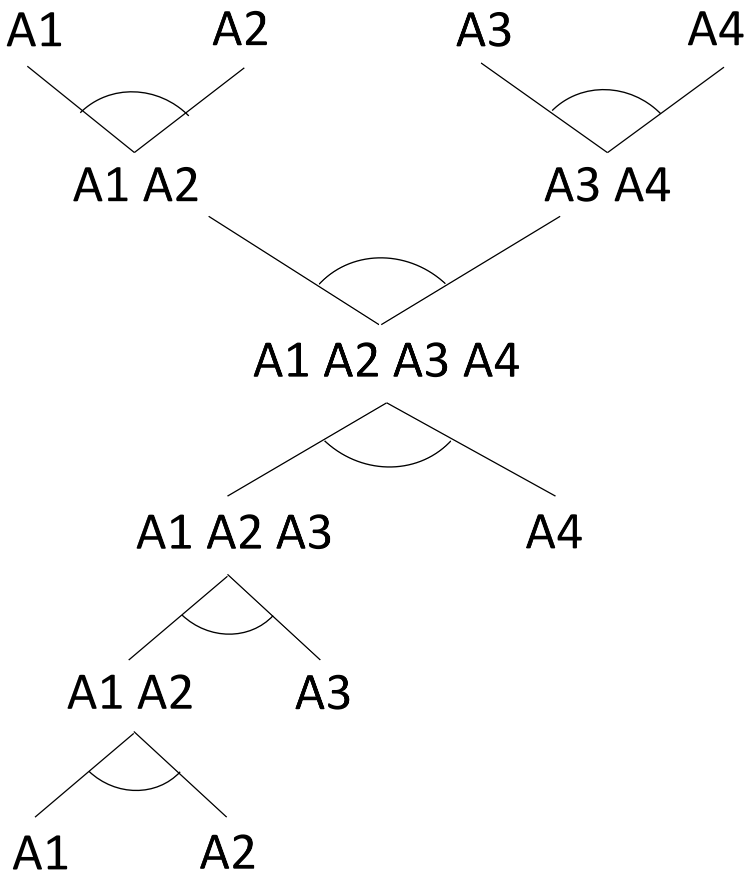
Searching Game Trees

Dr. Poulami Dalapati

Asst. Prof., Dept of CSE,

LNMIIT Jaipur

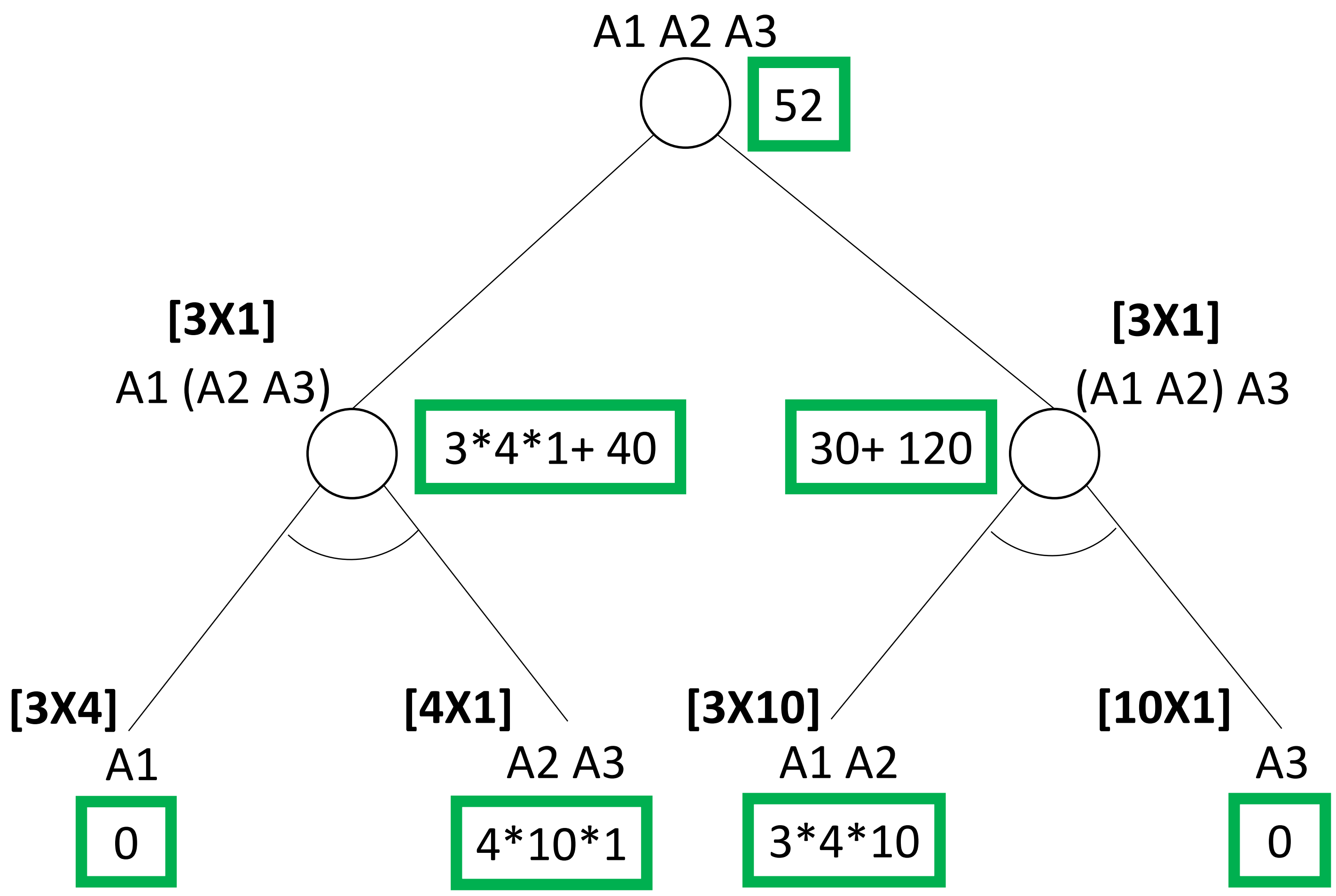
Problem Reduction Search



Matrix chain multiplication:

- Cost of multiplying two matrices A1 and A2 having order n x m and m x k respectively, is (n x m x k).
- We need to choose the optimal multiplication ordering so that cost can be minimized.

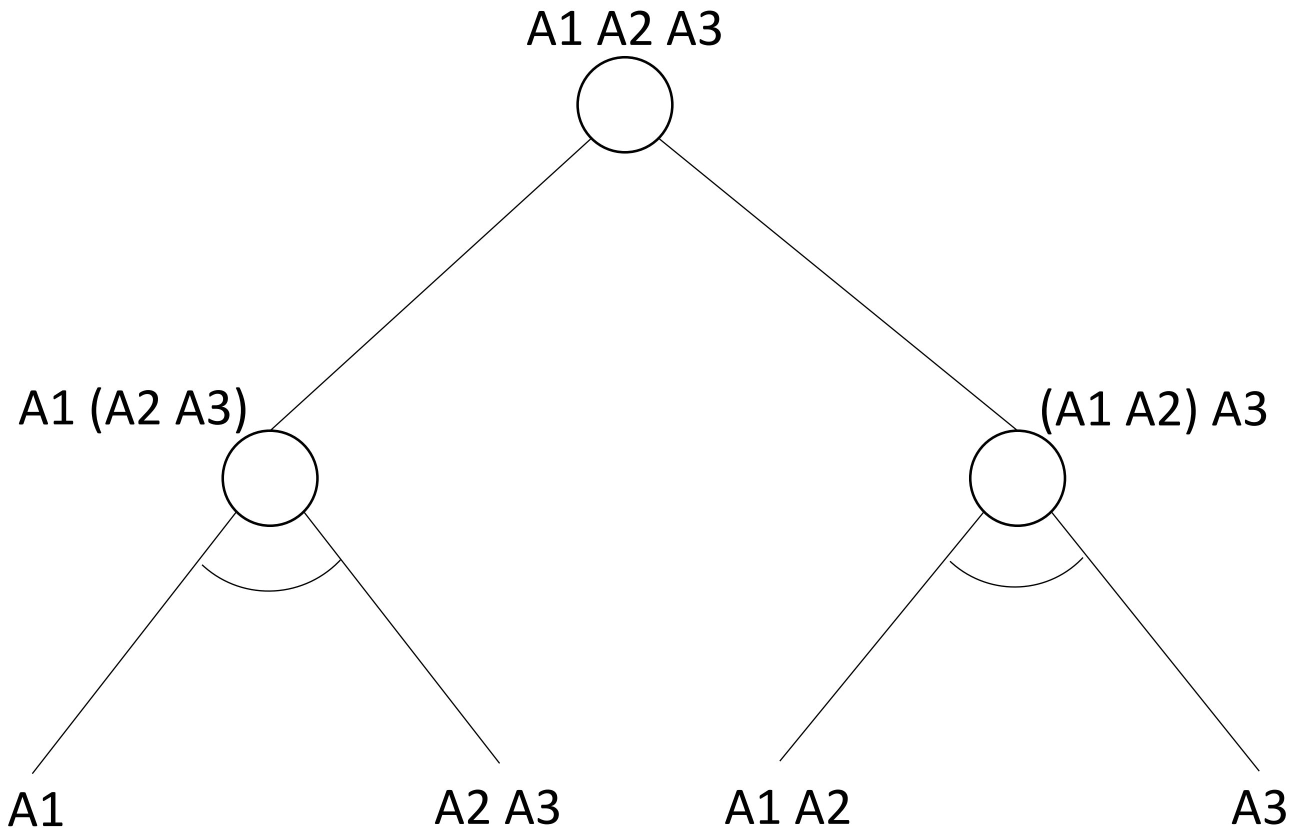
AND-OR Graph



$A1_{3x4} \times A2_{4x10} \times A3_{10x1}$

- An OR node represents a choice between possible decomposition.
- An AND node represents a given decomposition.
- Problem definition can be described using three tuples <G,S,T>, where G is AND-OR graph, S is the start state, T is the set of terminals.
- Our Goal is to find minimum cost solution tree.

AND-OR Graph

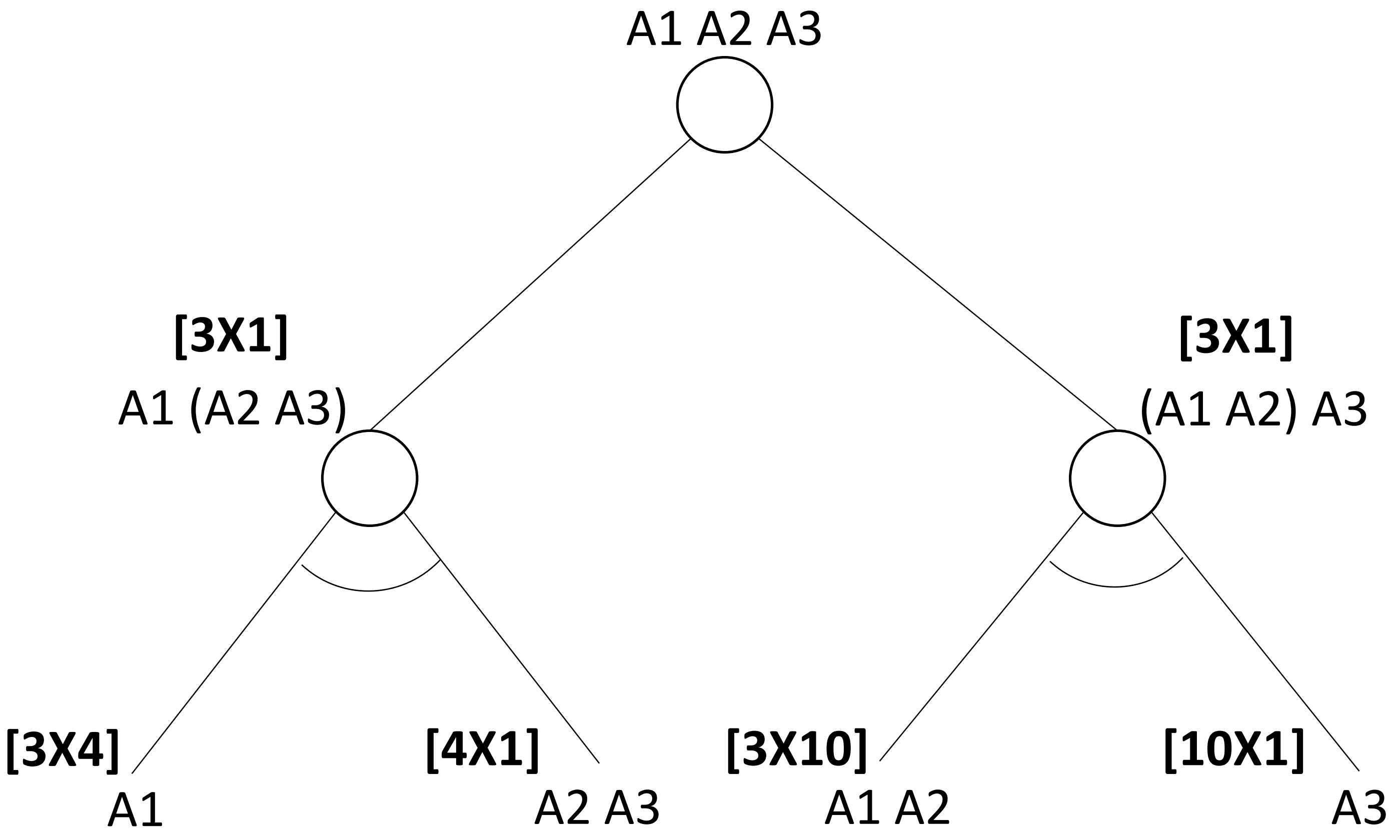


A1 -> 3 X 4

A2 -> 4 X 10

A3 -> 10 X 1

AND-OR Graph

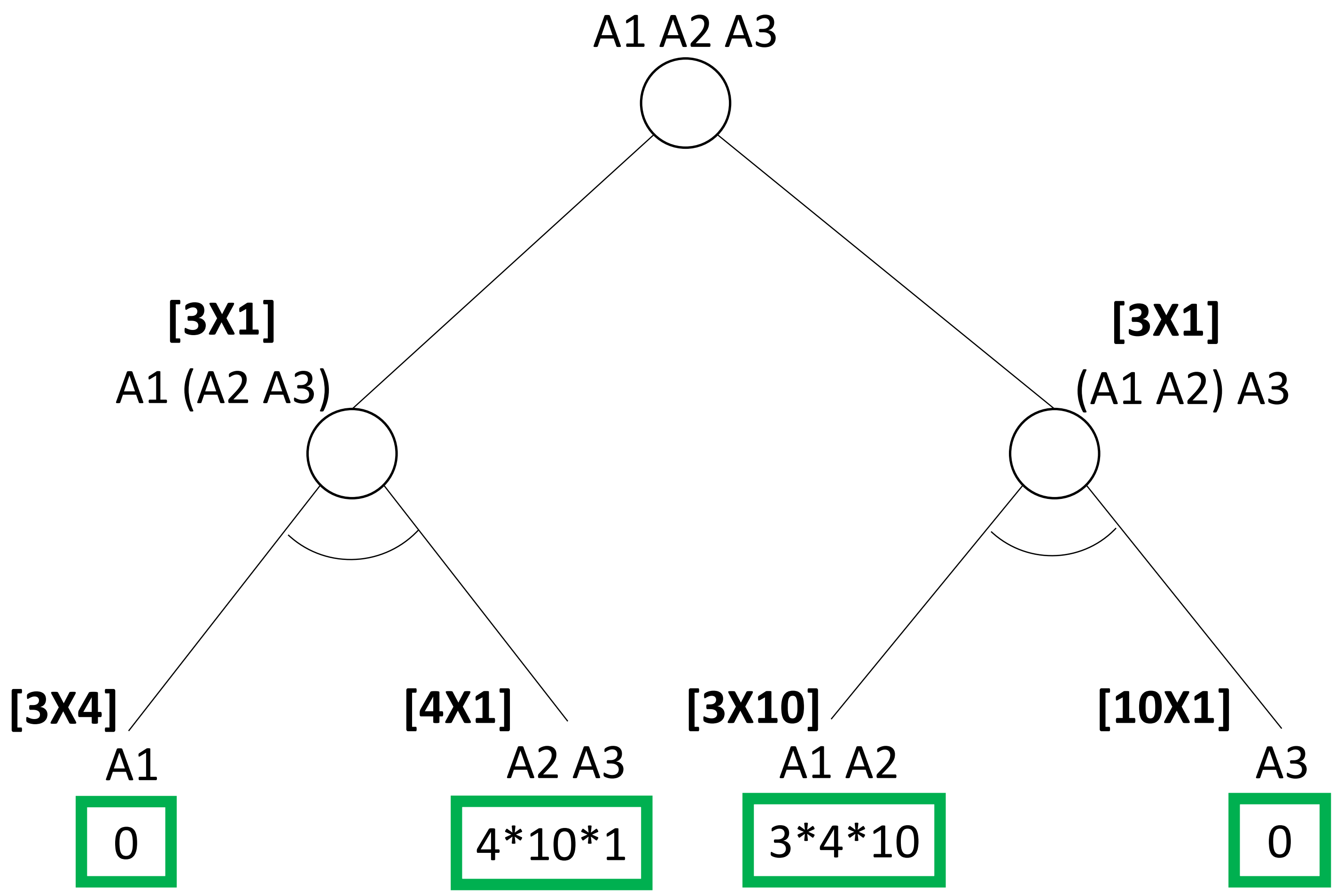


A1 -> 3 X 4

A2 -> 4 X 10

A3 -> 10 X 1

AND-OR Graph

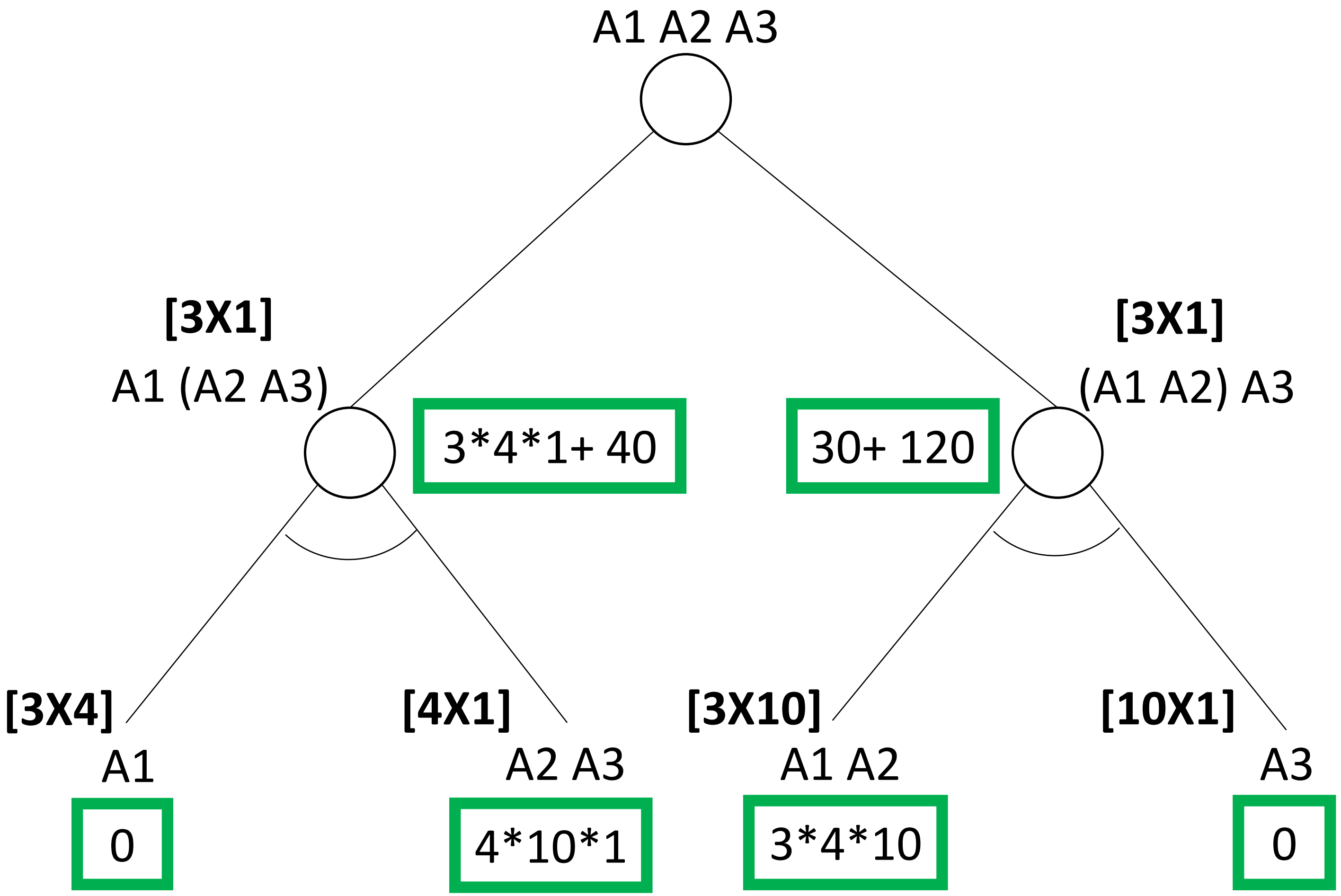


A1 -> 3 X 4

A2 -> 4 X 10

A3 -> 10 X 1

AND-OR Graph

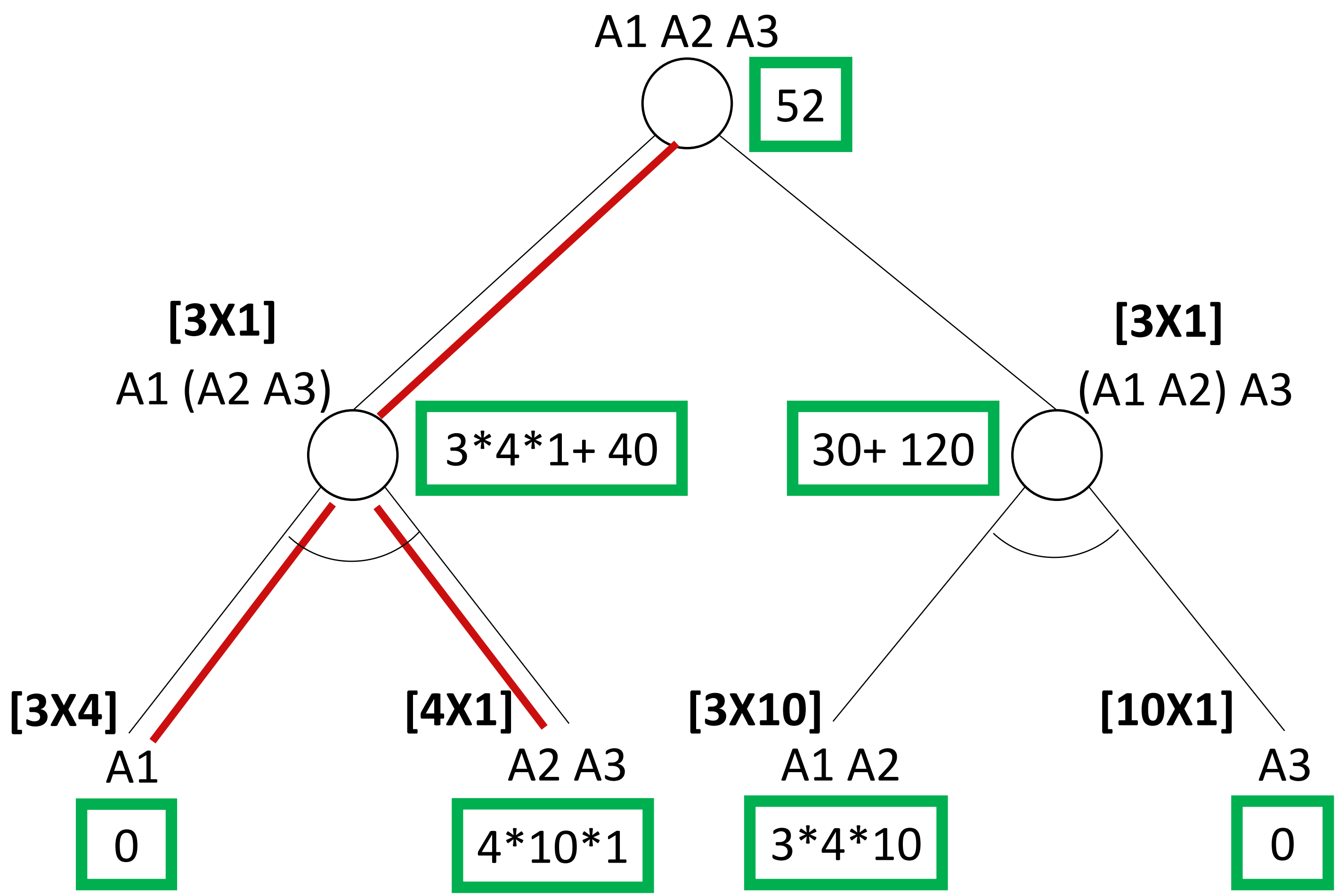


A1 -> 3 X 4

A2 -> 4 X 10

A3 -> 10 X 1

AND-OR Graph

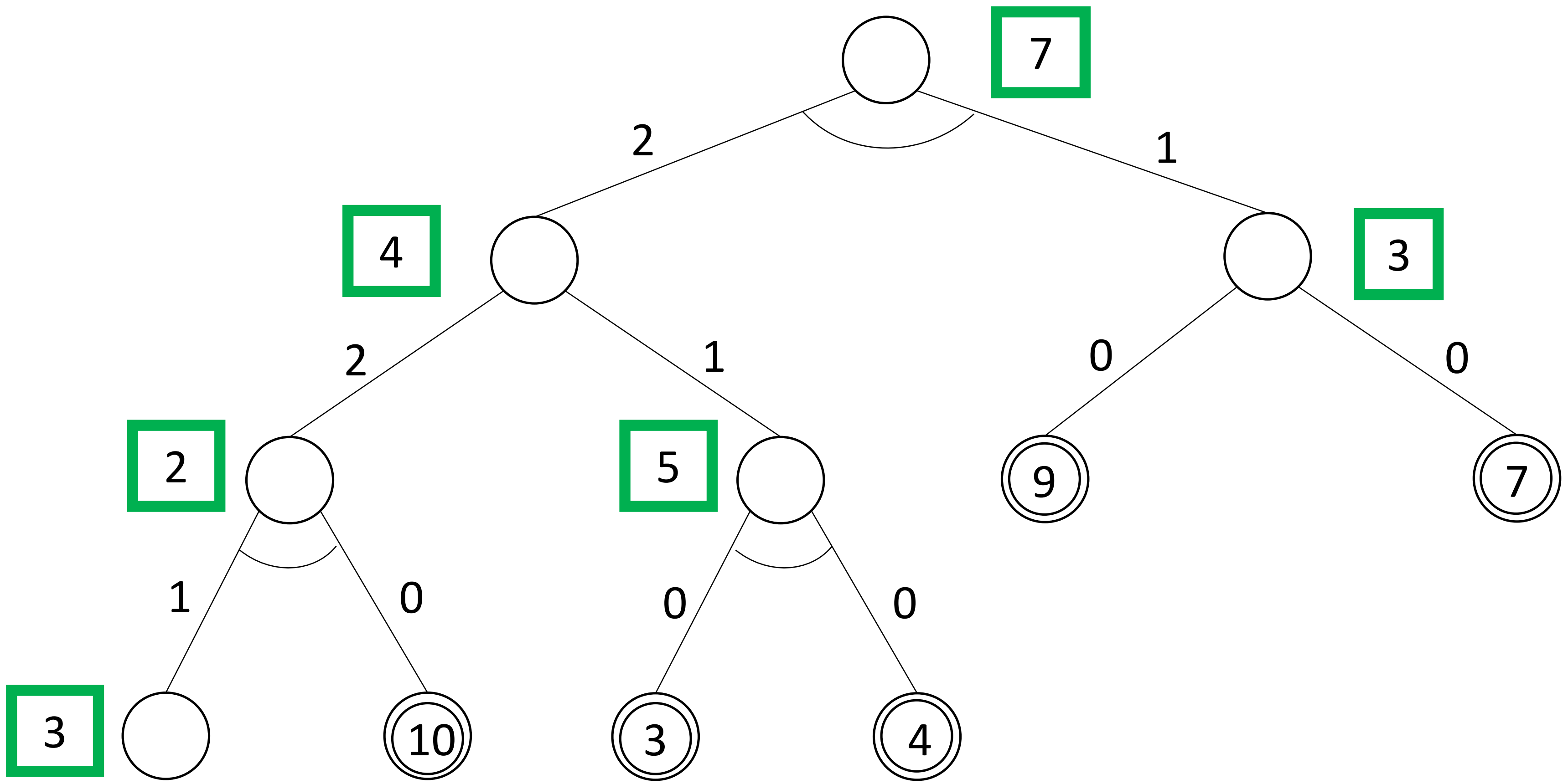


A1 -> 3 X 4

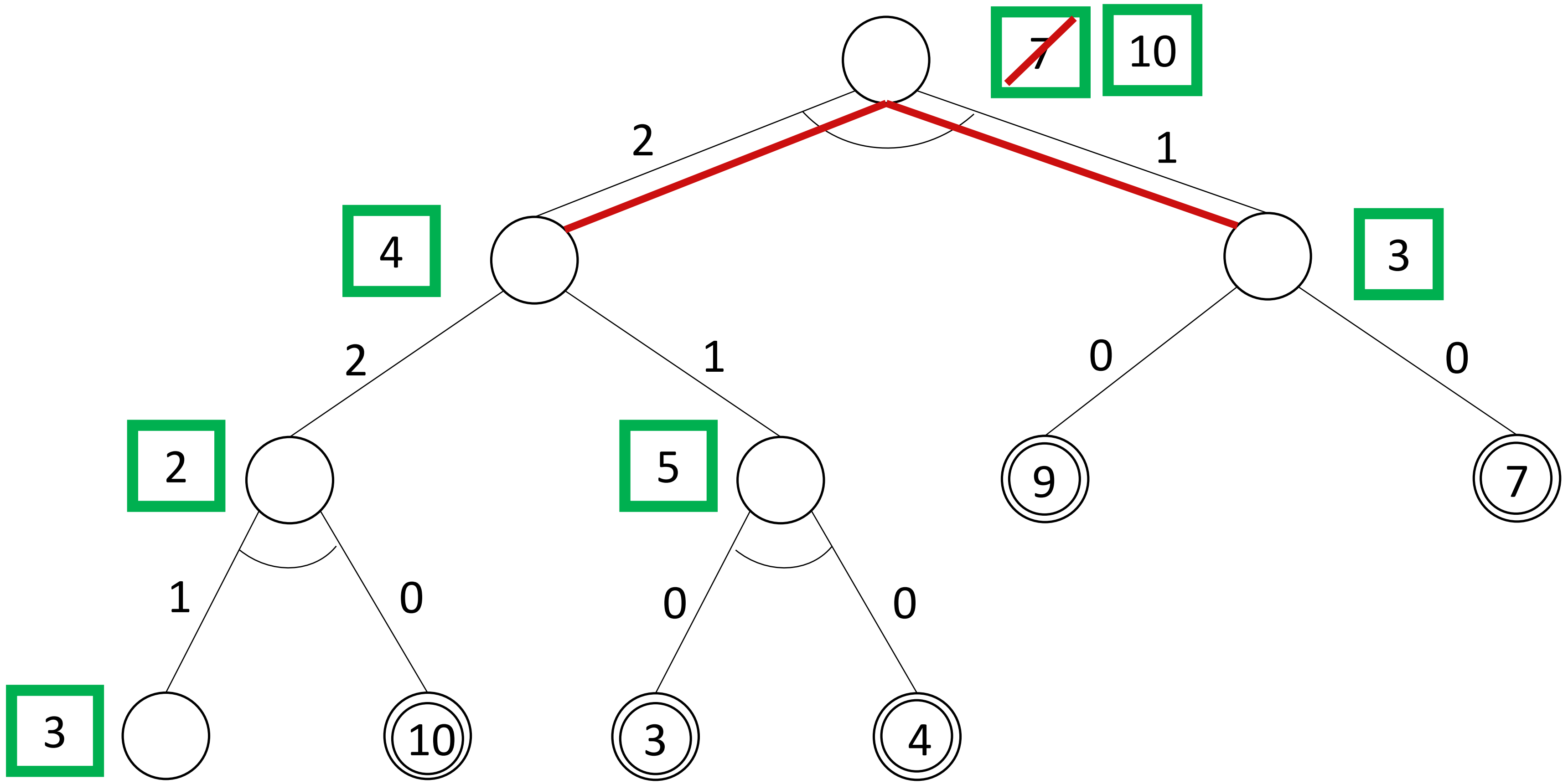
A2 -> 4 X 10

A3 -> 10 X 1

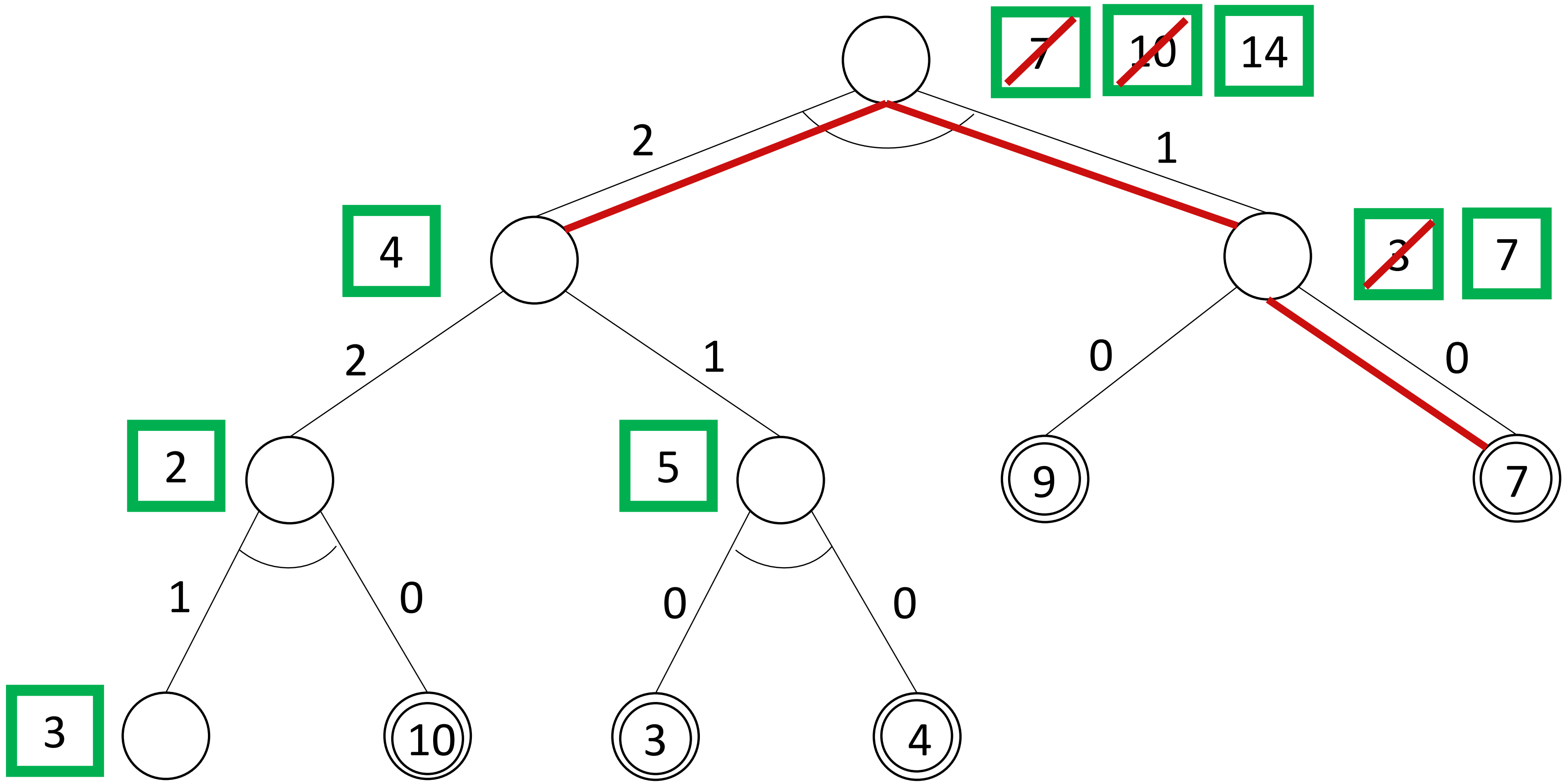
AND-OR Graph (with underestimated heuristic)



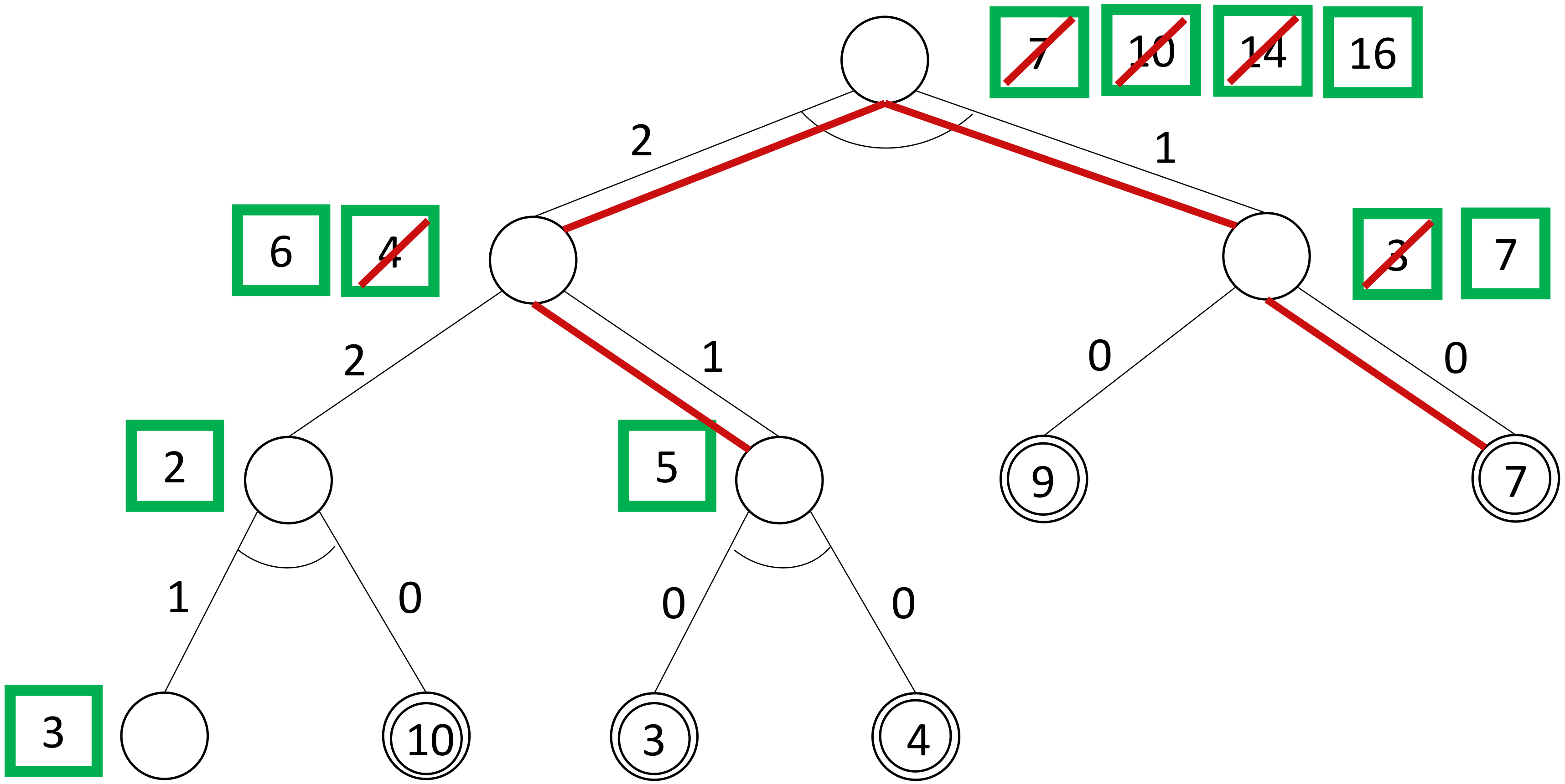
AND-OR Graph (with underestimated heuristic)



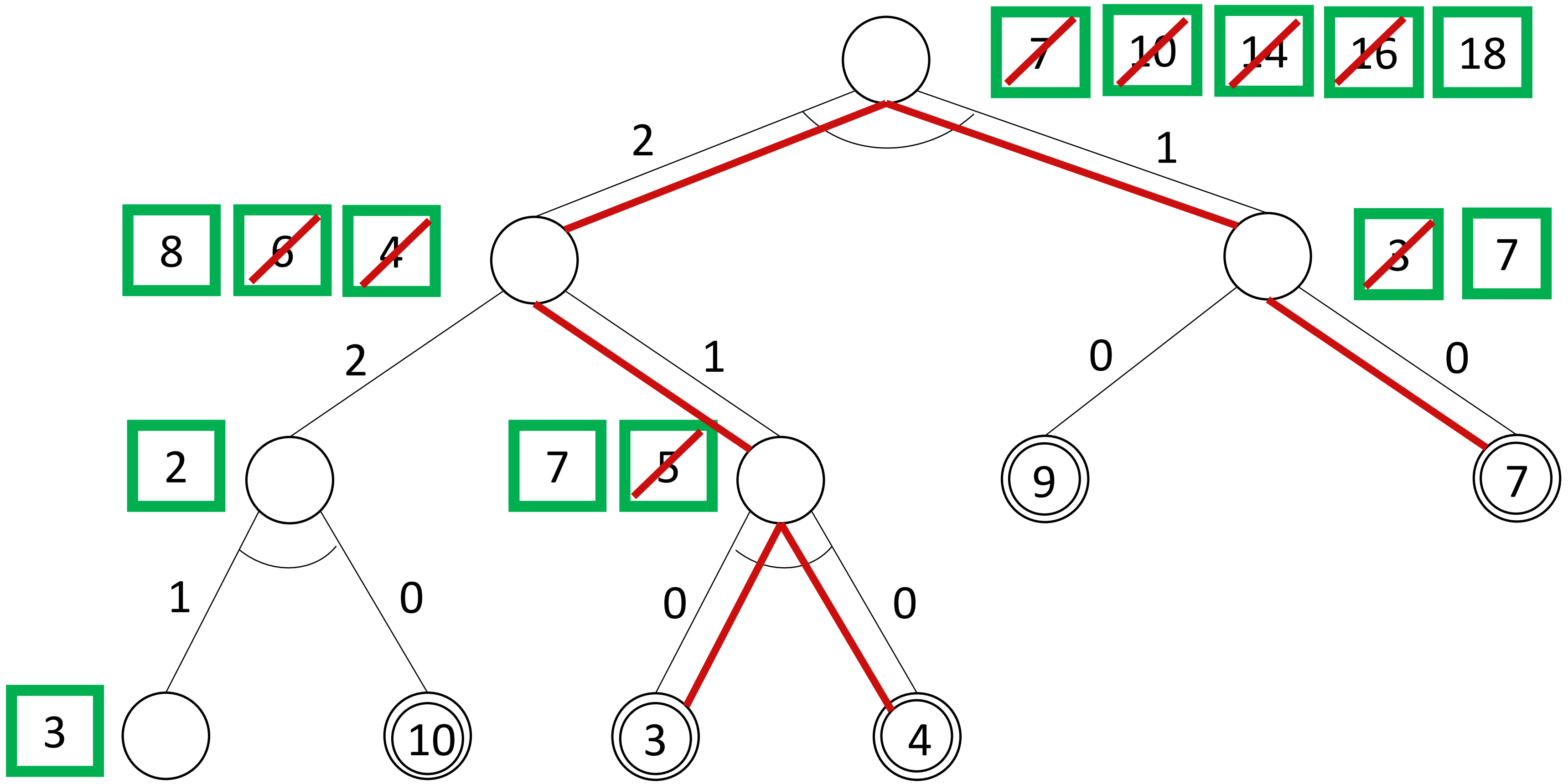
AND-OR Graph (with underestimated heuristic)



AND-OR Graph (with underestimated heuristic)

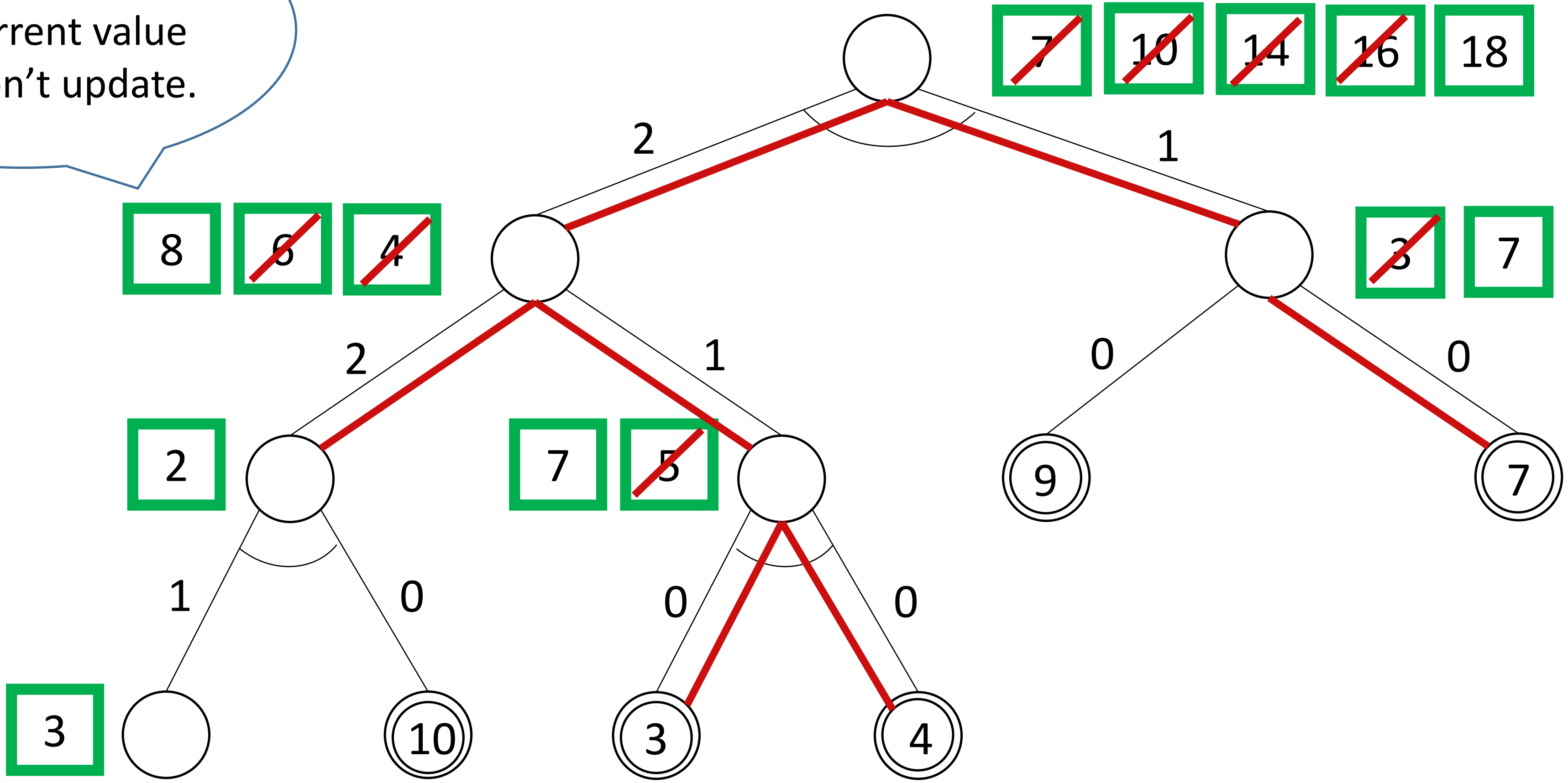


AND-OR Graph (with underestimated heuristic)



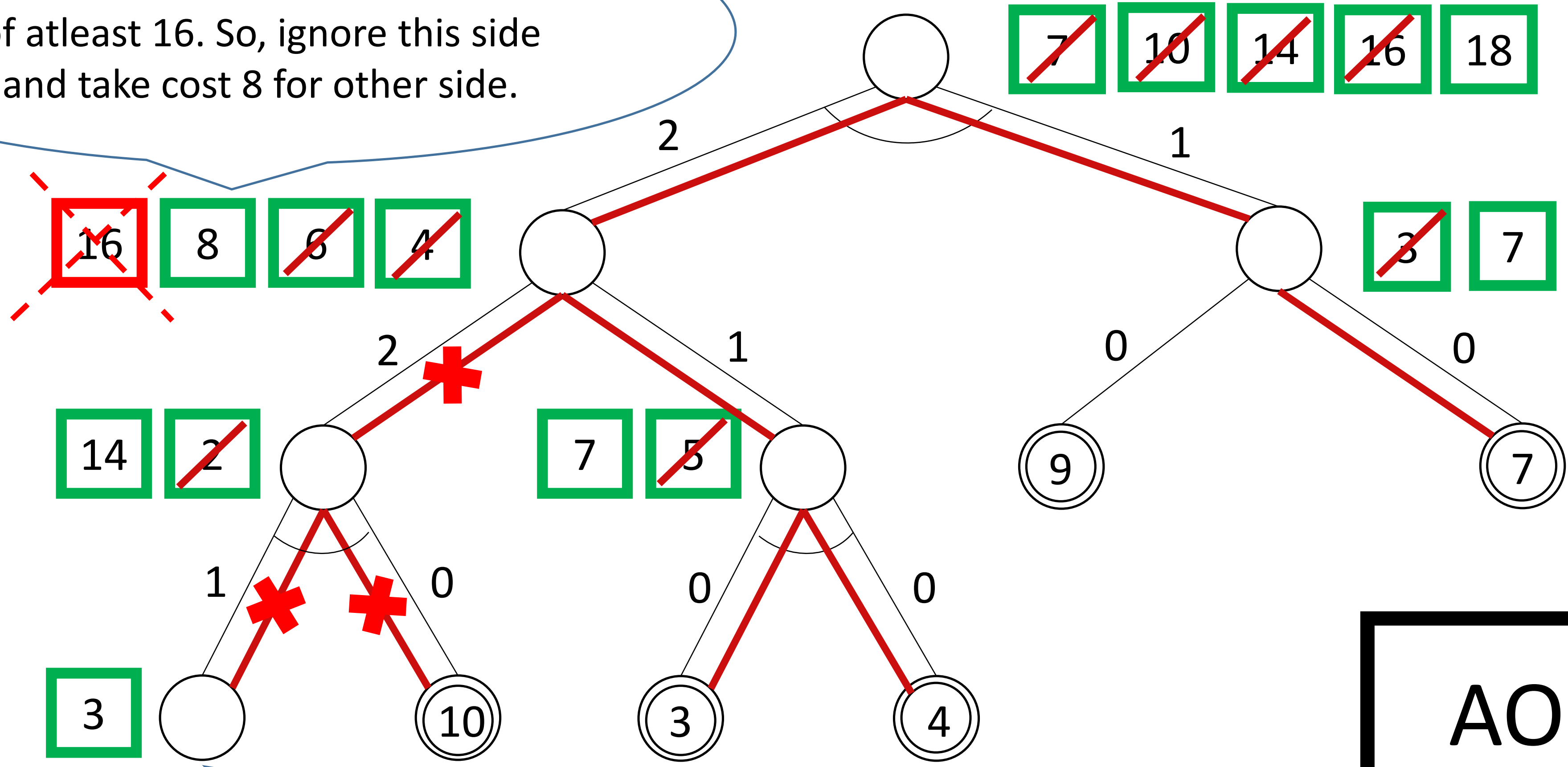
AND-OR Graph (with underestimated heuristic)

(2+2)=4 is lower than current value 8. So, don't update.



AND-OR Graph (with underestimated heuristic)

left side of this OR node has a cost of atleast 16. So, ignore this side and take cost 8 for other side.



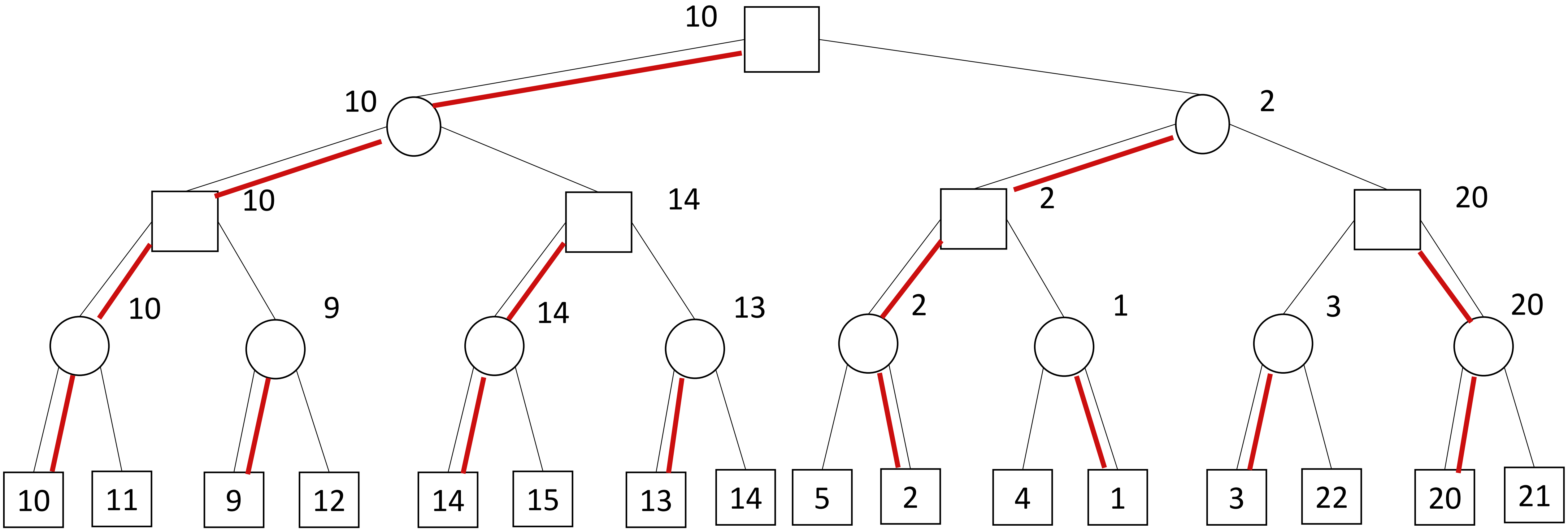
As we are ignoring this side we are not further decomposing this node.

AO*

Searching Game Trees- MINIMAX

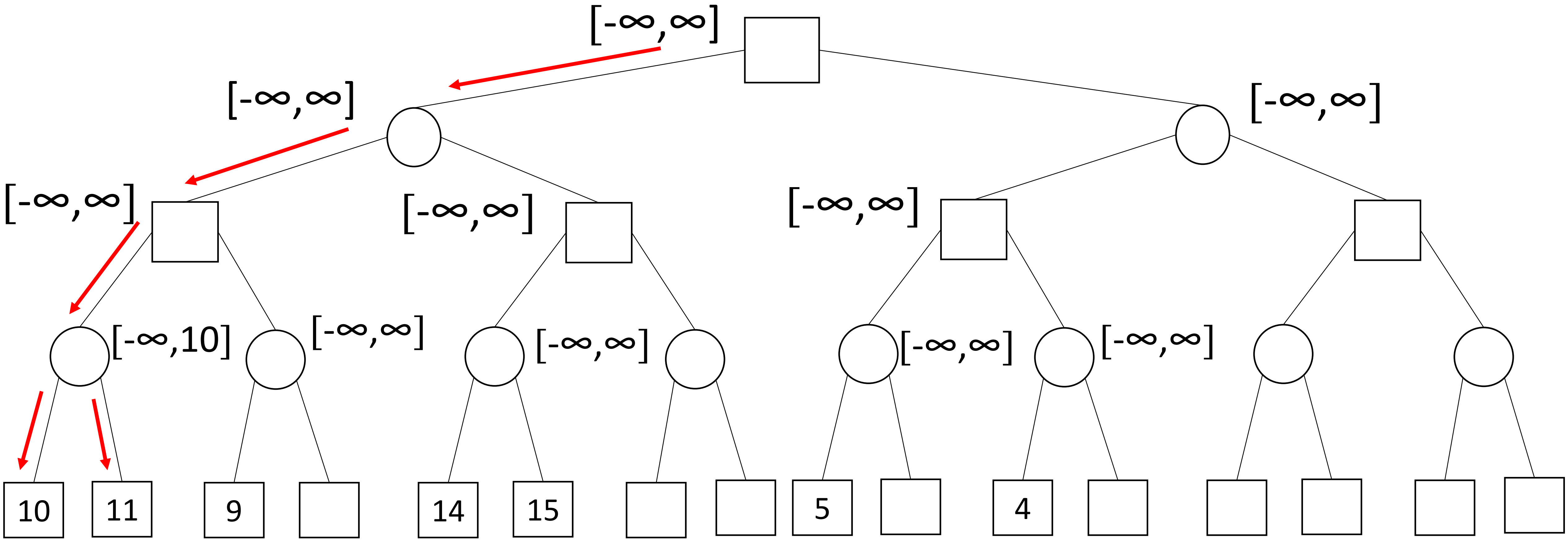
- Consider an OR tree with two types of OR nodes - Min node and Max node.
- In Min node, select the minimum cost successor.
- In Max node, select the maximum cost successor.
- Terminal nodes are either winning or loosing states.
- If terminal node searching is infeasible , then we use heuristic cost to compare non terminal nodes.
- Max node is denoted by a rectangle □ or by a triangle △
- Max node is denoted by a circle ○ or by a upside down triangle ▽
- In a Zero Sum Game if player A gains, other player B will loose.
- Cost indicates badness whereas utility indicates goodness.
- MINIMAX is applicable for Zero Sum Game.
- For Non-Zero Sum Game we have MAXIMIN.

MINIMAX

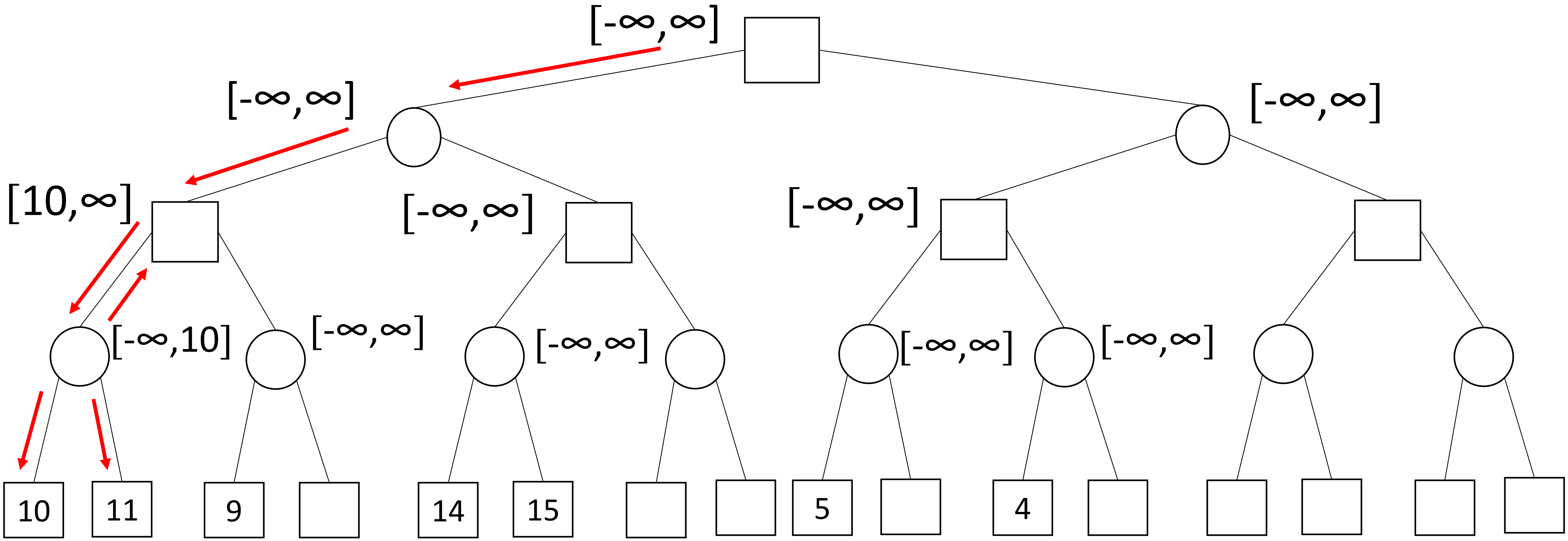


These values indicate the utility or goodness of player A. In max node, player A will try to maximize its utility. On the other hand player B will try to minimize the utility of player A.

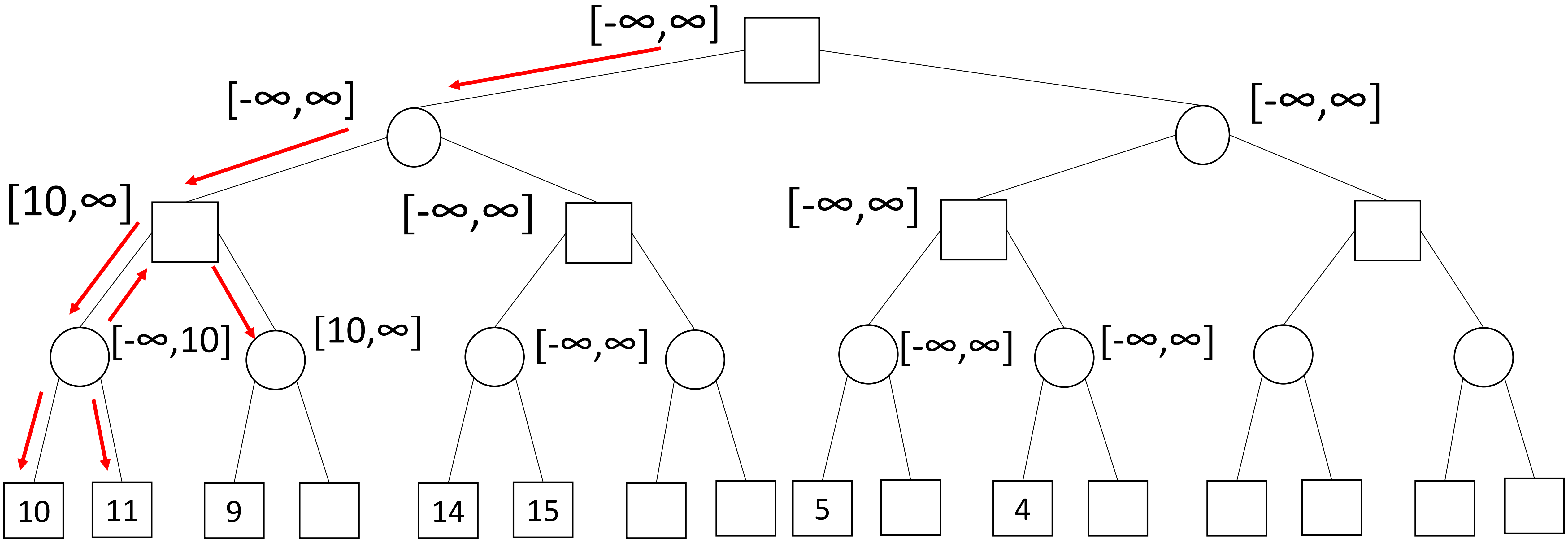
MINIMAX (with alpha-beta pruning)



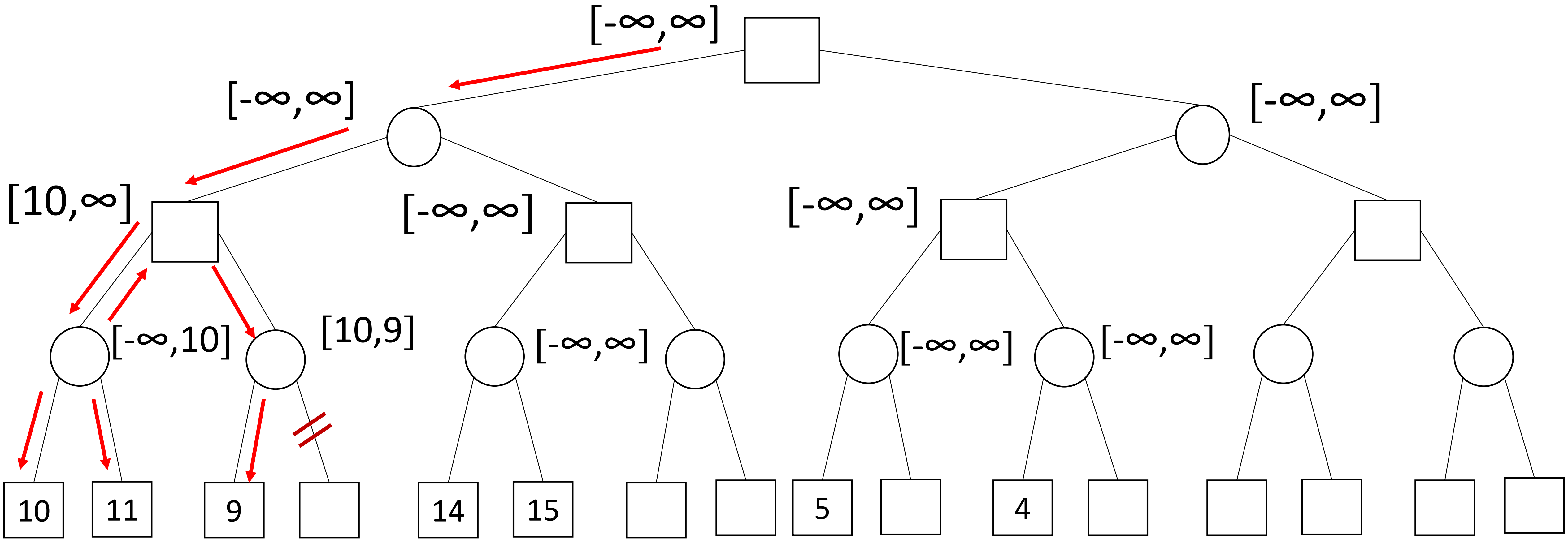
MINIMAX (with alpha-beta pruning)



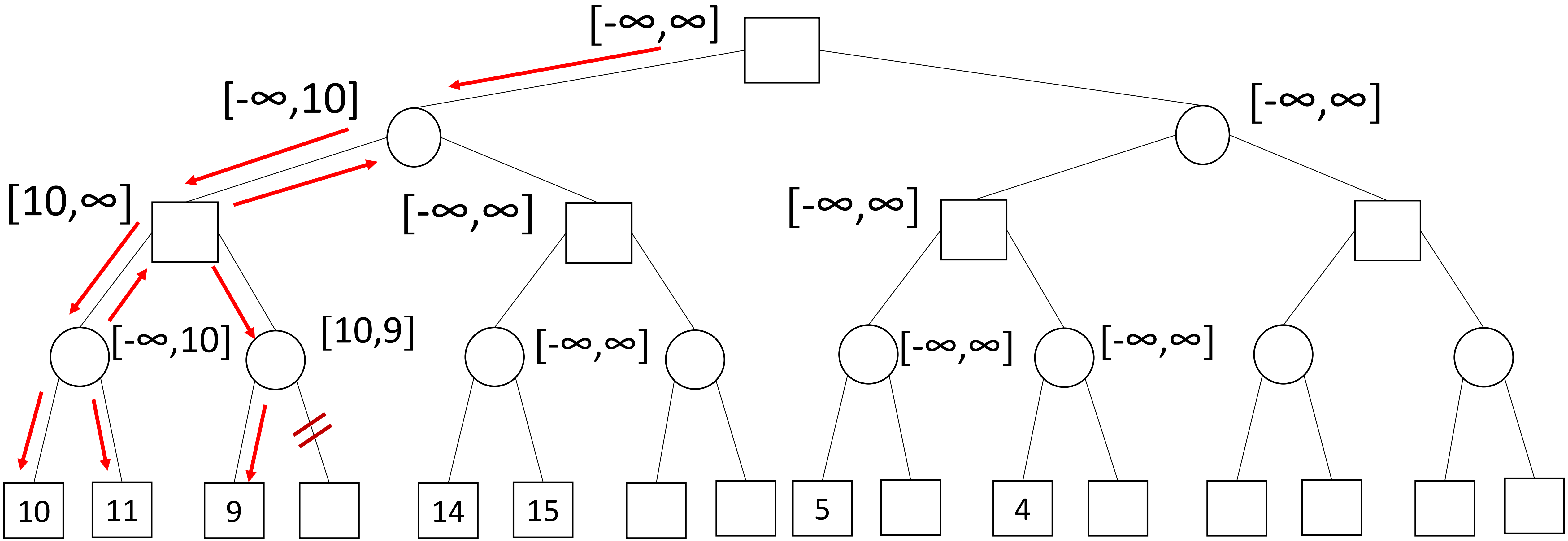
MINIMAX (with alpha-beta pruning)



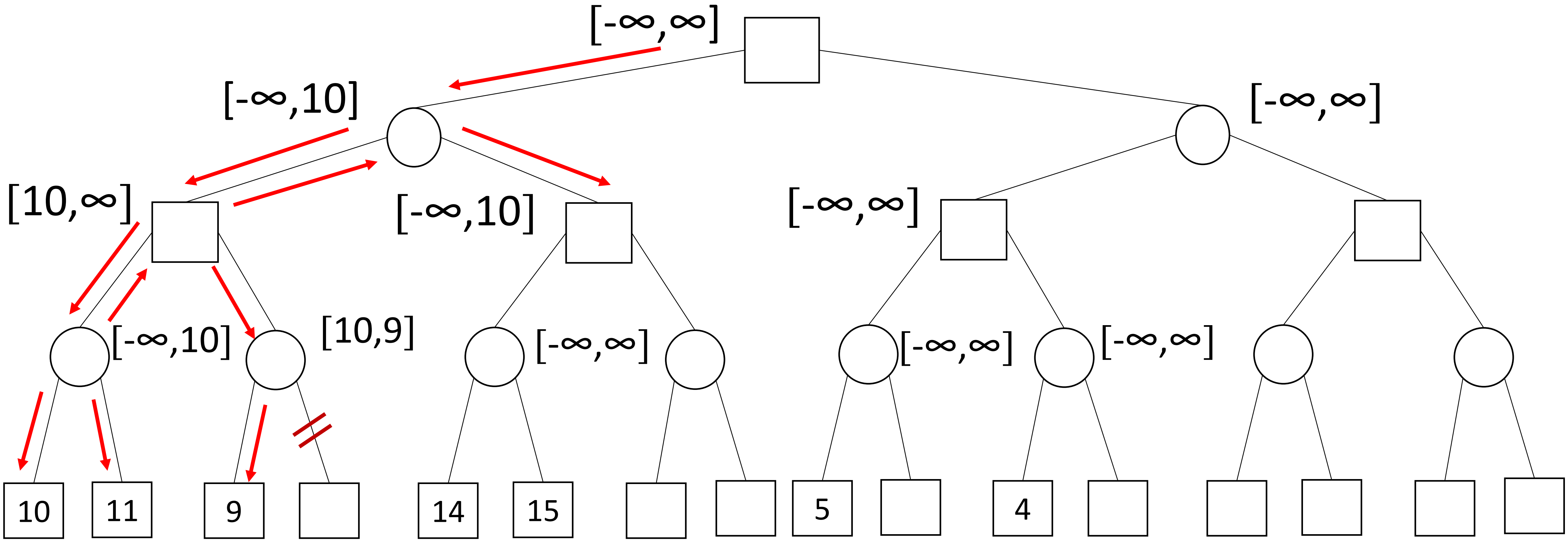
MINIMAX (with alpha-beta pruning)



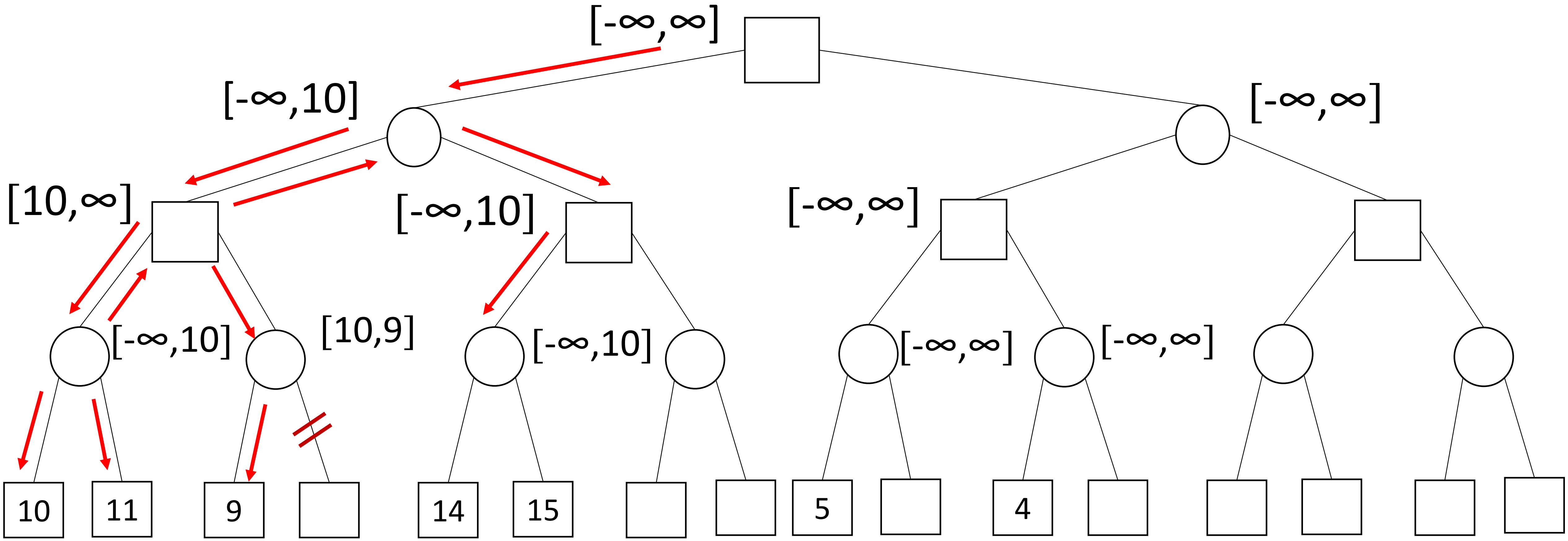
MINIMAX (with alpha-beta pruning)



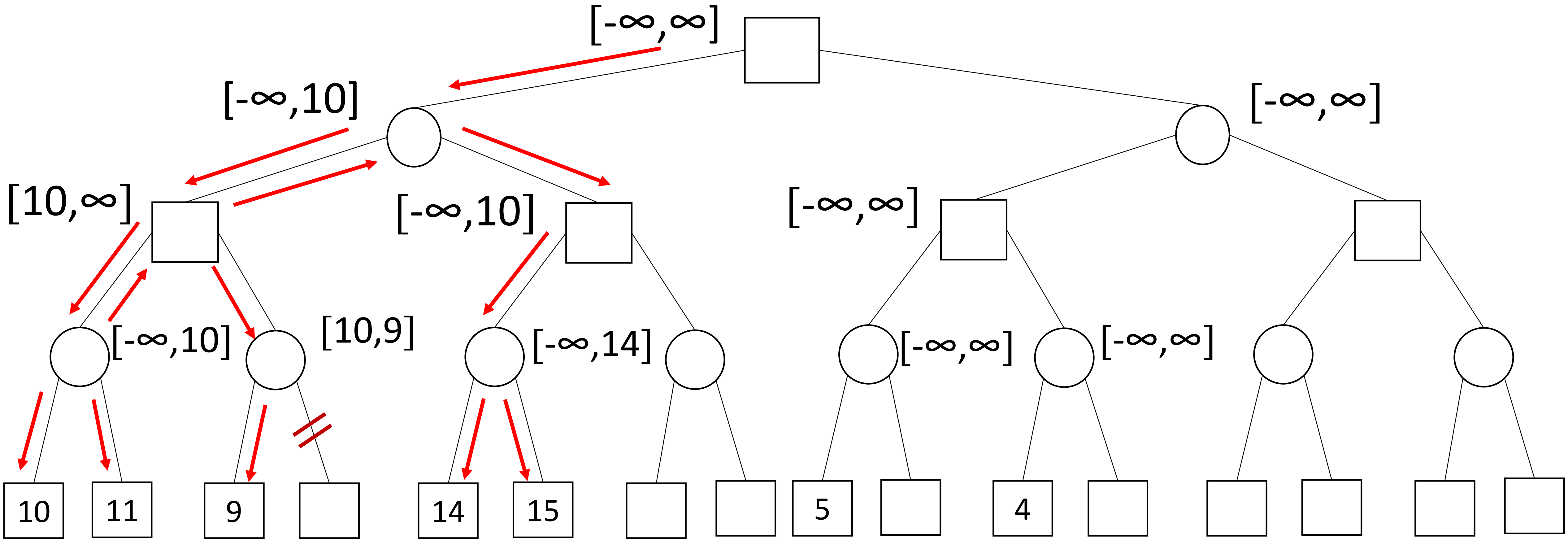
MINIMAX (with alpha-beta pruning)



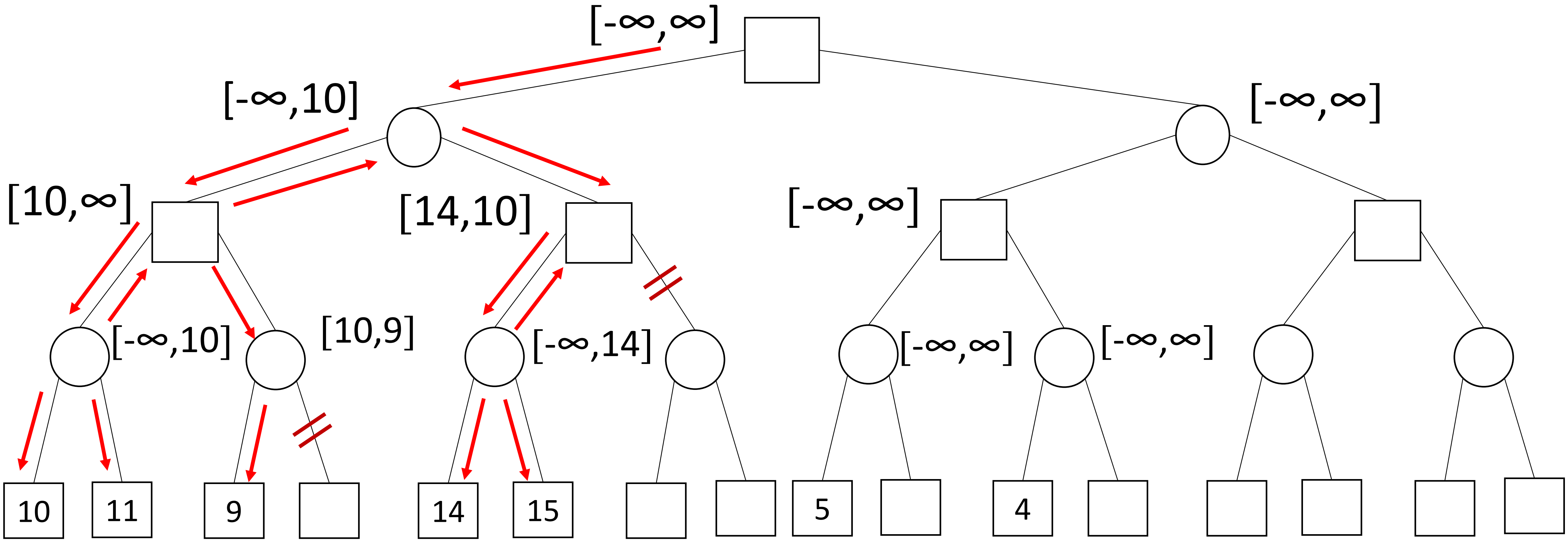
MINIMAX (with alpha-beta pruning)



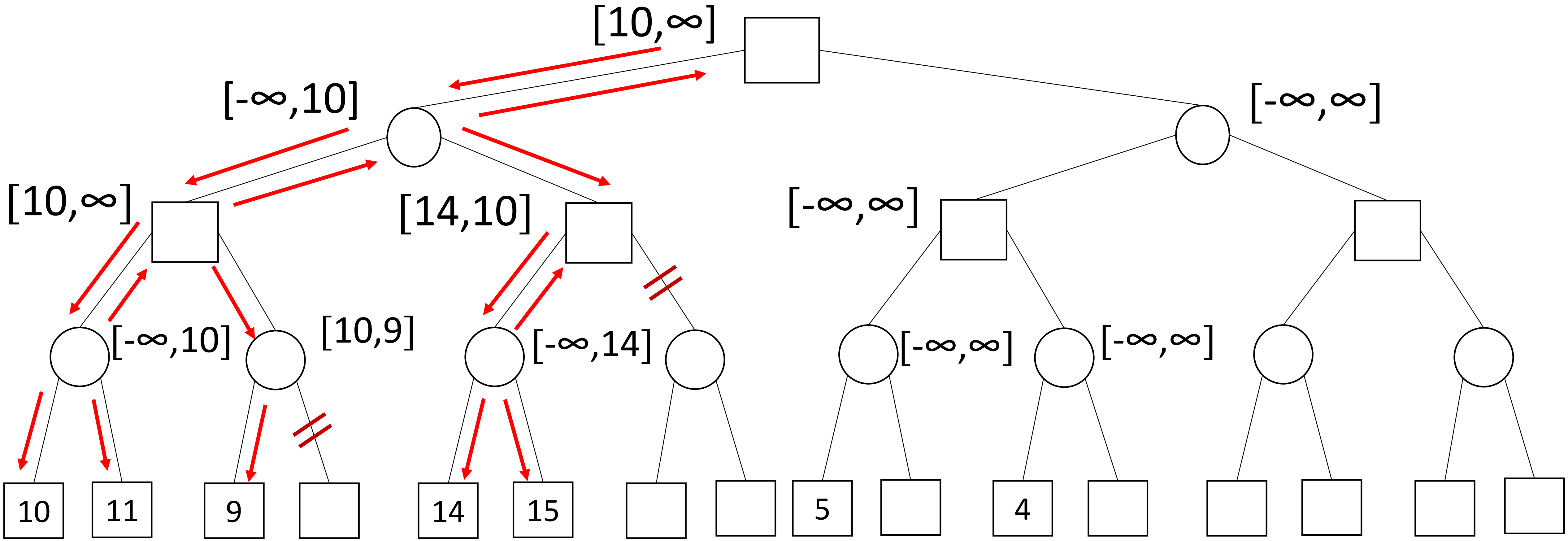
MINIMAX (with alpha-beta pruning)



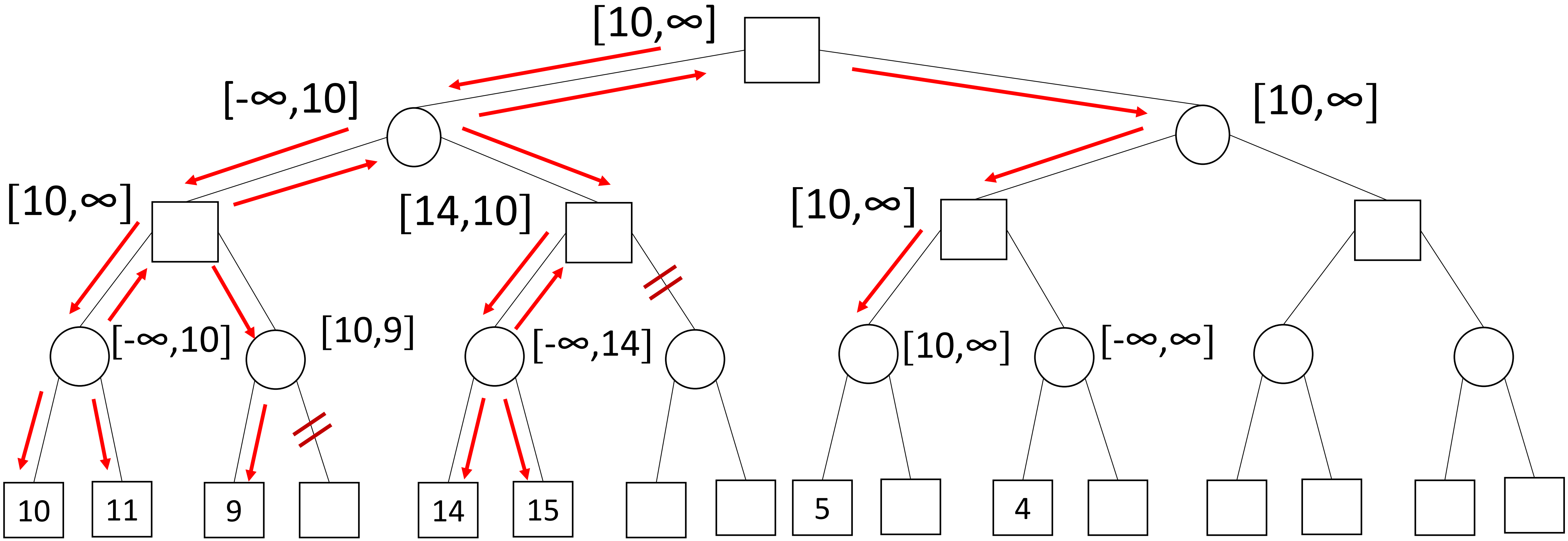
MINIMAX (with alpha-beta pruning)



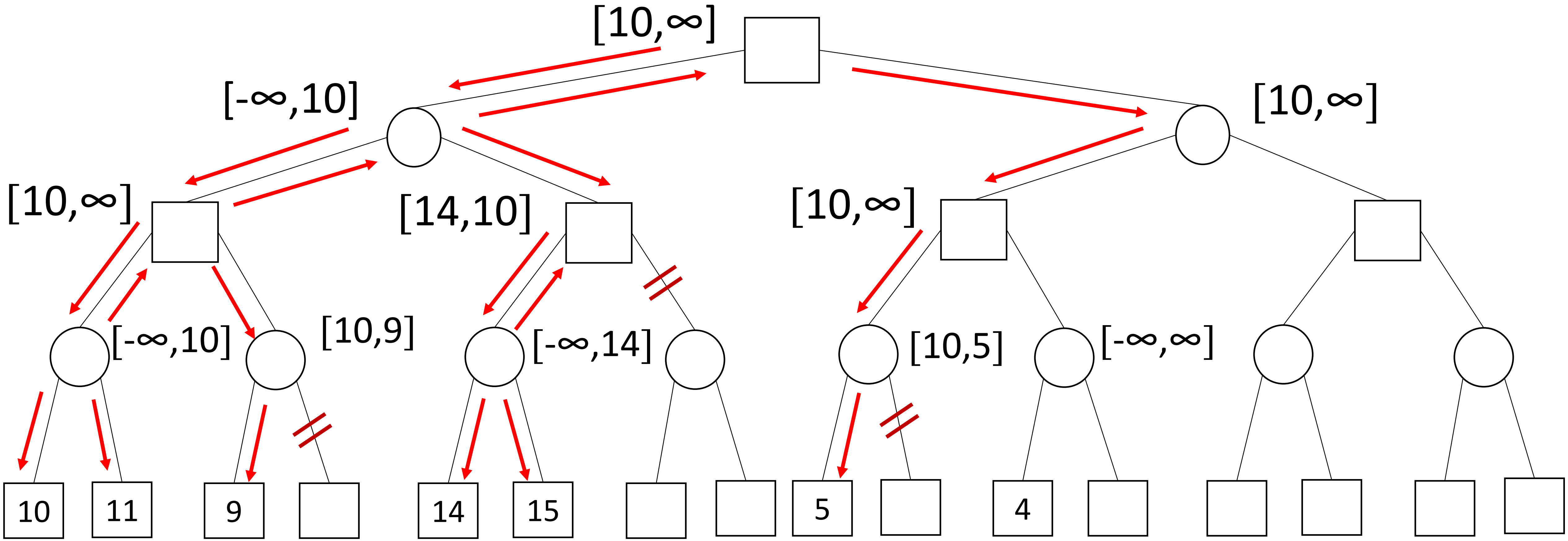
MINIMAX (with alpha-beta pruning)



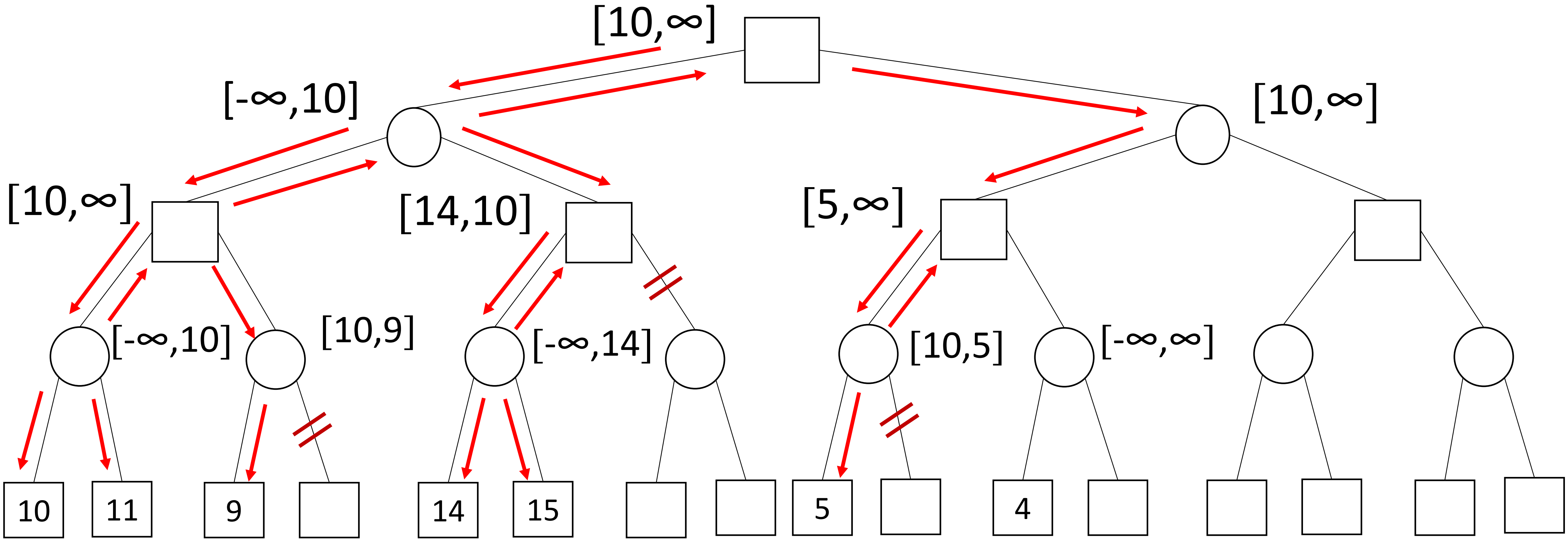
MINIMAX (with alpha-beta pruning)



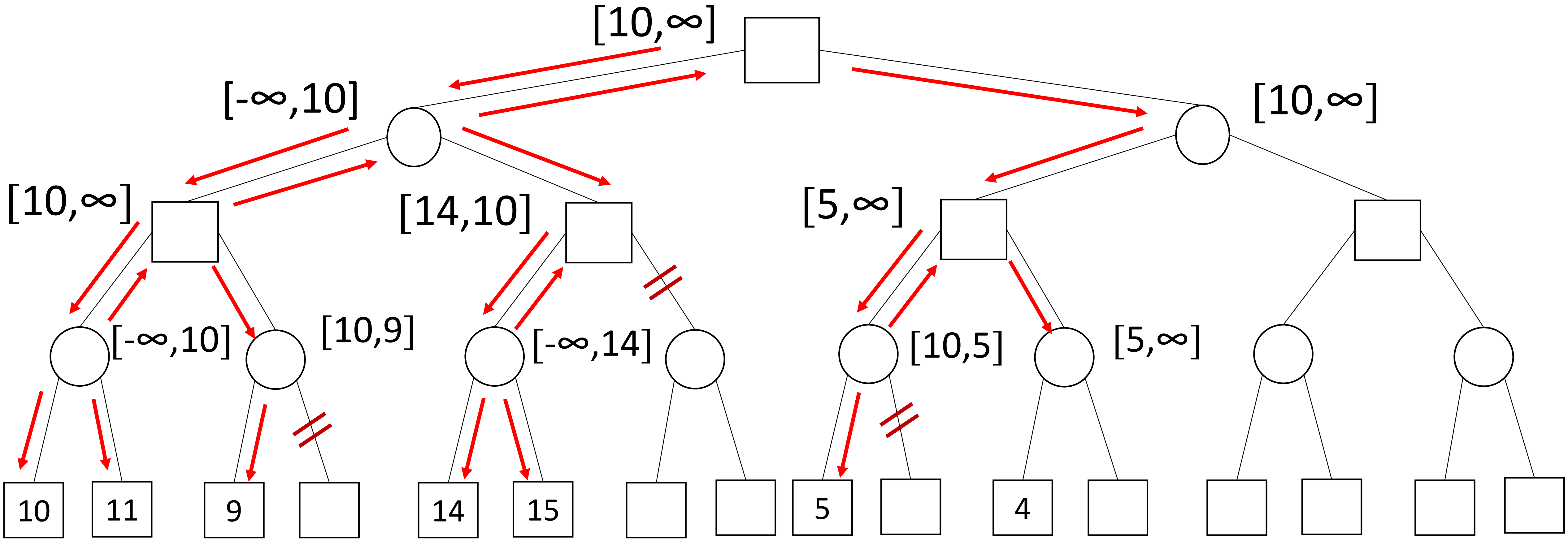
MINIMAX (with alpha-beta pruning)



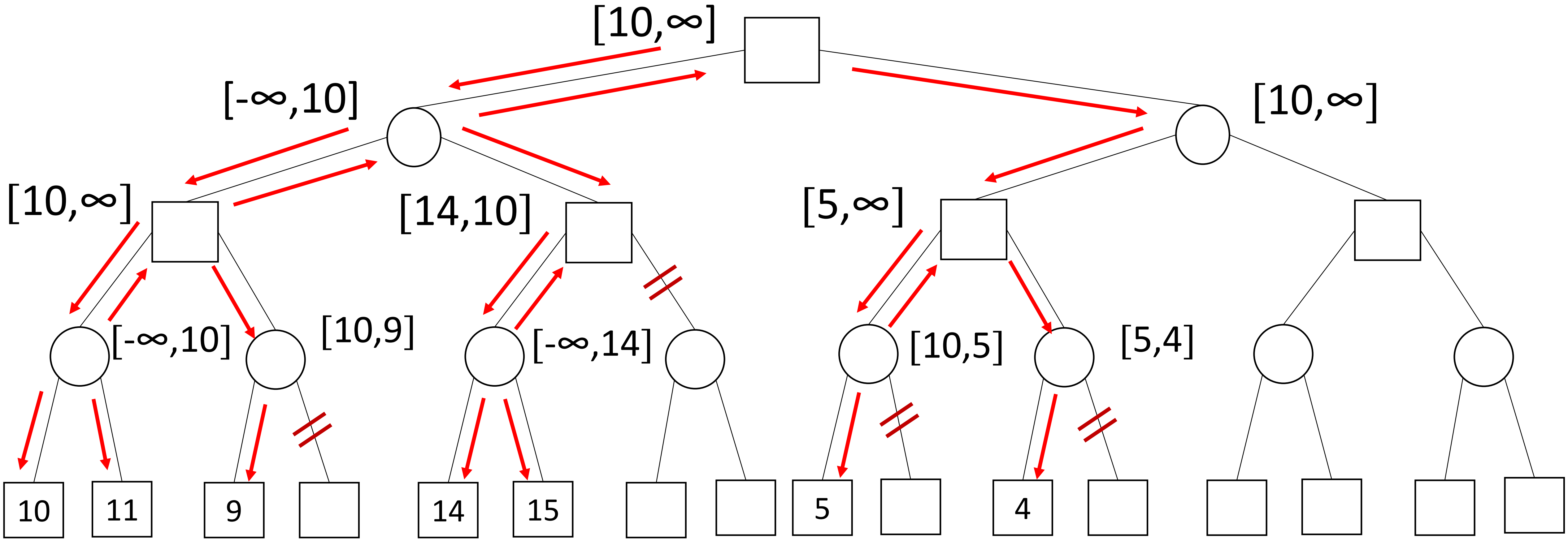
MINIMAX (with alpha-beta pruning)



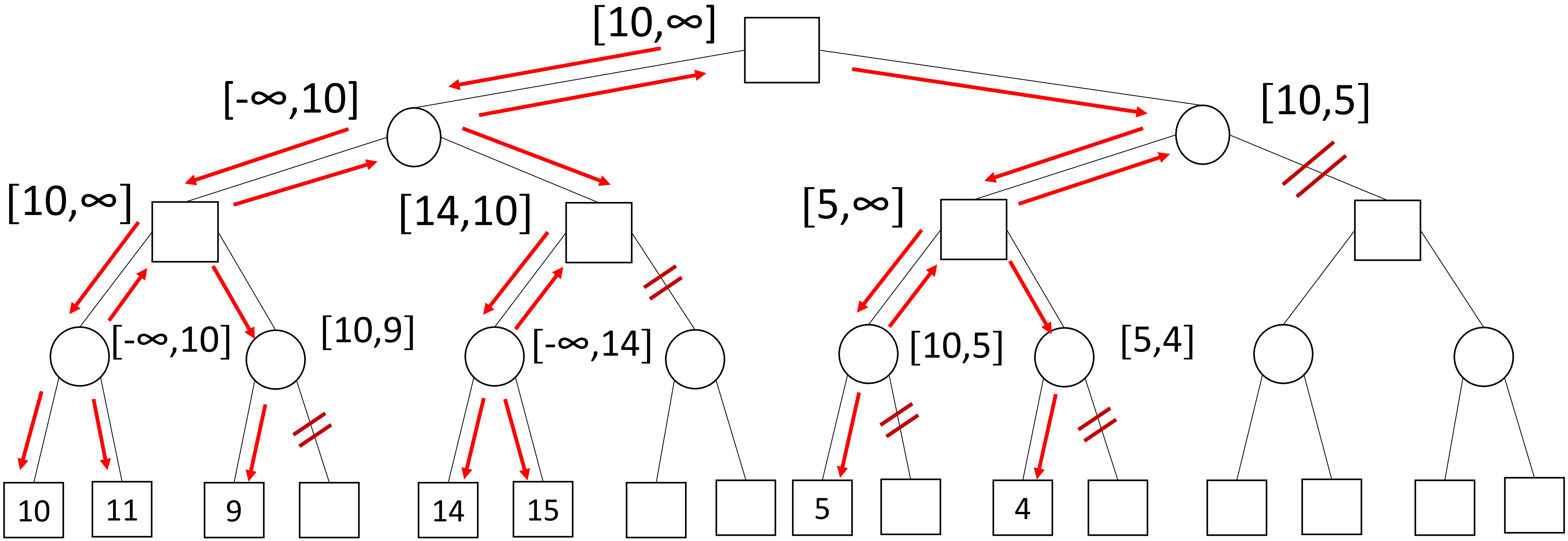
MINIMAX (with alpha-beta pruning)



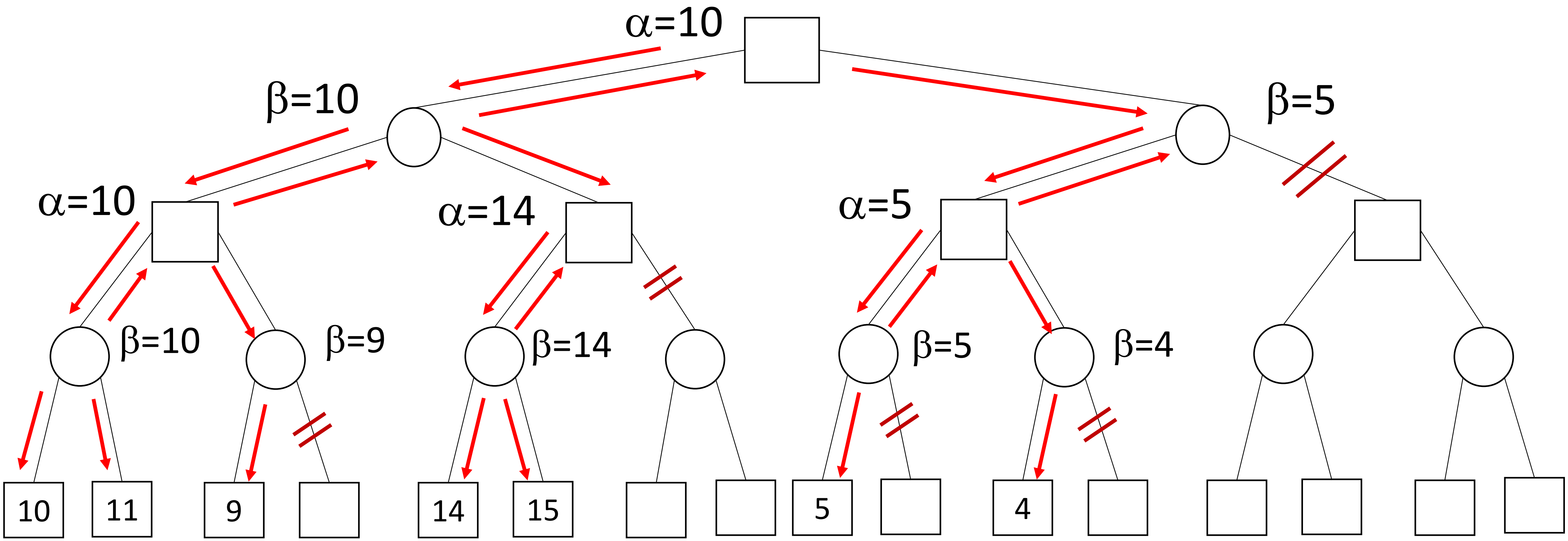
MINIMAX (with alpha-beta pruning)



MINIMAX (with alpha-beta pruning)



MINIMAX (with alpha-beta pruning)



Properties of MINIMAX

Complete?

- Yes

Optimal?

- Yes (against an optimal opponent)

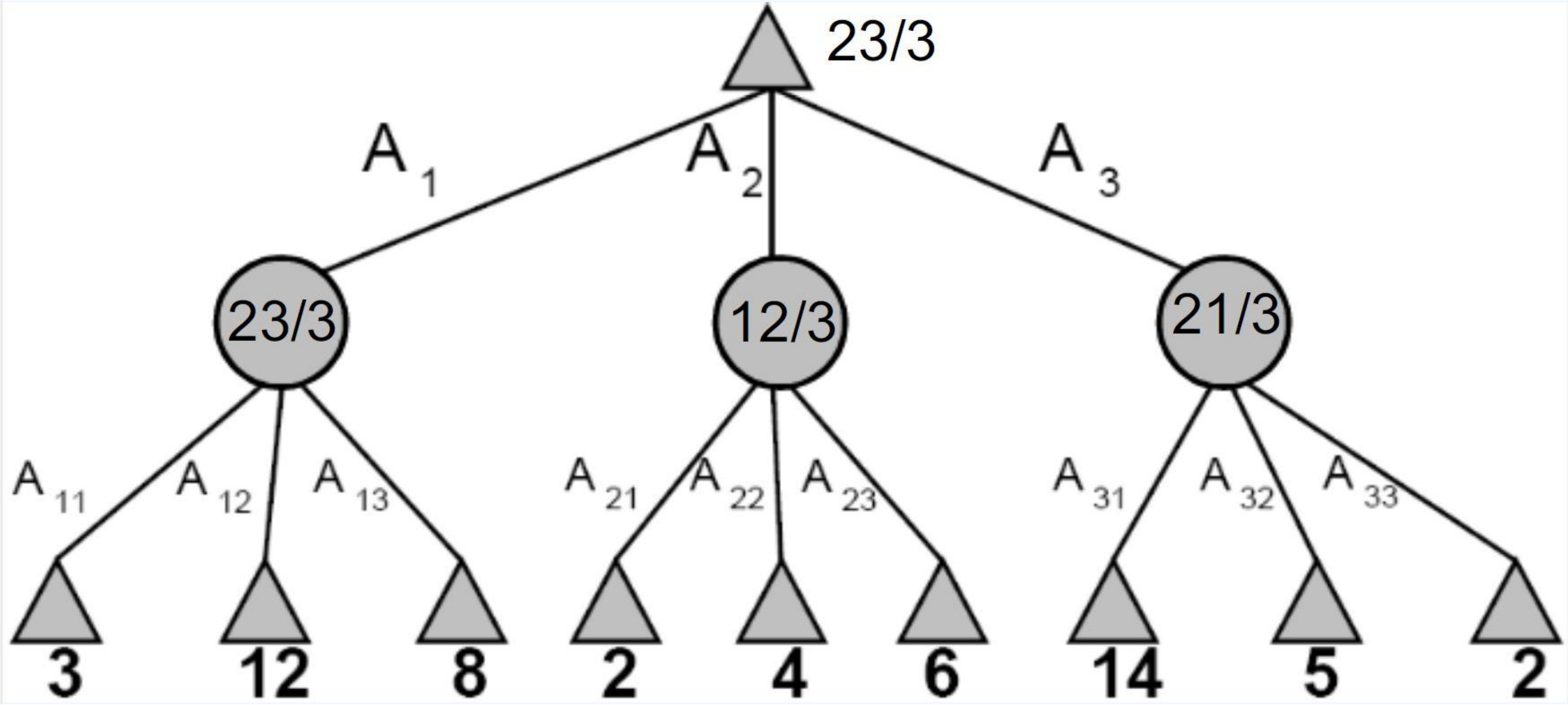
Time Complexity ?

- O(b^m) [*b* is branching factor and *m* is the maximum depth of the tree]

Space Complexity?

- O(*m*) [backtracking search, generates one action at a time]

EXPECTIMAX



23/3
A 1
A 2
A 3
23/3
12/3
21/3
A 11
A 12
A 13
A 21
A 22
A 23
A 31
A 32
A 33
3
12
8
2
4
6
14
5
2

EXPECTIMAX

Advantages of Expectimax over Minimax:

- Expectimax algorithm helps take advantage of non-optimal opponents.
- Unlike Minimax, Expectimax ‘can take a risk’ and end up in a state with a higher utility as opponents are random(not optimal).

Disadvantages:

- Expectimax is not optimal. It may lead to the agent losing (ending up in a state with lesser utility).
- Expectimax requires the full search tree to be explored. There is no type of pruning that can be done, as the value of a single unexplored utility can change the expectimax value drastically. Therefore it can be slow.

Properties of EXPECTIMAX

Time complexity: $O(b^m)$

Space complexity: $O(b*m)$, where *b* is branching factor and *m* is the maximum depth of the tree.

Applications: Expectimax can be used in environments where the actions of one of the agents are random. Following are a few examples,

- In **Pacman**, if we have random ghosts, we can model Pacman as the maximizer and ghosts as chance nodes. The utility values will be the values of the terminal states(win, lose or draw) or the evaluation function value for the set of possible states at a given depth.
- We can create a **minesweeper** AI by modelling the player agent as the maximizer and the mines as chance nodes.

Multi-agent Utilities