Adversarial Search: Mini-Max

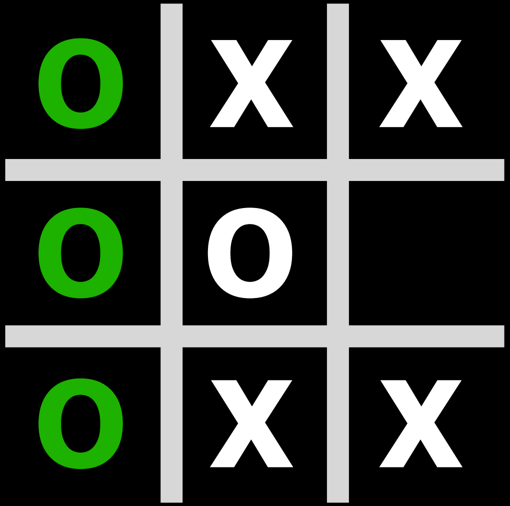
Arun Chauhan@Sitare

Computer Science and Engineering

	O	X
O	X	X
O		X

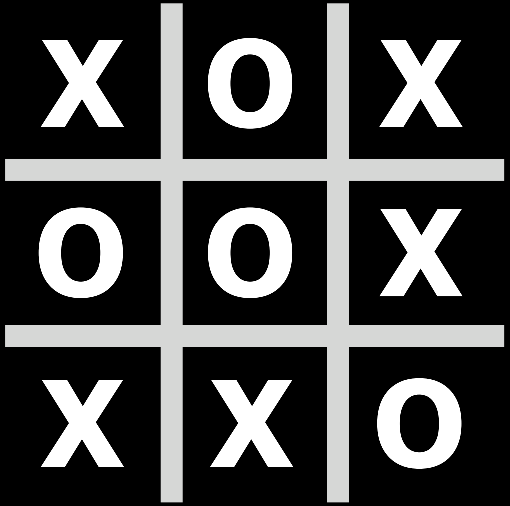
	O	X
O	X	X
O		X

Minimax



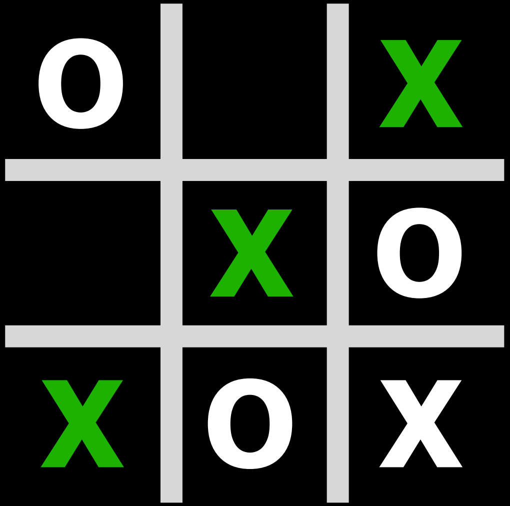
O	X	X
O	O	
O	X	X

-1



X	O	X
O	O	X
X	X	O

0



O		**X**
	X	O
X	O	X

1

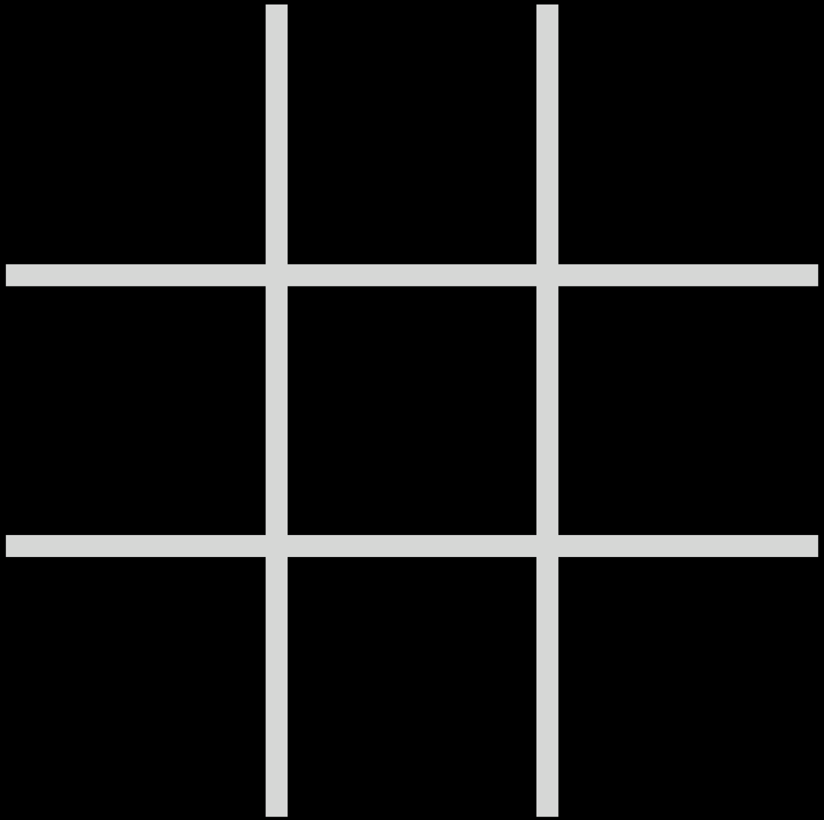
Minimax

- MAX (X) aims to maximize score.
- MIN (O) aims to minimize score.

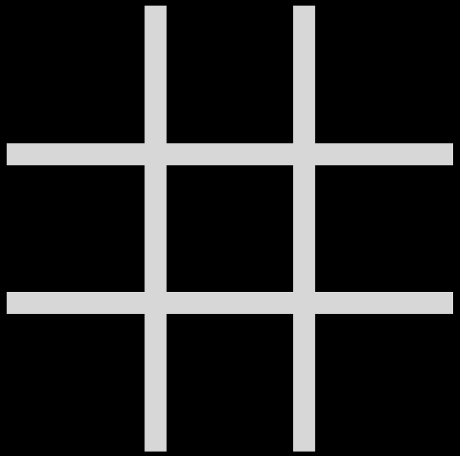
Game

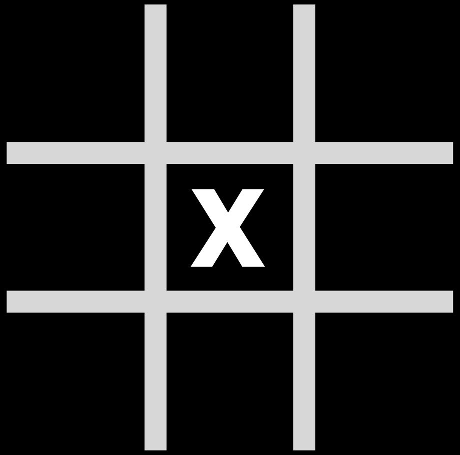
- S_0 : initial state
- PLAYER(s) : returns which player to move in state s
- ACTIONS(s) : returns legal moves in state s
- RESULT(s, a) : returns state after action a taken in state s
- TERMINAL(s) : checks if state s is a terminal state
- UTILITY(s) : final numerical value for terminal state s

Initial State

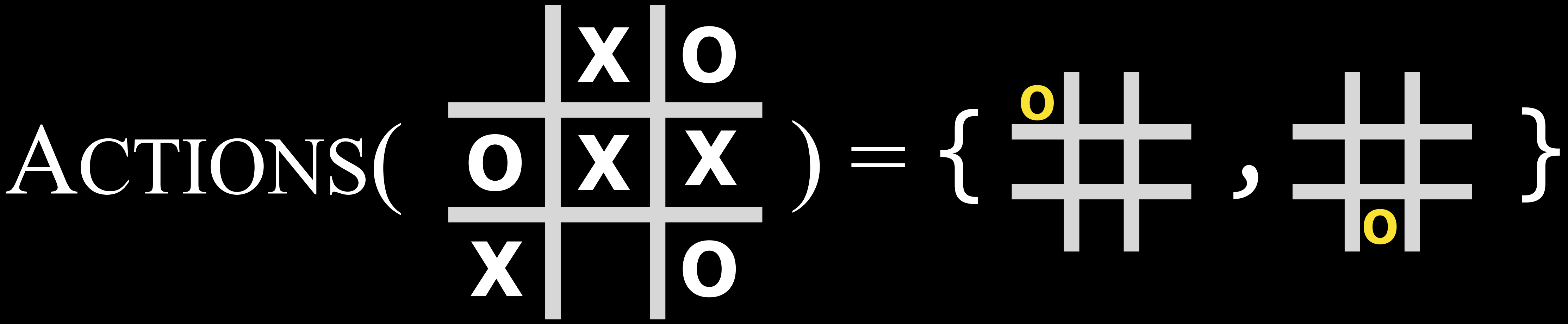


Player(*s*)

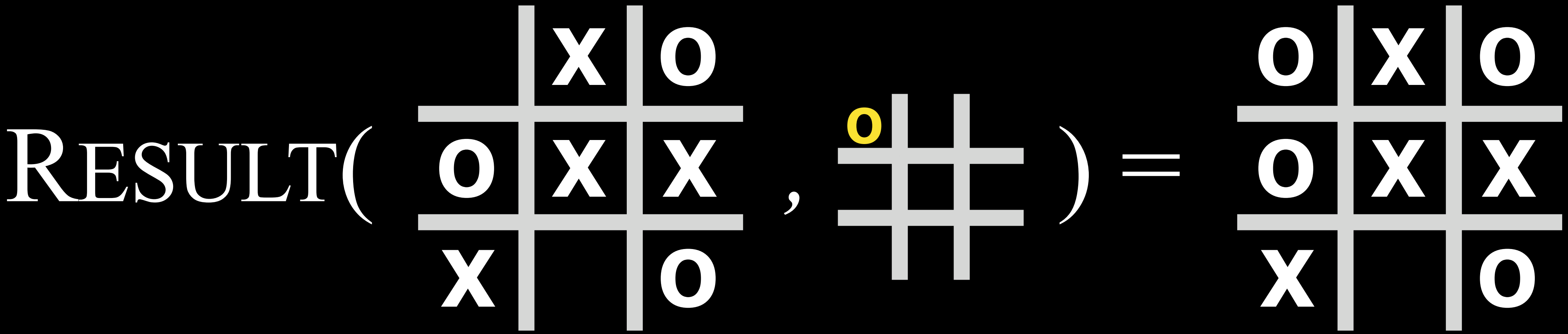
Player() = **X**

Player() = **O**

ACTIONS(s)



RESULT(*s*, *a*)



TERMINAL(s)

TERMINAL(

O		
O	X	
X	O	X

) = `false`

TERMINAL(

O		X
O	X	
X	O	X

) = `true`

$\textsc{Utility}(s)$

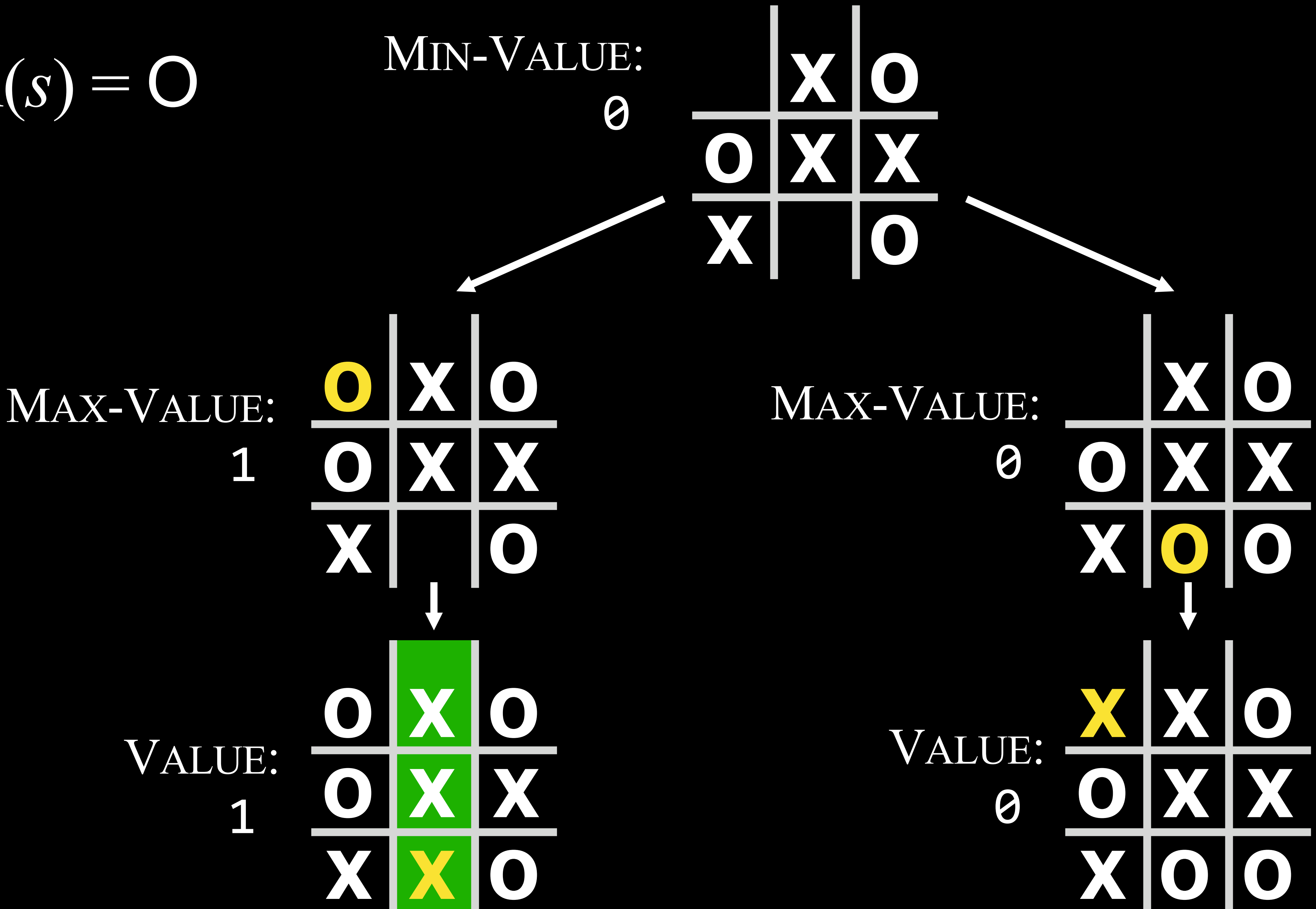
$$\textsc{Utility}\left(\begin{array}{c|c|c} \text{O} & & \text{X} \\ \hline \text{O} & \text{X} & \\ \hline \text{X} & \text{O} & \text{X} \end{array}\right) = 1$$

$$\textsc{Utility}\left(\begin{array}{c|c|c} \text{O} & \text{X} & \text{X} \\ \hline \text{X} & \text{O} & \\ \hline \text{O} & \text{X} & \text{O} \end{array}\right) = -1$$

O	X	O
O	X	X
X	X	O

Value: 1

PLAYER(s) = O



MIN-VALUE:
0

	X	O
O	X	X
X		O

MAX-VALUE:
1

O	X	O
O	X	X
X		O

MAX-VALUE:
0

	X	O
O	X	X
X	O	O

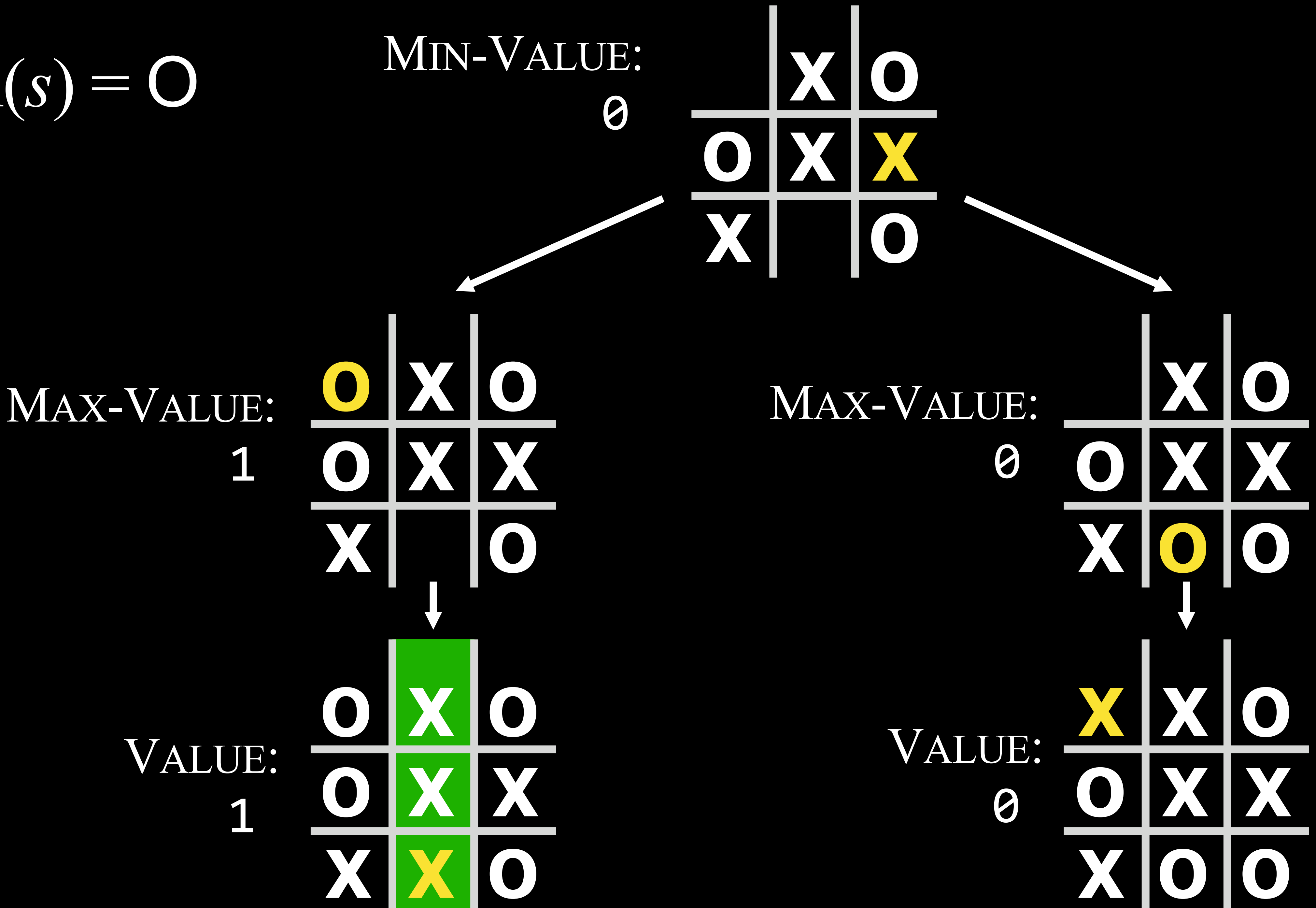
VALUE:
1

O	X	O
O	X	X
X	X	O

VALUE:
0

X	X	O
O	X	X
X	O	O

PLAYER(s) = O



MIN-VALUE:
0

	X	O
O	X	X
X		O

MAX-VALUE:
1

O	X	O
O	X	X
X		O

MAX-VALUE:
0

	X	O
O	X	X
X	O	O

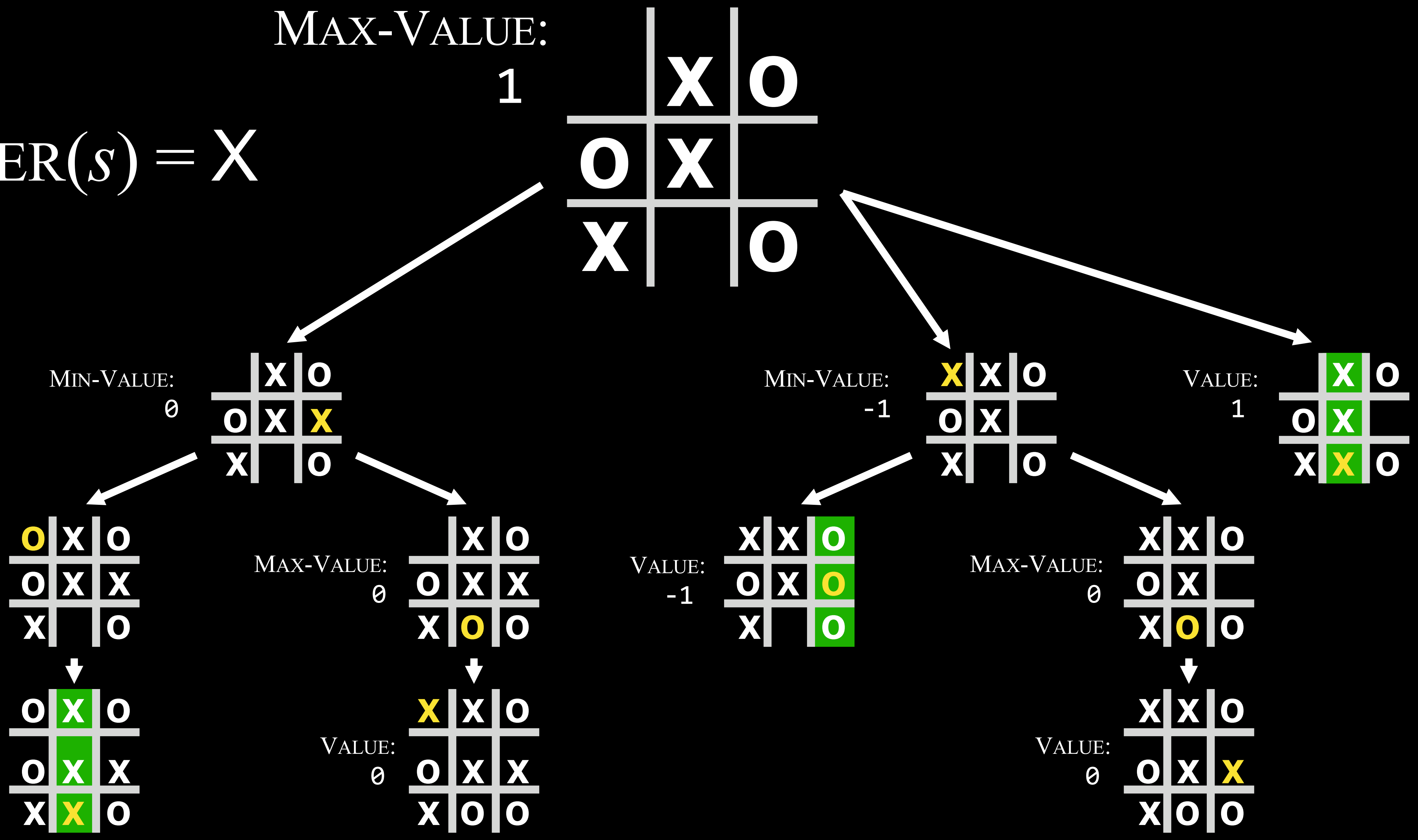
VALUE:
1

O	X	O
O	X	X
X	X	O

VALUE:
0

X	X	O
O	X	X
X	O	O

$\text{PLAYER}(s) = \text{X}$



MAX-VALUE:
1

MIN-VALUE:
0

MIN-VALUE:
-1

VALUE:
1

MAX-VALUE:
1

MAX-VALUE:
0

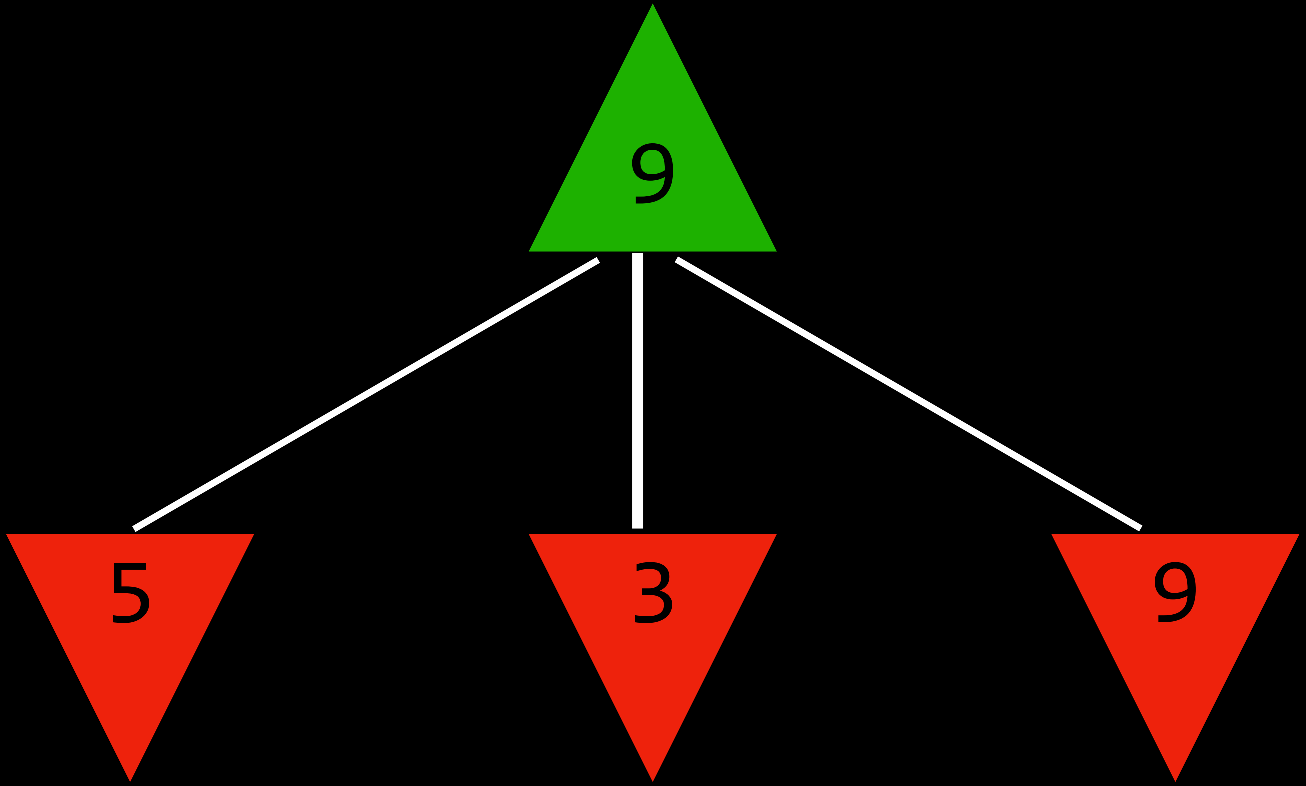
VALUE:
-1

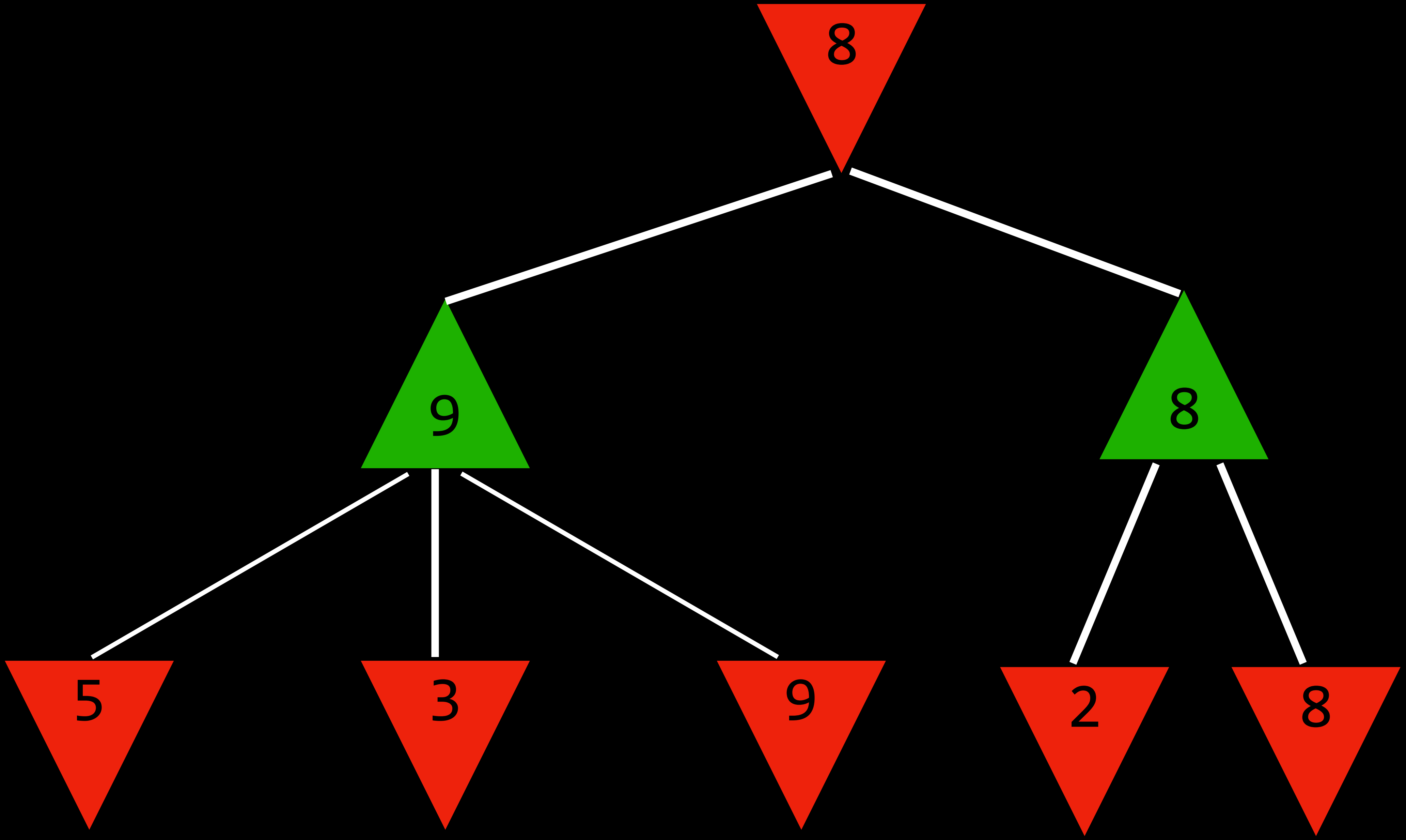
MAX-VALUE:
0

VALUE:
1

VALUE:
0

VALUE:
0


9
5
3
9


8
9
8
5
3
9
2
8

Minimax

- Given a state *s*:
  - MAX picks action *a* in ACTIONS(*s*) that produces highest value of MIN-VALUE(RESULT(*s*, *a*))
  - MIN picks action *a* in ACTIONS(*s*) that produces smallest value of MAX-VALUE(RESULT(*s*, *a*))

Minimax

```
function Max-Value(state):
    if Terminal(state):
        return Utility(state)
    $v = -\infty$
    for action in Actions(state):
        $v$ = Max($v$, Min-Value(Result(state, action)))
    return $v$
```

Minimax

```
function MIN-VALUE(state):
    if TERMINAL(state):
        return UTILITY(state)
    $v = \infty$
    for action in ACTIONS(state):
        $v$ = MIN($v$, MAX-VALUE(RESULT(state, action)))
    return $v$
```

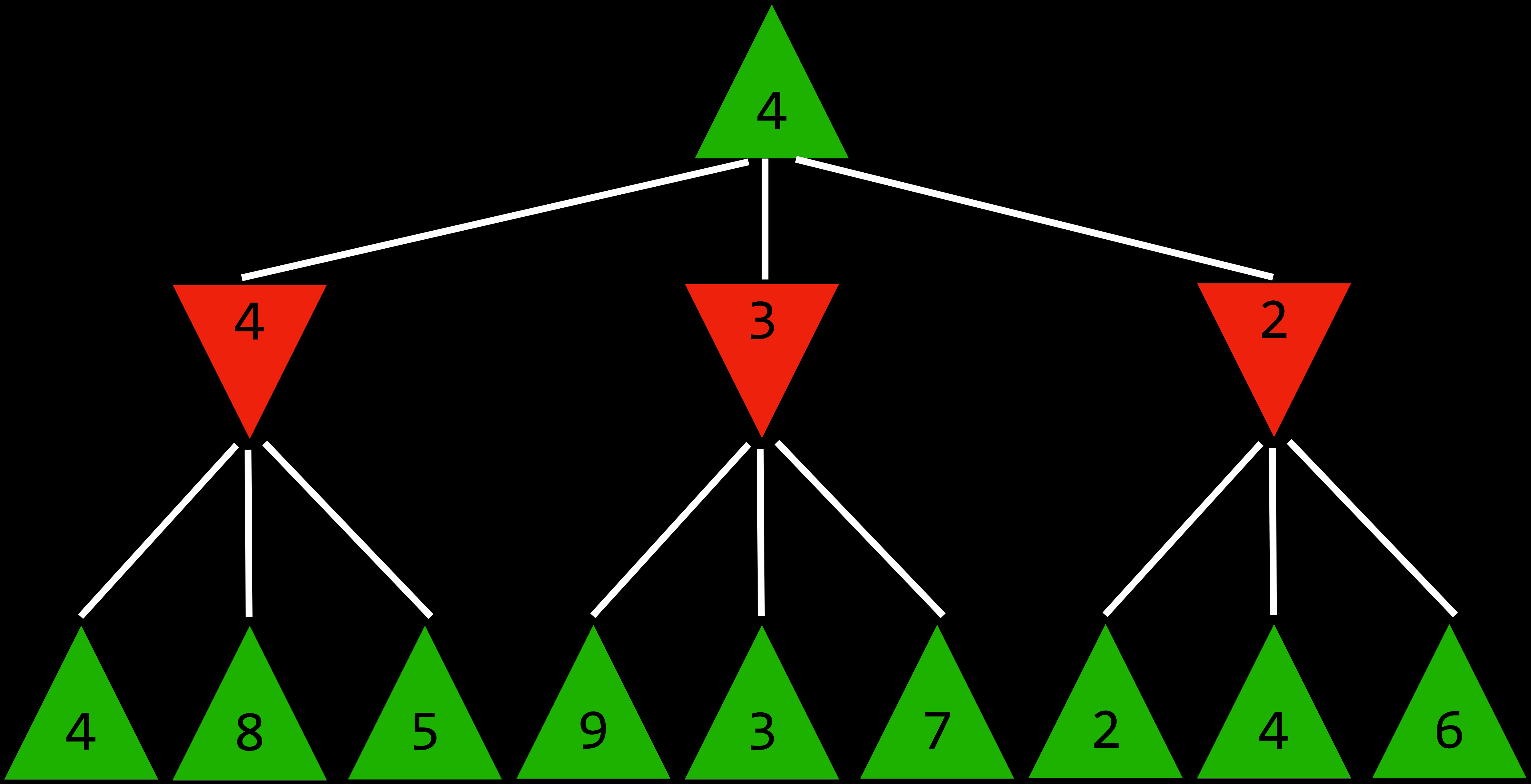
Lab-MinMax Algo

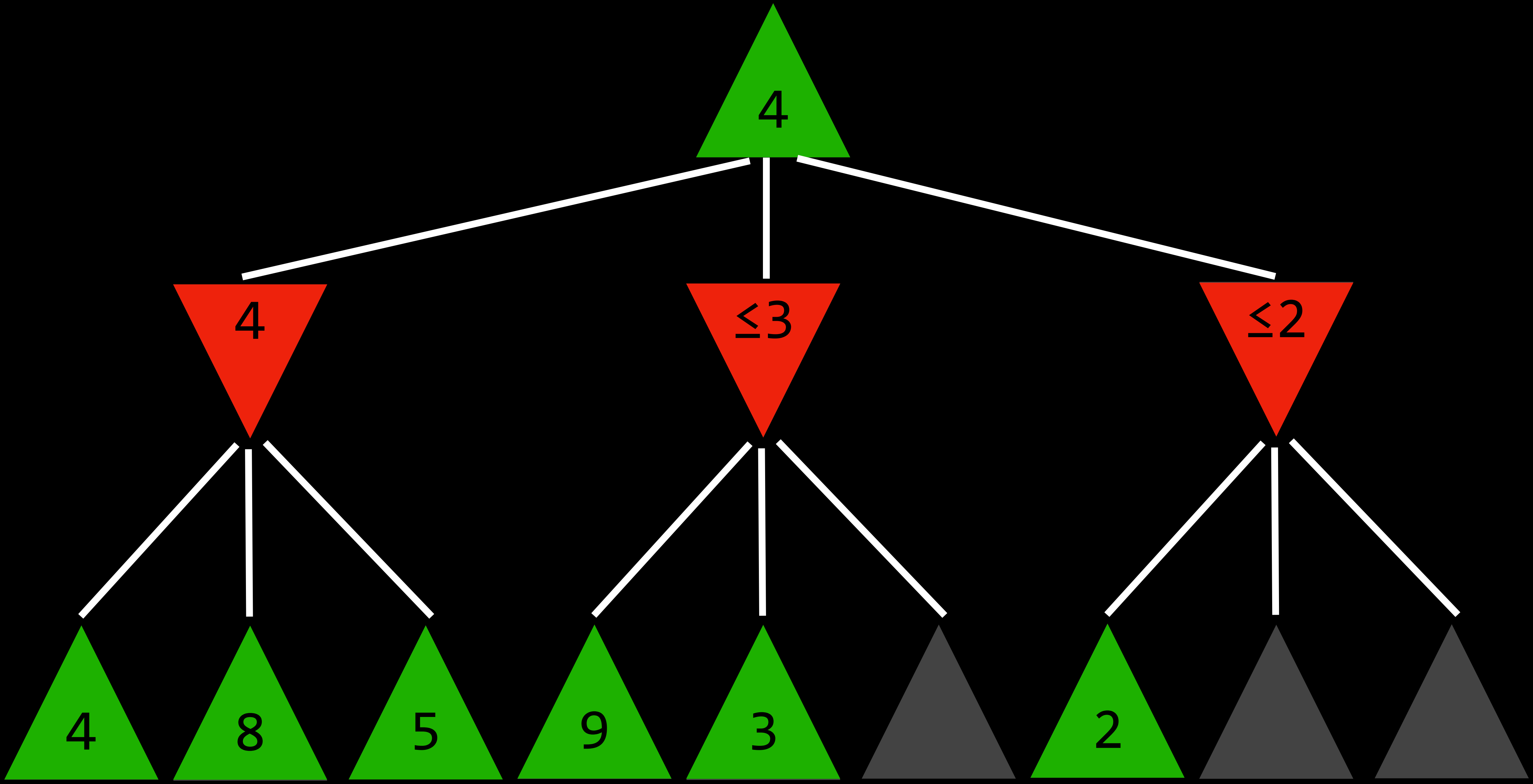
https://github.com/aruntakhur/SitareUniversity/blob/main/MinMax_Search_Lab.ipynb

Solution:

https://github.com/aruntakhur/SitareUniversity/blob/main/MinMaxSearch_AI_2025.ipynb

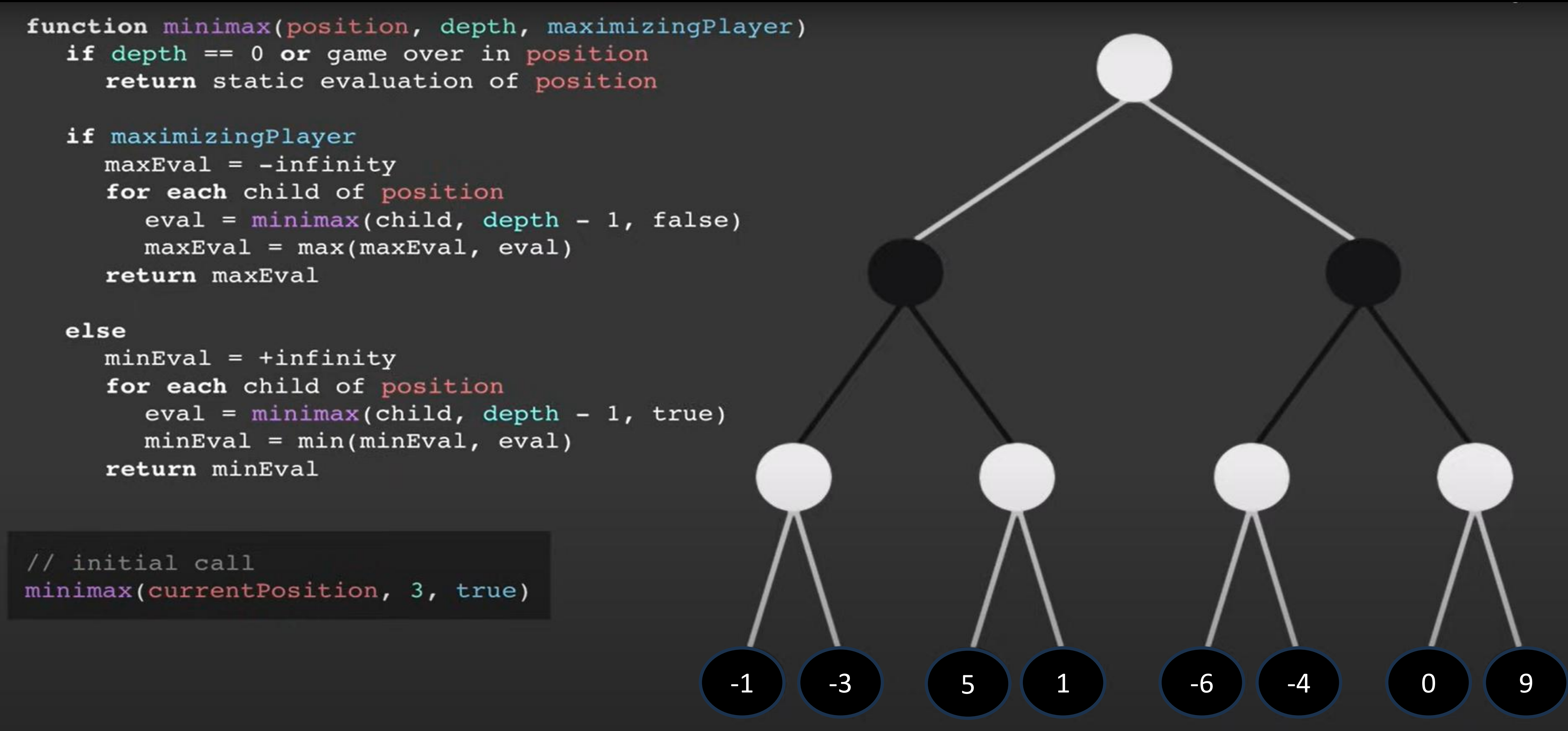
Optimizations


4
4
3
2
4
8
5
9
3
7
2
4
6


4
4
≤3
≤2
4
8
5
9
3
2

Alpha-Beta Pruning

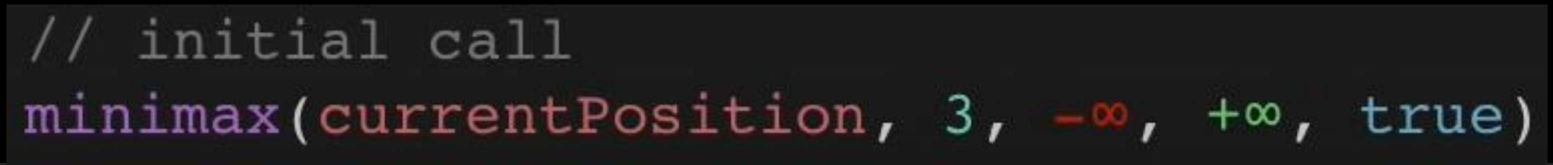
Understanding Alpha-Beta Pruning



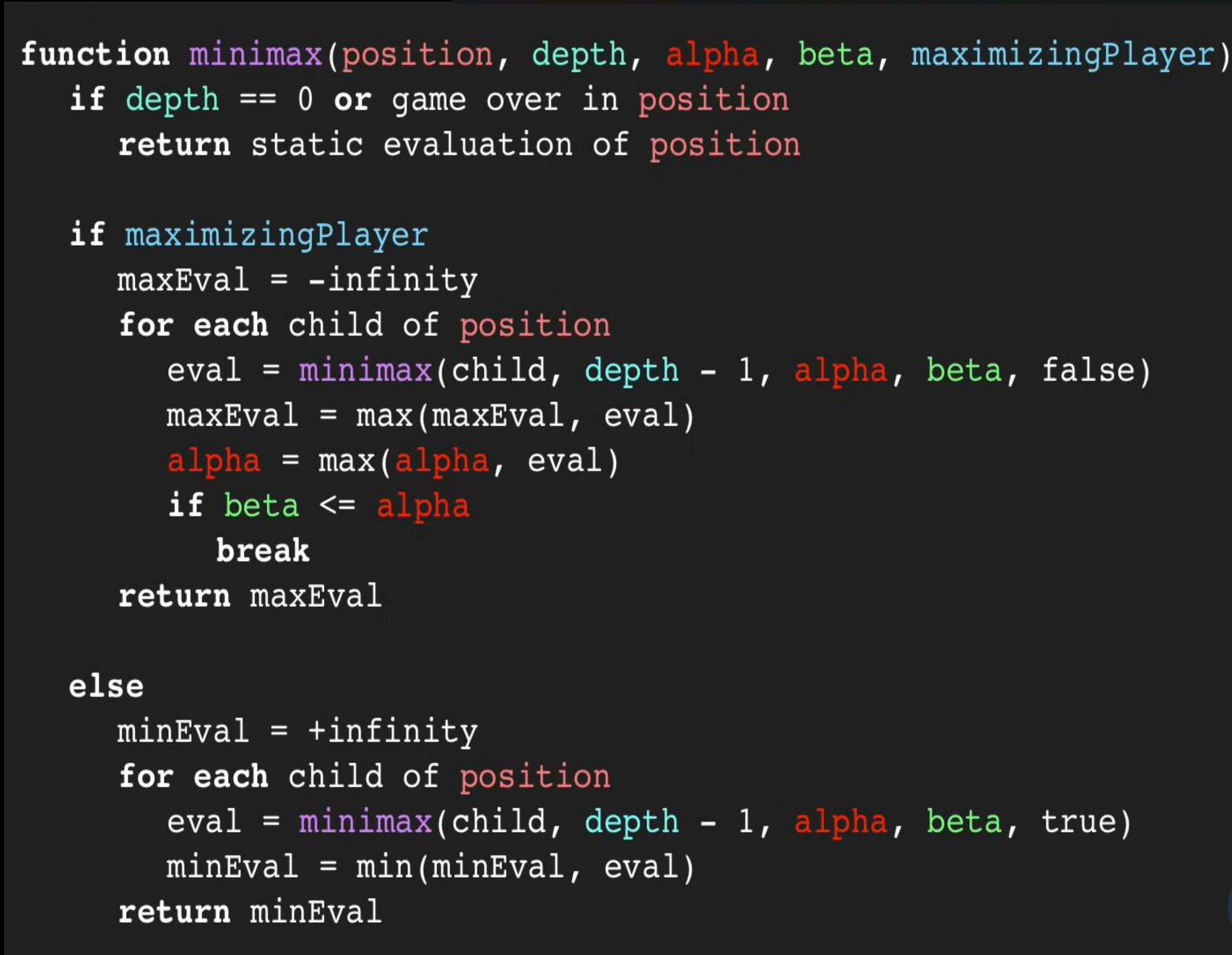
```
function minimax(position, depth, maximizingPlayer)
   if depth == 0 or game over in position
      return static evaluation of position

   if maximizingPlayer
      maxEval = -infinity
      for each child of position
         eval = minimax(child, depth - 1, false)
         maxEval = max(maxEval, eval)
      return maxEval

   else
      minEval = +infinity
      for each child of position
         eval = minimax(child, depth - 1, true)
         minEval = min(minEval, eval)
      return minEval
```

```
// initial call
minimax(currentPosition, 3, true)
```

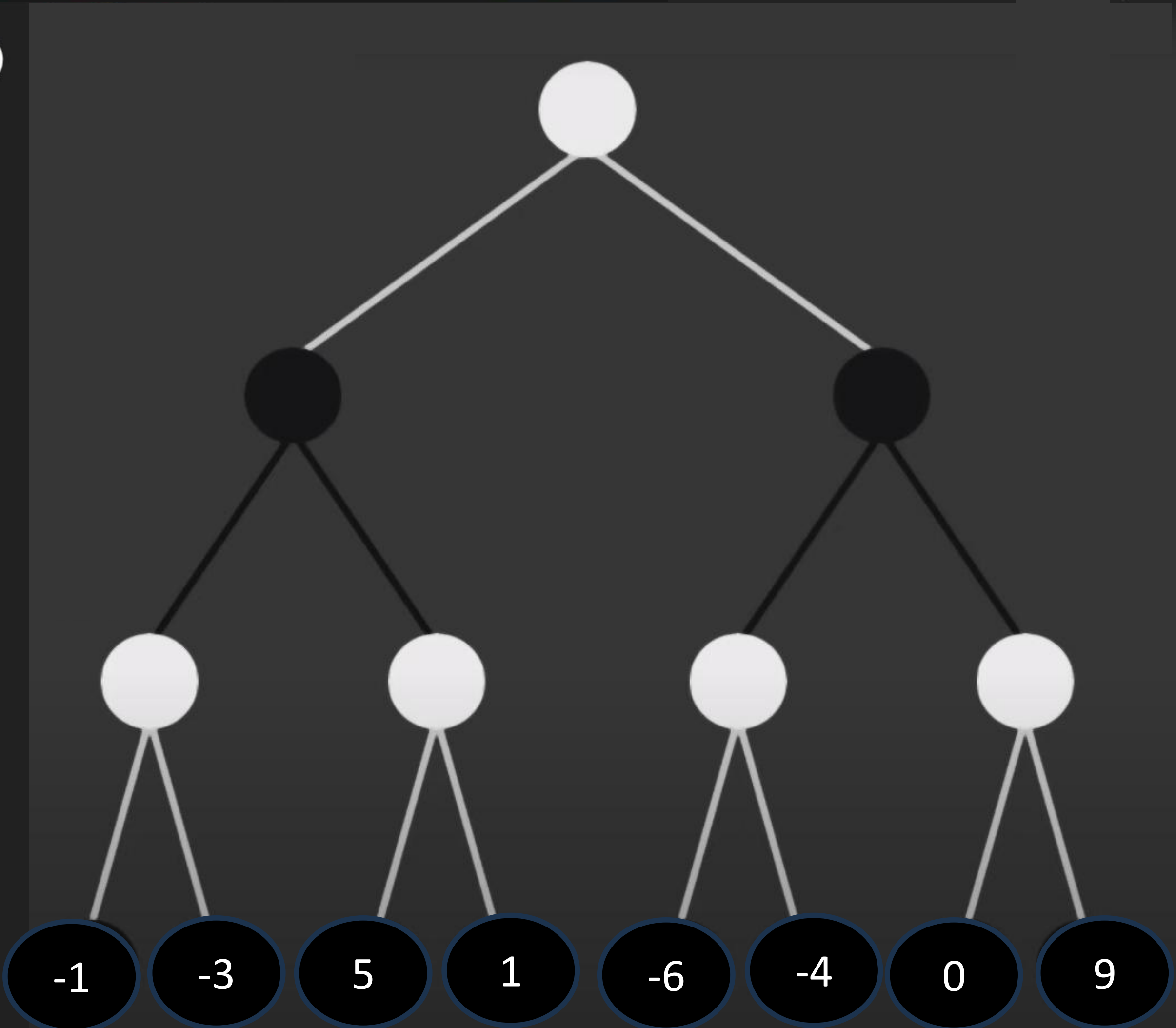
Understanding Alpha-Beta Pruning



```
// initial call
minimax(currentPosition, 3, -∞, +∞, true)
```



```
function minimax(position, depth, alpha, beta, maximizingPlayer)
  if depth == 0 or game over in position
    return static evaluation of position

  if maximizingPlayer
    maxEval = -infinity
    for each child of position
      eval = minimax(child, depth - 1, alpha, beta, false)
      maxEval = max(maxEval, eval)
      alpha = max(alpha, eval)
      if beta <= alpha
        break
    return maxEval

  else
    minEval = +infinity
    for each child of position
      eval = minimax(child, depth - 1, alpha, beta, true)
      minEval = min(minEval, eval)
    return minEval
```



Lab-MinMax Algo with Alpha-Beta Pruning

https://github.com/aruntakhur/SitareUniversity/blob/main/MinMax_AlphaPruning_AI_2025.ipynb

255,168

total possible Tic-Tac-Toe games

288,000,000,000

total possible chess games after four moves each

$$10^{29000}$$

total possible chess games
(lower bound)

Depth-Limited Minimax

evaluation function

function that estimates the expected utility of the game from a given state

Example-MinMax Algo with Alpha-Beta Pruning and Heurostic

https://github.com/aruntakhur/SitareUniversity/blob/main/MinMax_AlphaBeta_heuristic_AI_2025.ipynb

References

Book: Artificial Intelligence A Modern Approach (3rd Edition)

Harvard CS50's Artificial Intelligence with Python