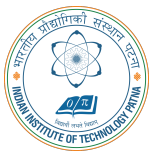
CS5201: Advanced Artificial Intelligence

Game tree search



Arijit Mondal
Dept of Computer Science and Engineering
Indian Institute of Technology Patna
www.iitp.ac.in/~arijit/

Games

- Chess — 2-player game, perfect information, win/loss/draw
- Tic-tac-toe — 2-player game, perfect information, win/loss/draw
- Bridge — 4-player game, two teams, incomplete information
- Ludo — 4-player game, perfect information, probabilistic outcome
- Control plant — incomplete, imperfect, probabilistic
- Robot games — multiple players, collaboration/competition

Prisoner's dilemma

- Two members of a criminal gang are arrested and imprisoned. Each prisoner is in solitary confinement with no means of communicating with the other.
- The prosecutors lack sufficient evidence to convict the pair on the principal charge, but they have enough to convict both on a lesser charge.
- Simultaneously, the prosecutors offer each prisoner a bargain.

		Prisoner B	
		Prisoner B stays silent (Cooperates)	Prisoner B betrays A (Defects)
Prisoner A	Prisoner A stays silent (Cooperates)	Each serves 1 year	Prisoner A: 3 years Prisoner B: goes free
	Prisoner A betrays B (Defects)	Prisoner A: goes free Prisoner B: 3 years	Each serves 2 years

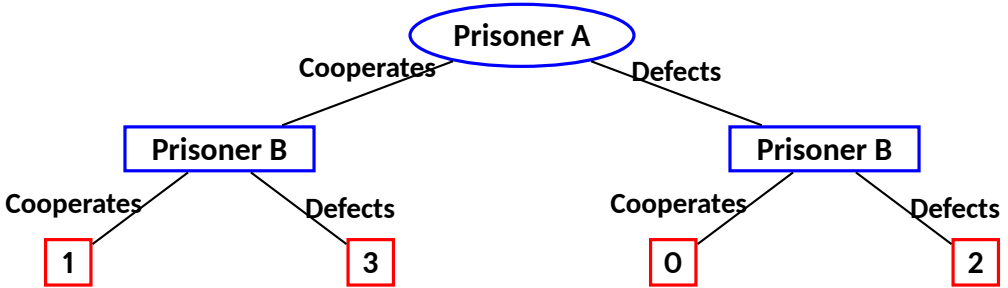
Prisoner's dilemma: solution argument

- Prisoner A will defect. Why?
  - Prisoner-A knows that Prisoner-B may cooperate or defect
    - If Prisoner-B cooperates, then by defecting, Prisoner-A will go free
    - If Prisoner-B defects, then Prisoner-A will face a longer sentence by staying silent
  - In both cases Prisoner-A gains by defecting
  - Therefore both prisoners will defect, although they would have gained from cooperating

		Prisoner B	
		Prisoner B stays silent (Cooperates)	Prisoner B betrays A (Defects)
Prisoner A	Prisoner A stays silent (Cooperates)	Each serves 1 year	Prisoner A: 3 years Prisoner B: goes free
	Prisoner A betrays B (Defects)	Prisoner A: goes free Prisoner B: 3 years	Each serves 2 years

Prisoner's dilemma: Game tree

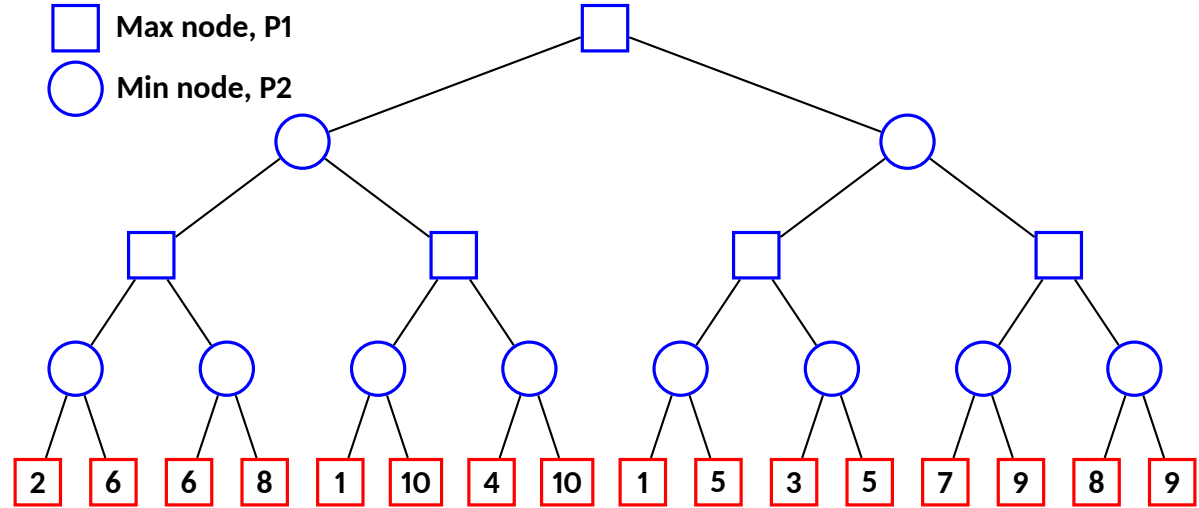
		Prisoner B	
		Prisoner B stays silent (Cooperates)	Prisoner B betrays A (Defects)
Prisoner A	Prisoner A stays silent (Cooperates)	Each serves 1 year	Prisoner A: 3 years Prisoner B: goes free
	Prisoner A betrays B (Defects)	Prisoner A: goes free Prisoner B: 3 years	Each serves 2 years



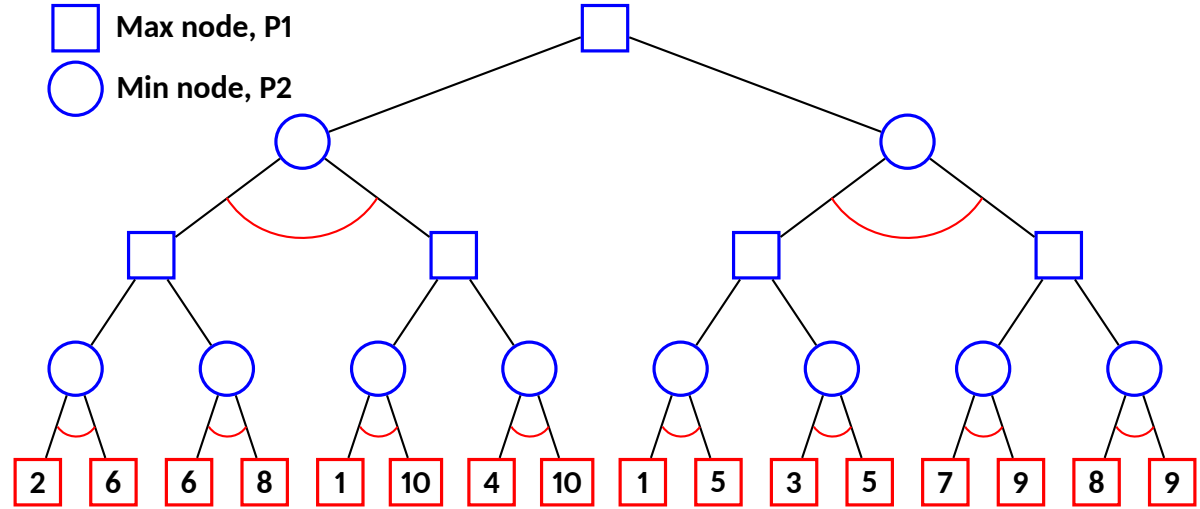
Game tree

- A tree with three types of nodes, namely Terminal nodes, Min nodes and Max nodes. Terminal nodes have no children. The tree has alternating levels of Max and Min nodes, representing the turns of Player-1 and Player-2 in making moves
- All nodes represent some state of the game
- Terminal nodes are labeled with the payoff for Player-1. It could be Boolean (such as WON or LOST). In large games, where looking ahead up to the WON / LOST states is not feasible, the payoff at a terminal node may represent a heuristic cost representing the quality of the state of the game from Player-1's perspective
- The payoff at a Min node is the minimum among the payoffs of its successors
- The payoff at a Max node is the maximum among the payoffs of its successors
- If Player-1 aims to maximize its payoff, then it represents Max nodes, else it represents Min nodes.

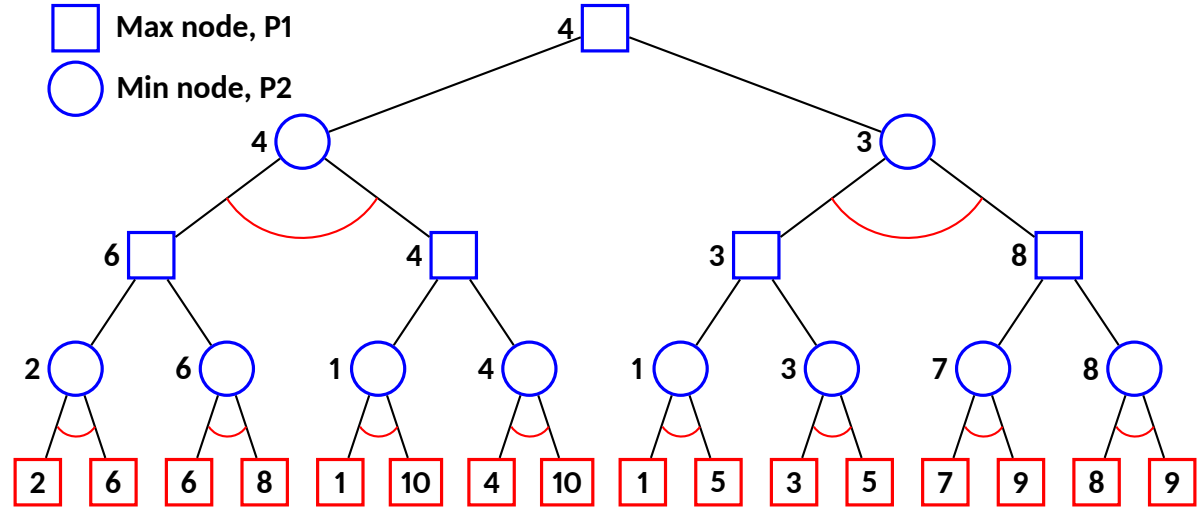
Game tree



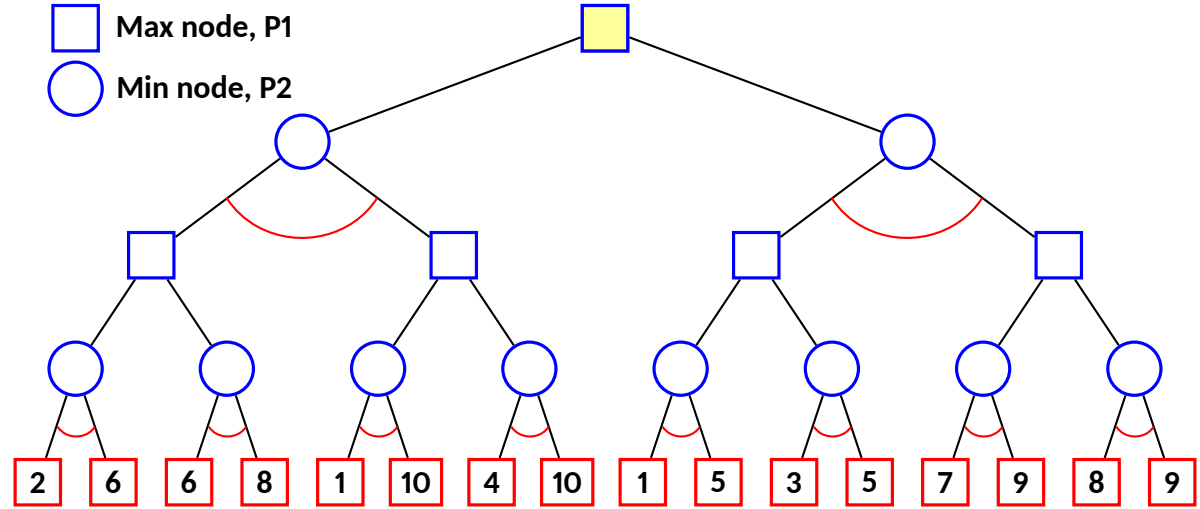
Game tree: MinMax value



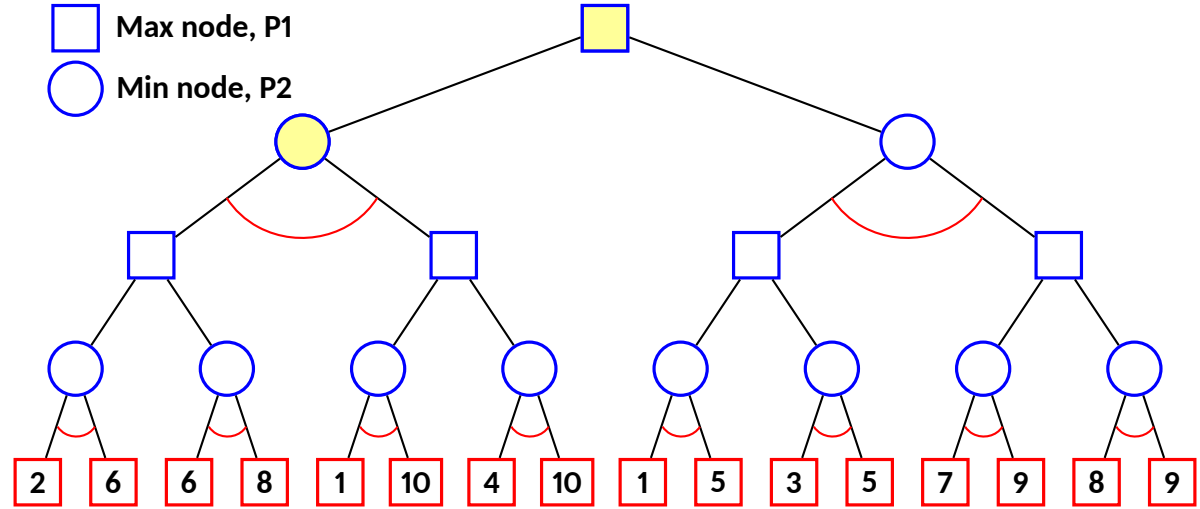
Game tree: MinMax value



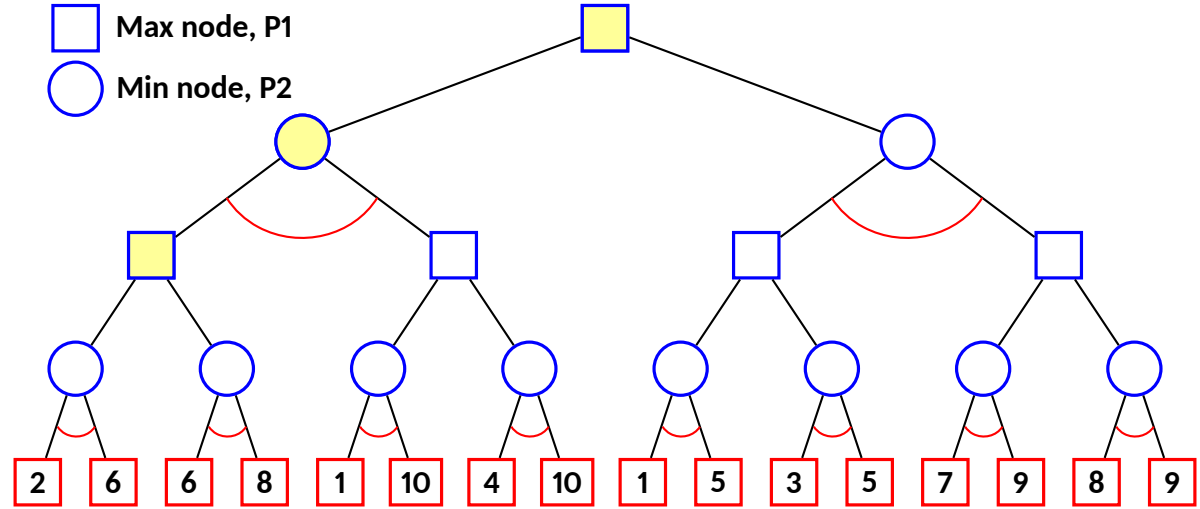
Game tree: Depth first search



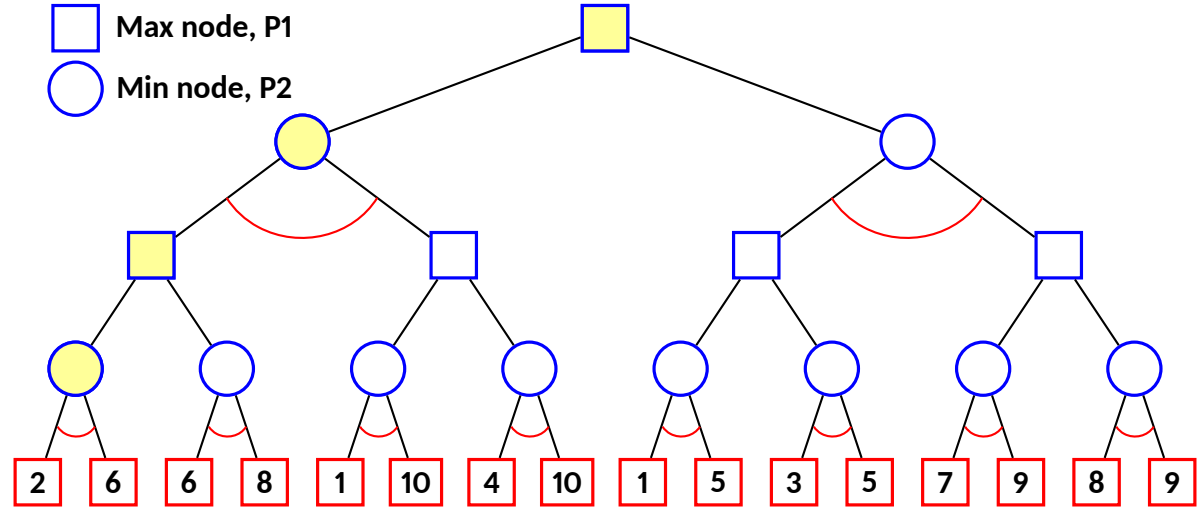
Game tree: Depth first search



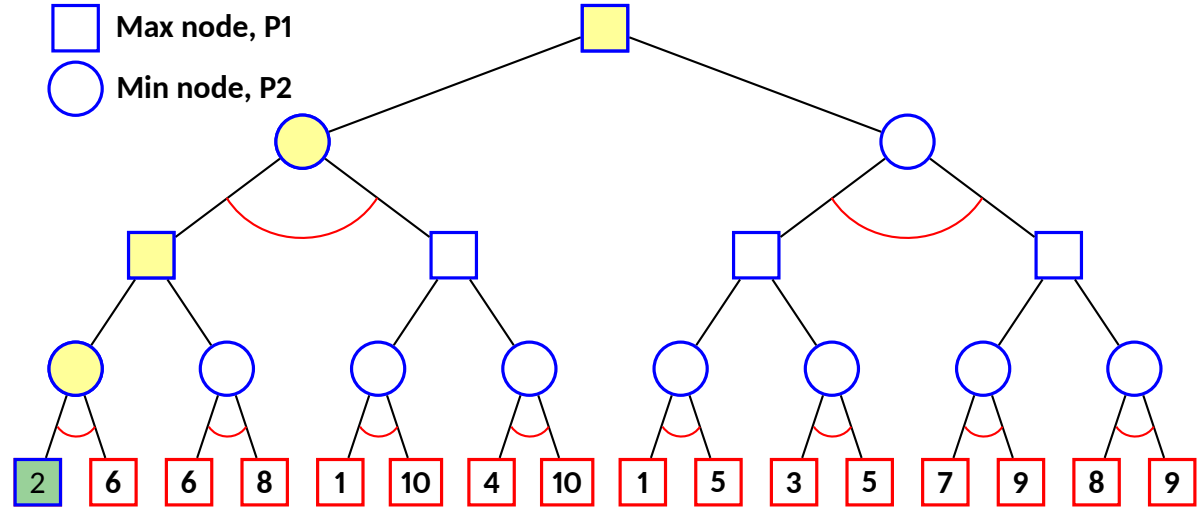
Game tree: Depth first search



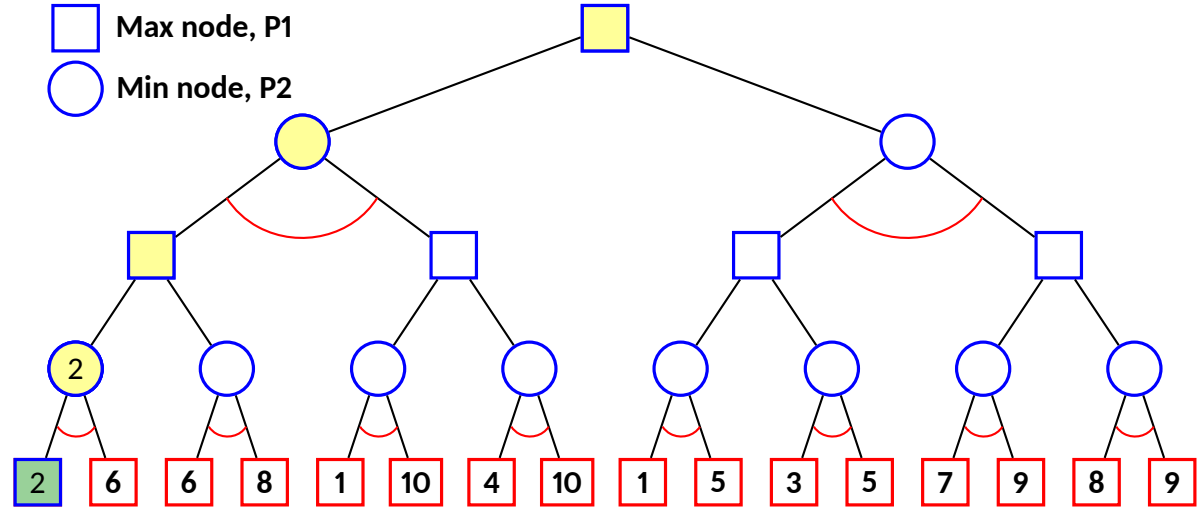
Game tree: Depth first search



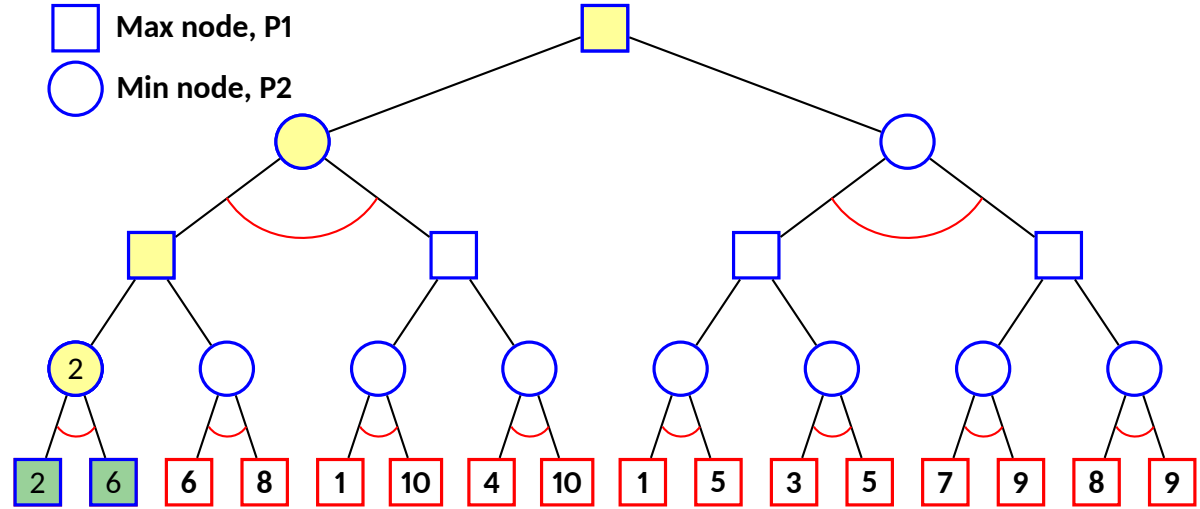
Game tree: Depth first search



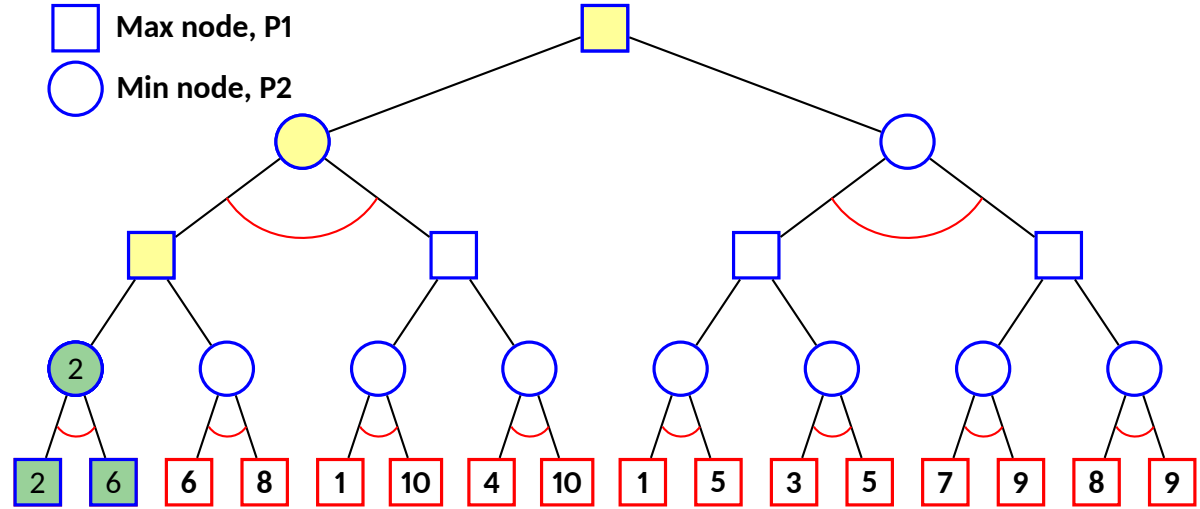
Game tree: Depth first search



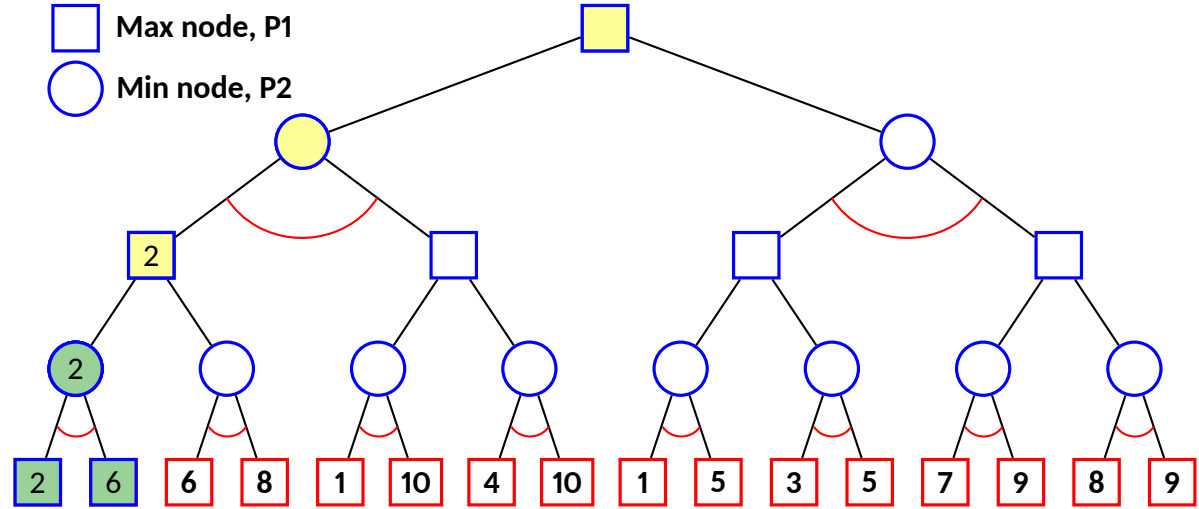
Game tree: Depth first search



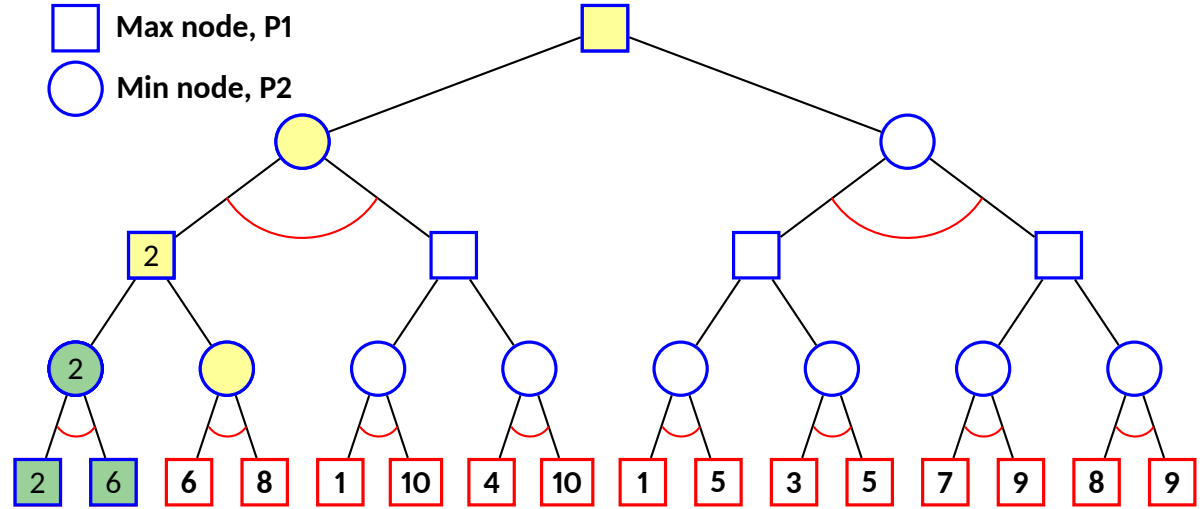
Game tree: Depth first search



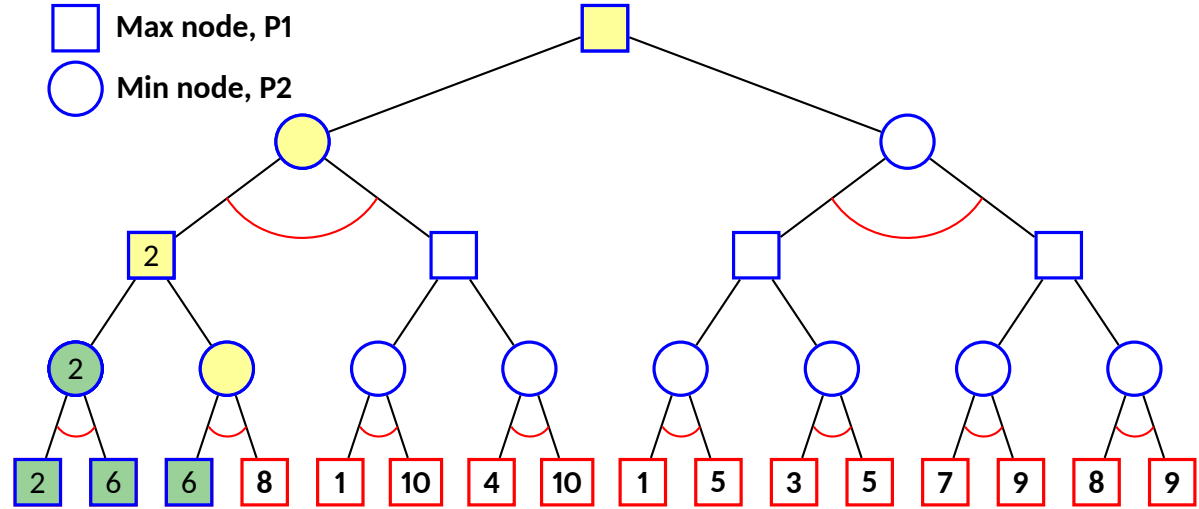
Game tree: Depth first search



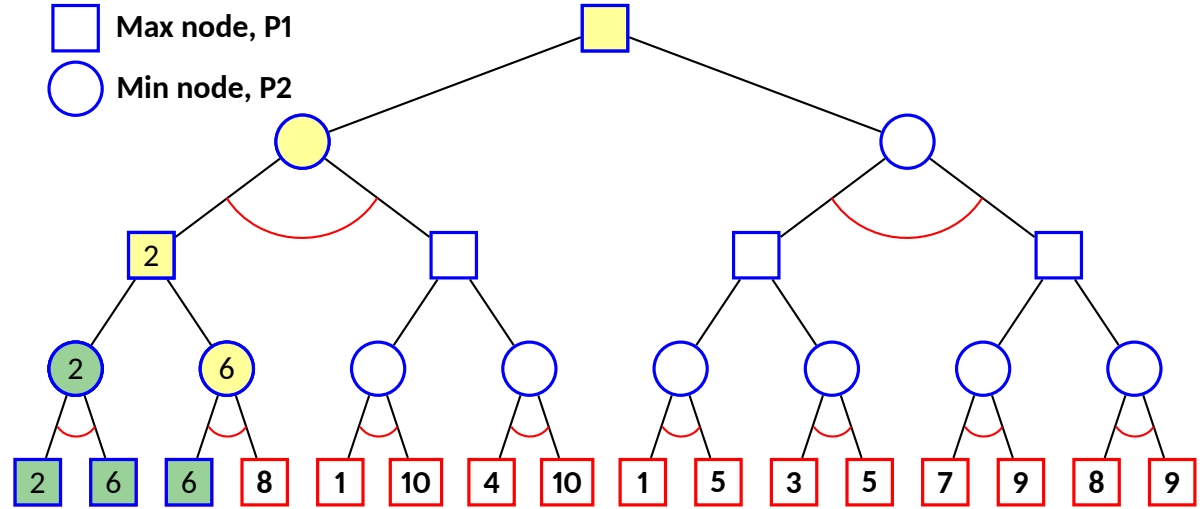
Game tree: Depth first search



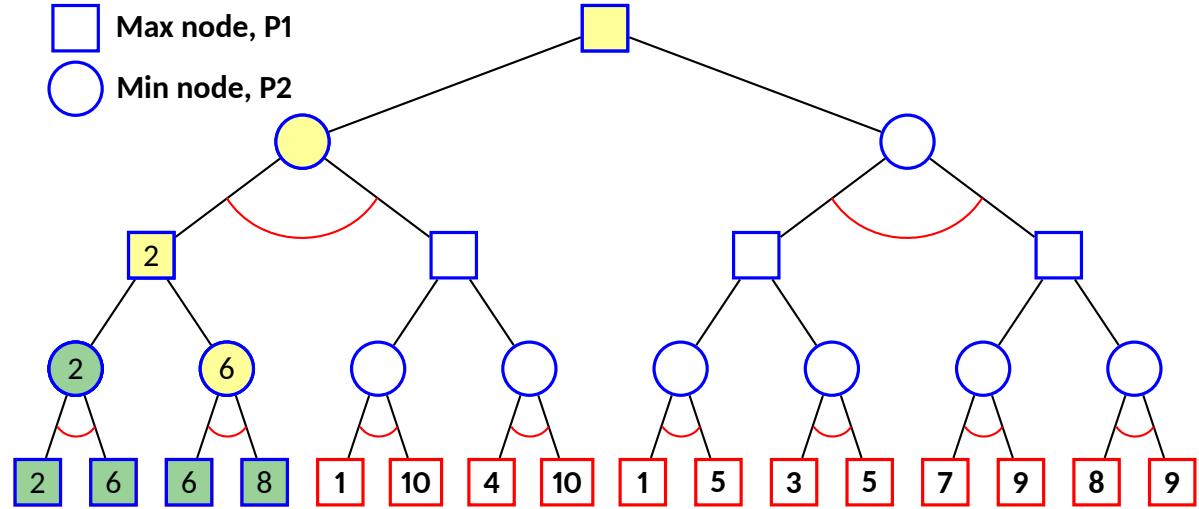
Game tree: Depth first search



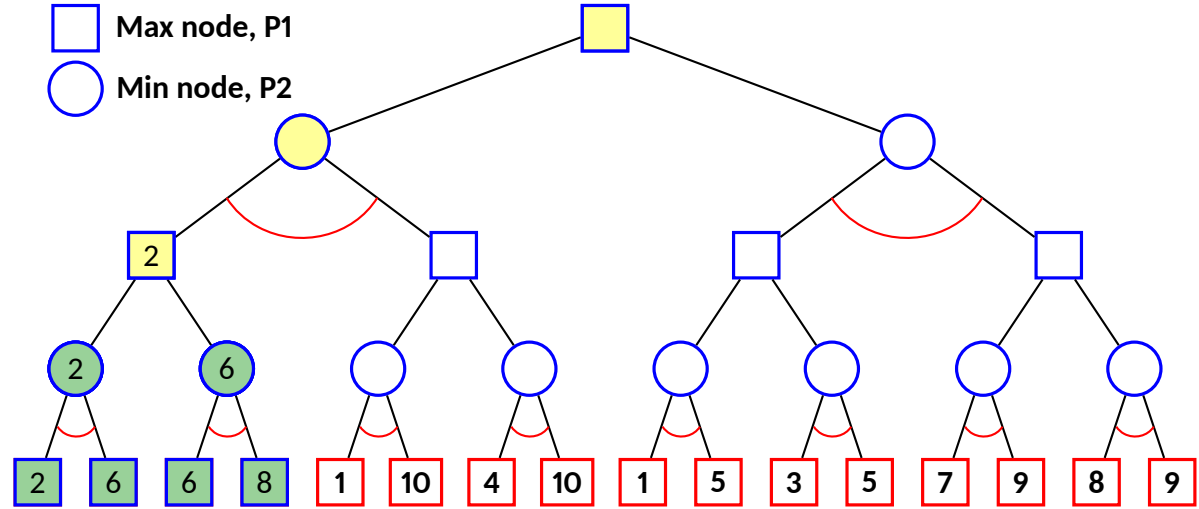
Game tree: Depth first search



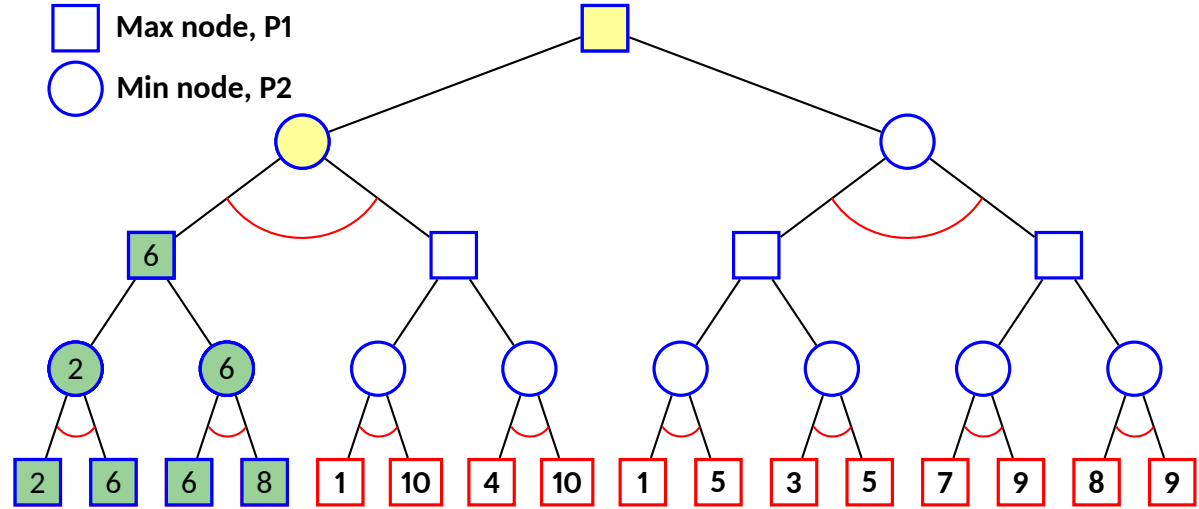
Game tree: Depth first search



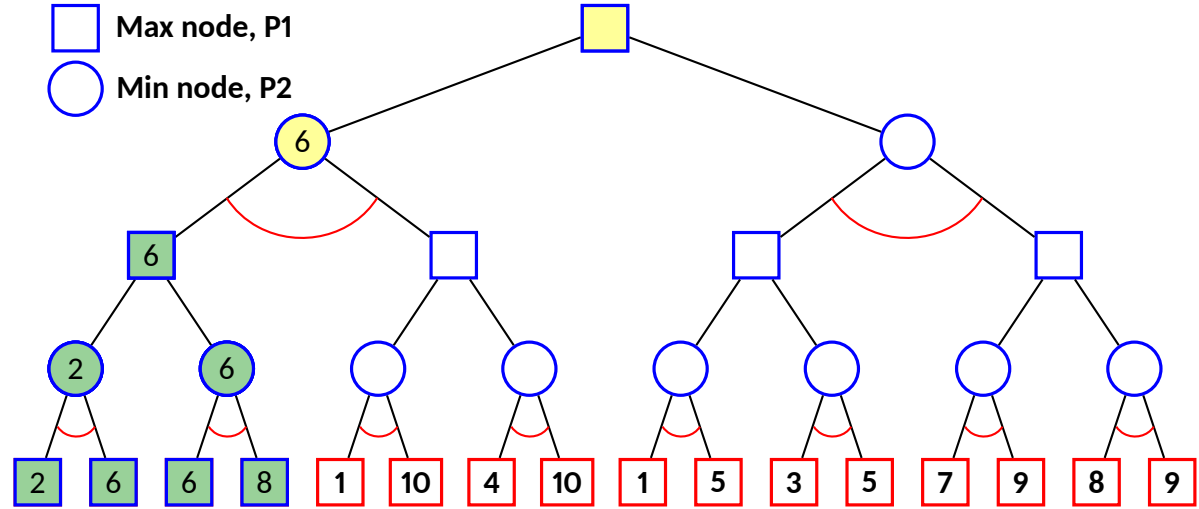
Game tree: Depth first search



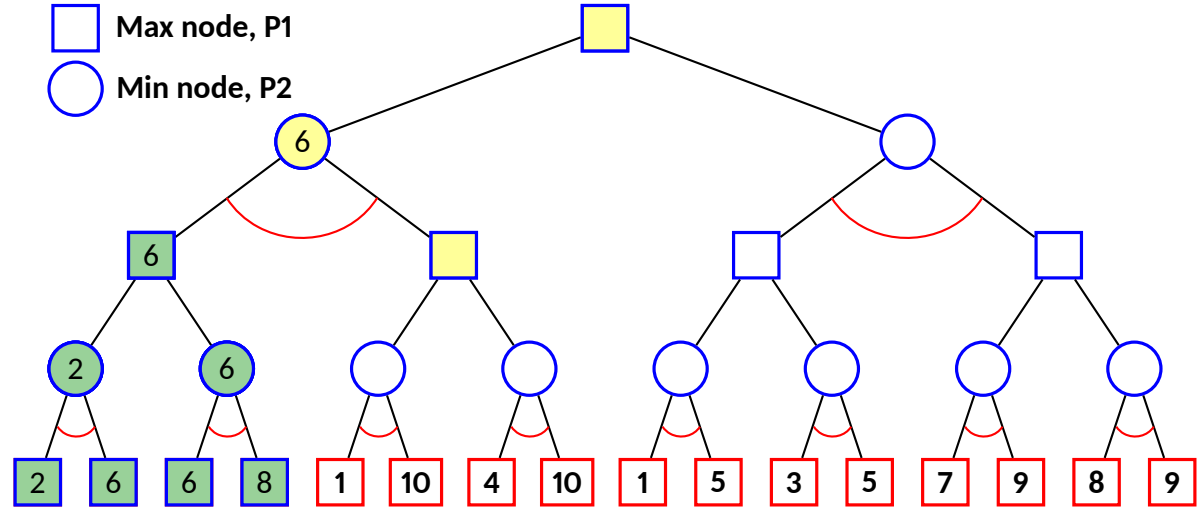
Game tree: Depth first search



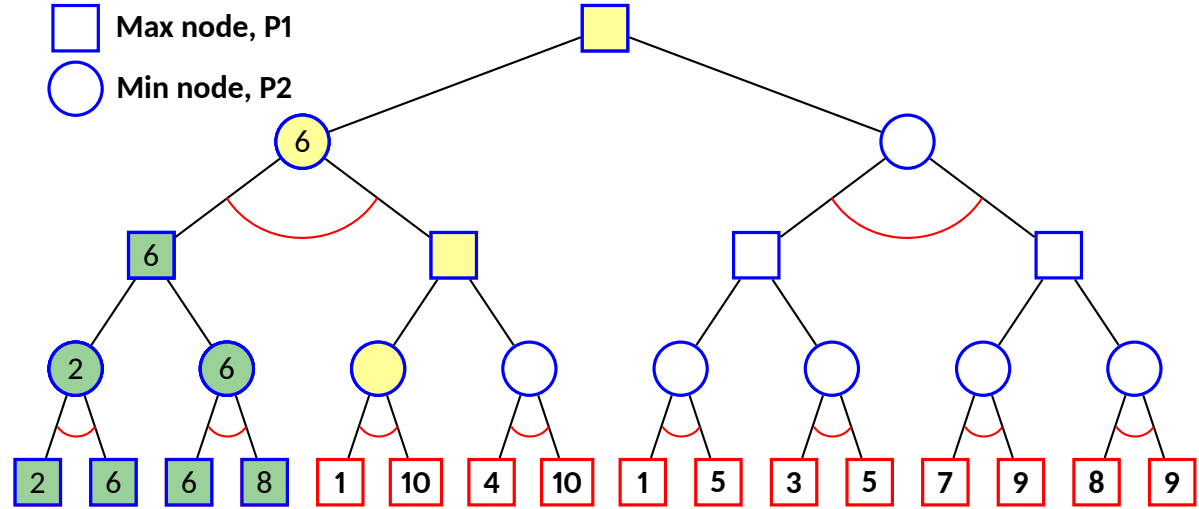
Game tree: Depth first search



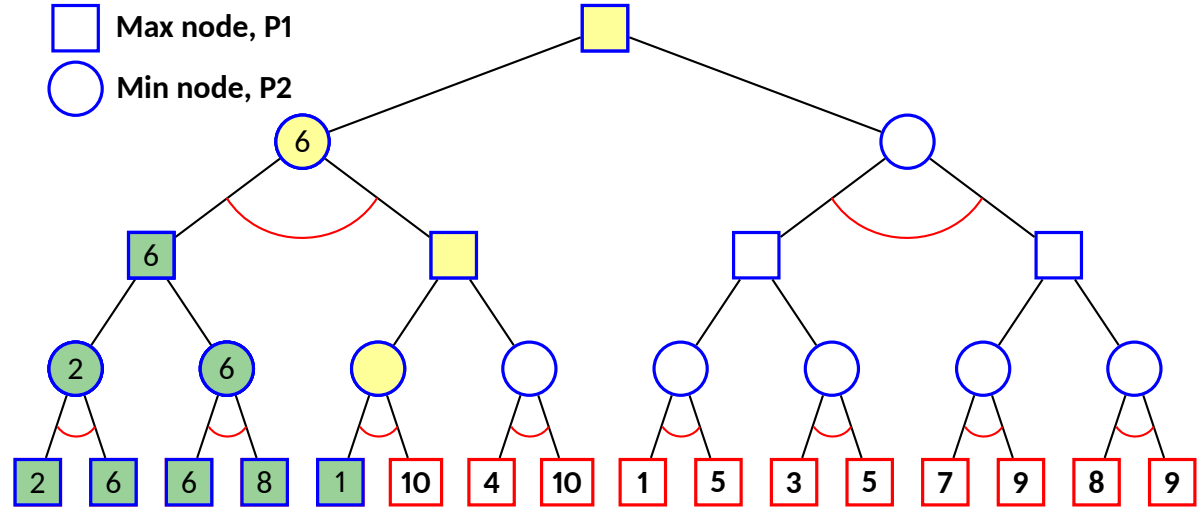
Game tree: Depth first search



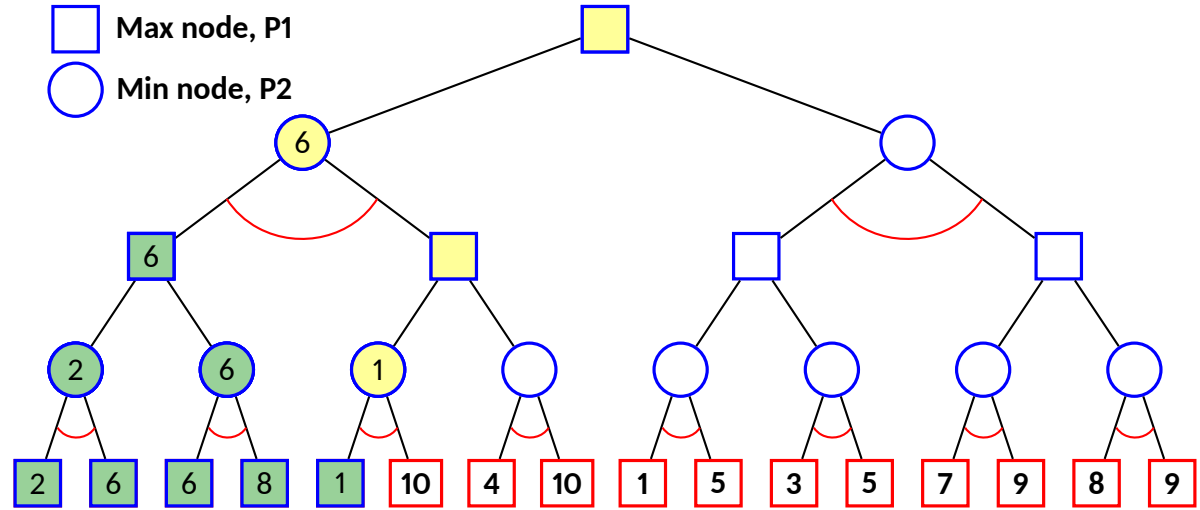
Game tree: Depth first search



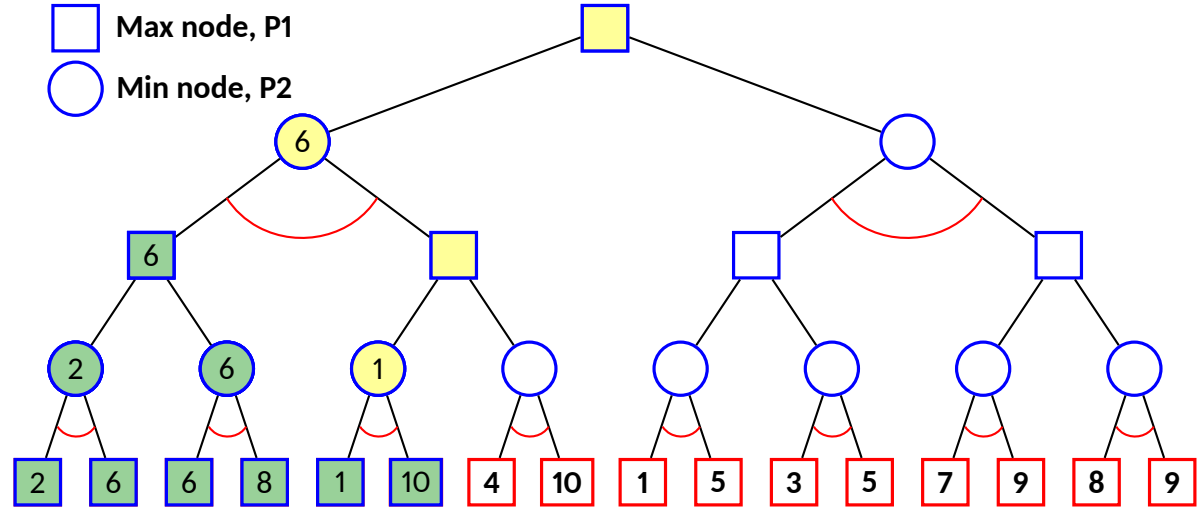
Game tree: Depth first search



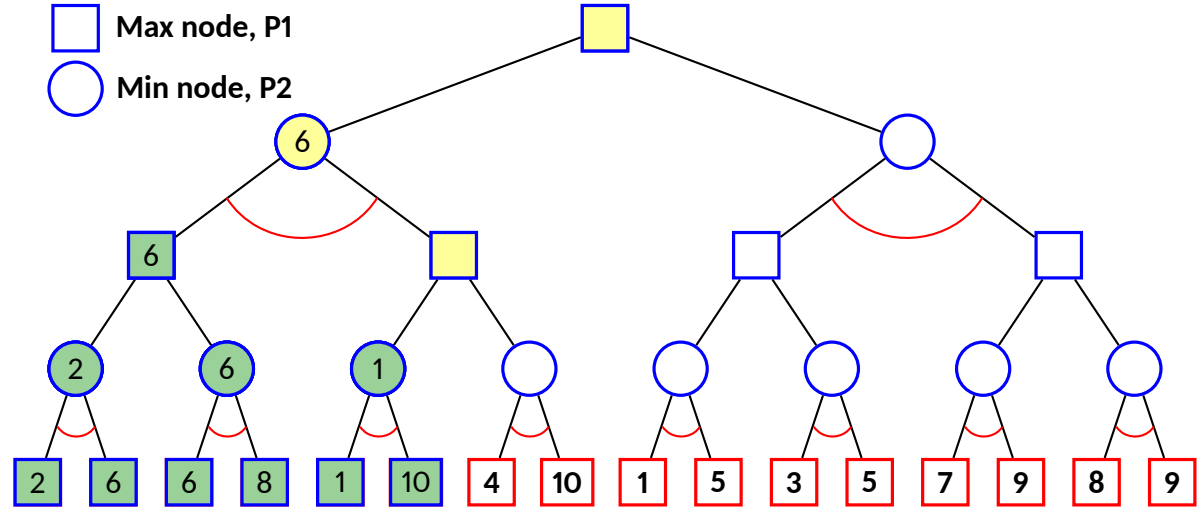
Game tree: Depth first search



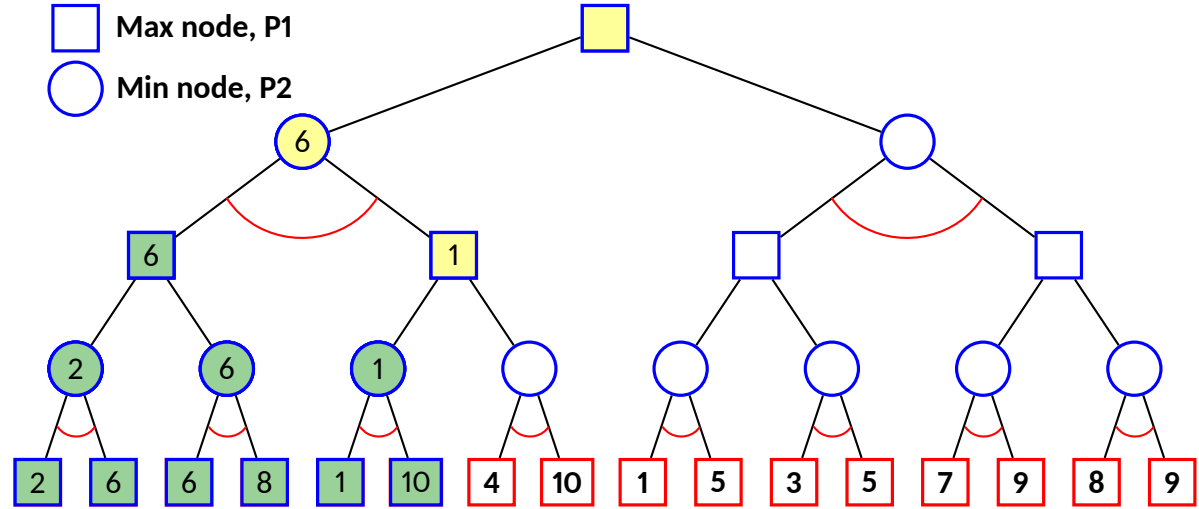
Game tree: Depth first search



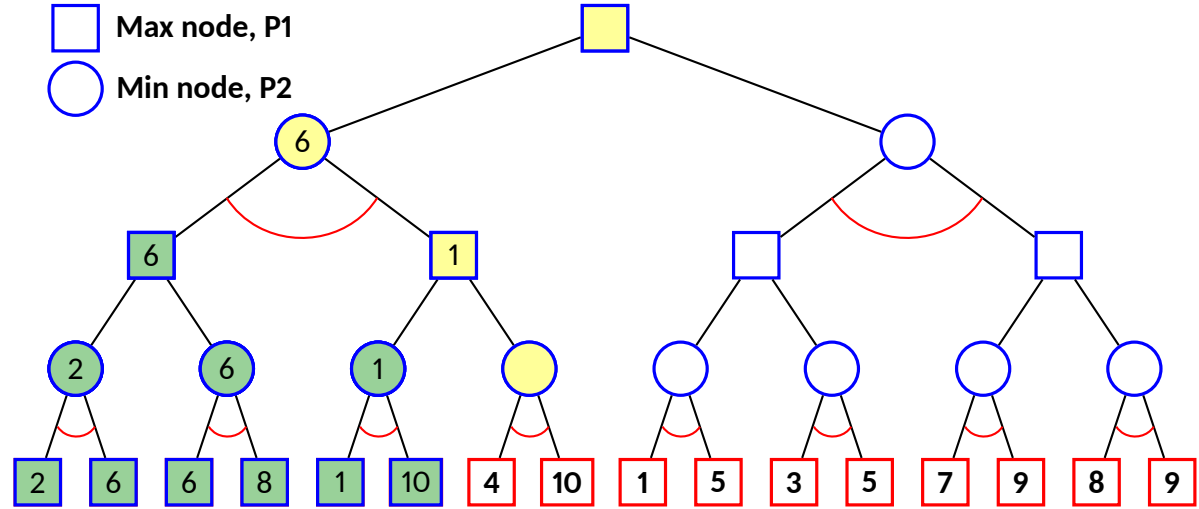
Game tree: Depth first search



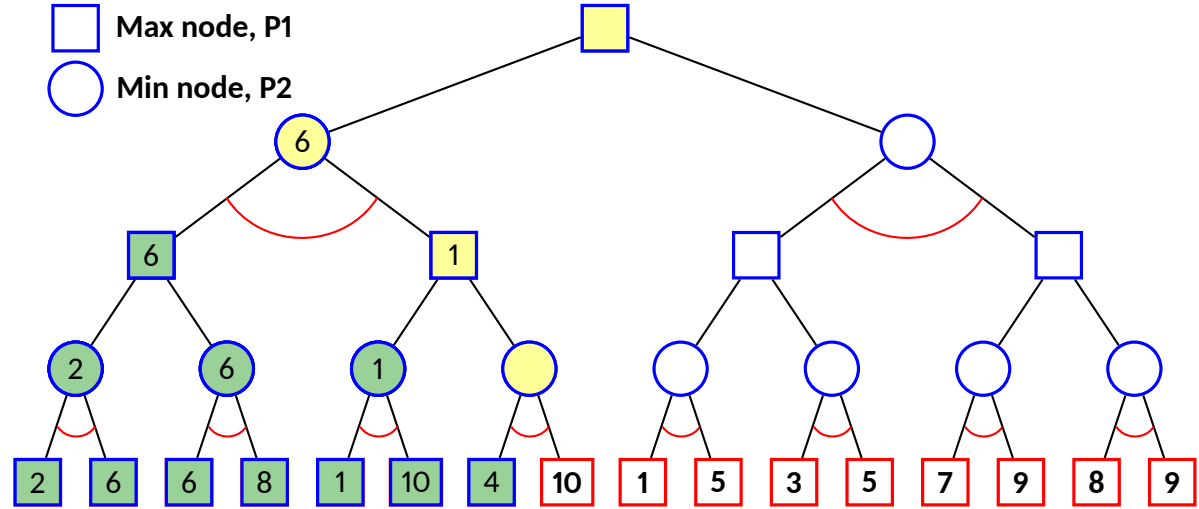
Game tree: Depth first search



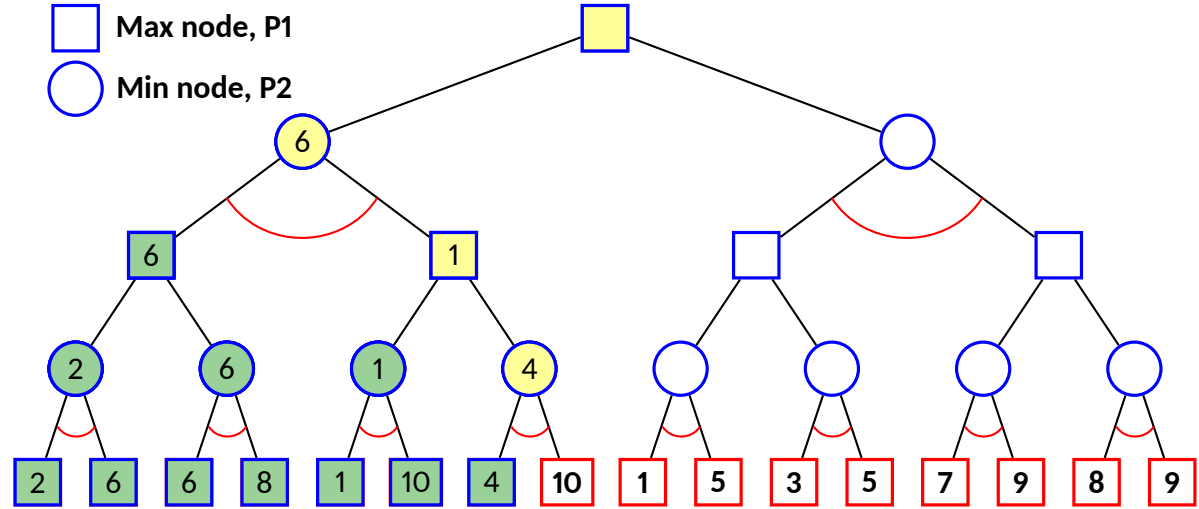
Game tree: Depth first search



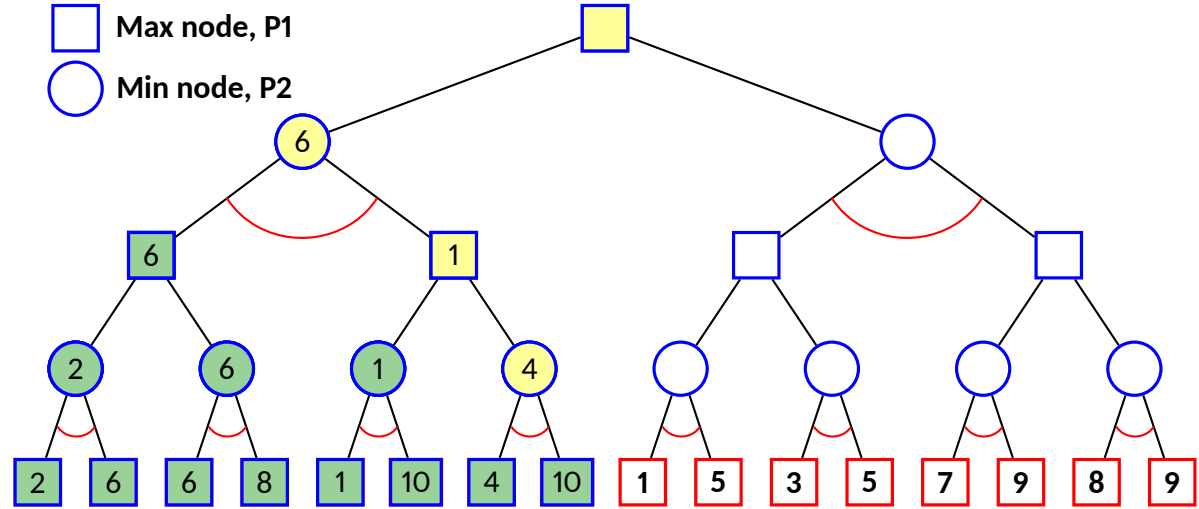
Game tree: Depth first search



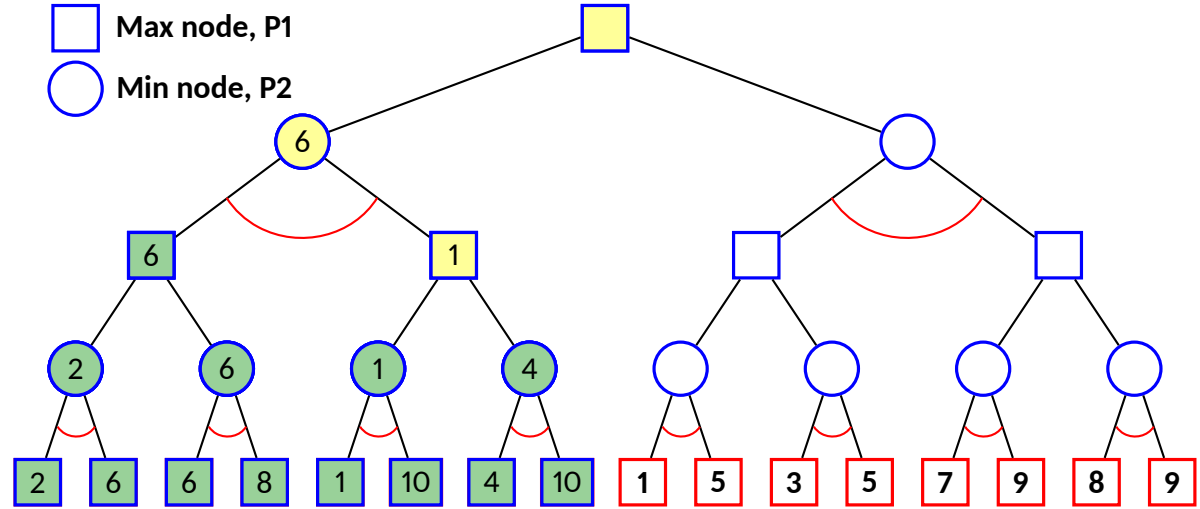
Game tree: Depth first search



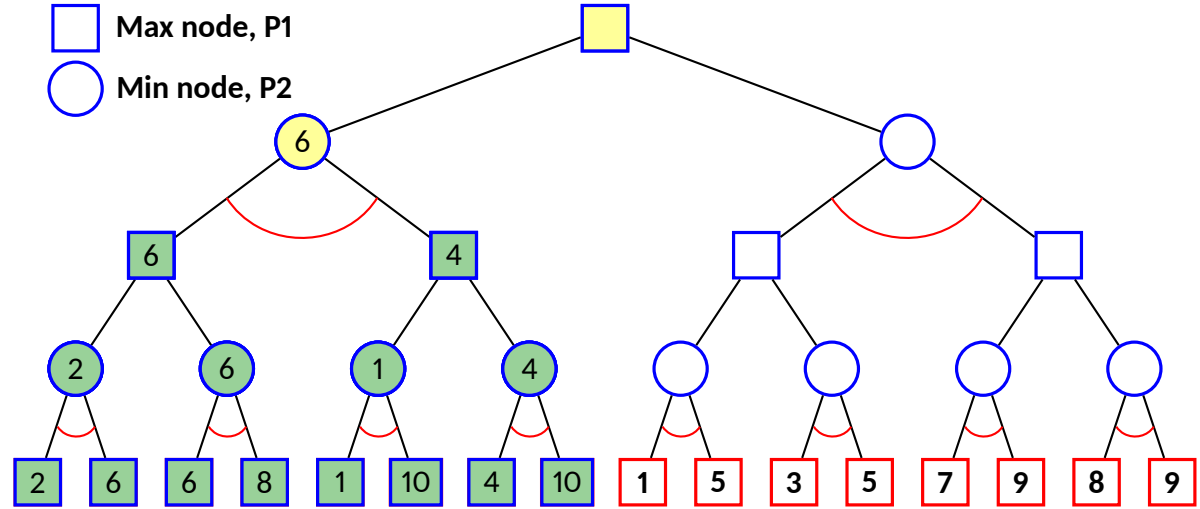
Game tree: Depth first search



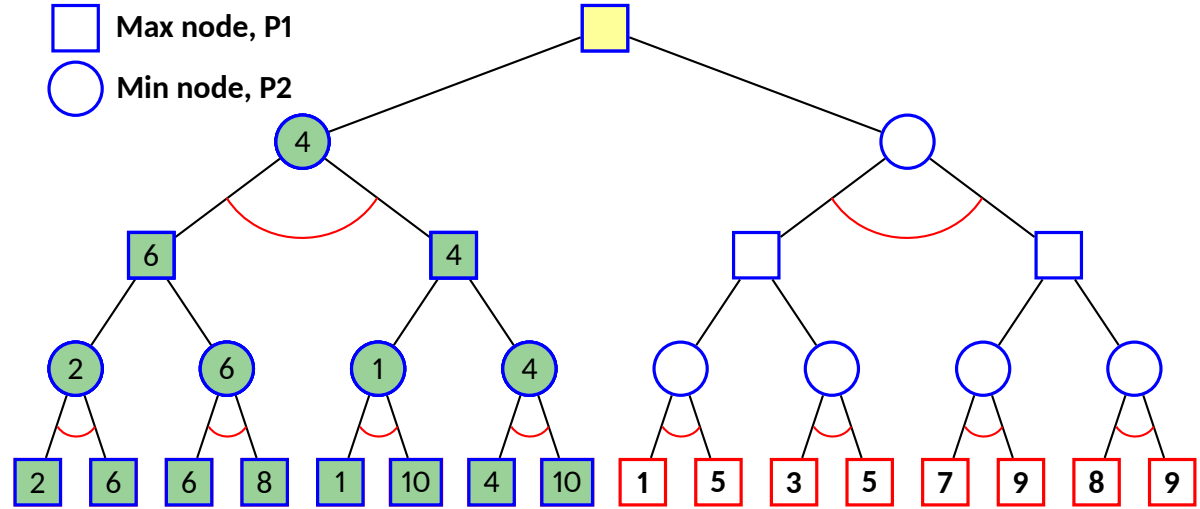
Game tree: Depth first search



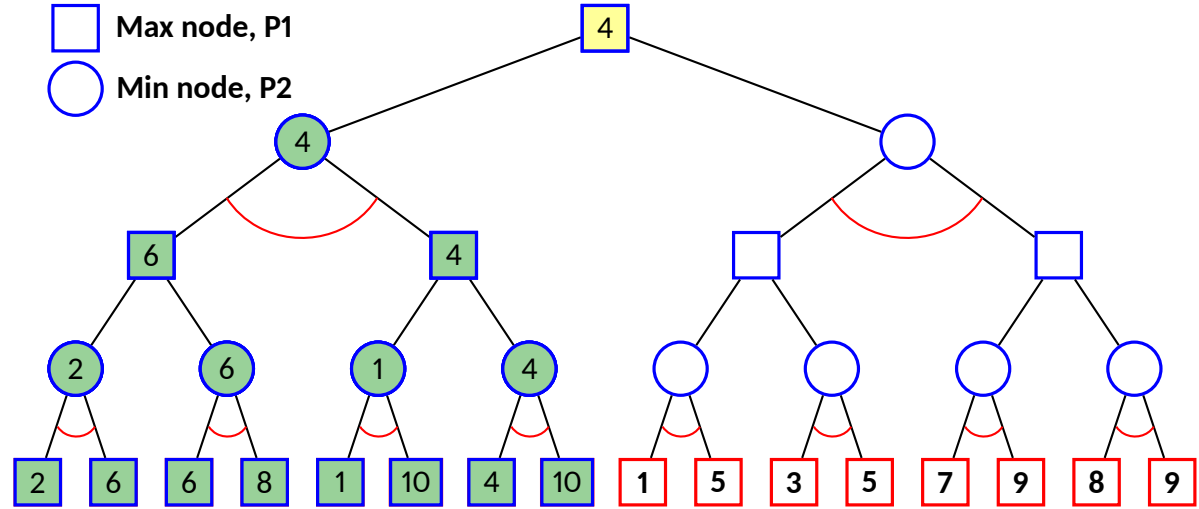
Game tree: Depth first search



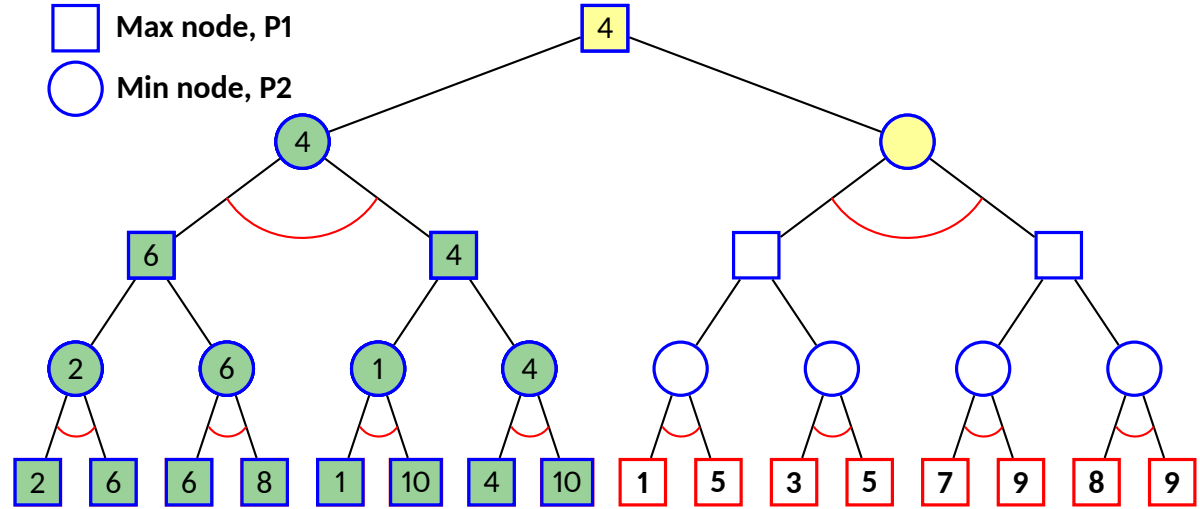
Game tree: Depth first search



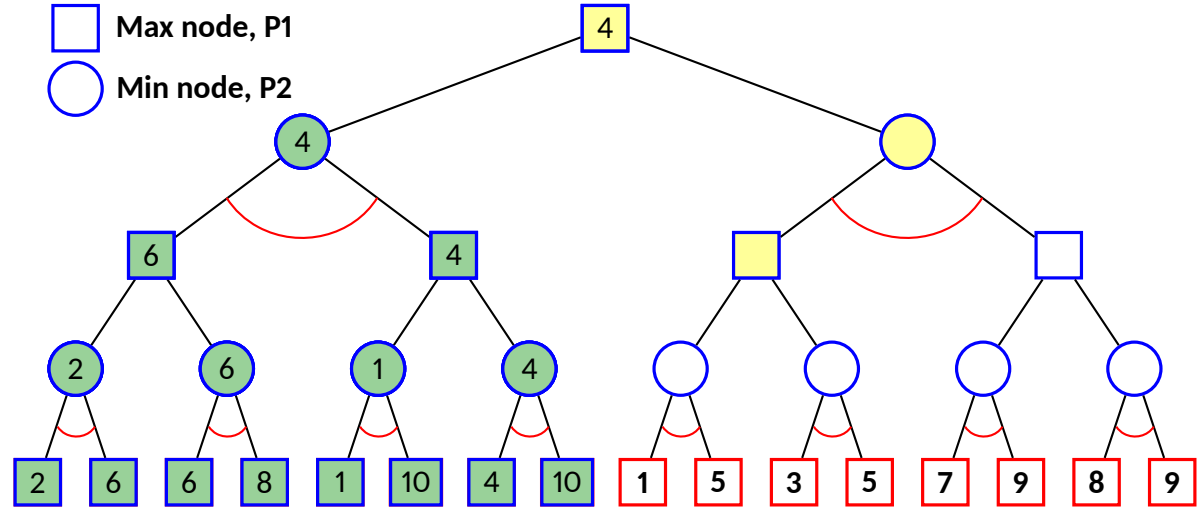
Game tree: Depth first search



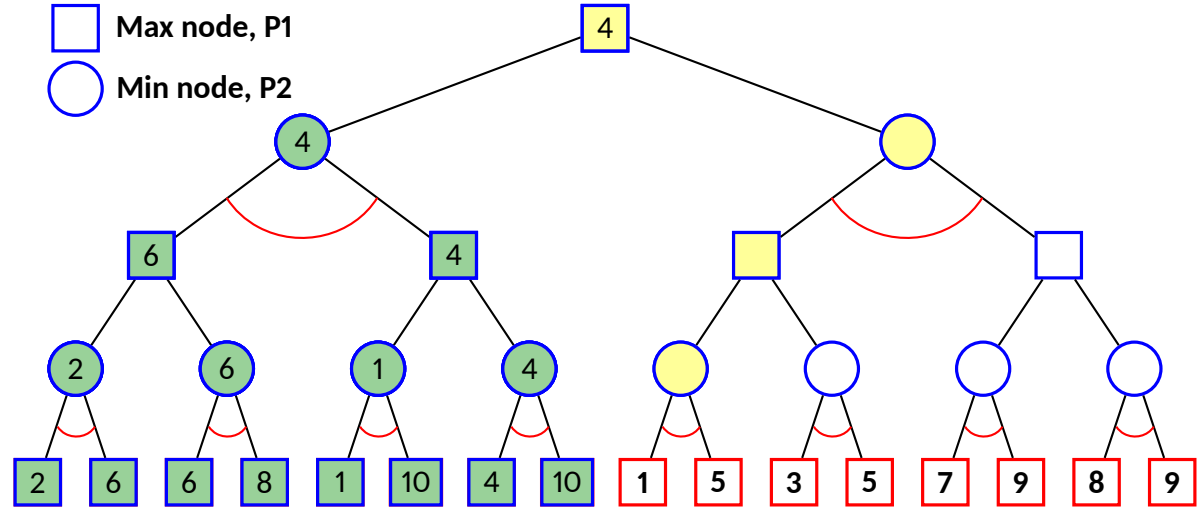
Game tree: Depth first search



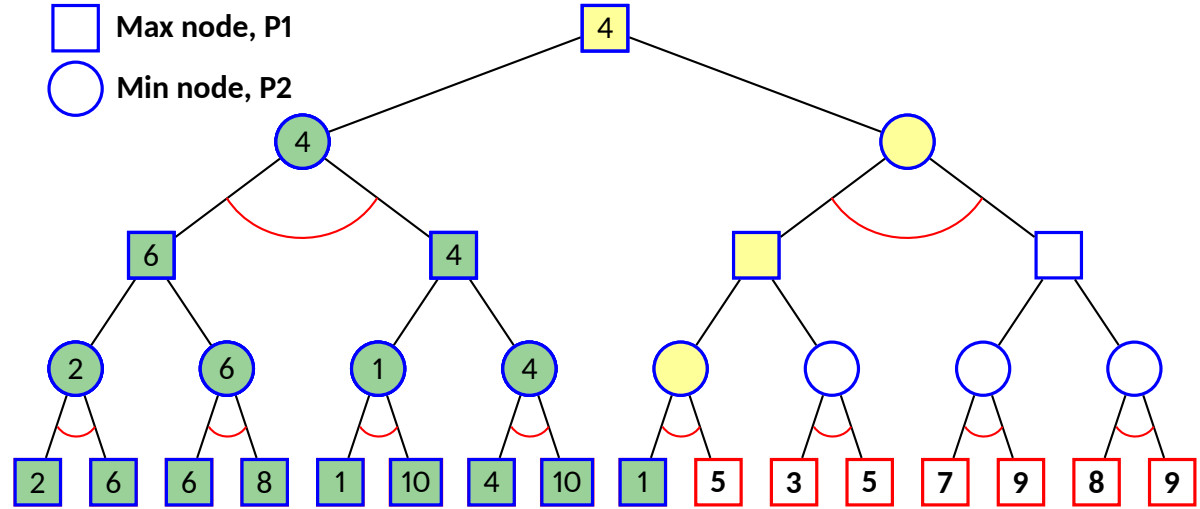
Game tree: Depth first search



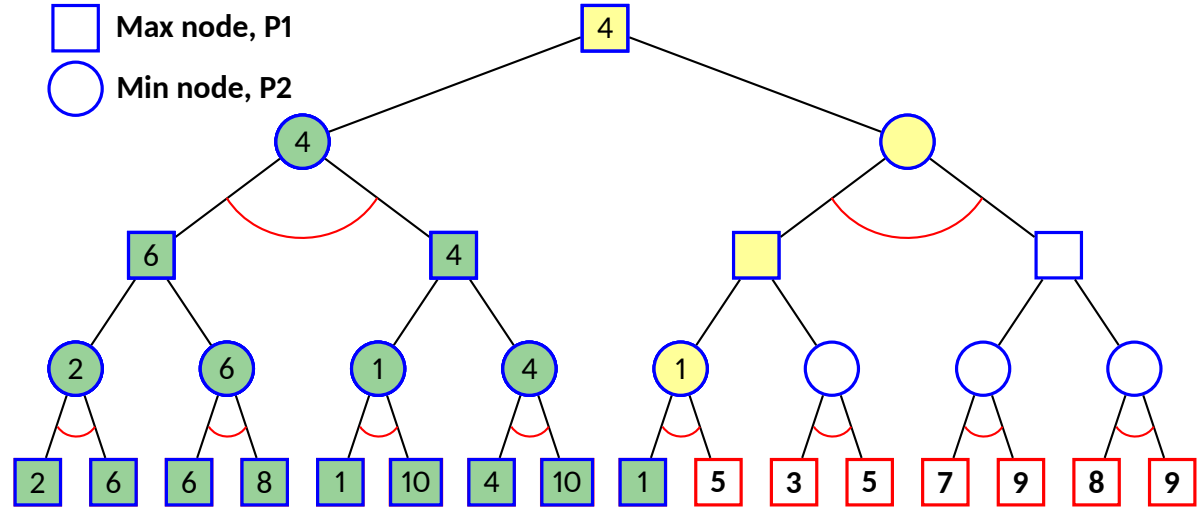
Game tree: Depth first search



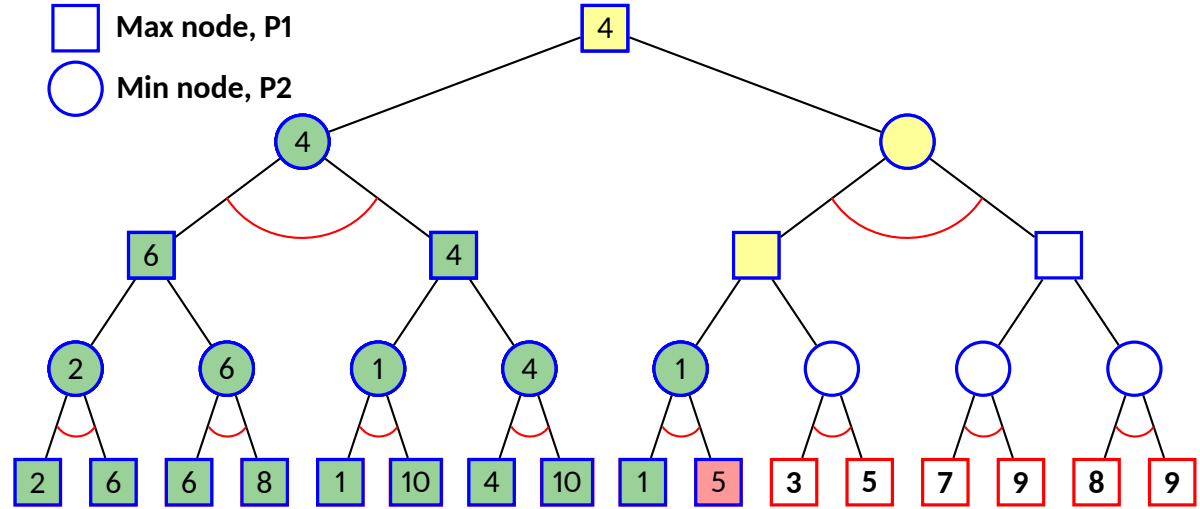
Game tree: Depth first search



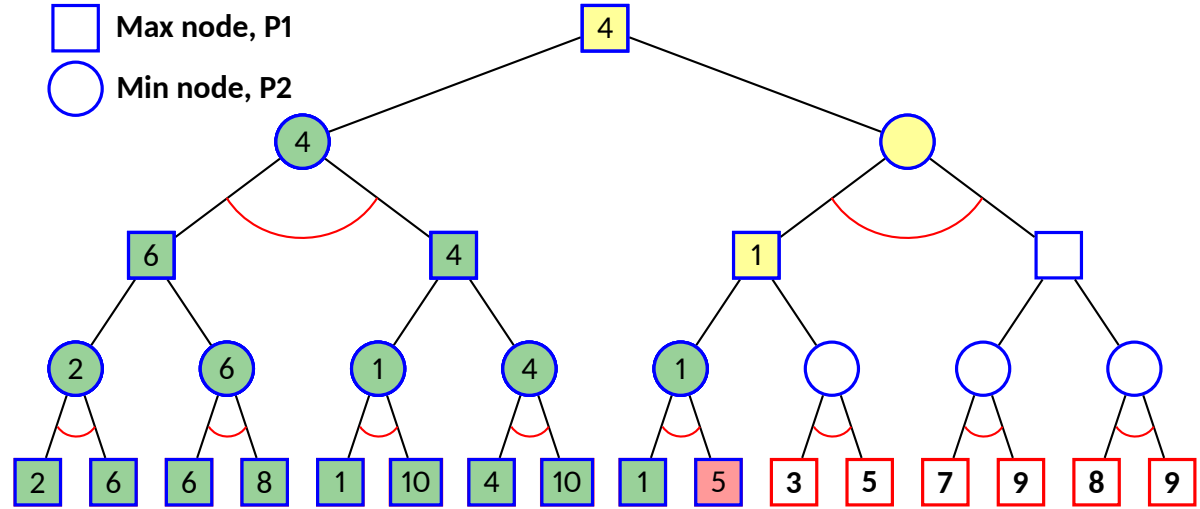
Game tree: Depth first search



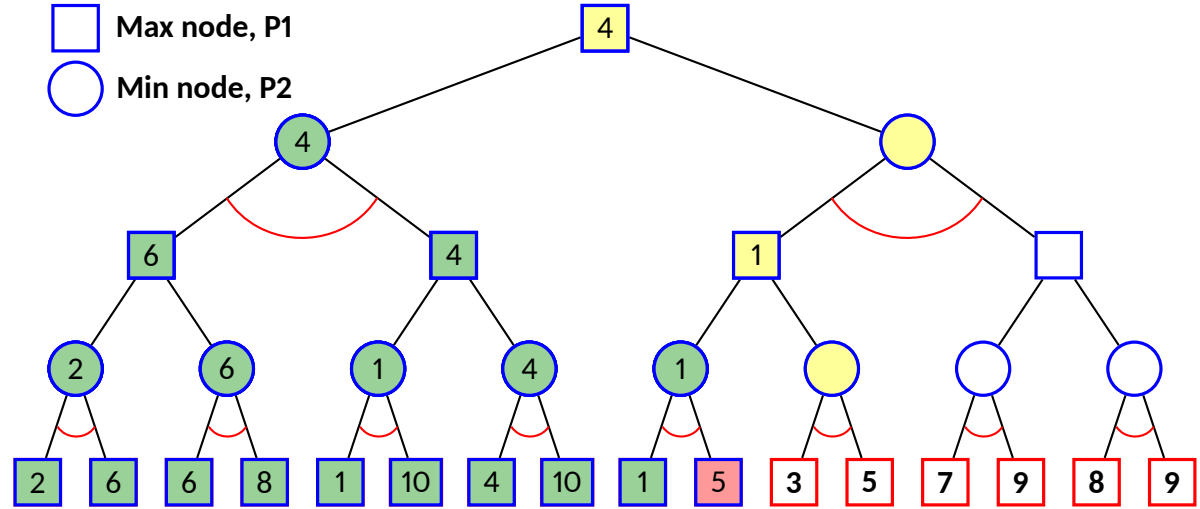
Game tree: Depth first search



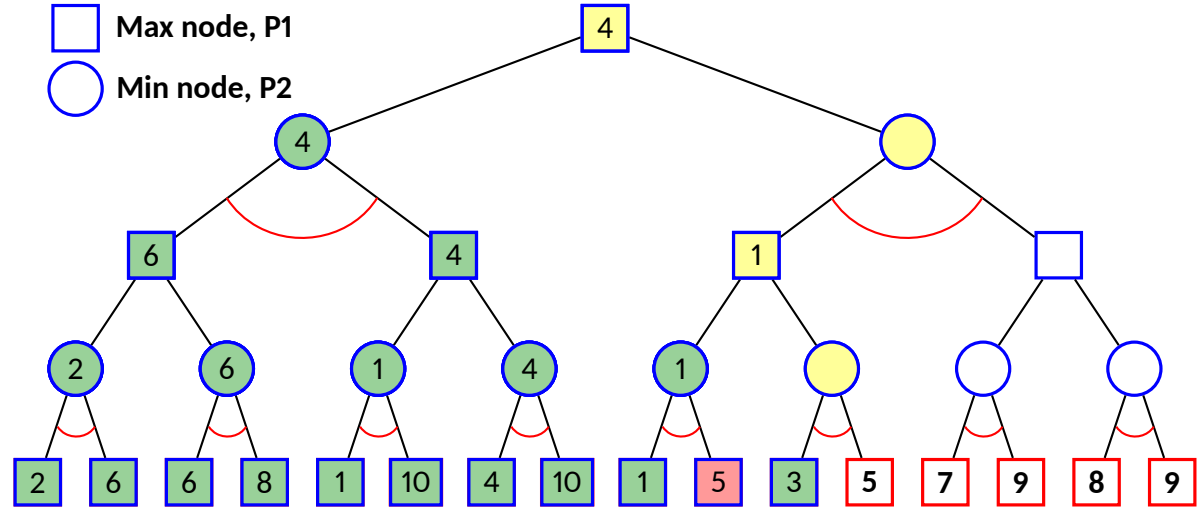
Game tree: Depth first search



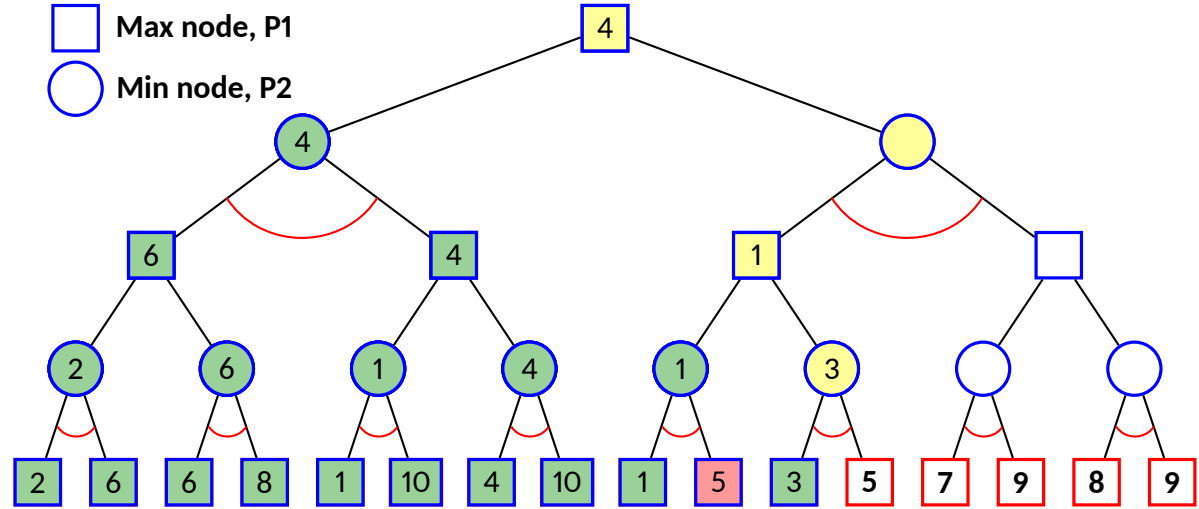
Game tree: Depth first search



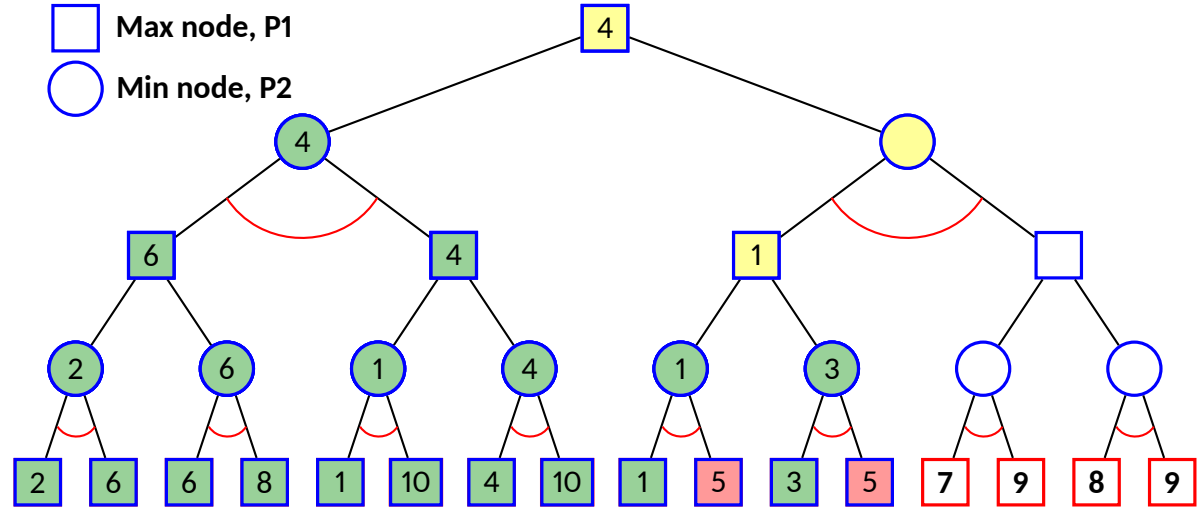
Game tree: Depth first search



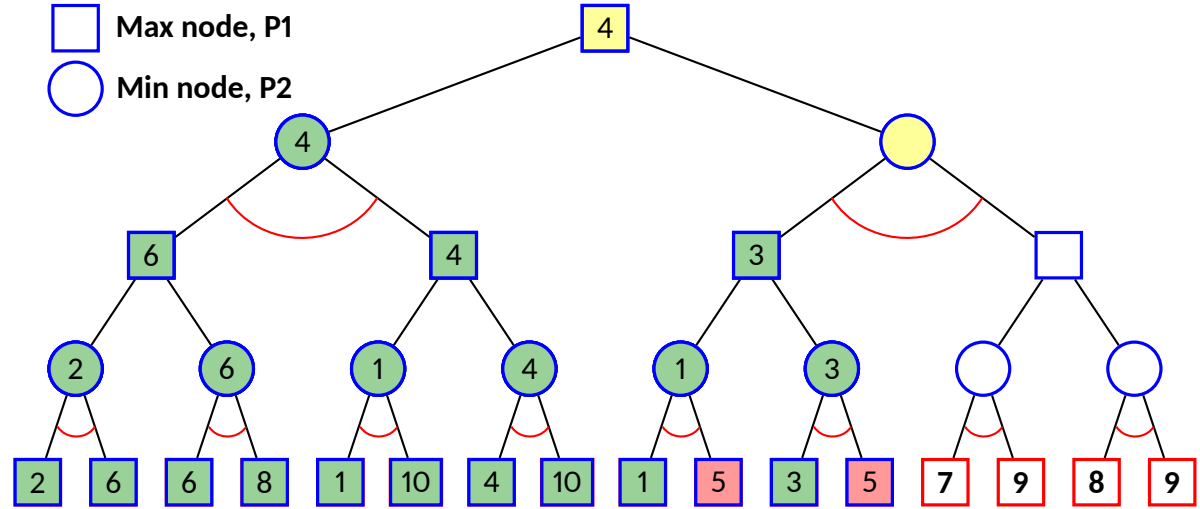
Game tree: Depth first search



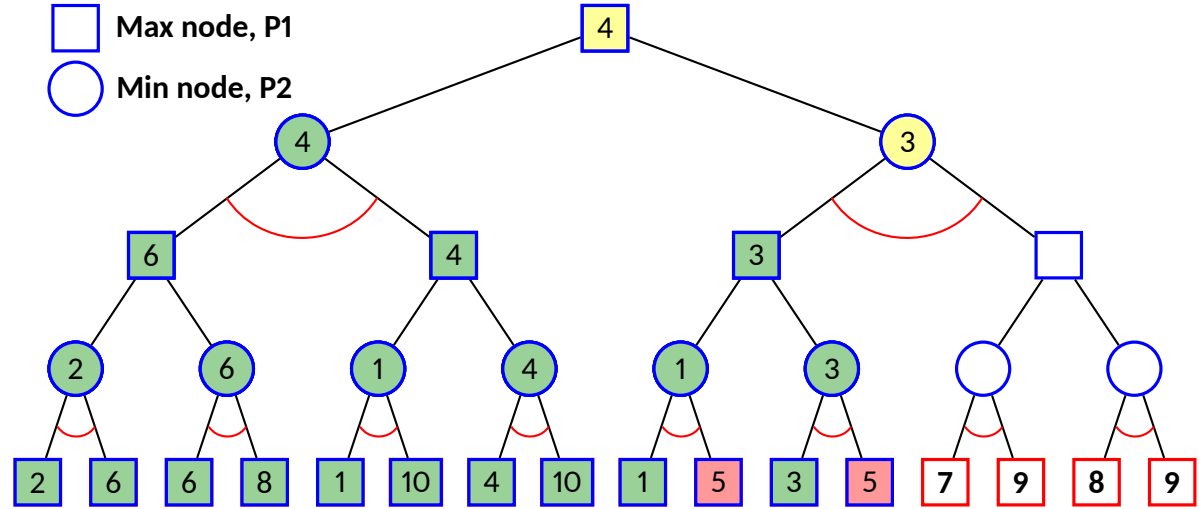
Game tree: Depth first search



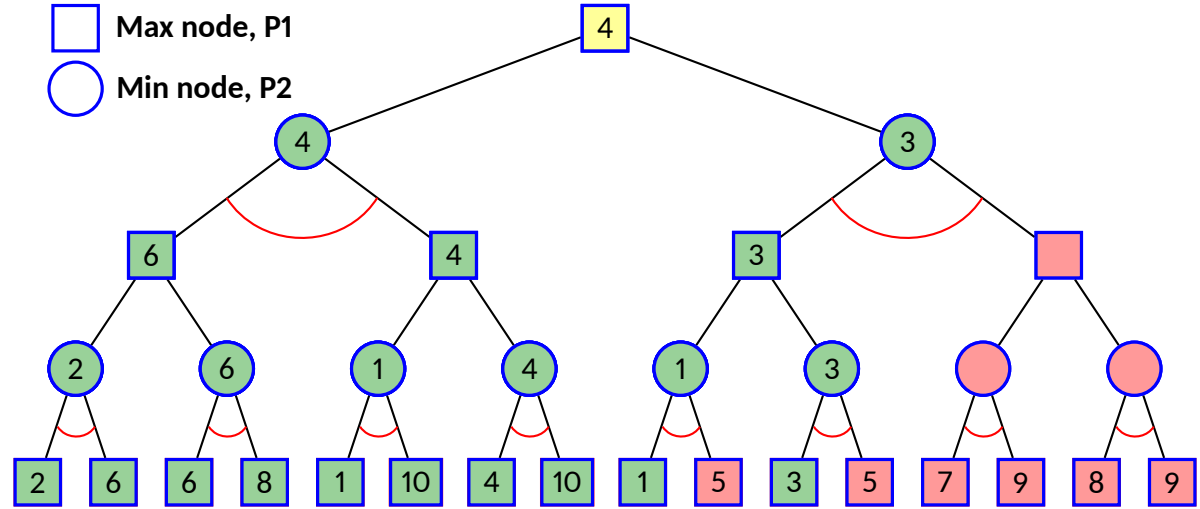
Game tree: Depth first search



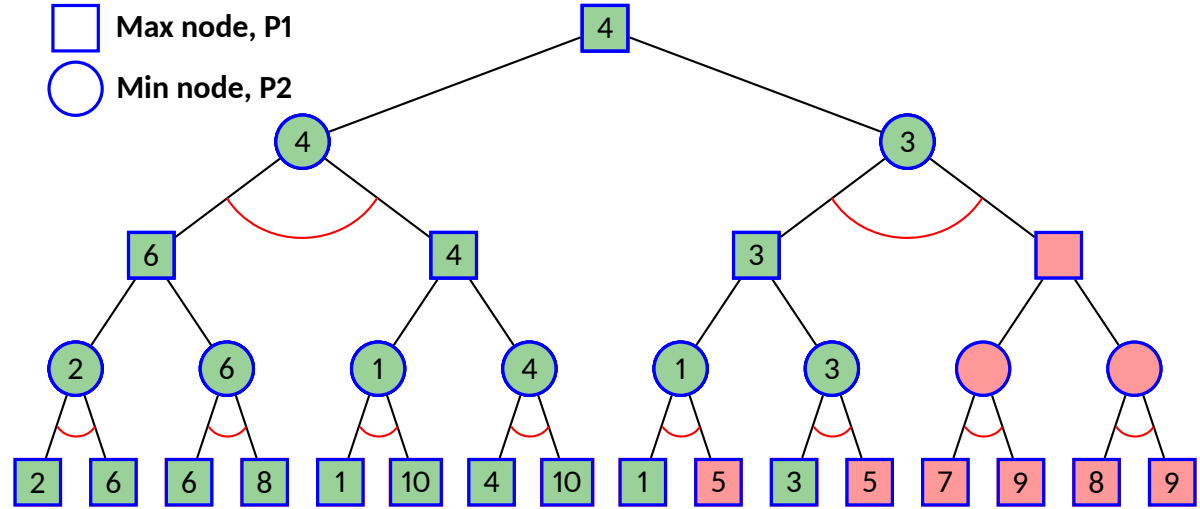
Game tree: Depth first search



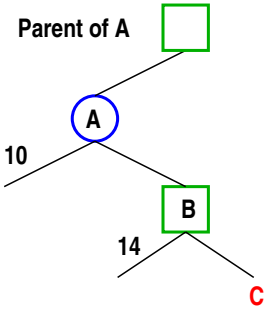
Game tree: Depth first search

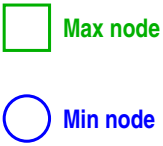


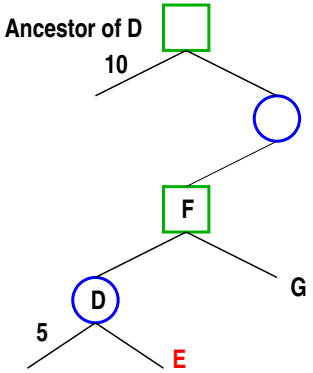
Game tree: Depth first search



Pruning rules in game tree search







Shallow cut-off:

Exploring node C and its successor is meaningless because payoff at B is at least 14, and hence A will never choose the move to B.

Deep cut-off:

Exploring node E and its successors is meaningless because the payoff at node D is at most 5, whereas the ROOT can already guarantee a payoff of 10 by choosing the left move.

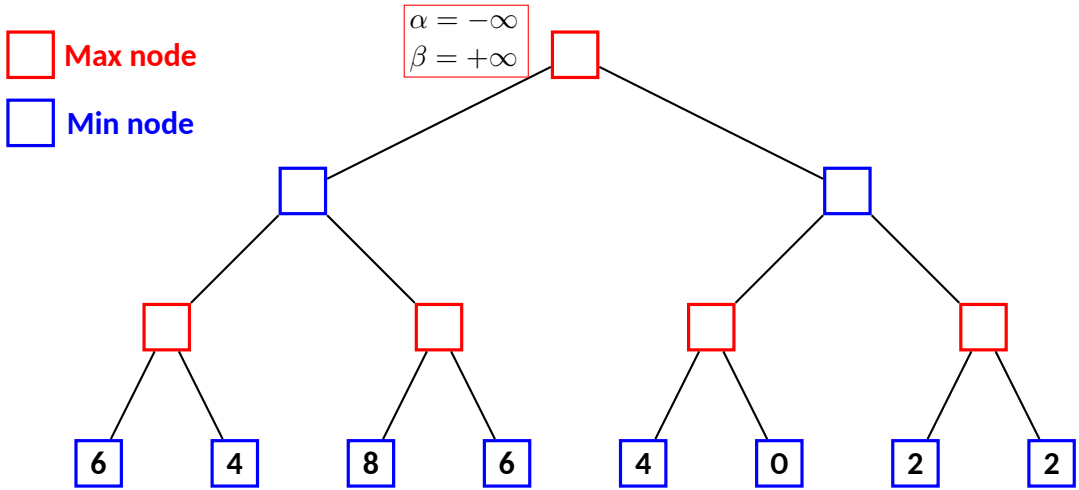
Alpha-Beta pruning

- **Alpha Bound of J(α):**
  - **The max current payoff of all MAX ancestors of J (Lower Bound)**
  - **Exploration of a min node, J, is stopped when its payoff β (Upper Bound) equals or falls below alpha.**
- **Beta Bound of J(β):**
  - **The min current payoff of all MIN ancestors of J (Upper Bound)**
  - **Exploration of a max node, J, is stopped when its payoff α (Lower Bound) equals or exceeds beta**
- **In a max node, we update alpha or Lower Bound**
- **In a min node, we update beta or Upper Bound**
- **In both min and max nodes, we return when $\alpha \geq \beta$**

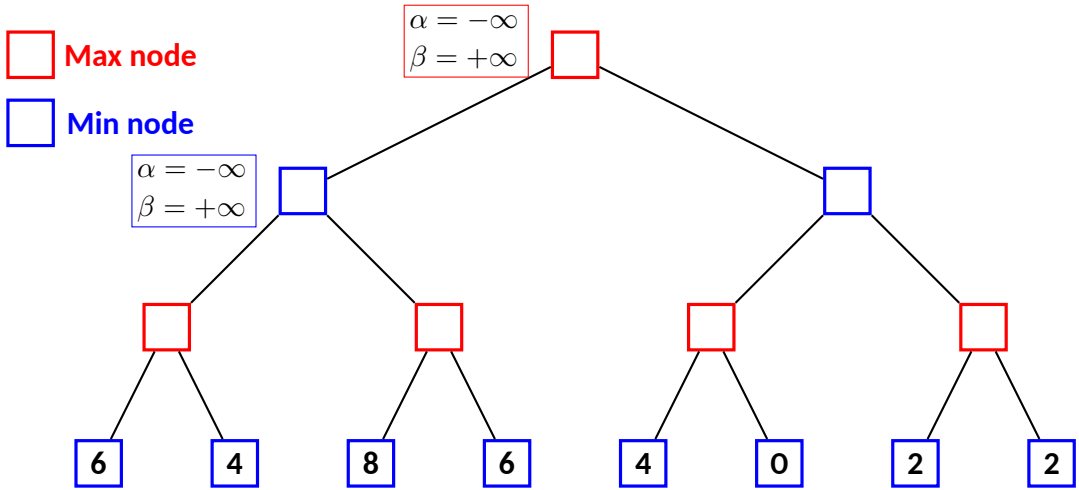
Alpha-Beta pruning procedure $V(J, \alpha, \beta)$

1. If J is a terminal node, return $V(J) = h(J)$
2. If J is max node
   a. For each successor J_i of J
      (i) Set $\alpha = \max\{\alpha, V(J_i, \alpha, \beta)\}$
      (ii) If $\alpha \geq \beta$ then return β else continue
   b. Return α
3. If J is min node
   a. For each successor J_i of J
      (i) Set $\beta = \min\{\beta, V(J_i, \alpha, \beta)\}$
      (ii) If $\alpha \geq \beta$ then return α else continue
   b. Return β

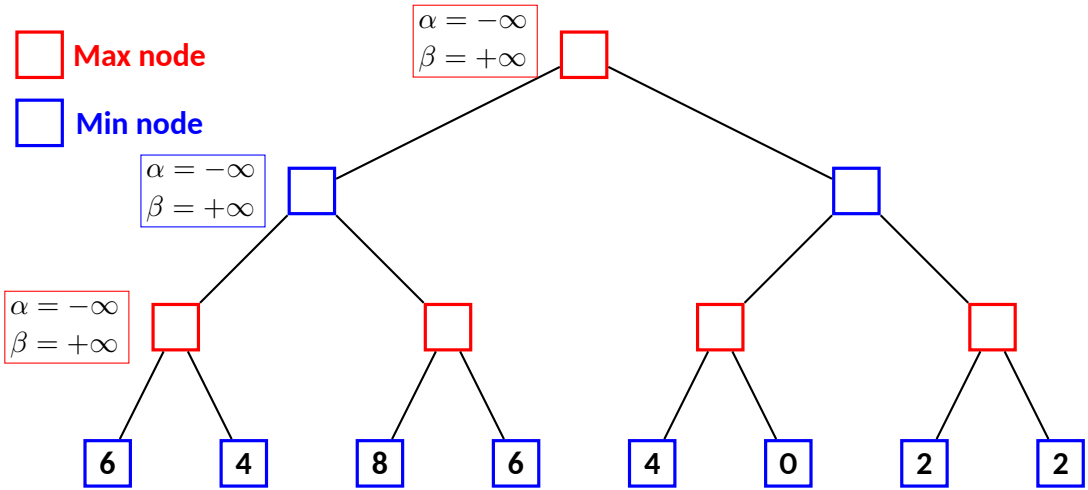
Alpha-Beta pruning



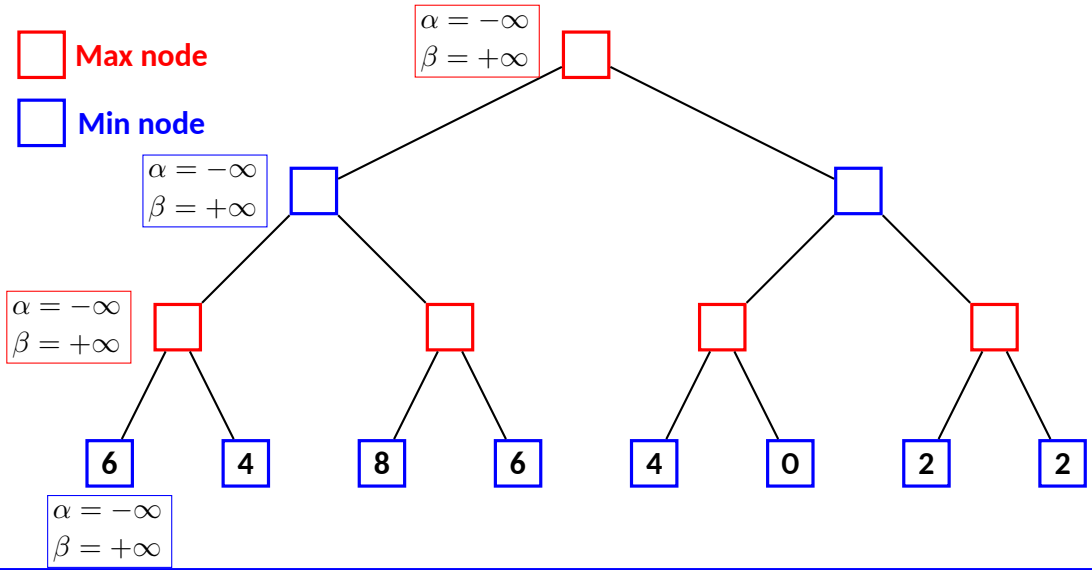
Alpha-Beta pruning



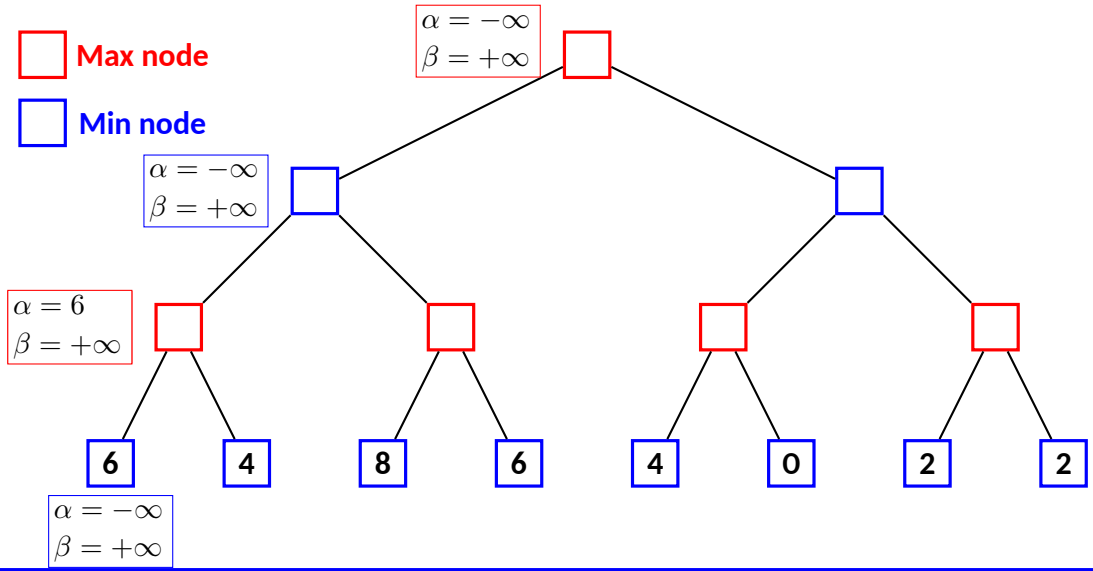
Alpha-Beta pruning



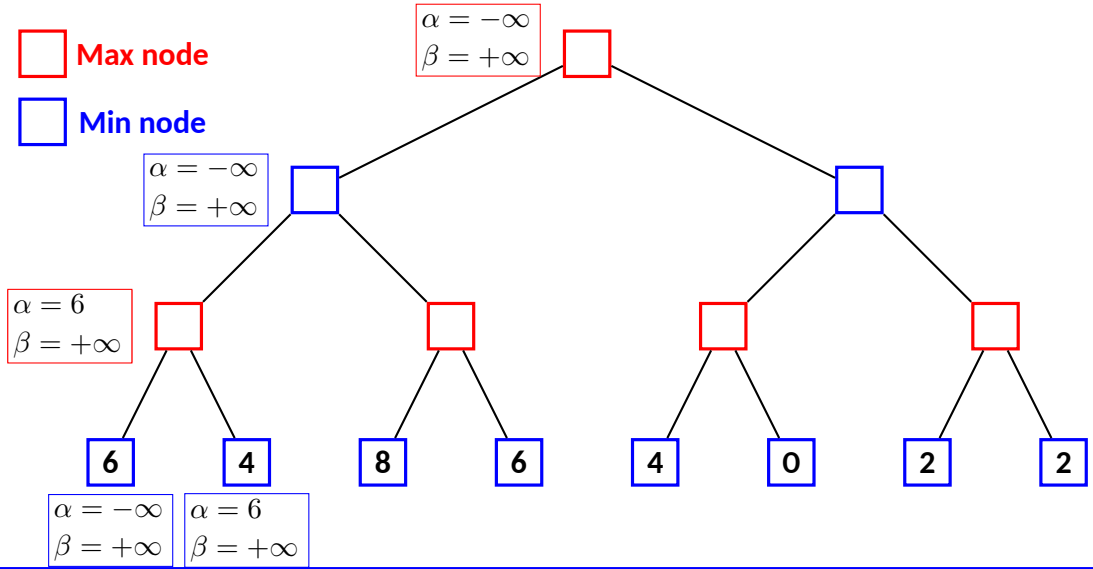
Alpha-Beta pruning



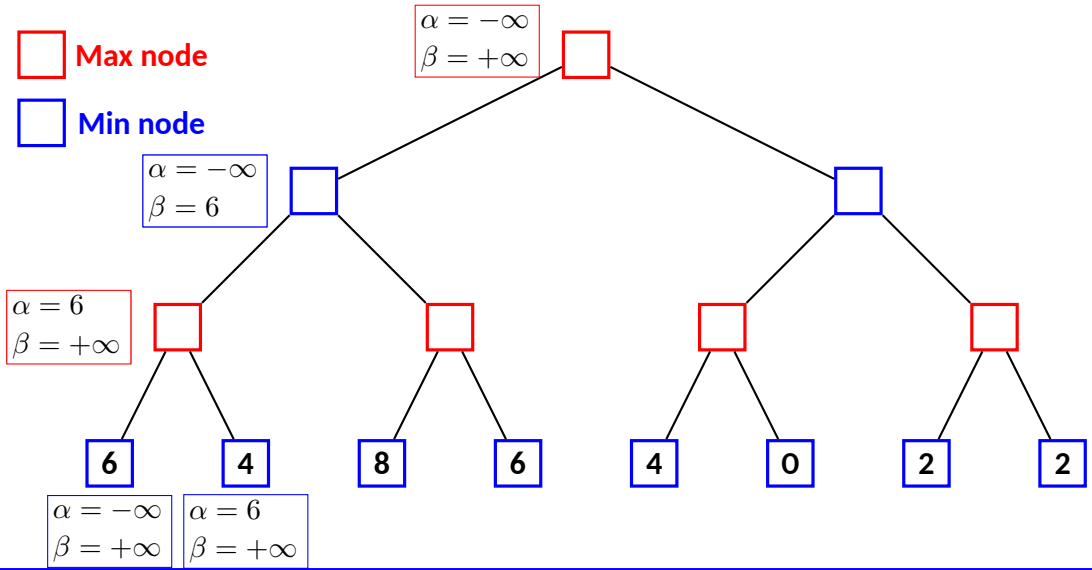
Alpha-Beta pruning



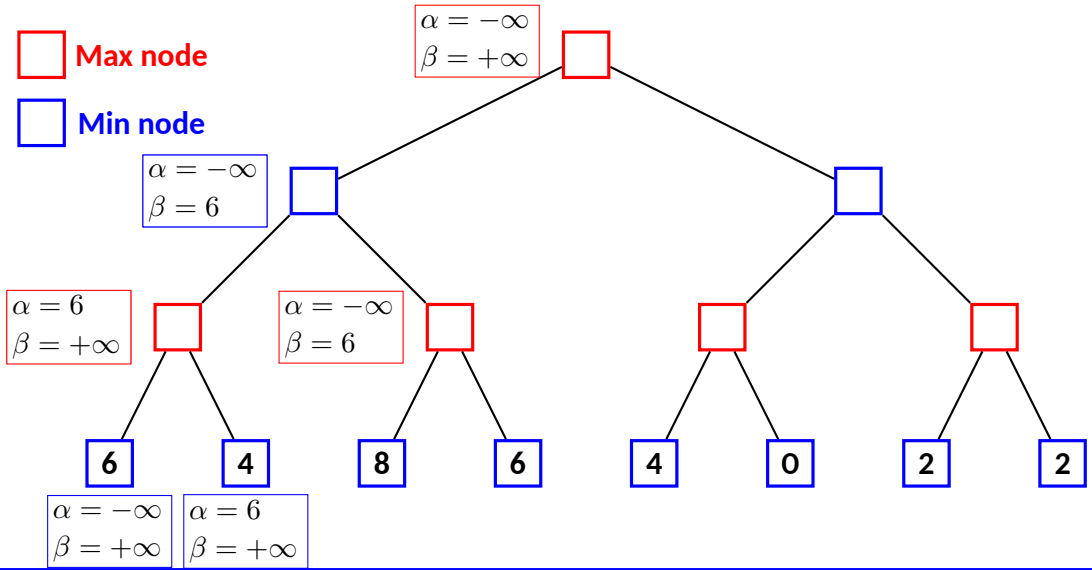
Alpha-Beta pruning



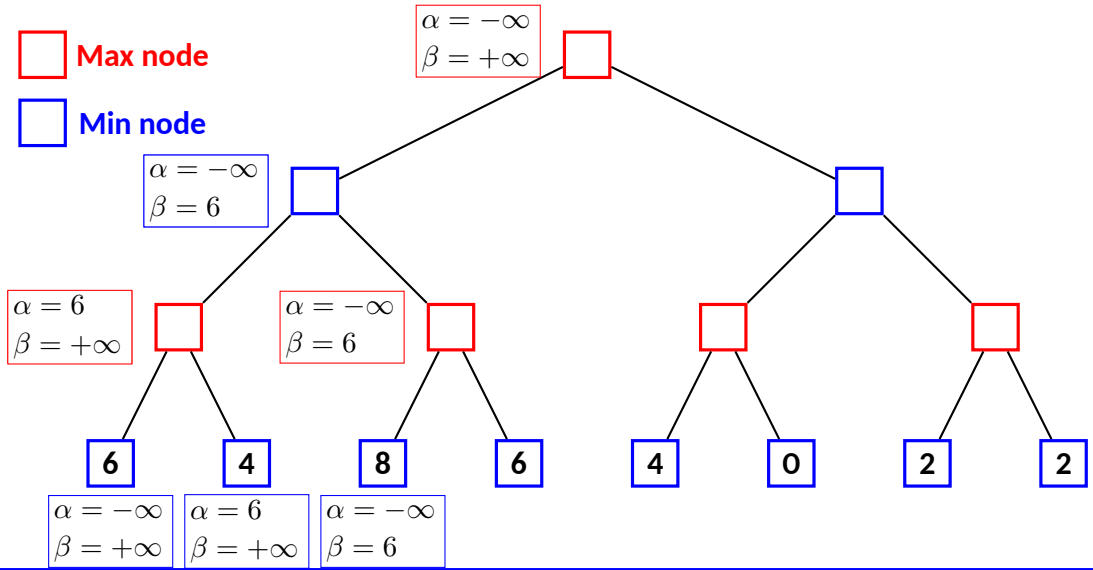
Alpha-Beta pruning



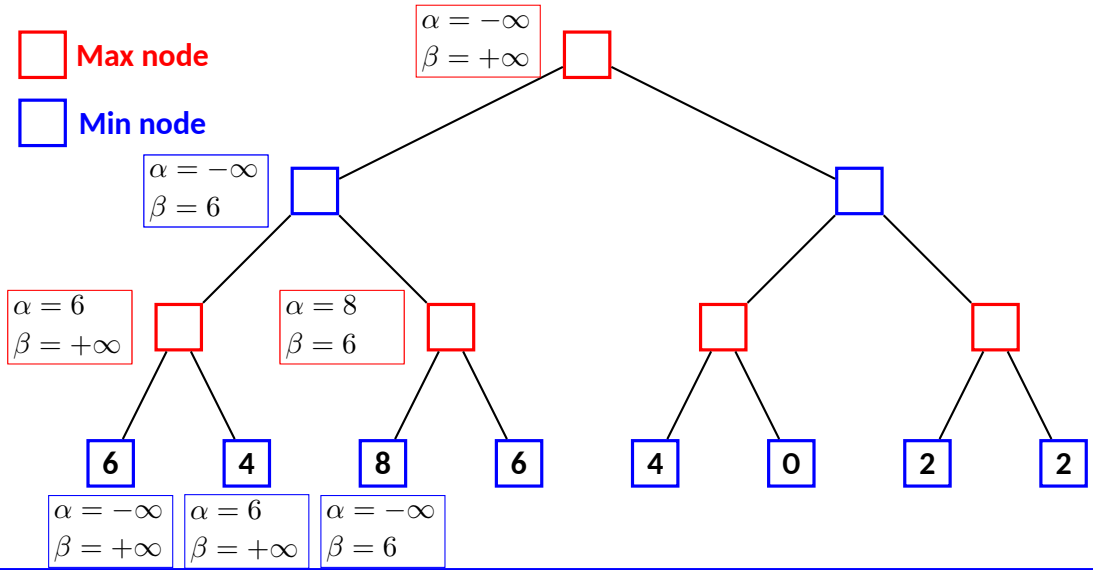
Alpha-Beta pruning



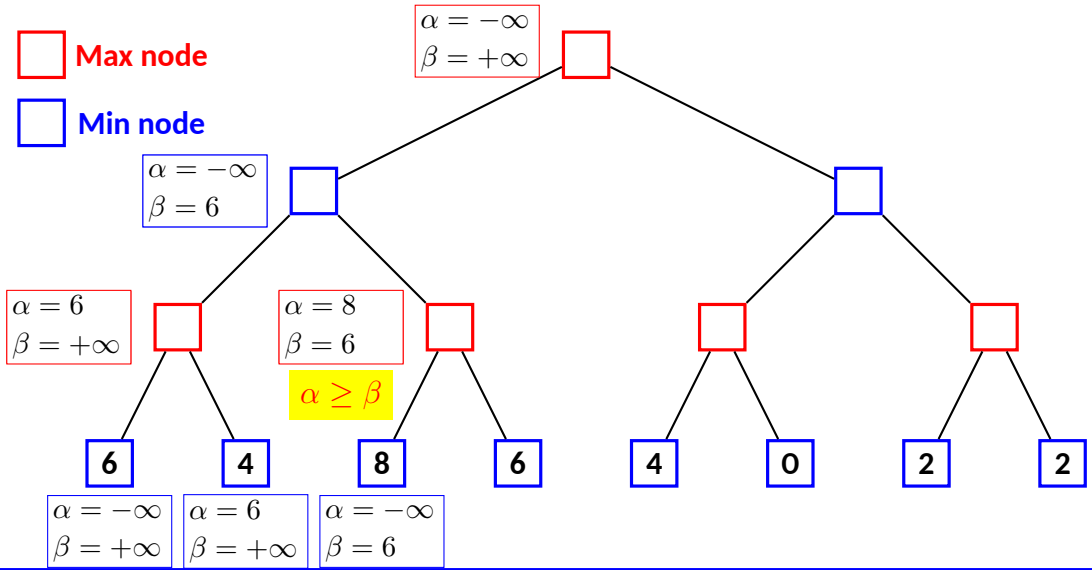
Alpha-Beta pruning



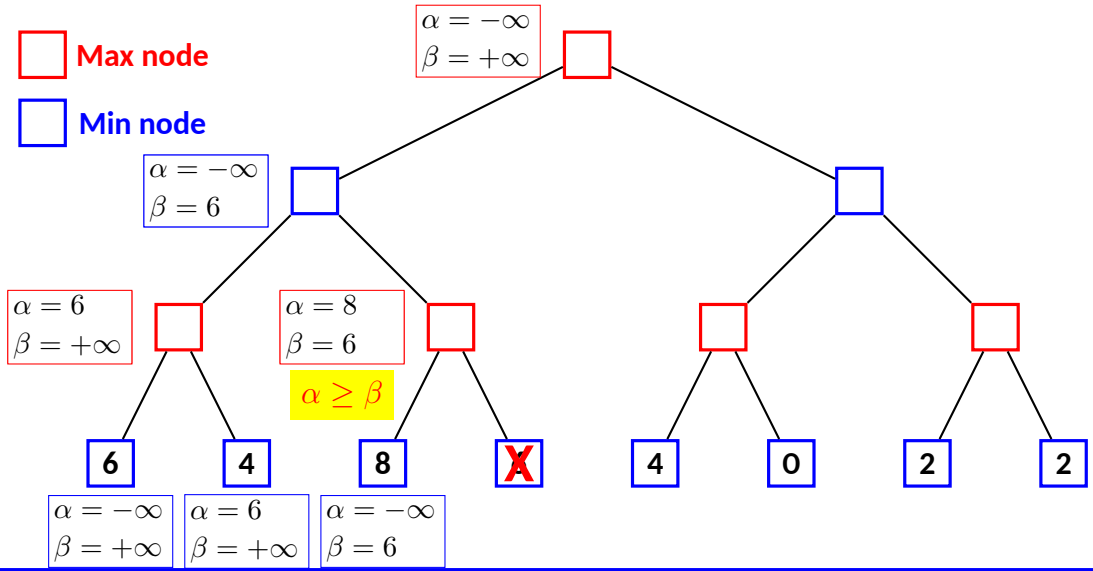
Alpha-Beta pruning



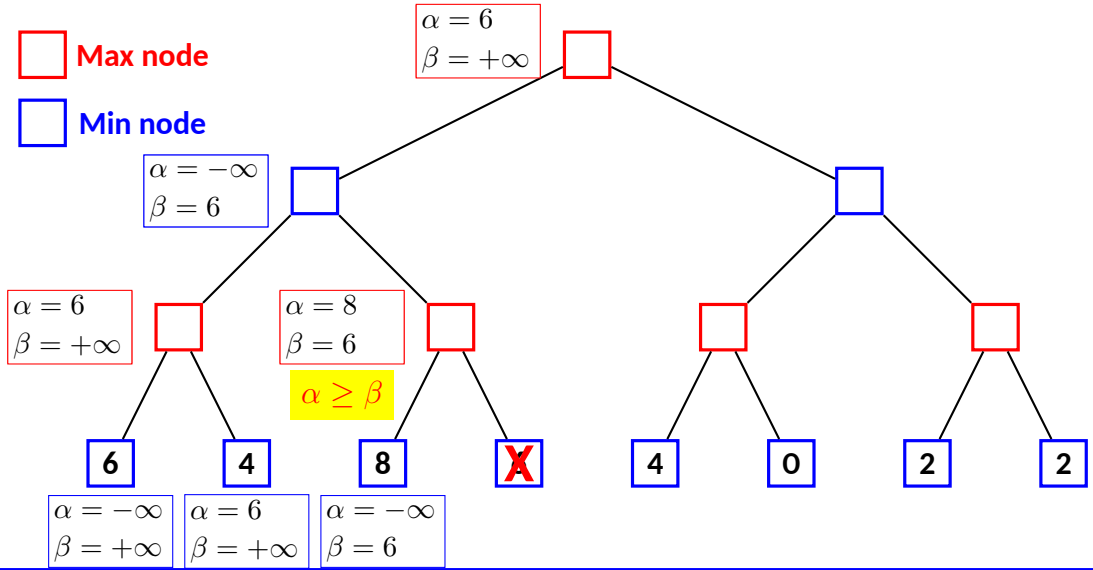
Alpha-Beta pruning



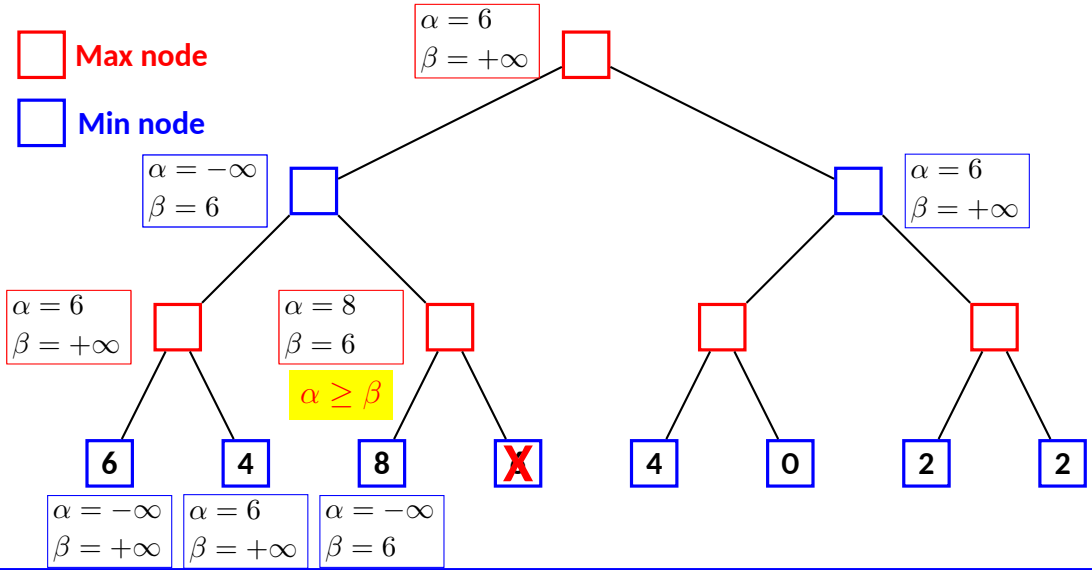
Alpha-Beta pruning



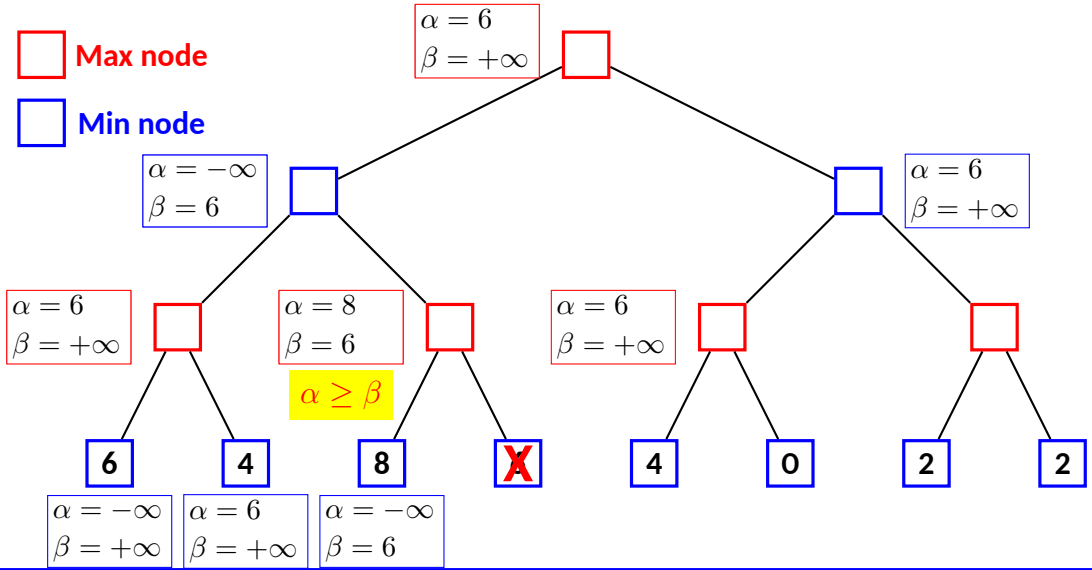
Alpha-Beta pruning



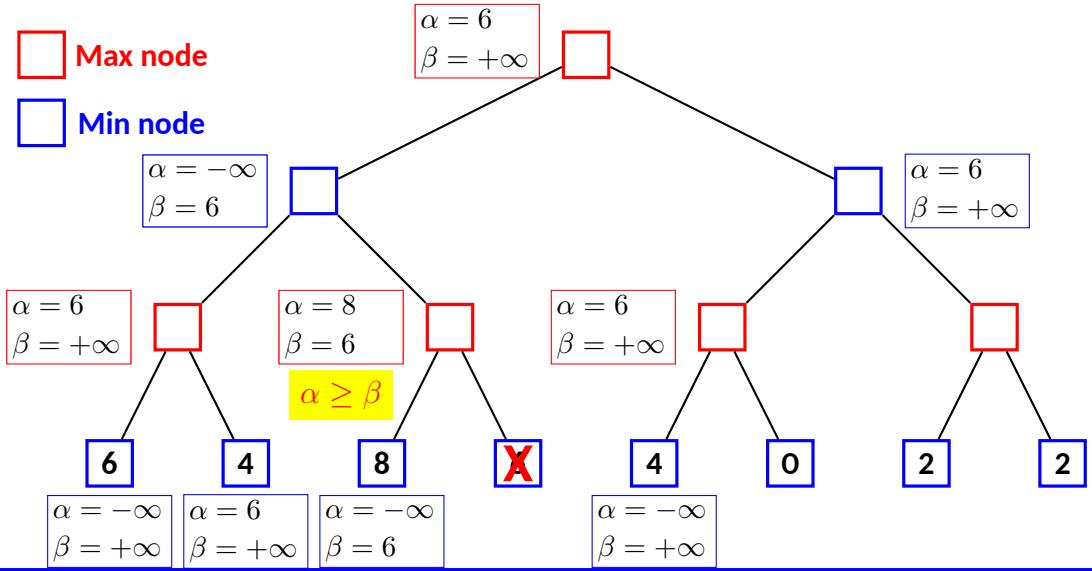
Alpha-Beta pruning



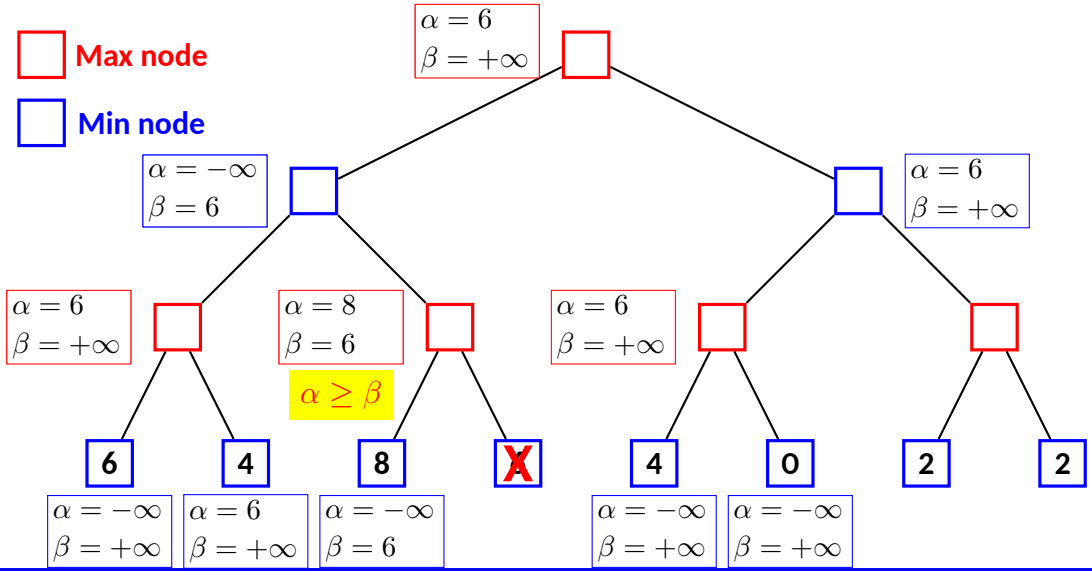
Alpha-Beta pruning



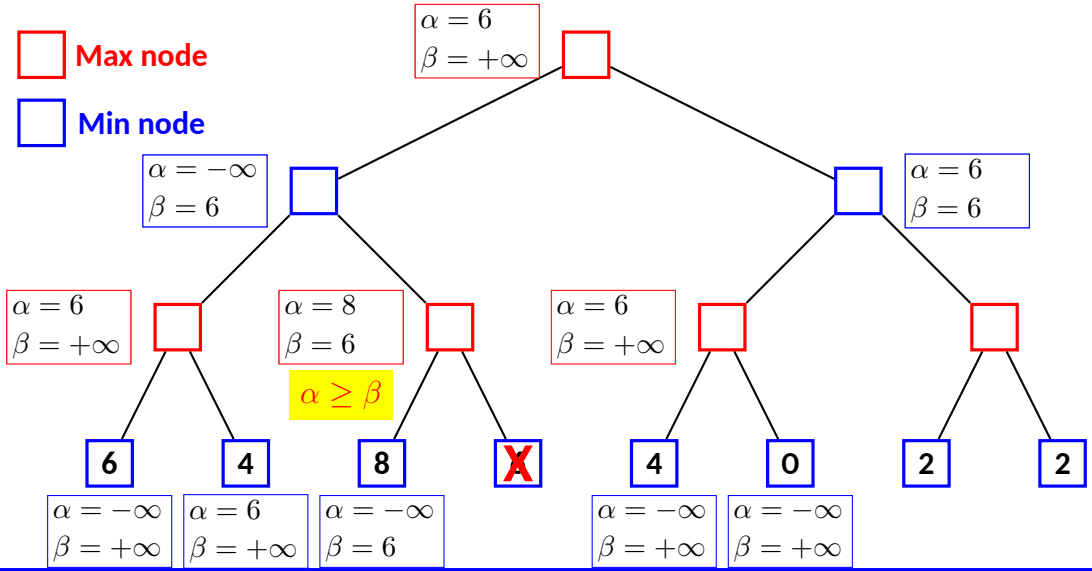
Alpha-Beta pruning



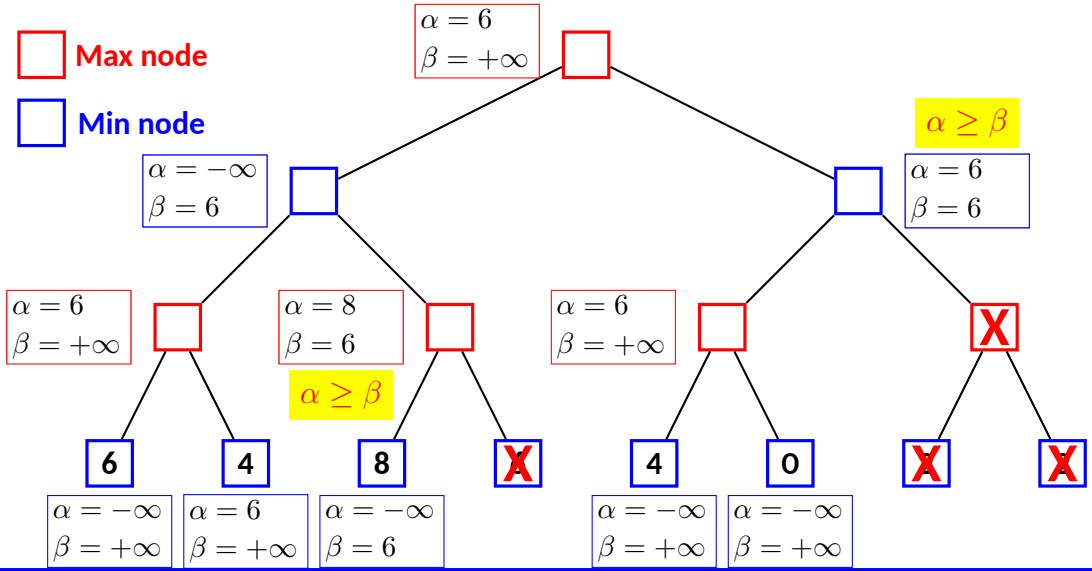
Alpha-Beta pruning



Alpha-Beta pruning



Alpha-Beta pruning



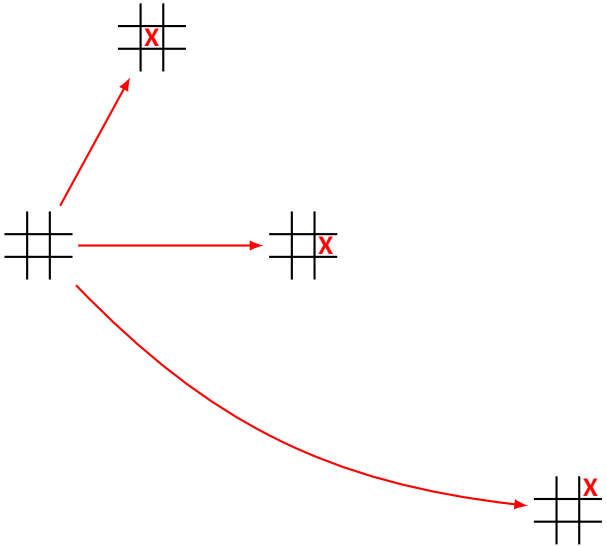
Other challenges in game tree search

- Design of heuristics in game trees
- Perform best first search in game tree
- Multi-player games
- Team games
- Probabilistic games
- Real life applications

Tic-tac-toe - 1

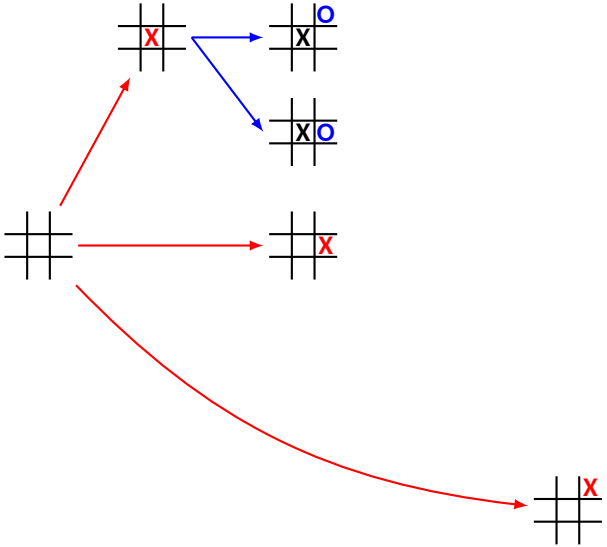
h(s) = (Number of complete cols, rows, diagonals open for MAX) — (Number of complete cols, rows, diagonals open for MIN)

Tic-tac-toe - 1



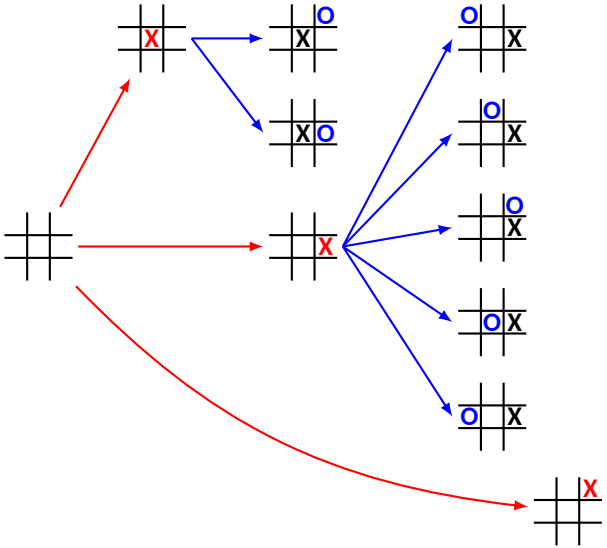
h(s) = (Number of complete cols, rows, diagonals open for MAX) — (Number of complete cols, rows, diagonals open for MIN)

Tic-tac-toe - 1



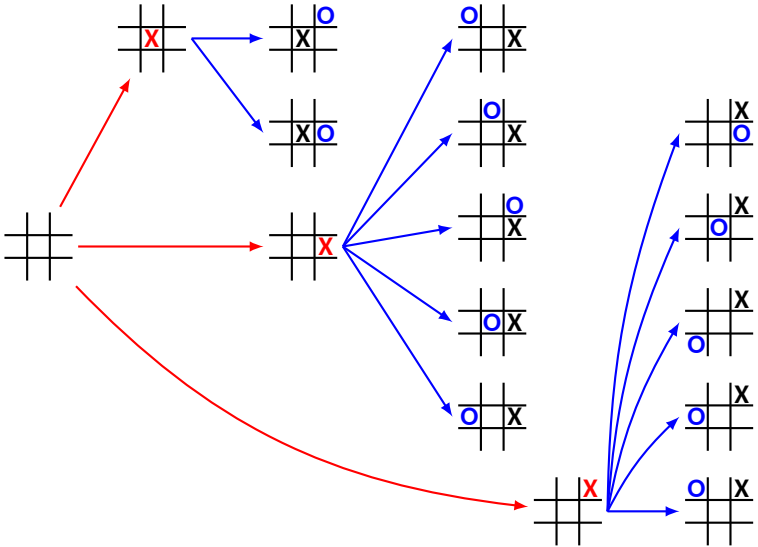
h(s) = (Number of complete cols, rows, diagonals open for MAX) — (Number of complete cols, rows, diagonals open for MIN)

Tic-tac-toe - 1



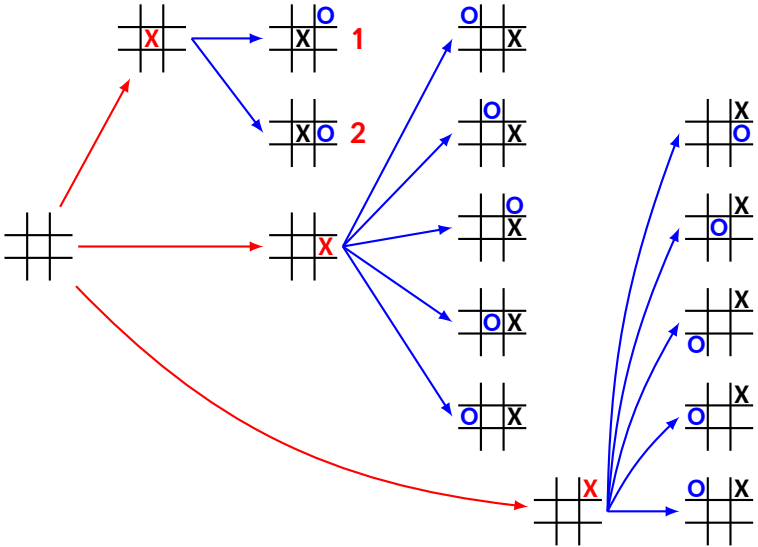
h(s) = (Number of complete cols, rows, diagonals open for MAX) — (Number of complete cols, rows, diagonals open for MIN)

Tic-tac-toe - 1



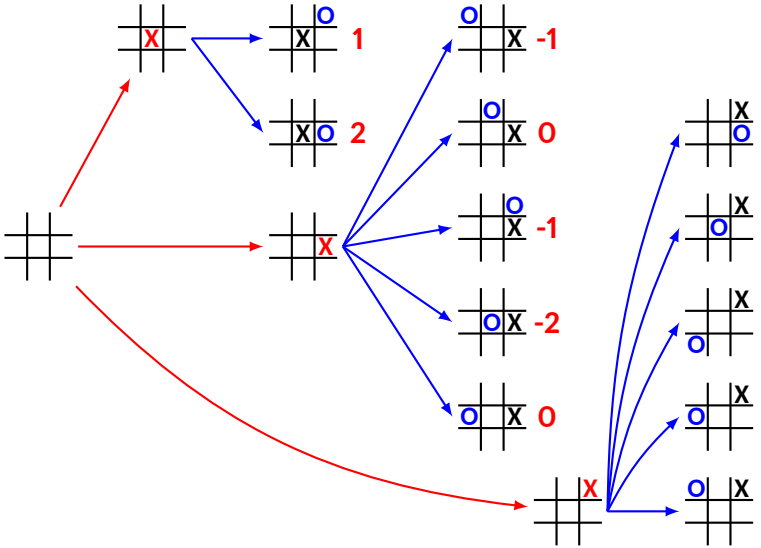
h(s) = (Number of complete cols, rows, diagonals open for MAX) — (Number of complete cols, rows, diagonals open for MIN)

Tic-tac-toe - 1



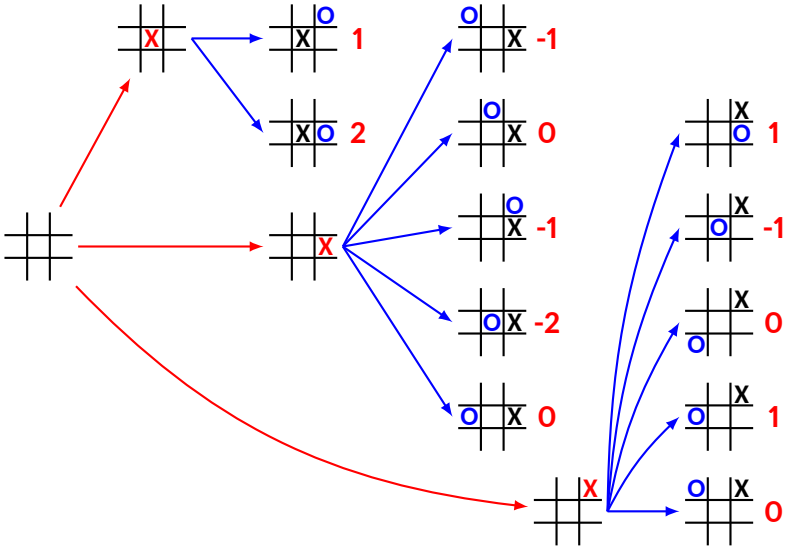
h(s) = (Number of complete cols, rows, diagonals open for MAX) — (Number of complete cols, rows, diagonals open for MIN)

Tic-tac-toe - 1



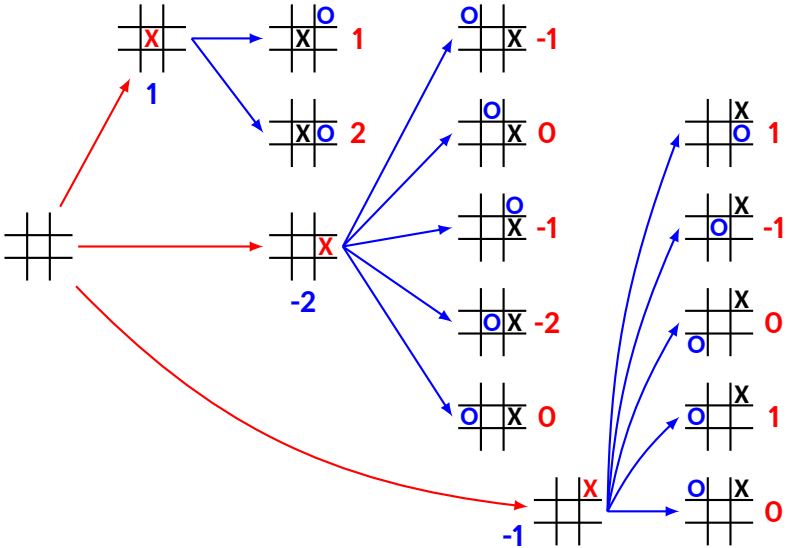
h(s) = (Number of complete cols, rows, diagonals open for MAX) — (Number of complete cols, rows, diagonals open for MIN)

Tic-tac-toe - 1



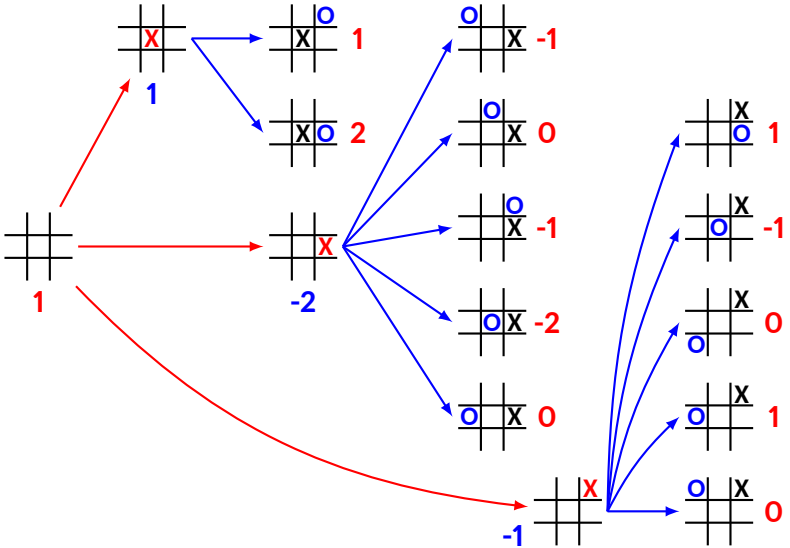
h(s) = (Number of complete cols, rows, diagonals open for MAX) — (Number of complete cols, rows, diagonals open for MIN)

Tic-tac-toe - 1



h(s) = (Number of complete cols, rows, diagonals open for MAX) — (Number of complete cols, rows, diagonals open for MIN)

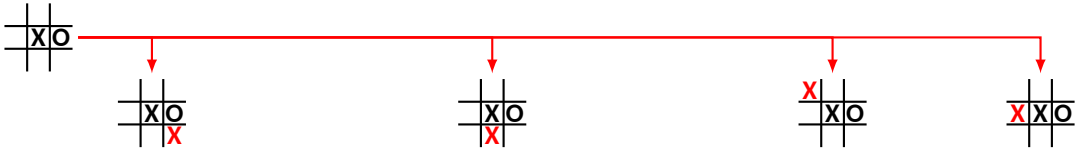
Tic-tac-toe - 1



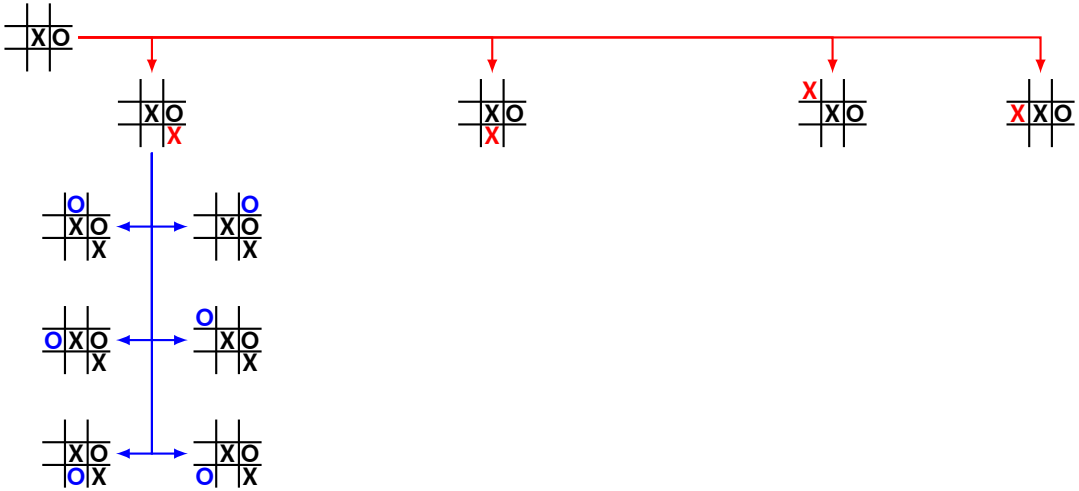
h(s) = (Number of complete cols, rows, diagonals open for MAX) — (Number of complete cols, rows, diagonals open for MIN)

Tic-tac-toe - 2

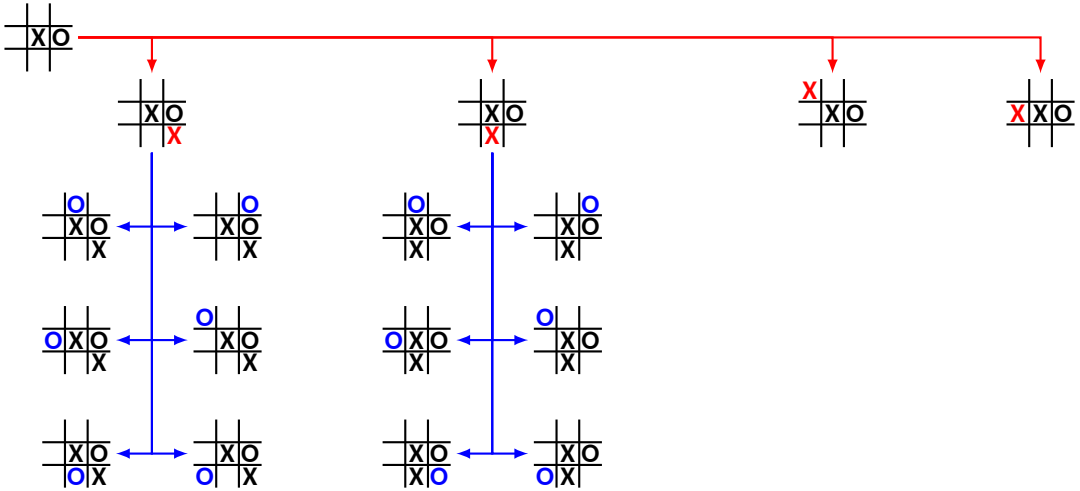
Tic-tac-toe - 2



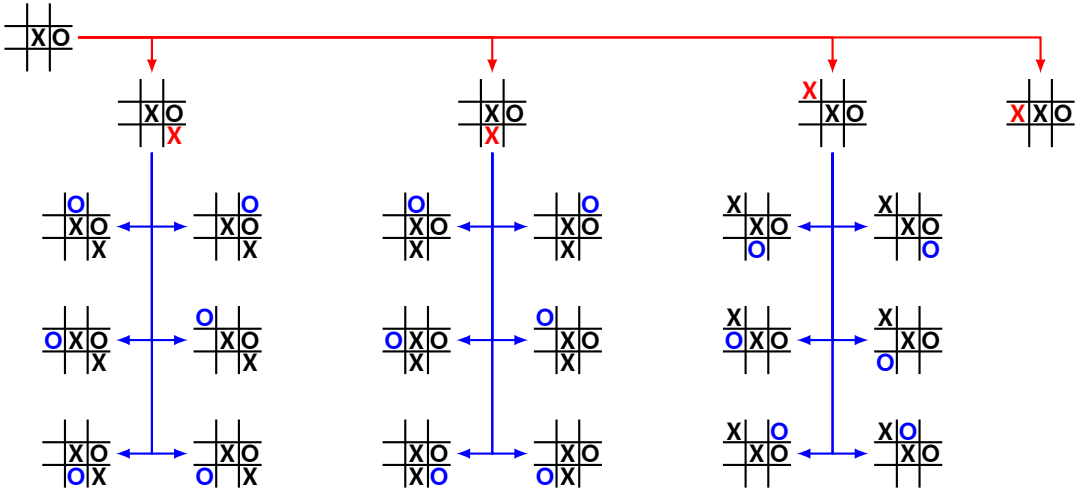
Tic-tac-toe - 2



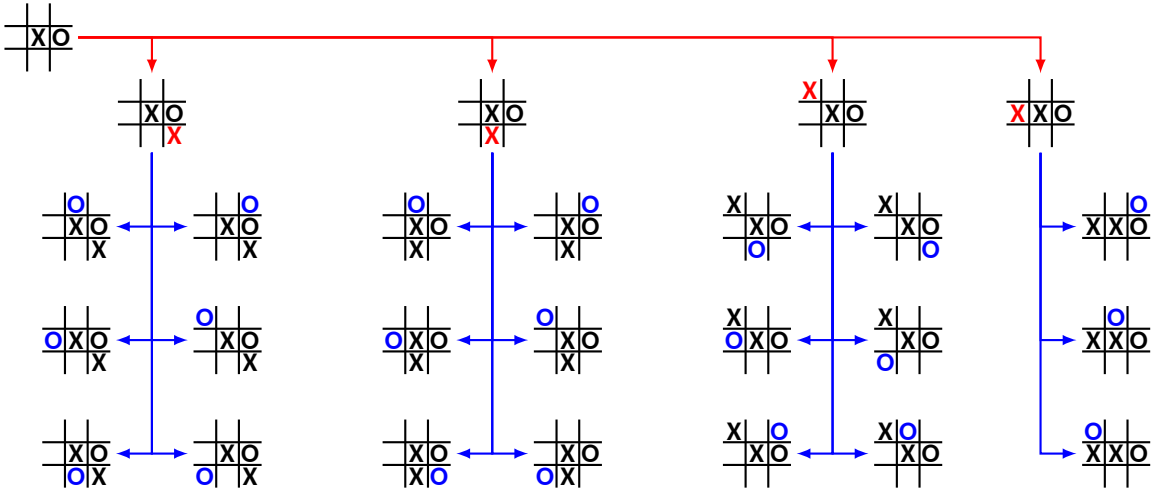
Tic-tac-toe - 2



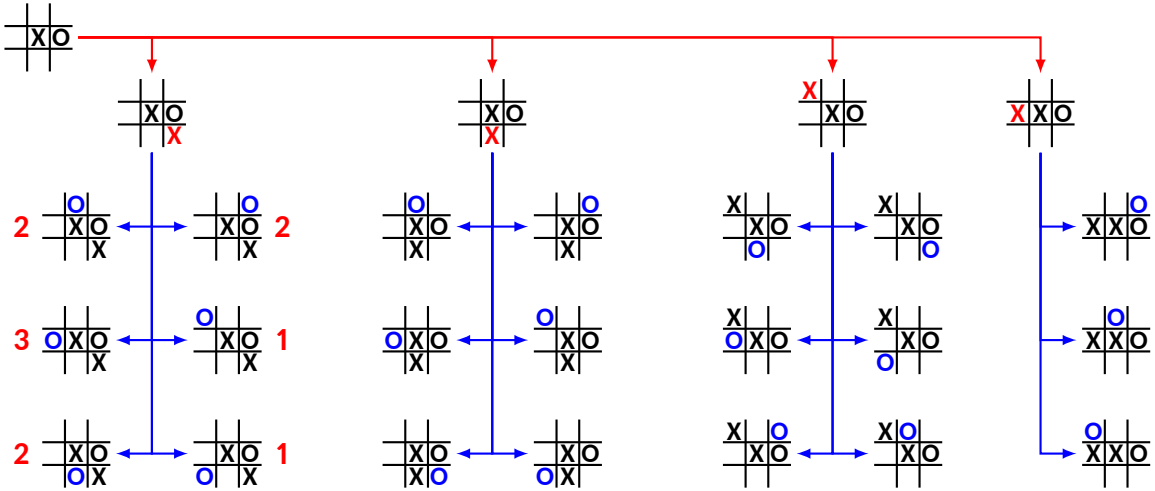
Tic-tac-toe - 2



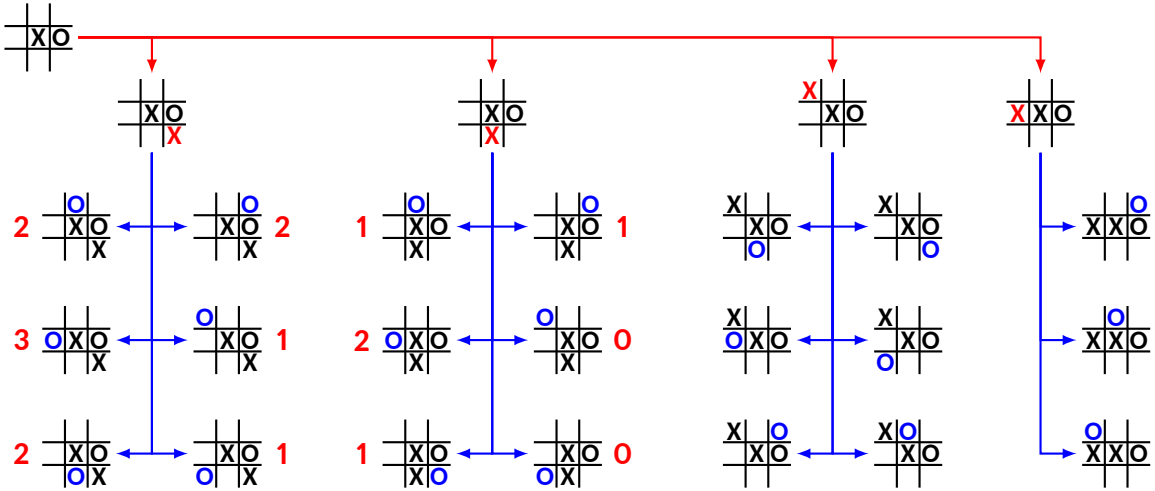
Tic-tac-toe - 2



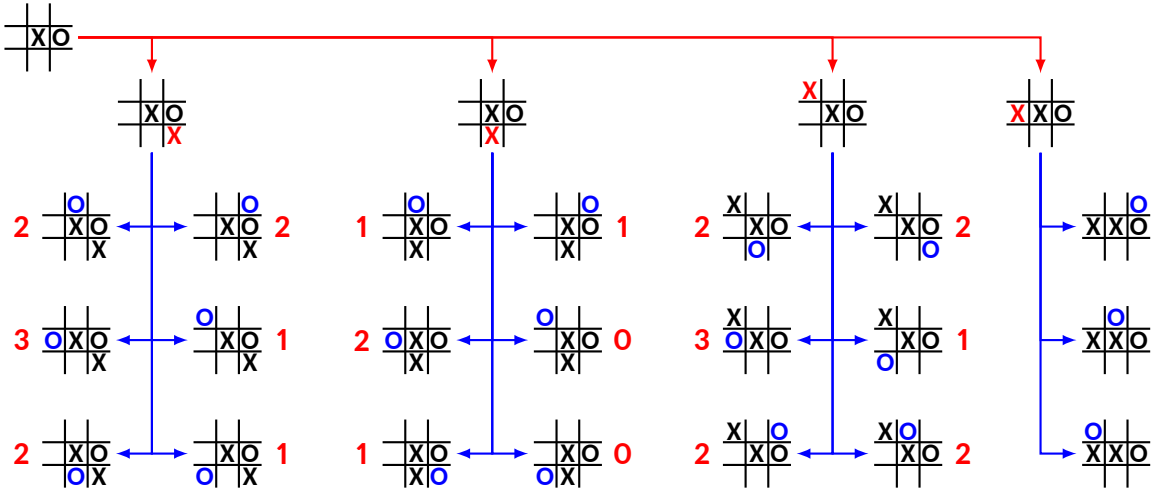
Tic-tac-toe - 2



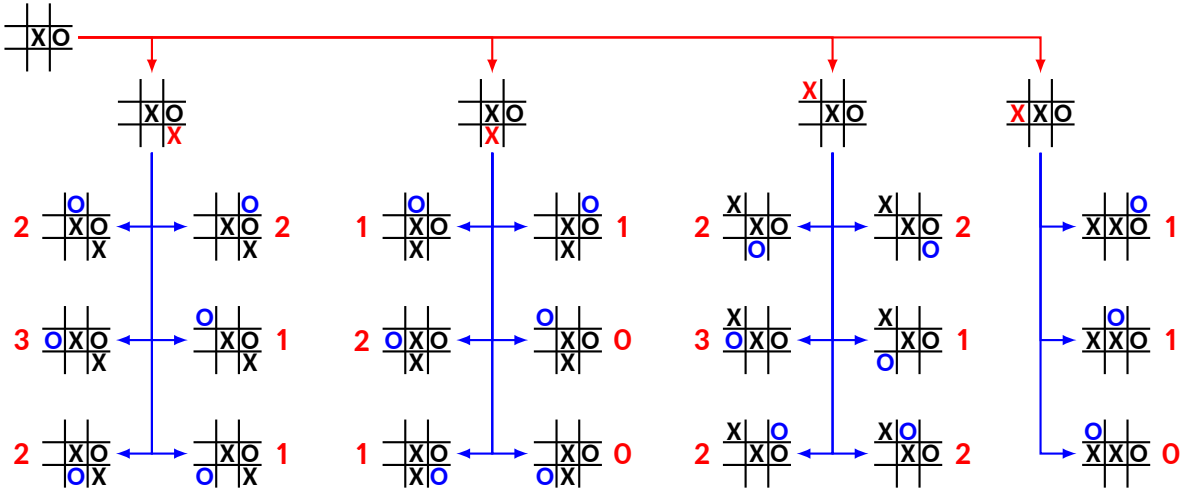
Tic-tac-toe - 2



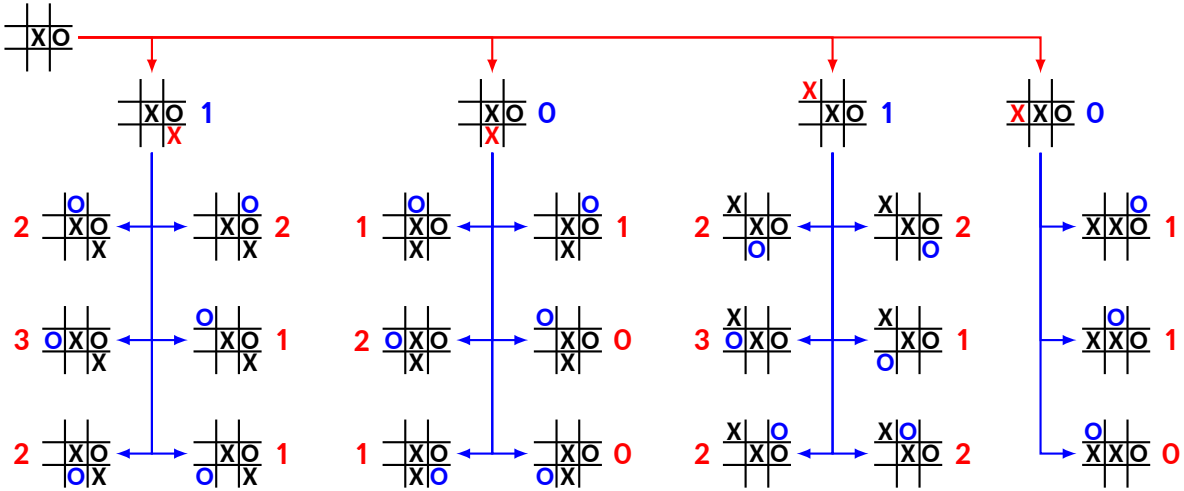
Tic-tac-toe - 2



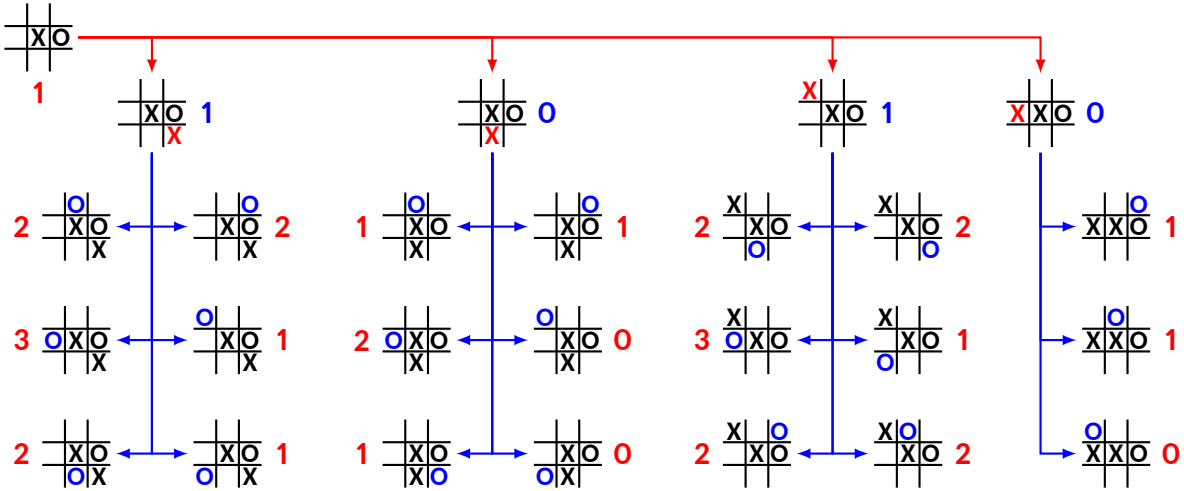
Tic-tac-toe - 2



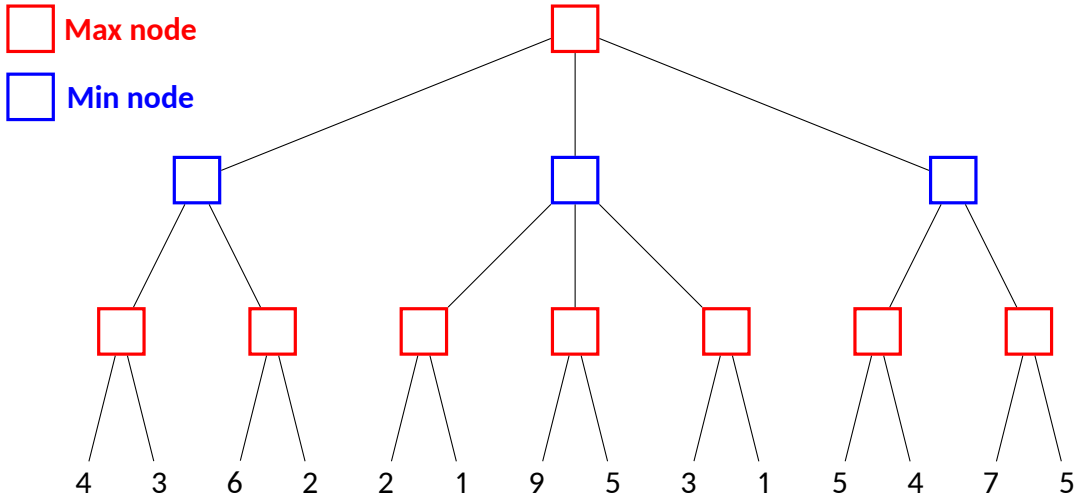
Tic-tac-toe - 2



Tic-tac-toe - 2



Alpha-Beta pruning (Exercise)



Thank you!