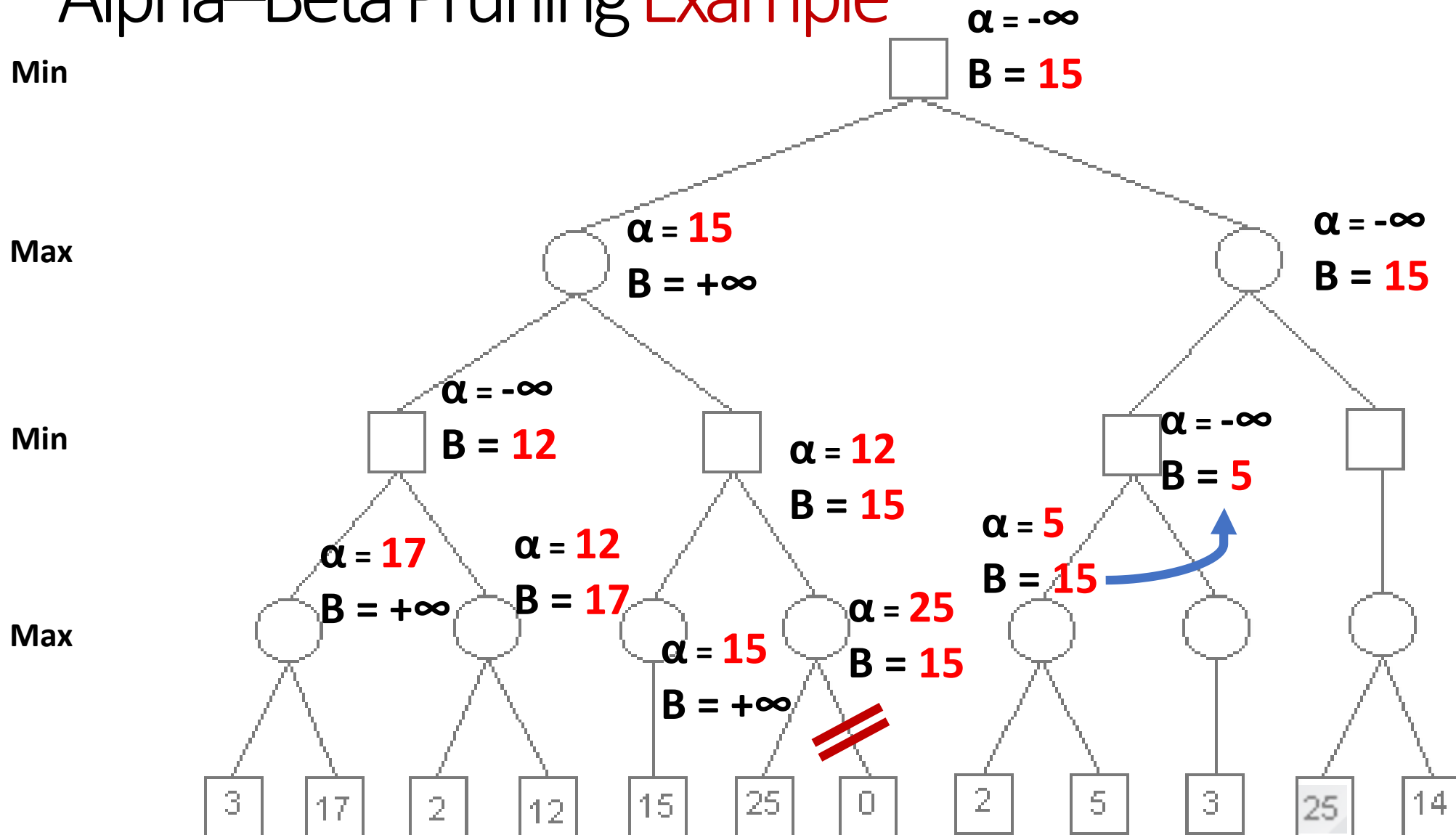
$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



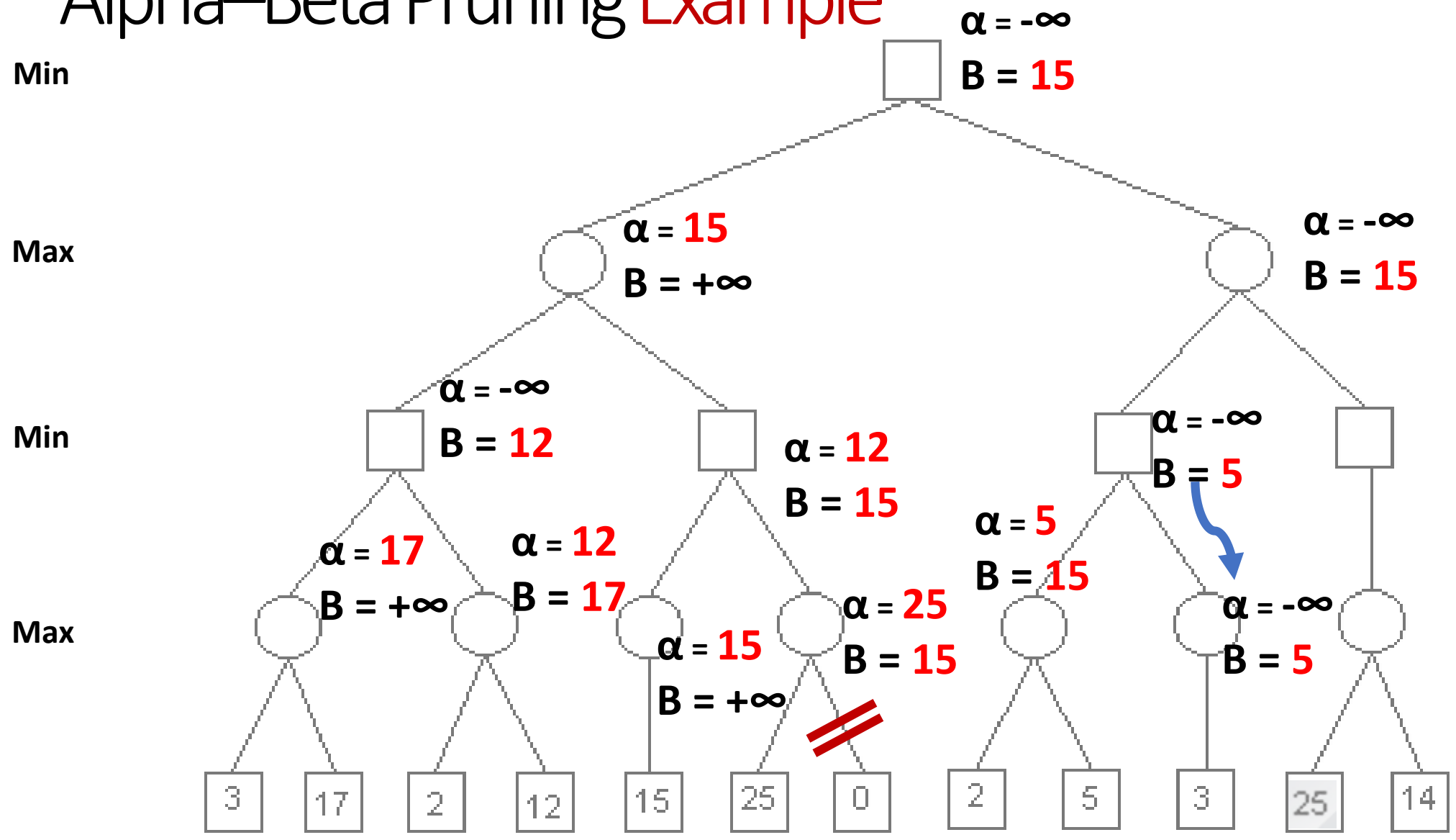
$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



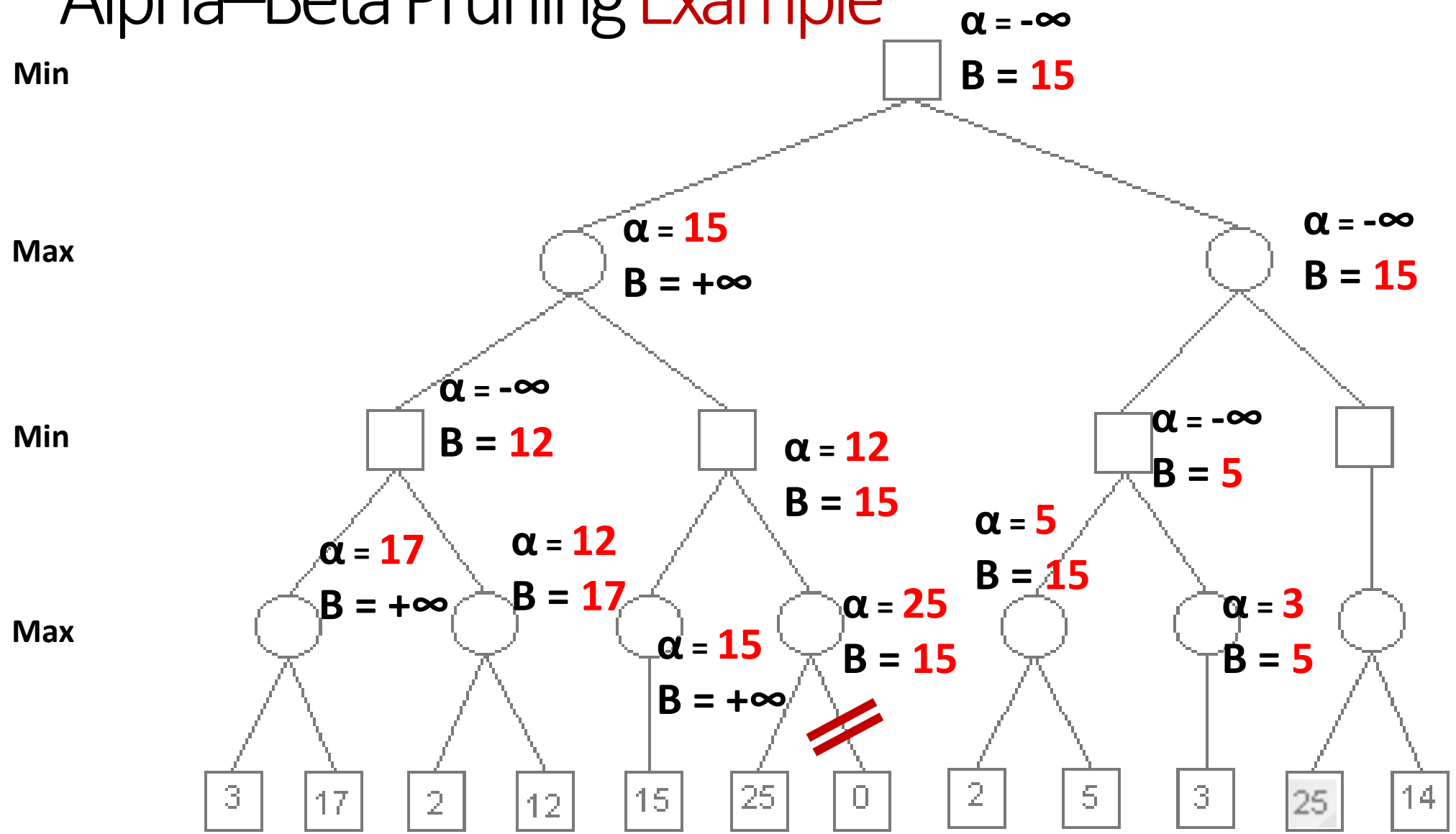
$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



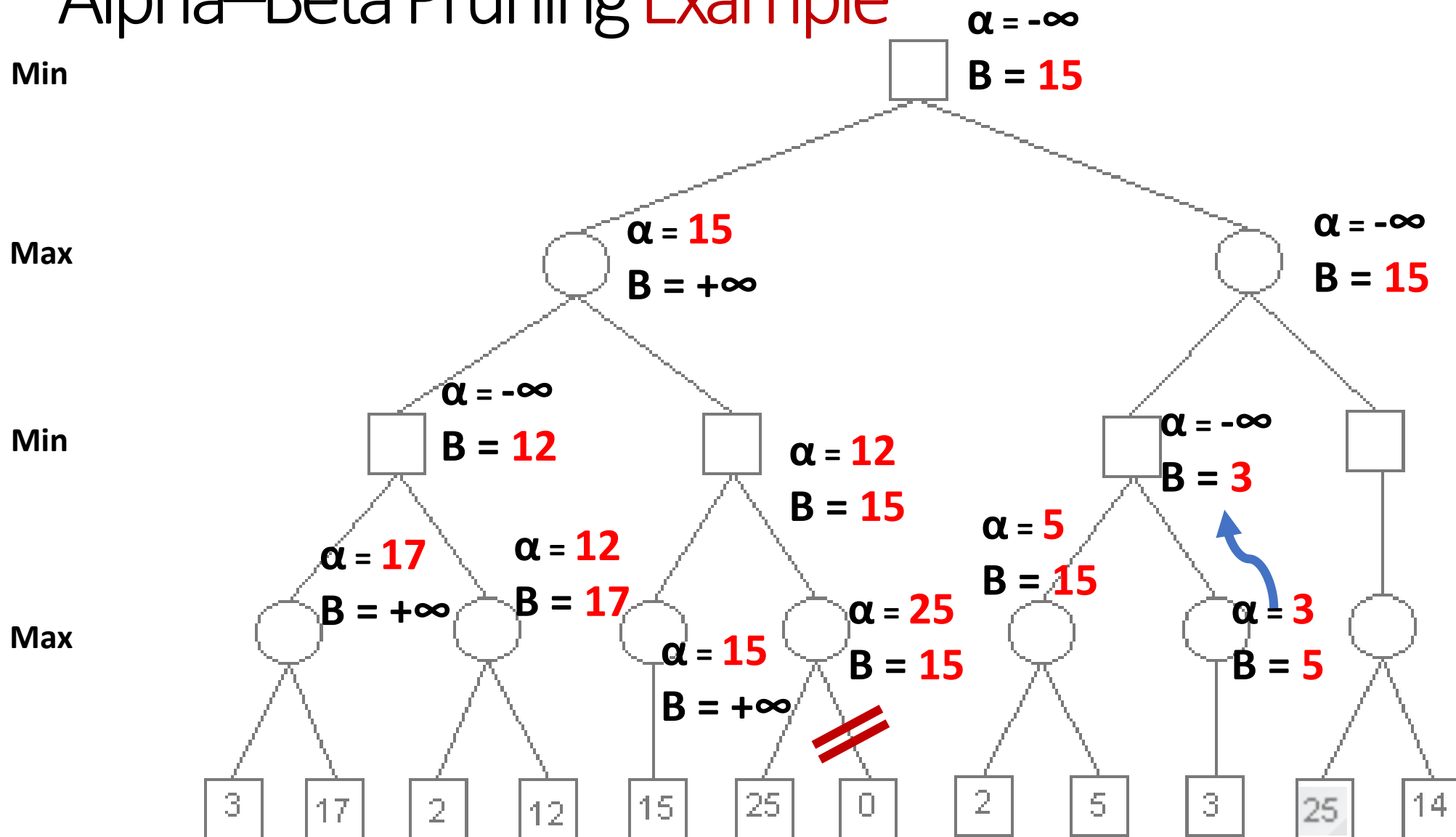
$\alpha \geq \beta$
 Max = α
 Min = β

Alpha-Beta Pruning Example



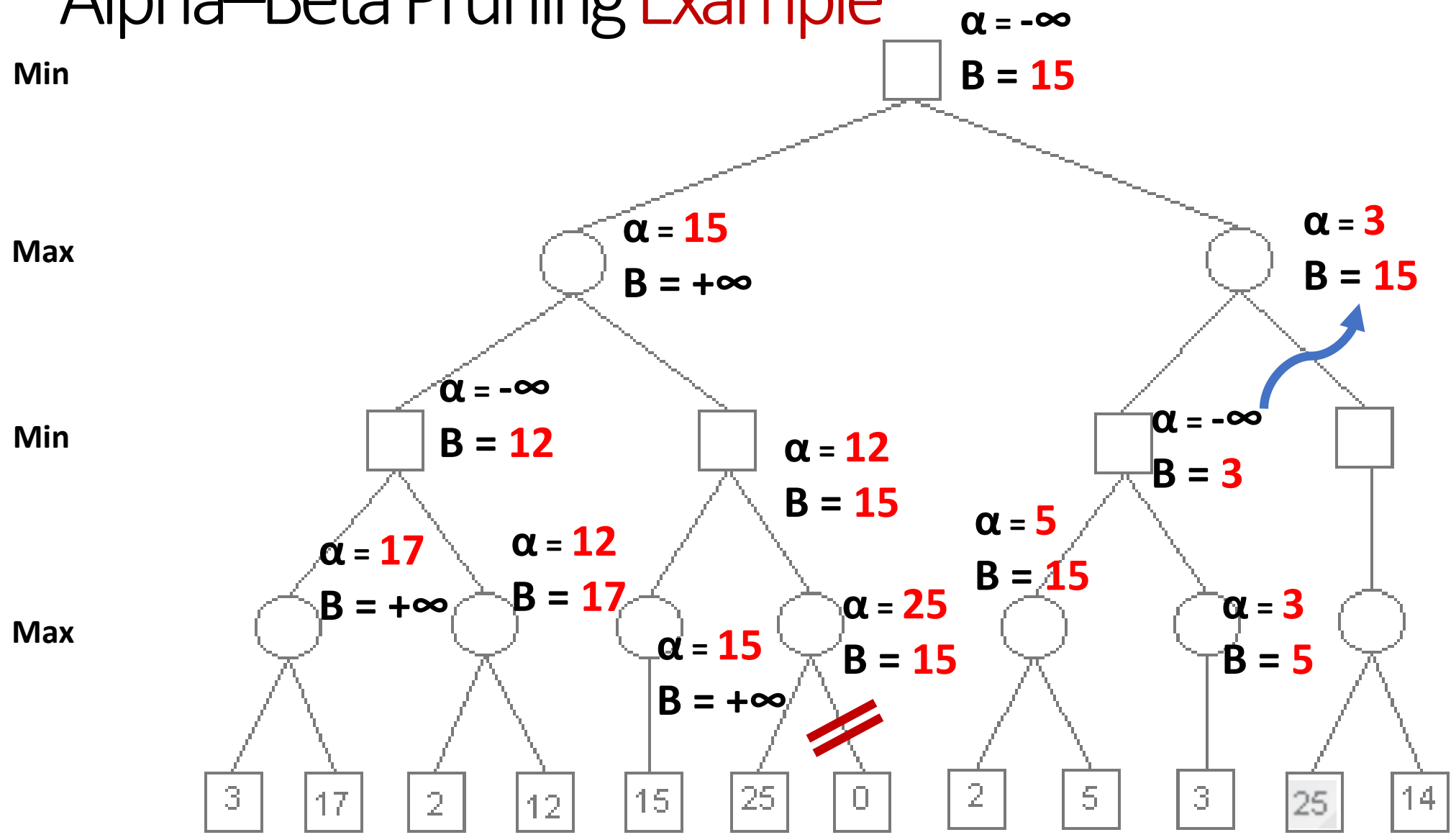
$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



$\alpha \geq \beta$
Max = α
Min = β

Alpha-Beta Pruning Example



$\alpha \geq \beta$
Max = α
Min = β

$$\alpha = -\infty$$
$$B = 15$$

$\alpha = 15$
 $B = +\infty$

$\alpha = 3$
 $B = 15$

$$\alpha = -\infty$$
$$B = 12$$

$\alpha = 12$
 $B = 15$

$$\alpha = -\infty$$
$$B = 3$$

$\alpha = 3$
 $B = 15$

$\alpha = 17$
 $B = +\infty$

$\alpha = 12$
 $B = 17$

$\alpha = 25$
 $B = 15$

$\alpha = 5$
 $B = 15$

$\alpha = 3$
 $B = 5$

$$\alpha = 15$$
$$B = +\infty$$

2

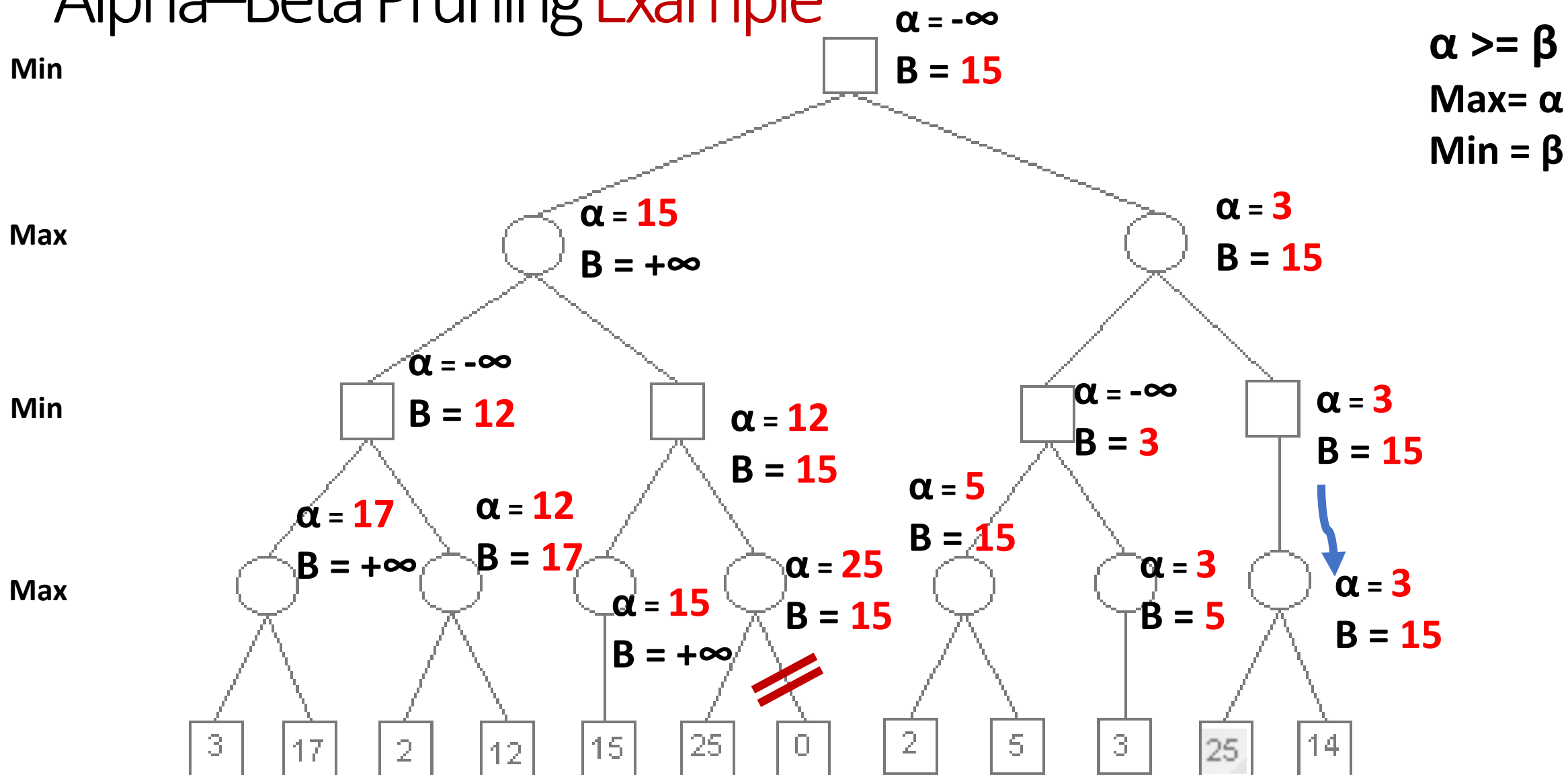
5

3

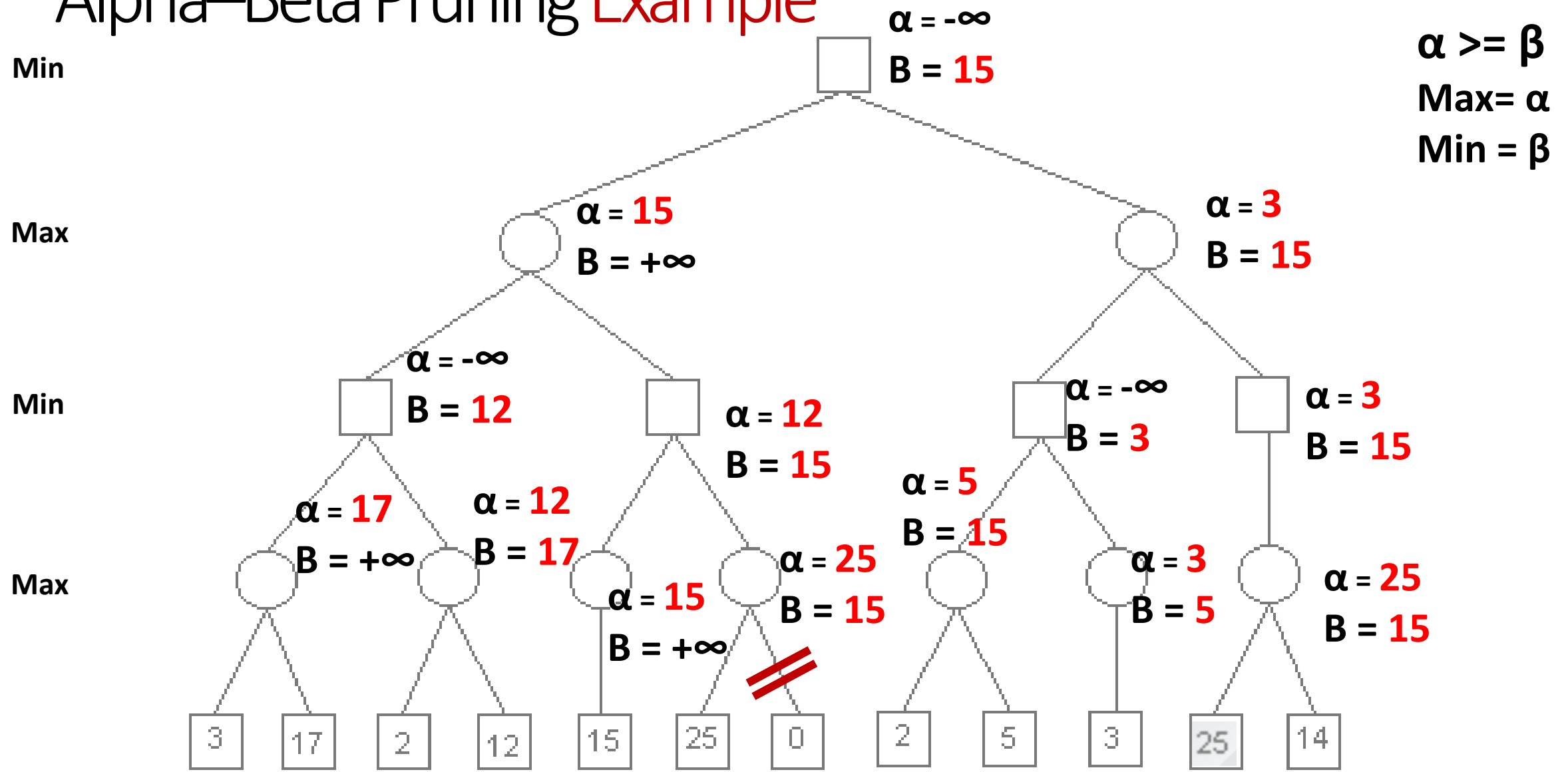
24

$\alpha \geq \beta$
Max = α
Min = β

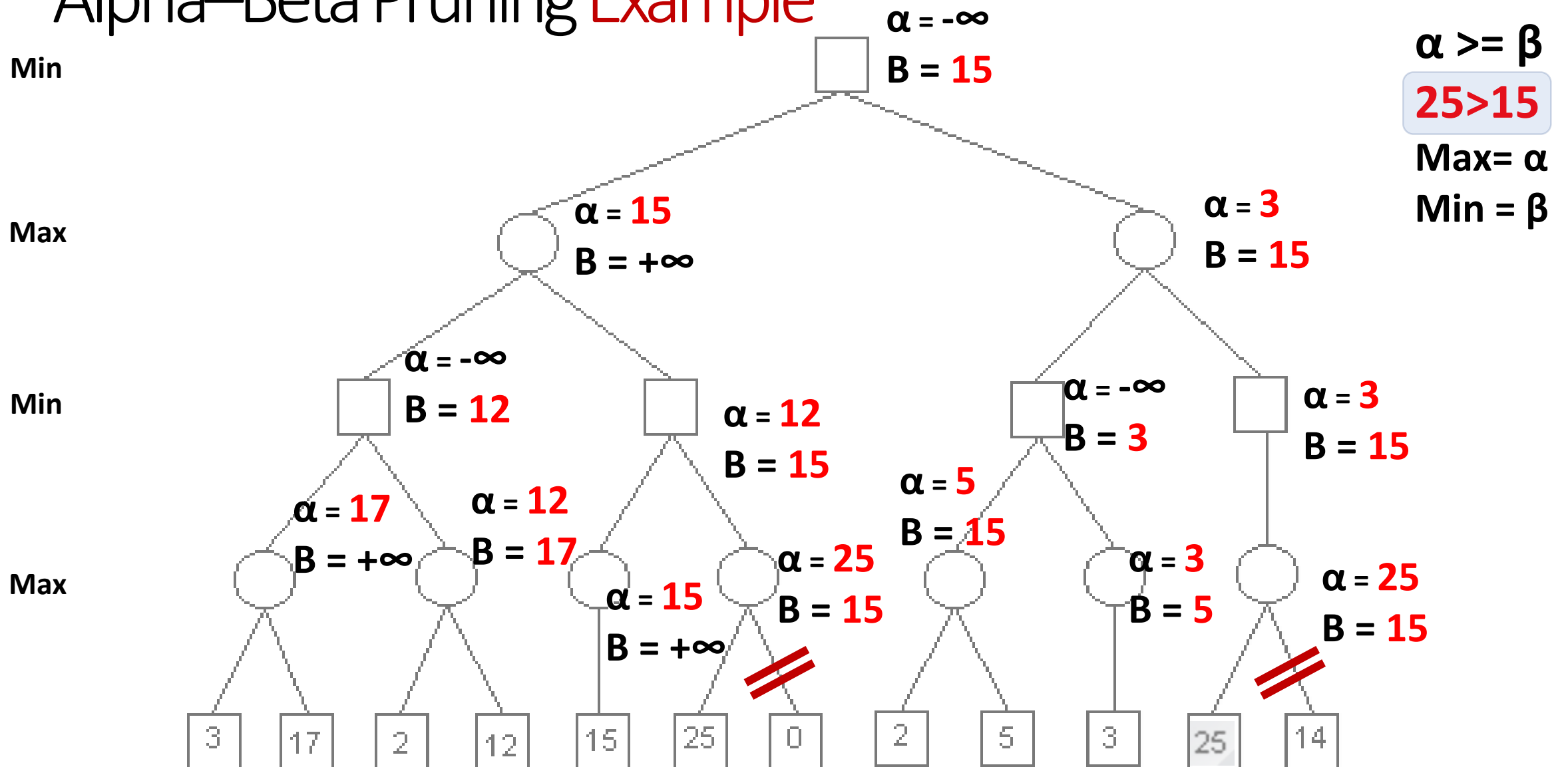
Alpha-Beta Pruning Example



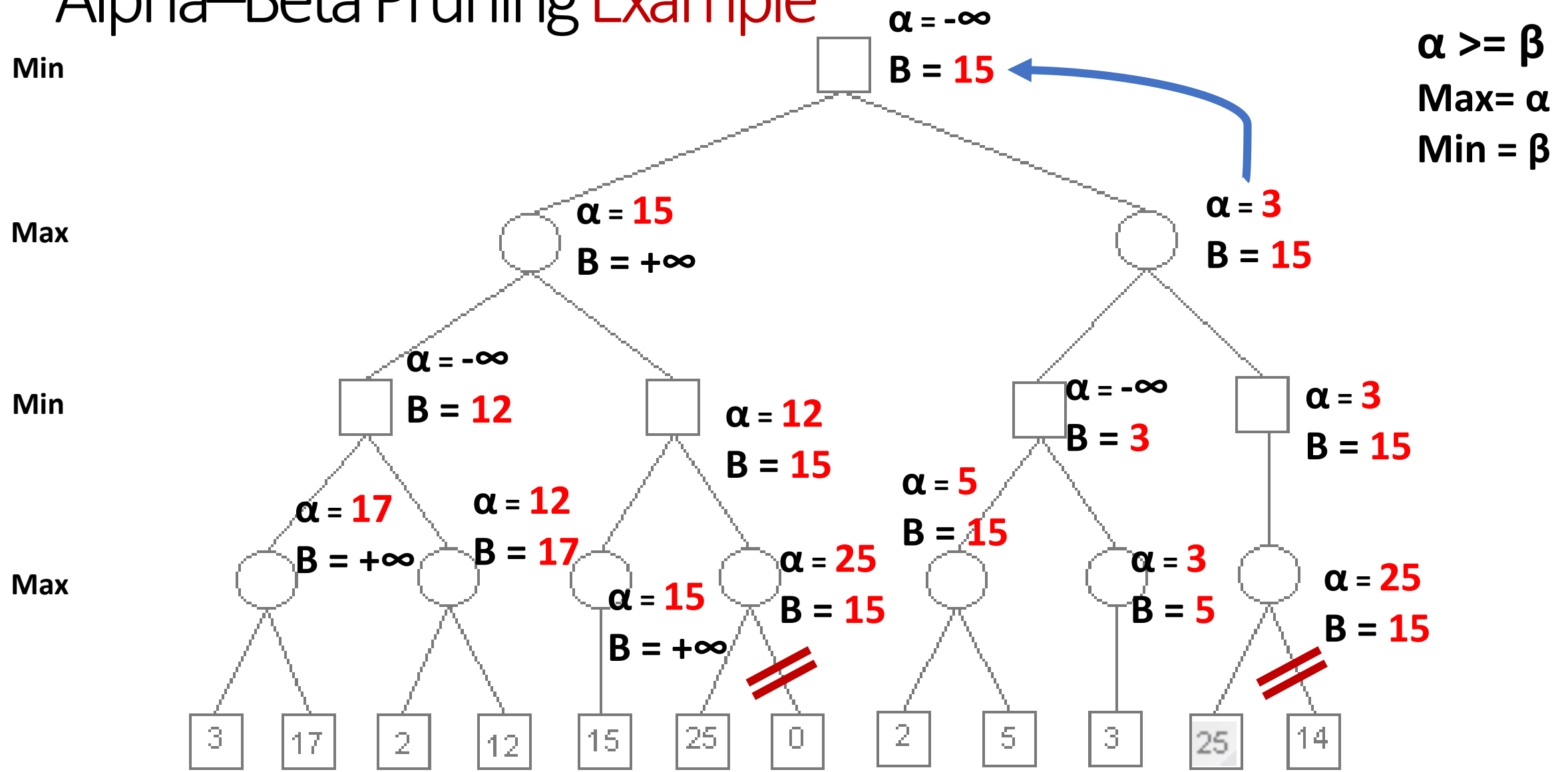
Alpha-Beta Pruning Example



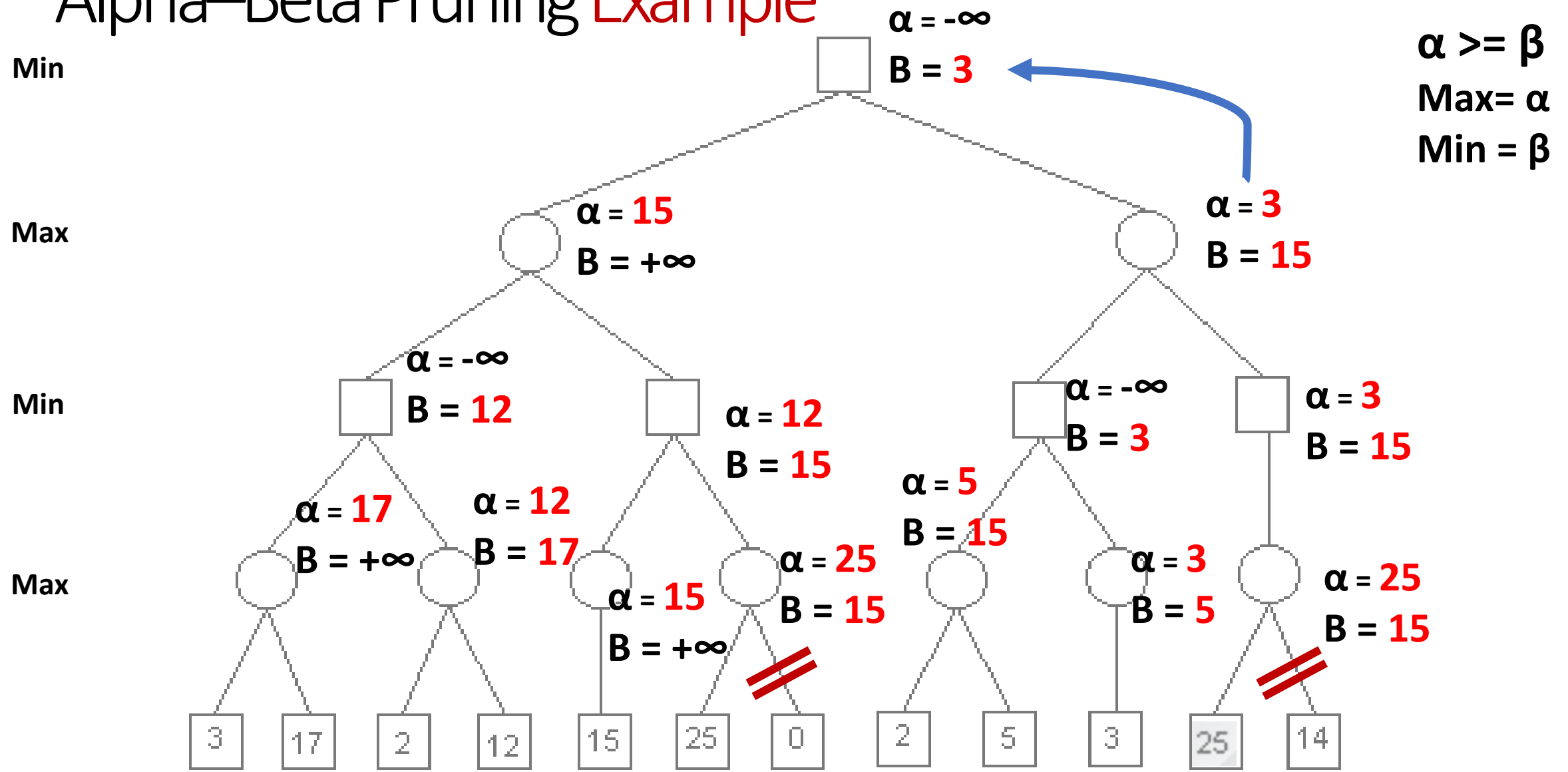
Alpha-Beta Pruning Example



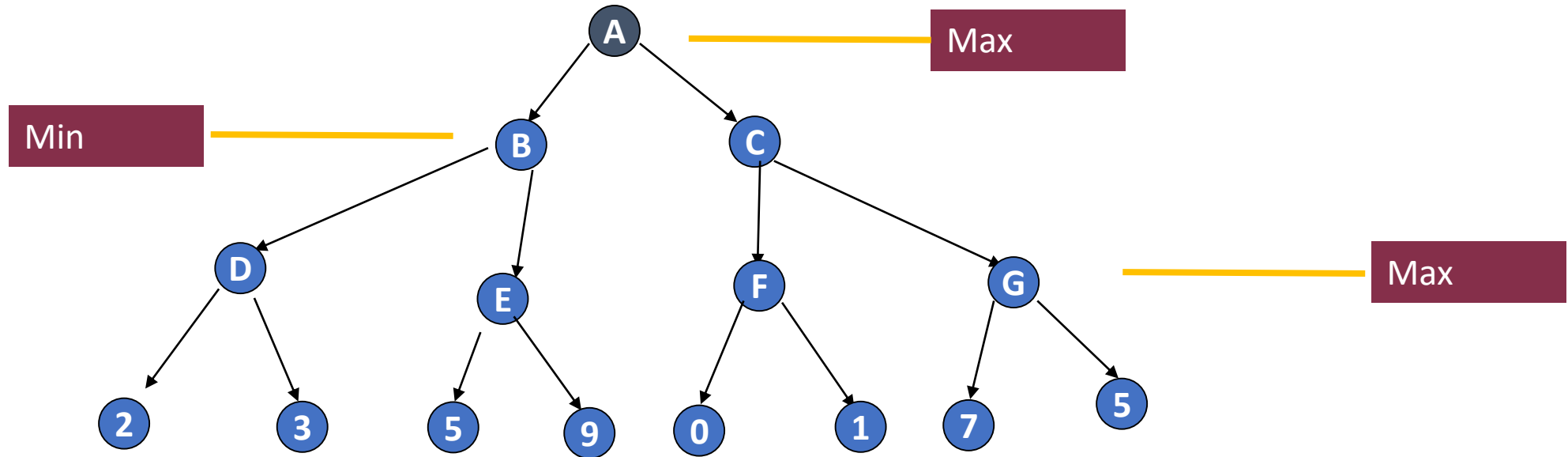
Alpha-Beta Pruning Example



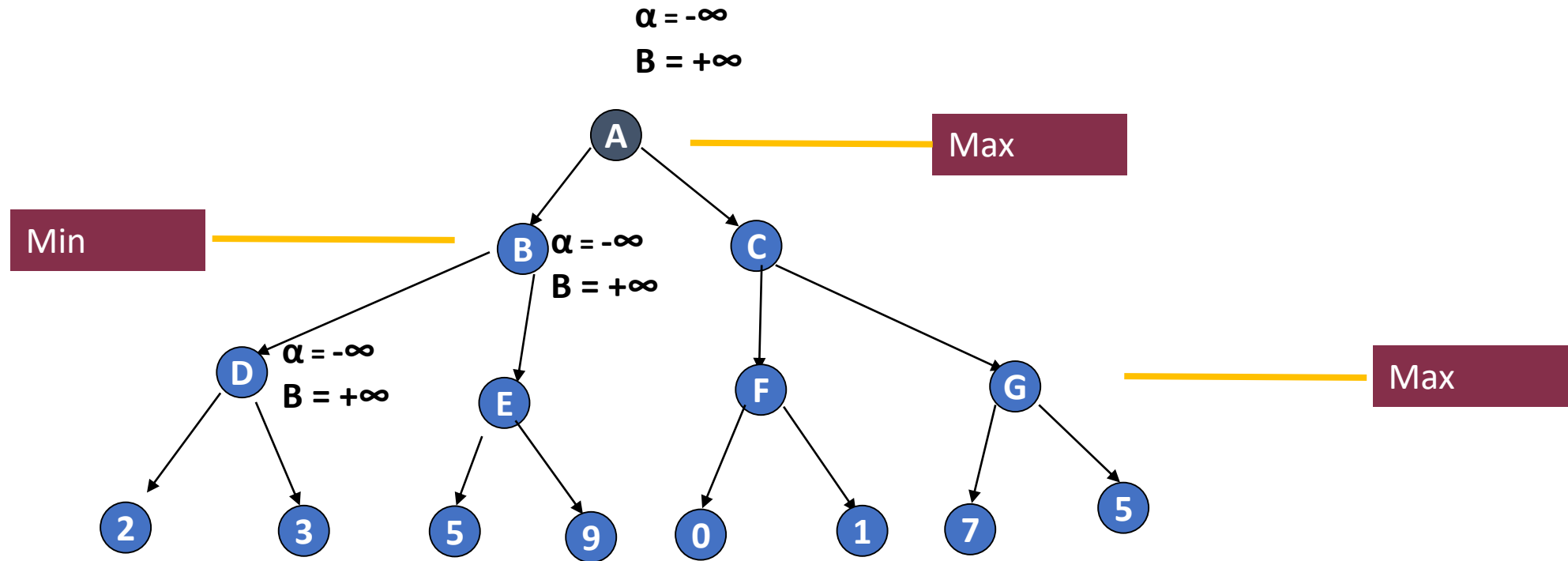
Alpha-Beta Pruning Example



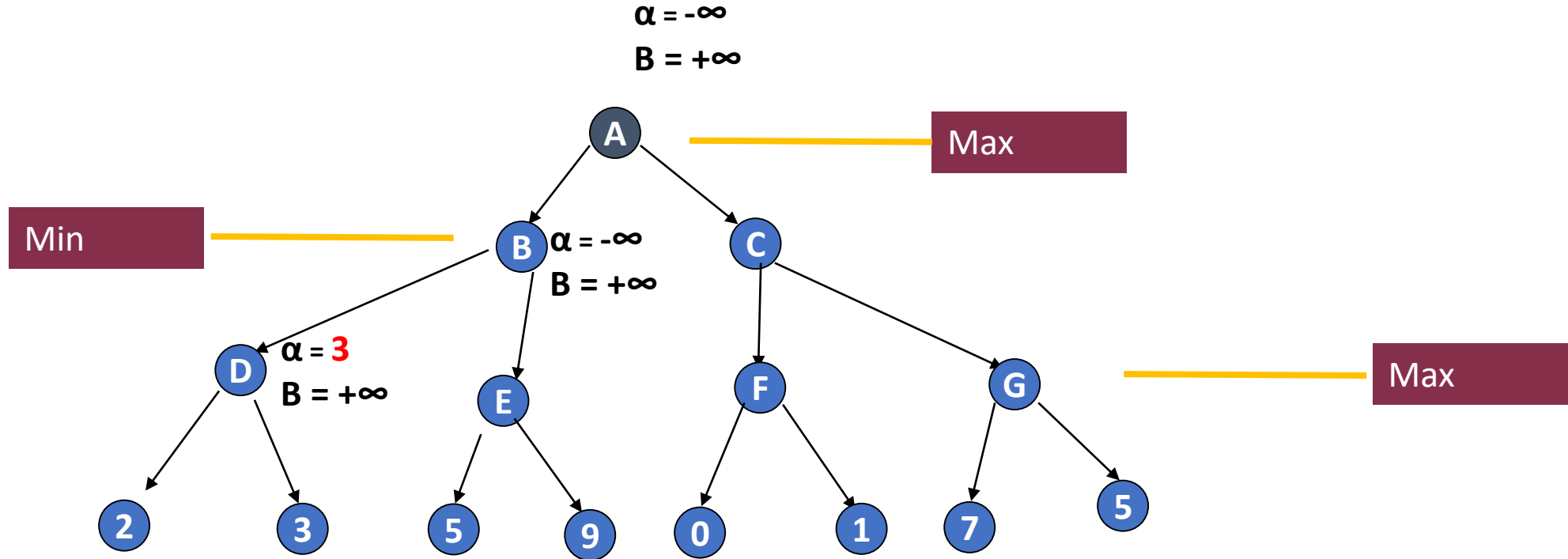
Alpha Beta Pruning Example



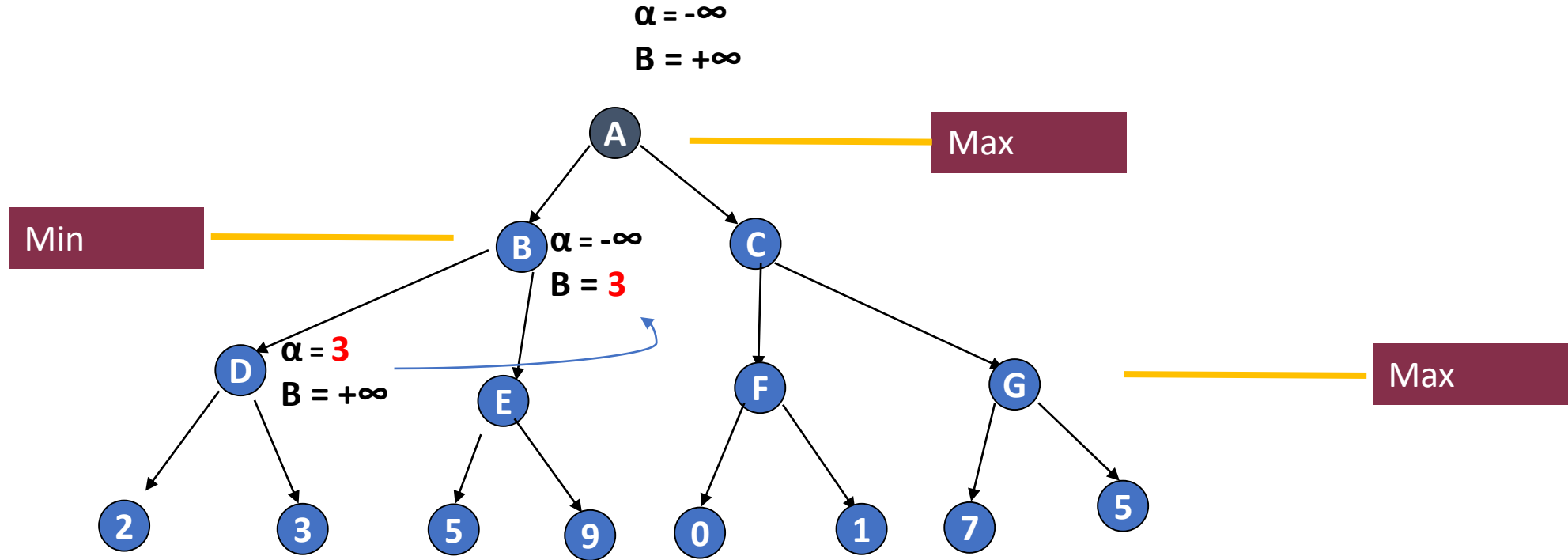
Alpha Beta Pruning Example



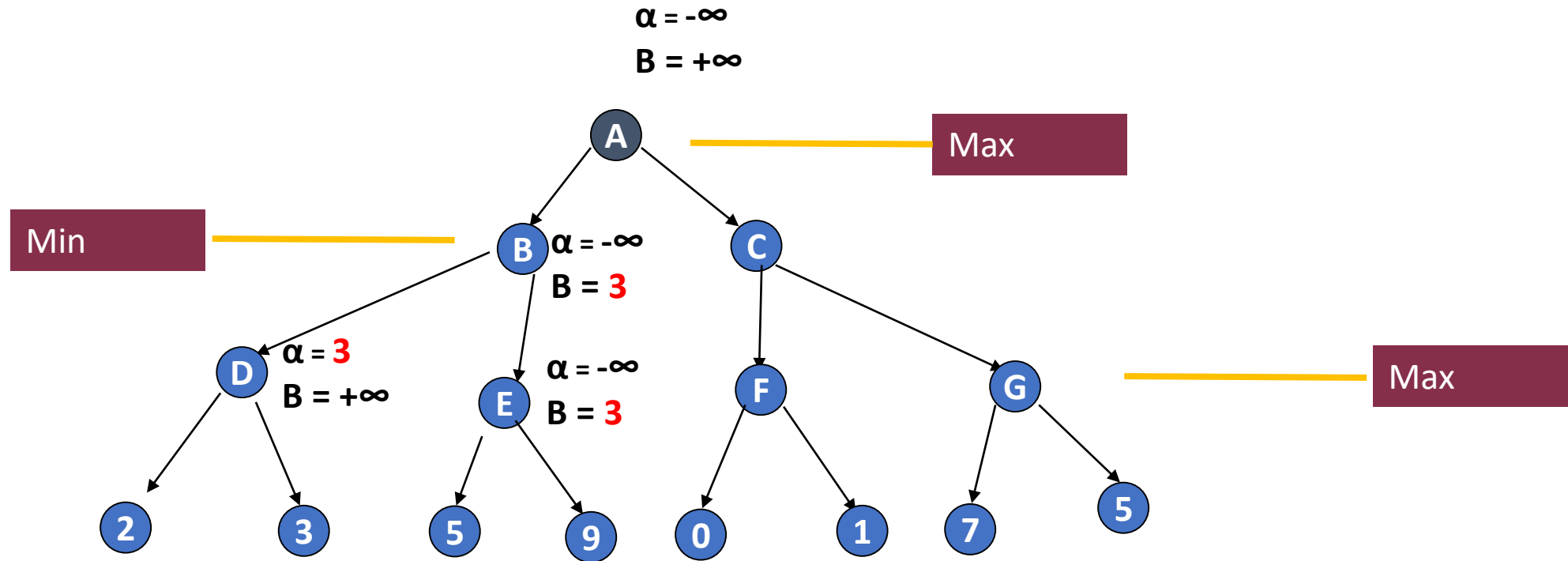
Alpha Beta Pruning Example



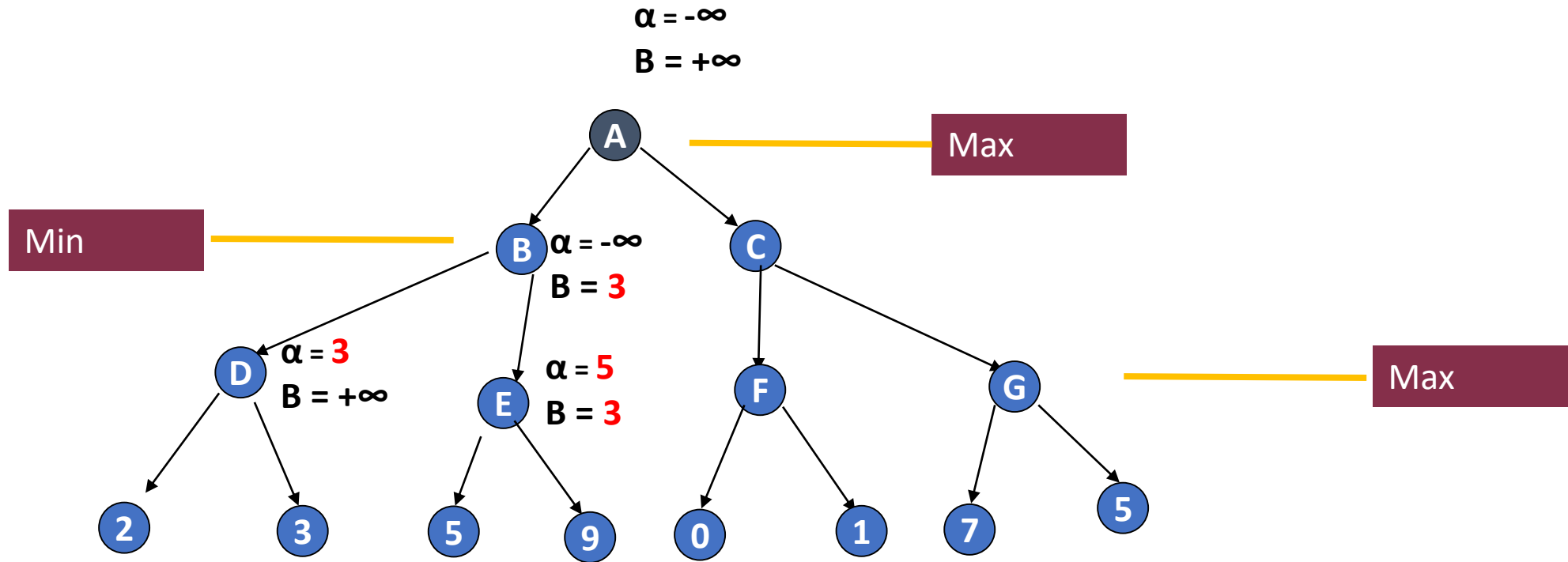
Alpha Beta Pruning Example



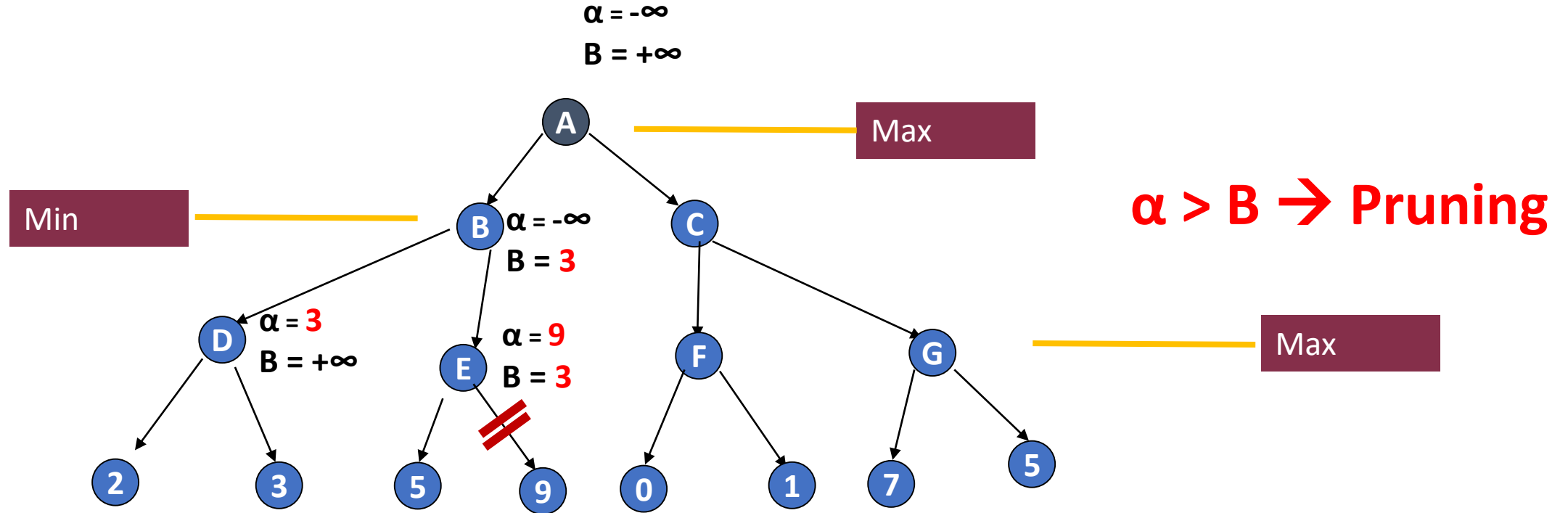
Alpha Beta Pruning Example



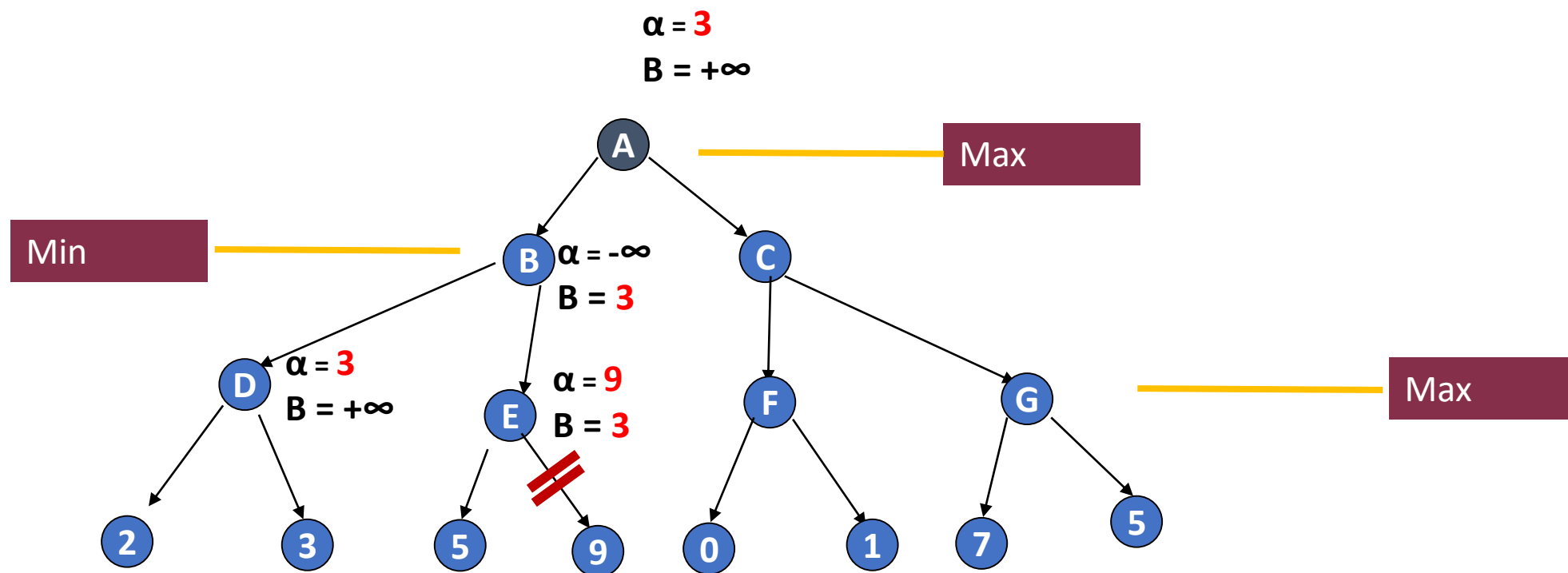
Alpha Beta Pruning Example



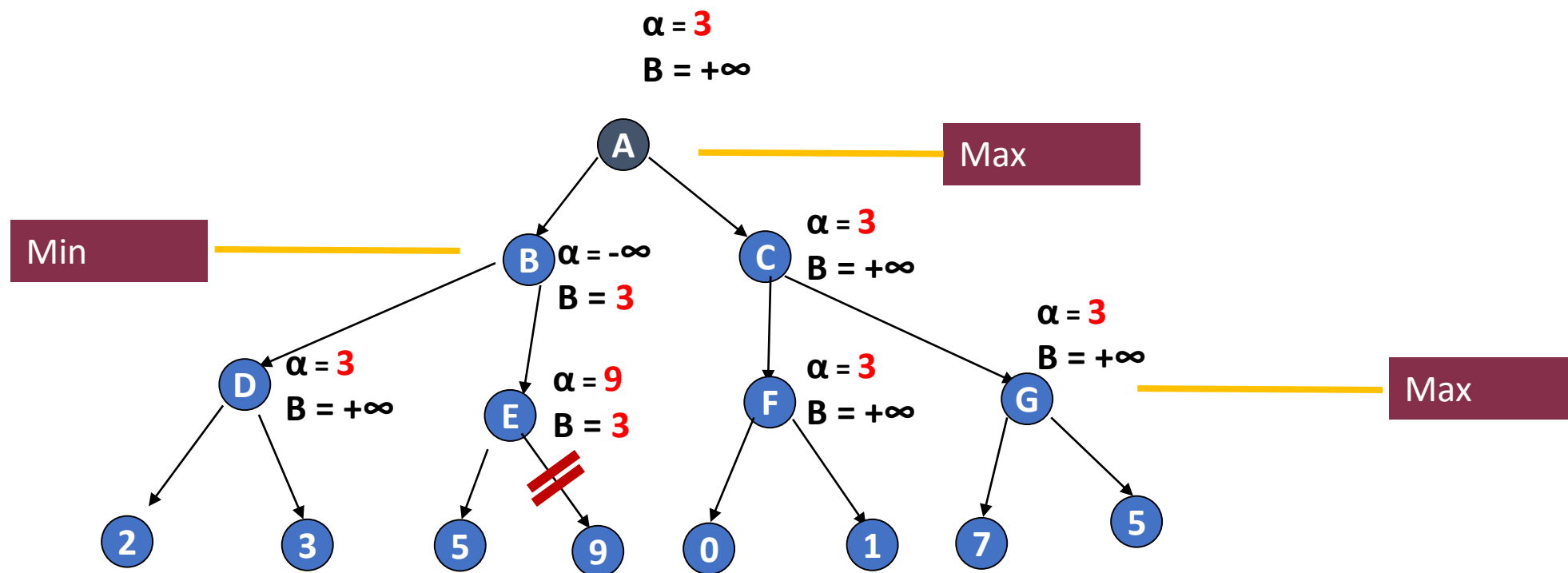
Alpha Beta Pruning Example



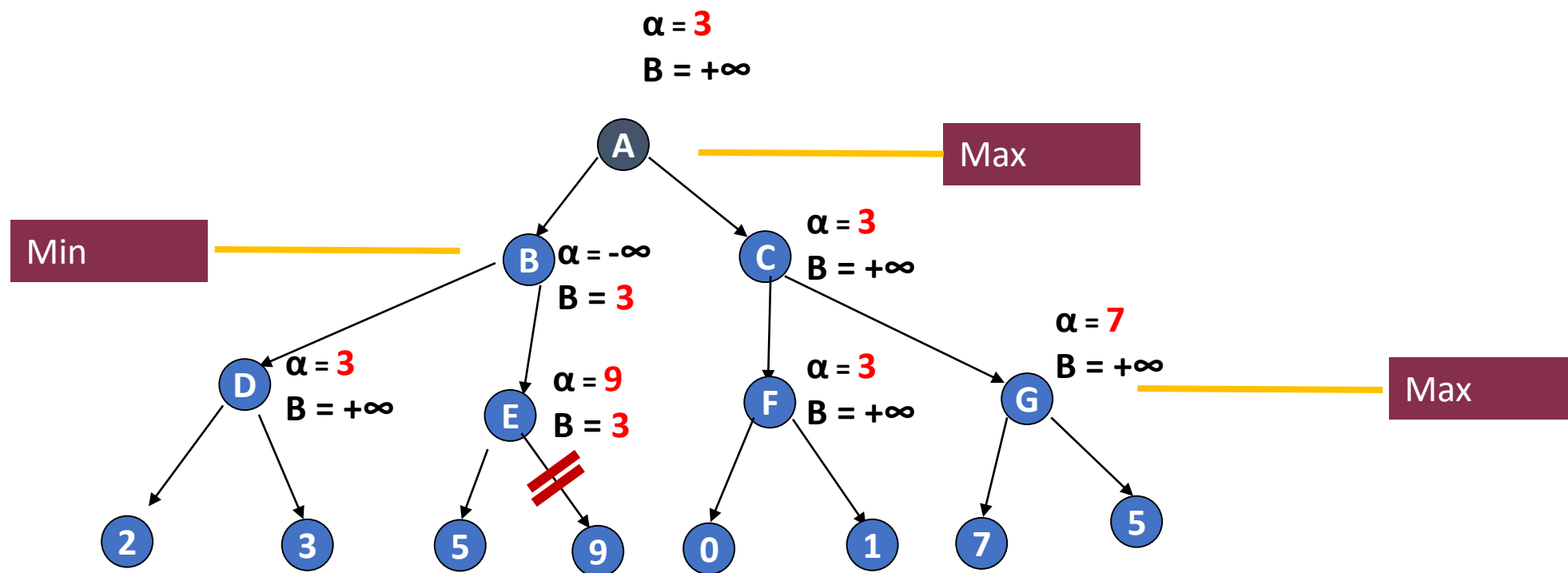
Alpha Beta Pruning Example



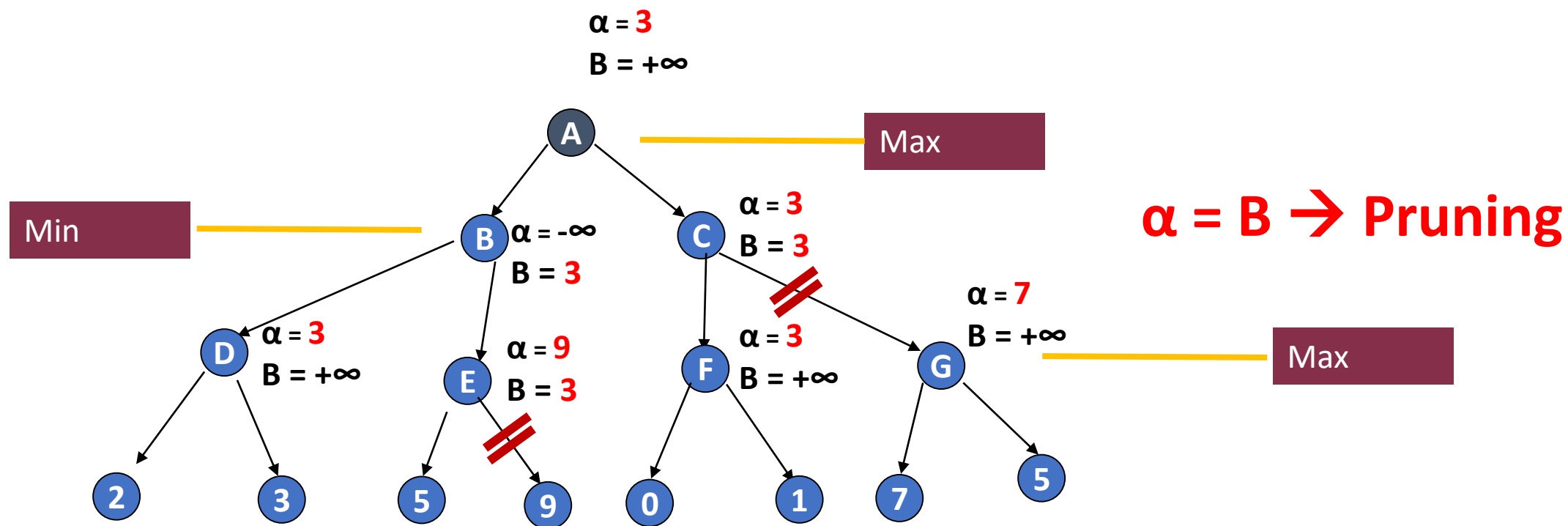
Alpha Beta Pruning Example



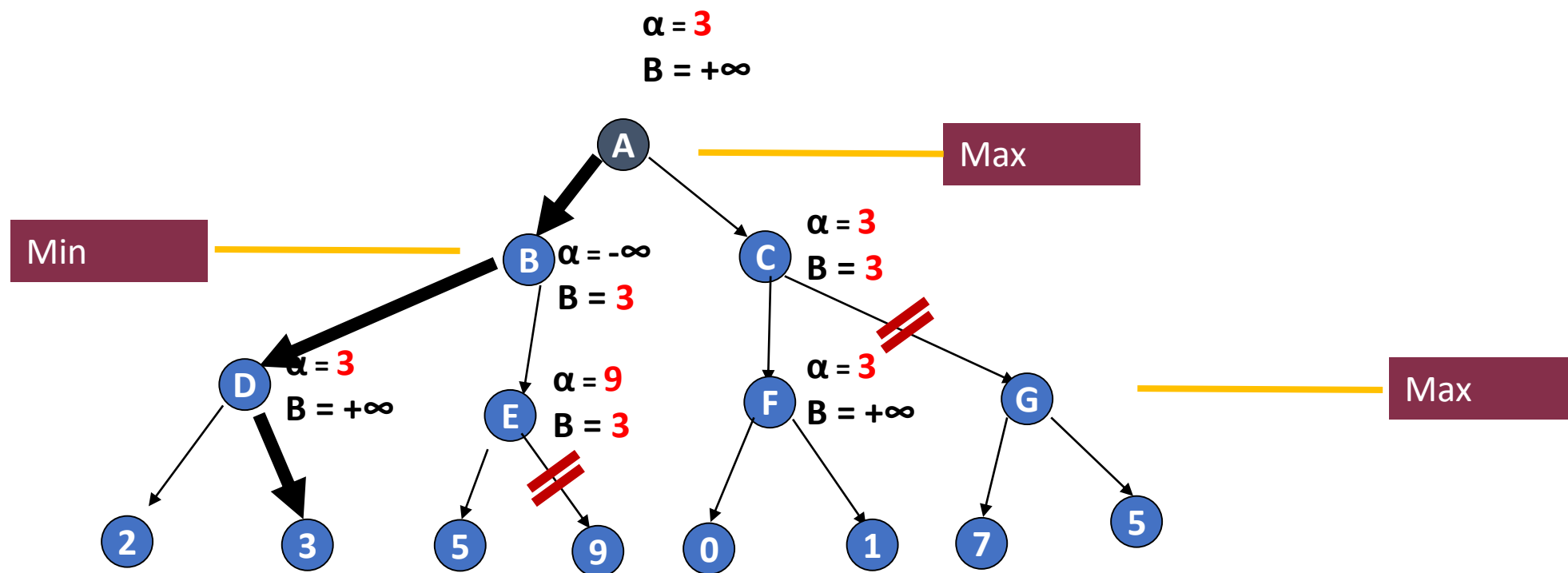
Alpha Beta Pruning Example



Alpha Beta Pruning Example



Alpha Beta Pruning Example



Additional Resources

- Game Playing

<https://pages.cs.wisc.edu/~dyer/cs540/notes/games.html>

- Alpha–Beta Pruning in Chess Game

<https://www.youtube.com/watch?v=l-hh51ncgDI>



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY



Thank You

