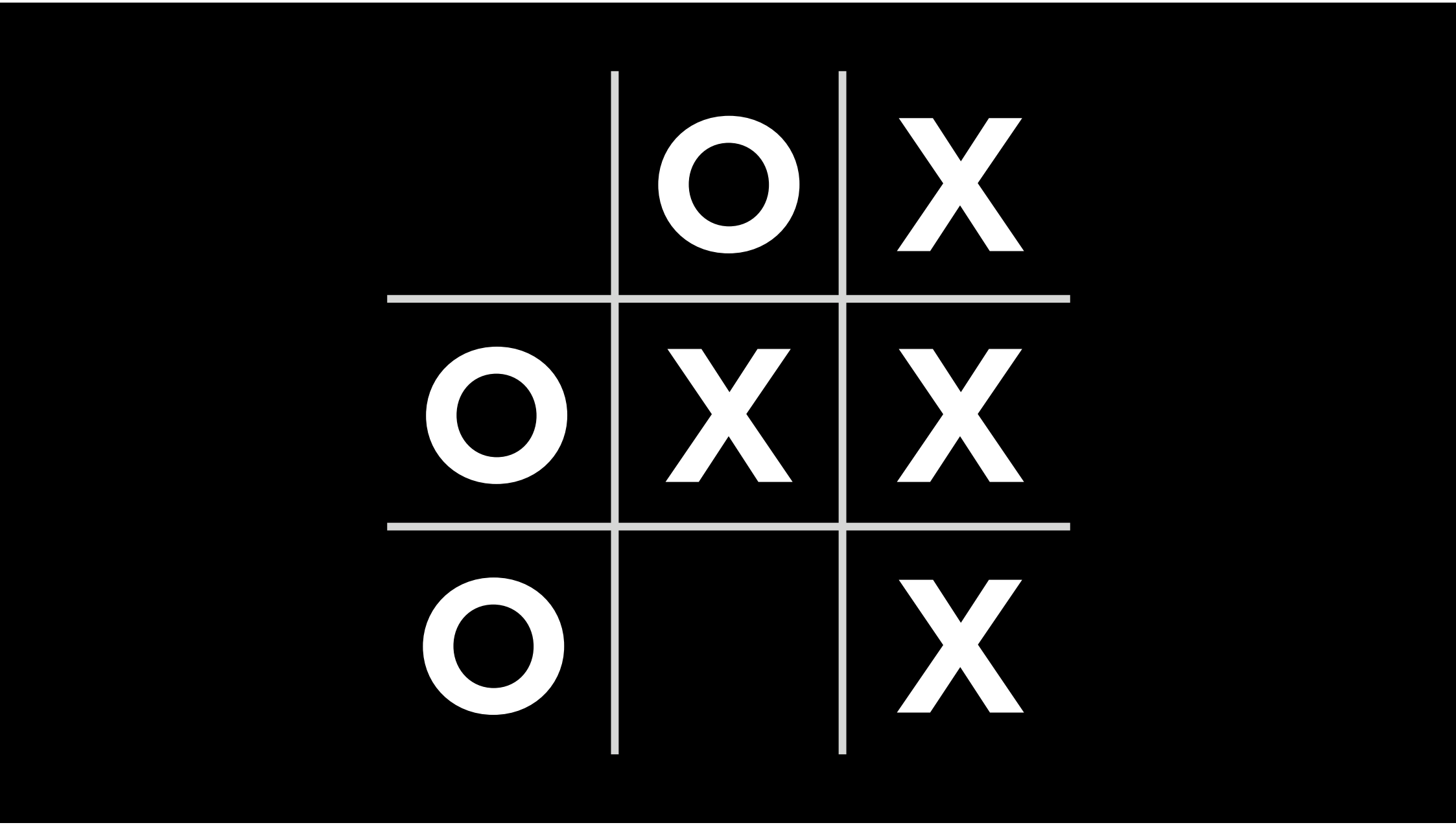
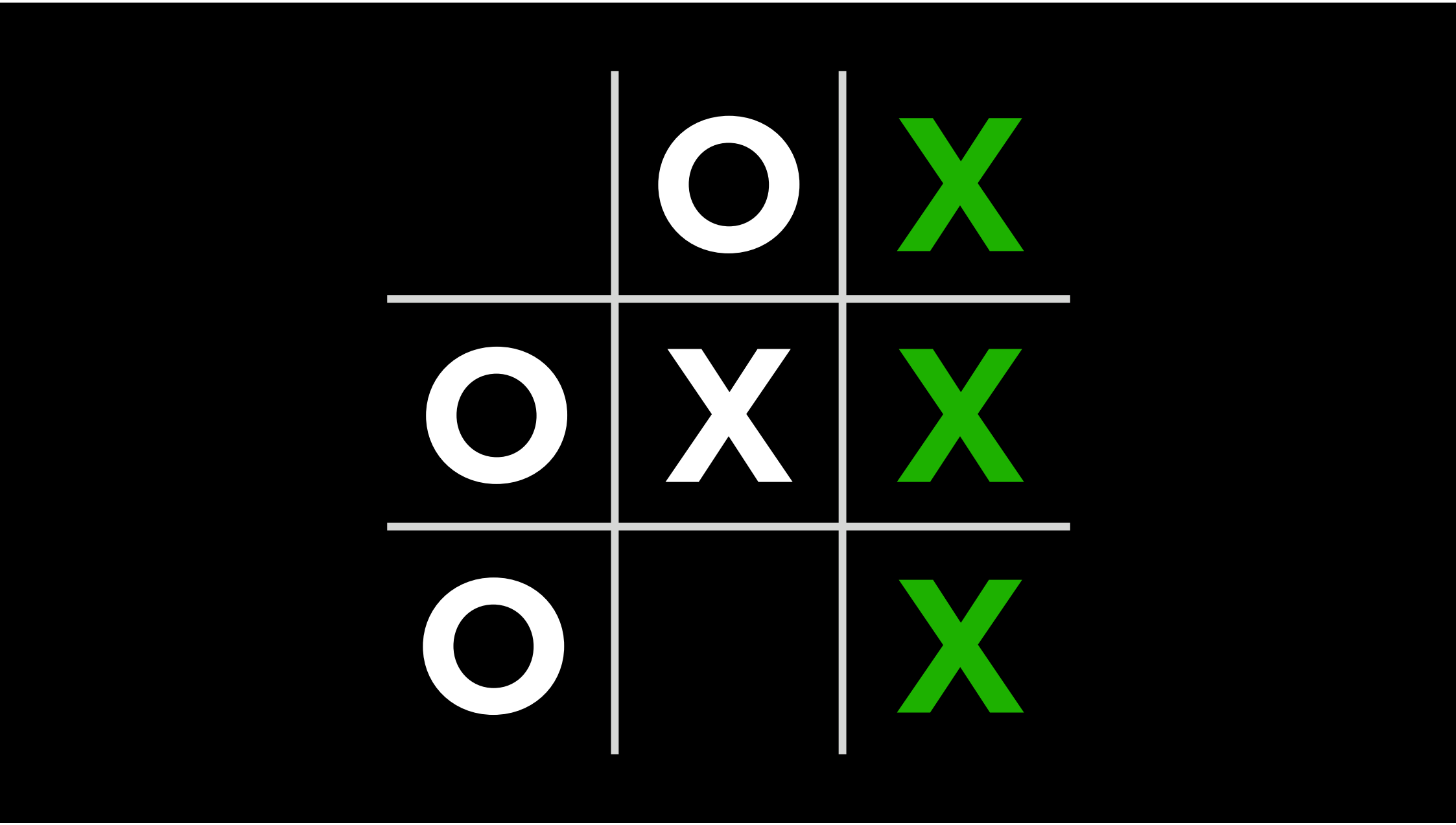
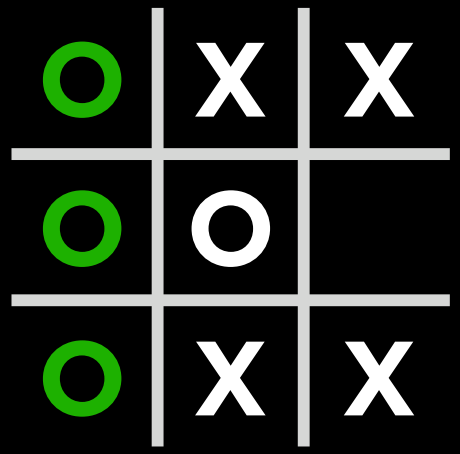
Adversarial Search

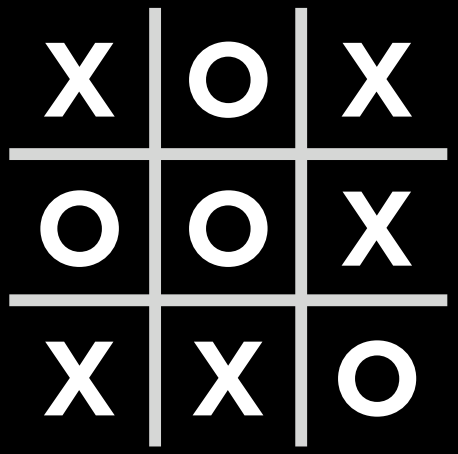

O X
O X X
O X


O X
O X X
O X

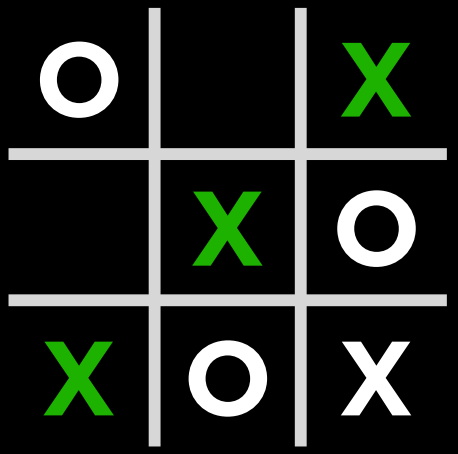
Minimax



-1



0



1

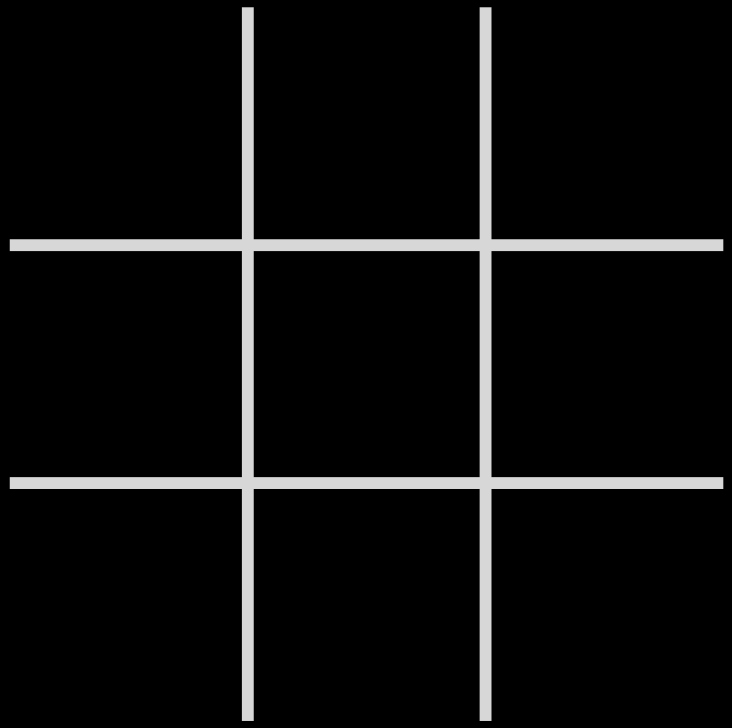
Minimax

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

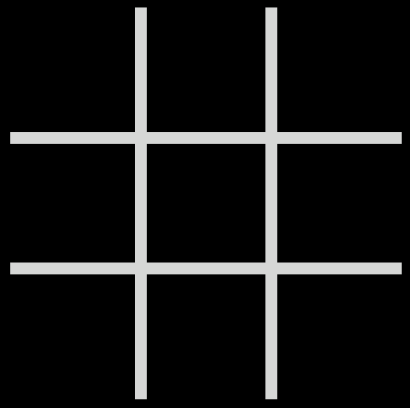
Game

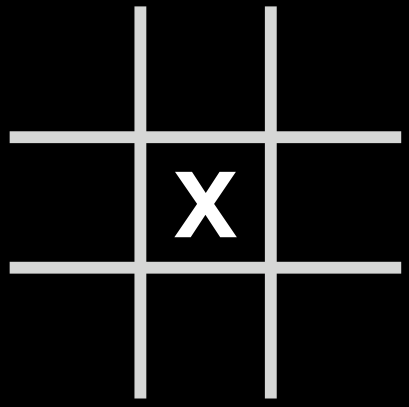
- S_0 : initial state
- PLAYER(s) : returns which player to move in state s
- ACTIONS(s) : returns legal moves in state s
- RESULT(s, a) : returns state after action a taken in state s
- TERMINAL(s) : checks if state s is a terminal state
- UTILITY(s) : final numerical value for terminal state s

Initial State



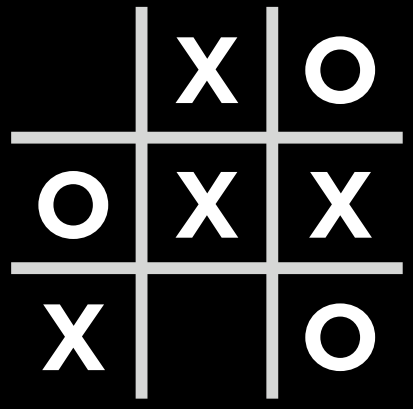
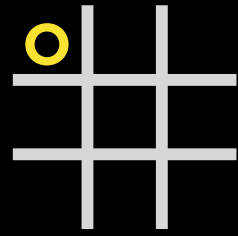
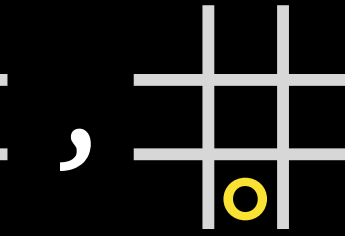
PLAYER(*s*)

PLAYER() = X

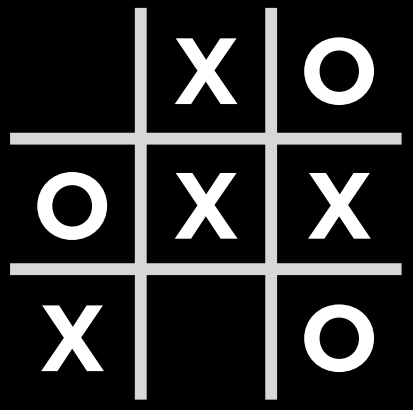
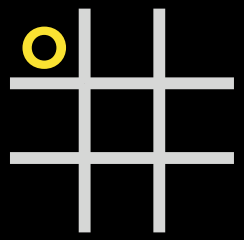
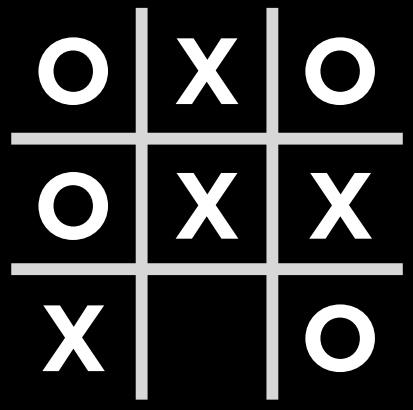
PLAYER(

) = O

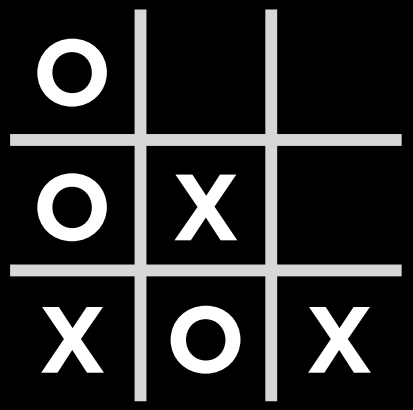
ACTIONS(s)

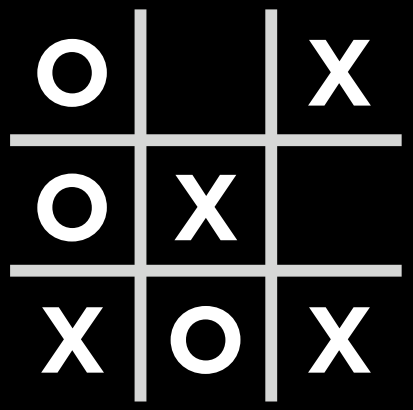
ACTIONS() = {  ,  }

RESULT(s, a)

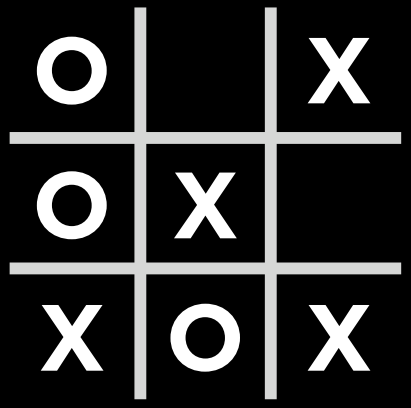
RESULT( , ) = 

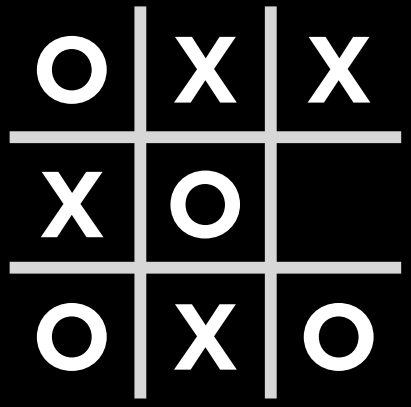
TERMINAL(s)

TERMINAL() = false

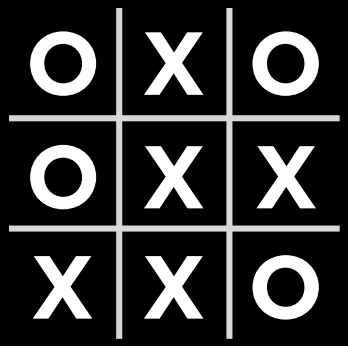
TERMINAL() = true

UTILITY(s)

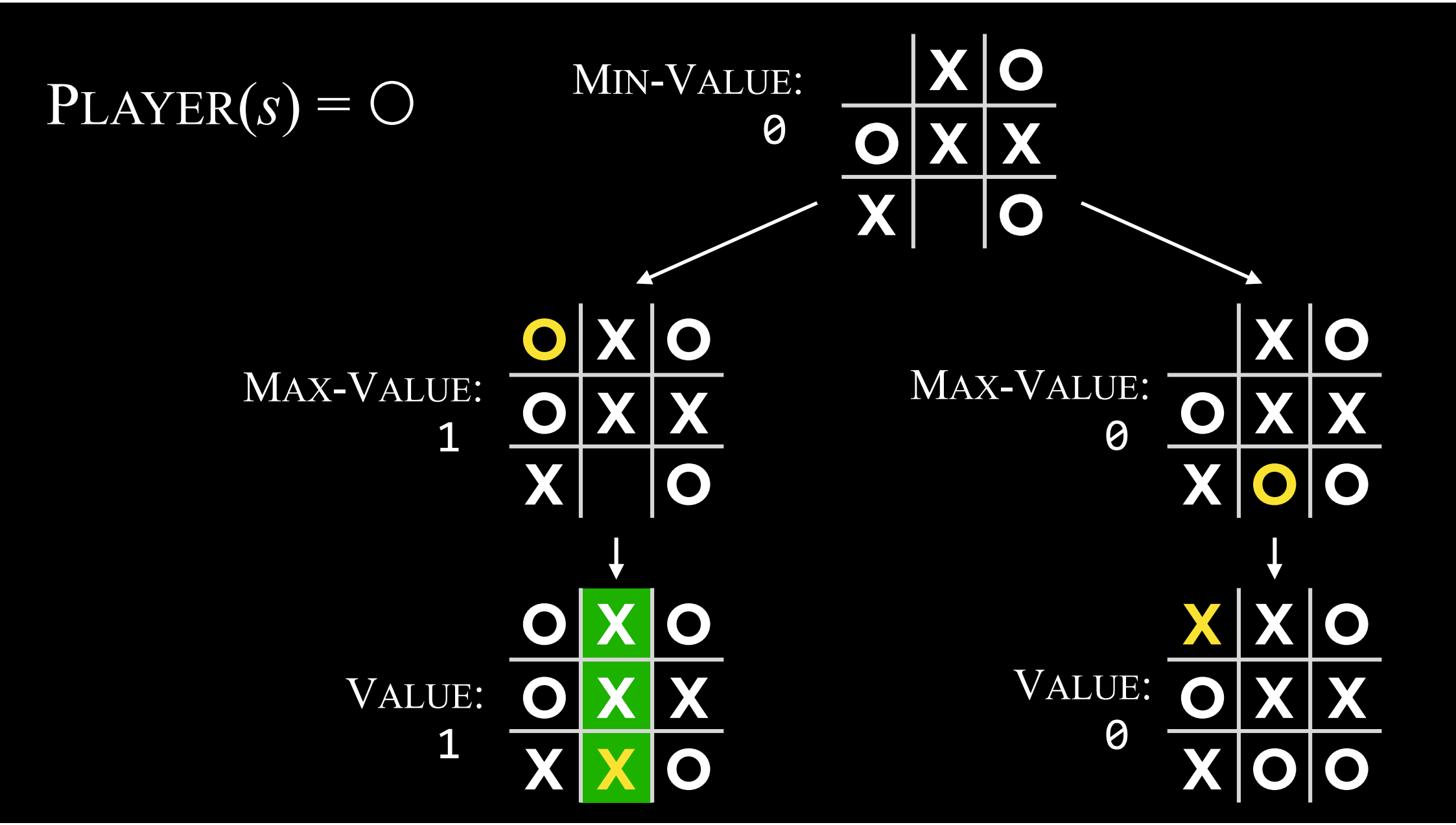
UTILITY() = 1

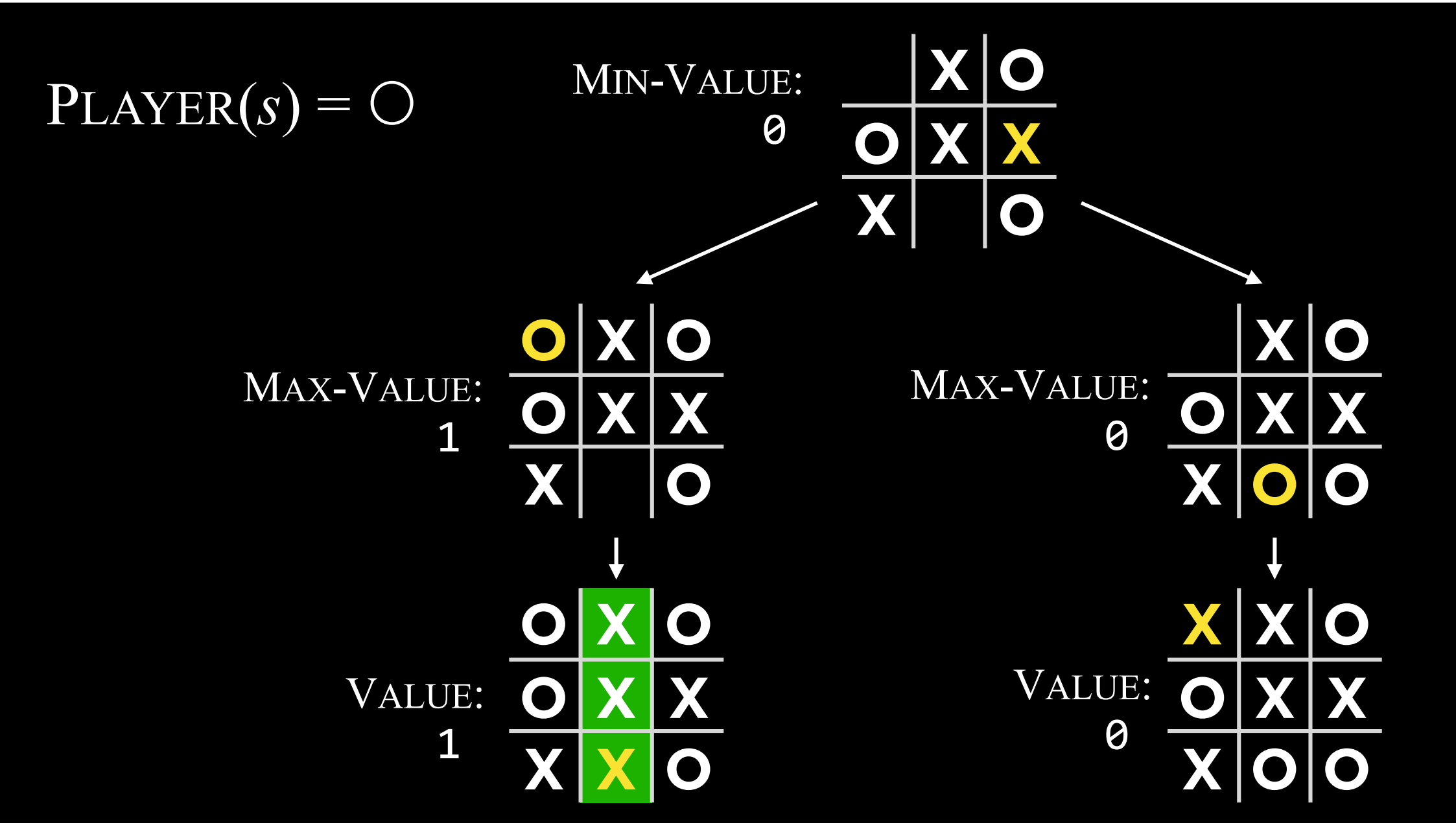


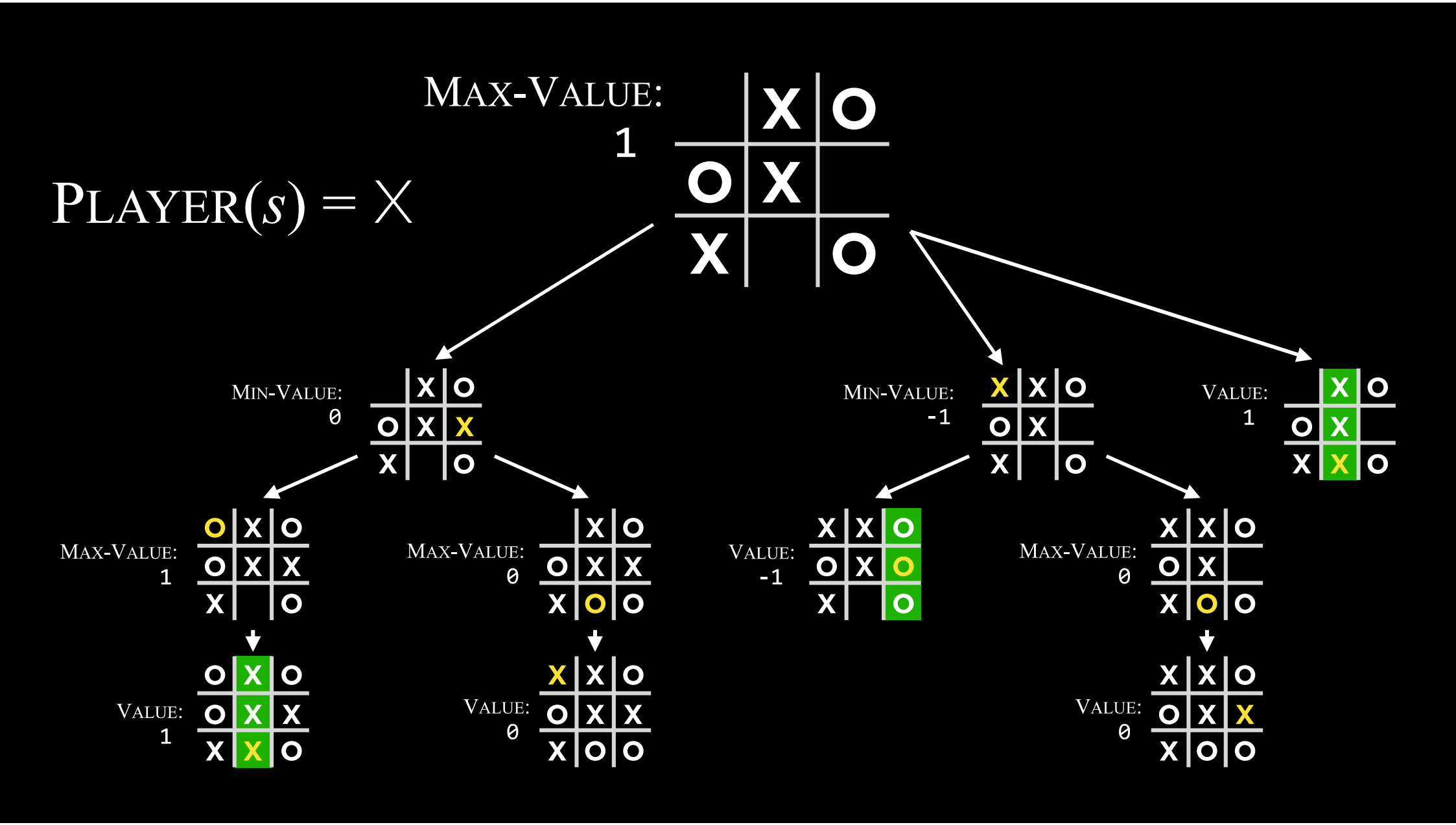
UTILITY( ) = -1

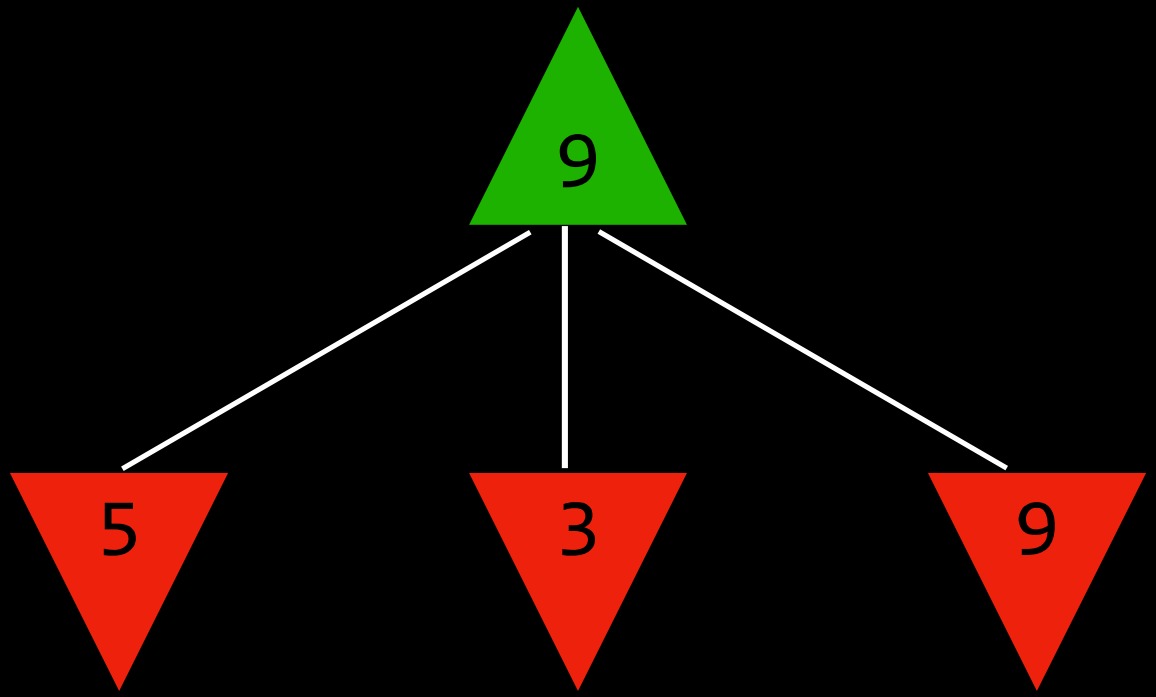


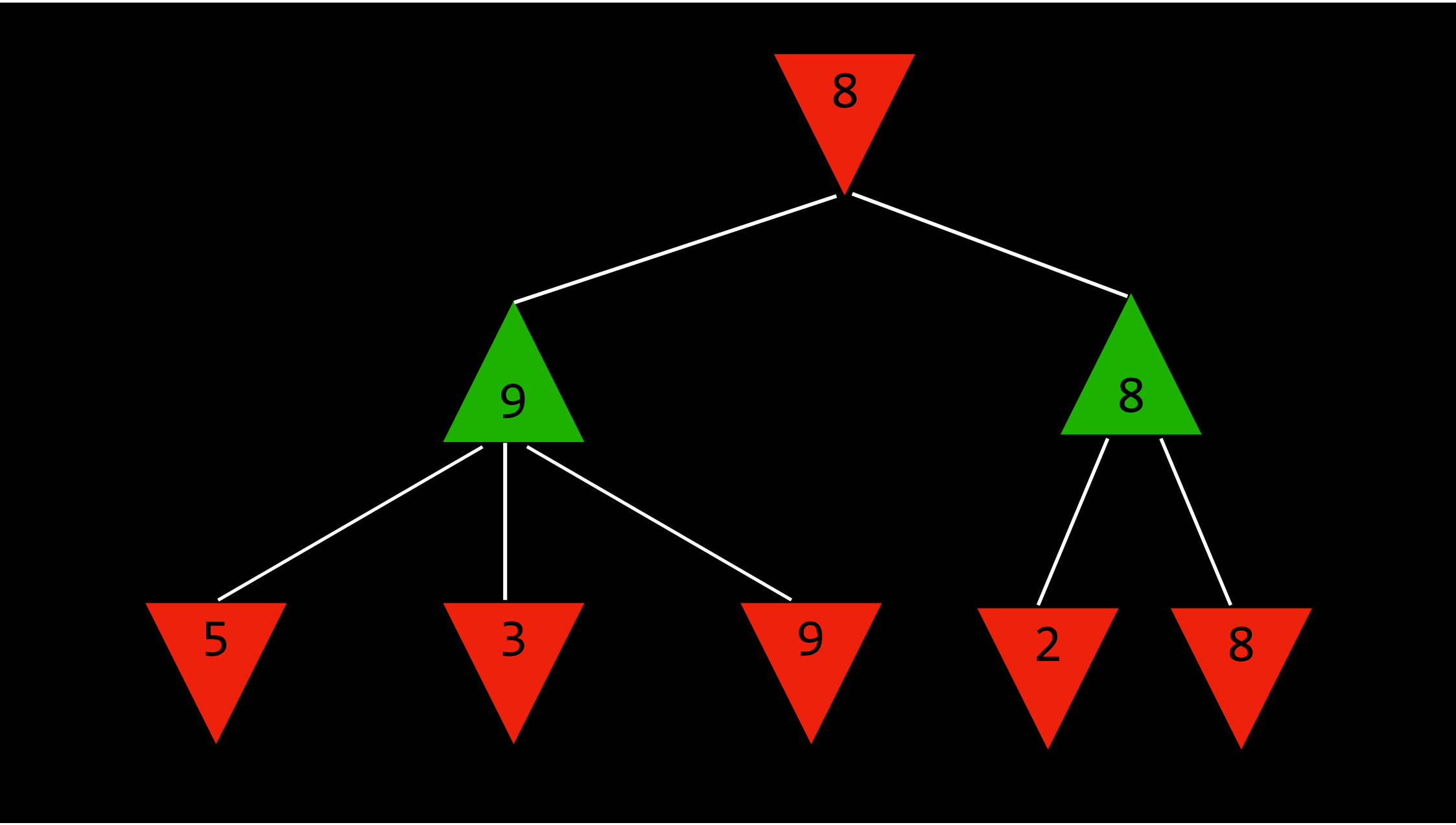
VALUE: 1


PLAYER(s) = O
MIN-VALUE:
0
MAX-VALUE:
1
MAX-VALUE:
0
VALUE:
1
VALUE:
0


PLAYER(s) = O
MIN-VALUE:
0
MAX-VALUE:
1
MAX-VALUE:
0
VALUE:
1
VALUE:
0


PLAYER(s) = 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))

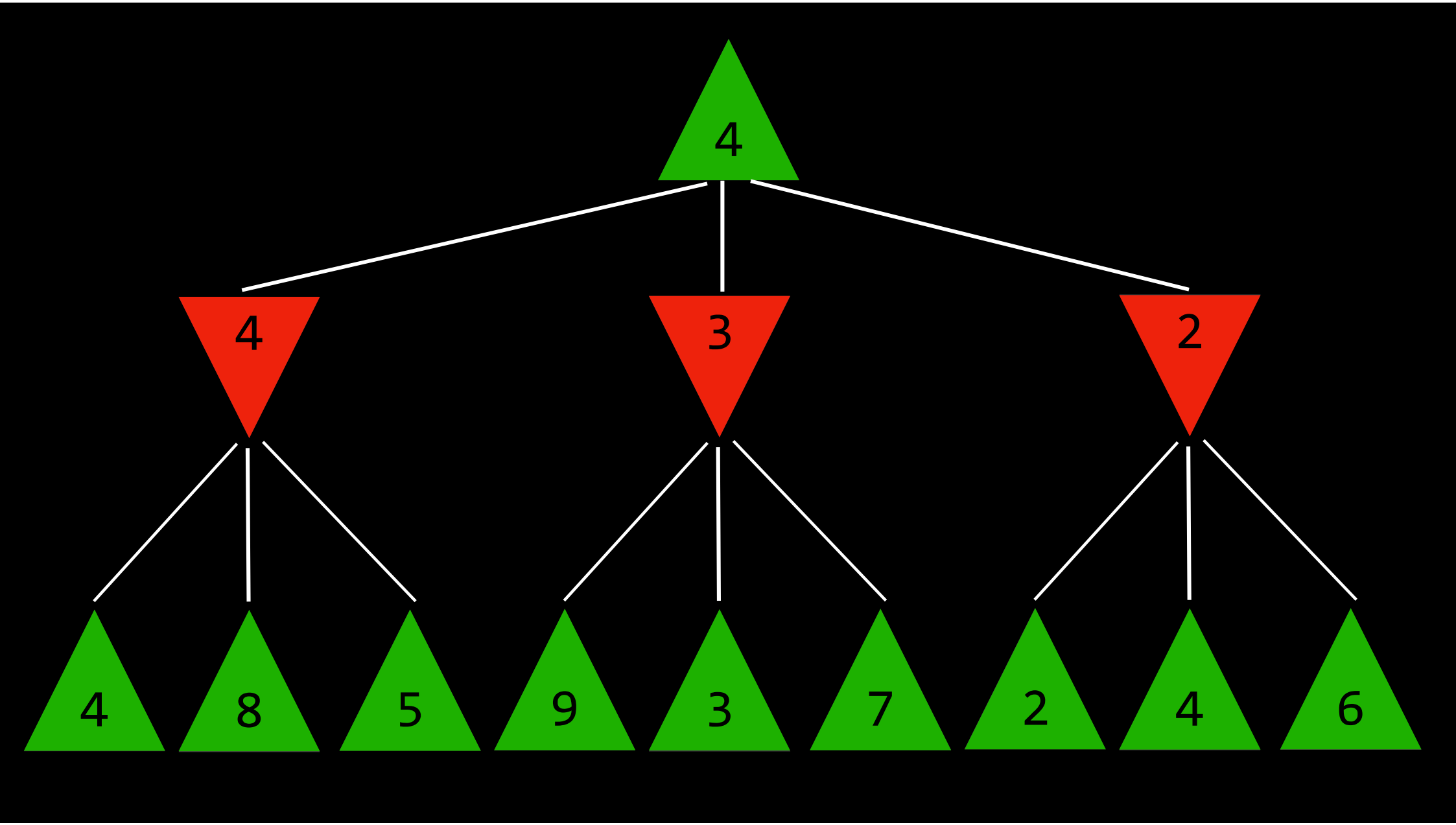
Minimax

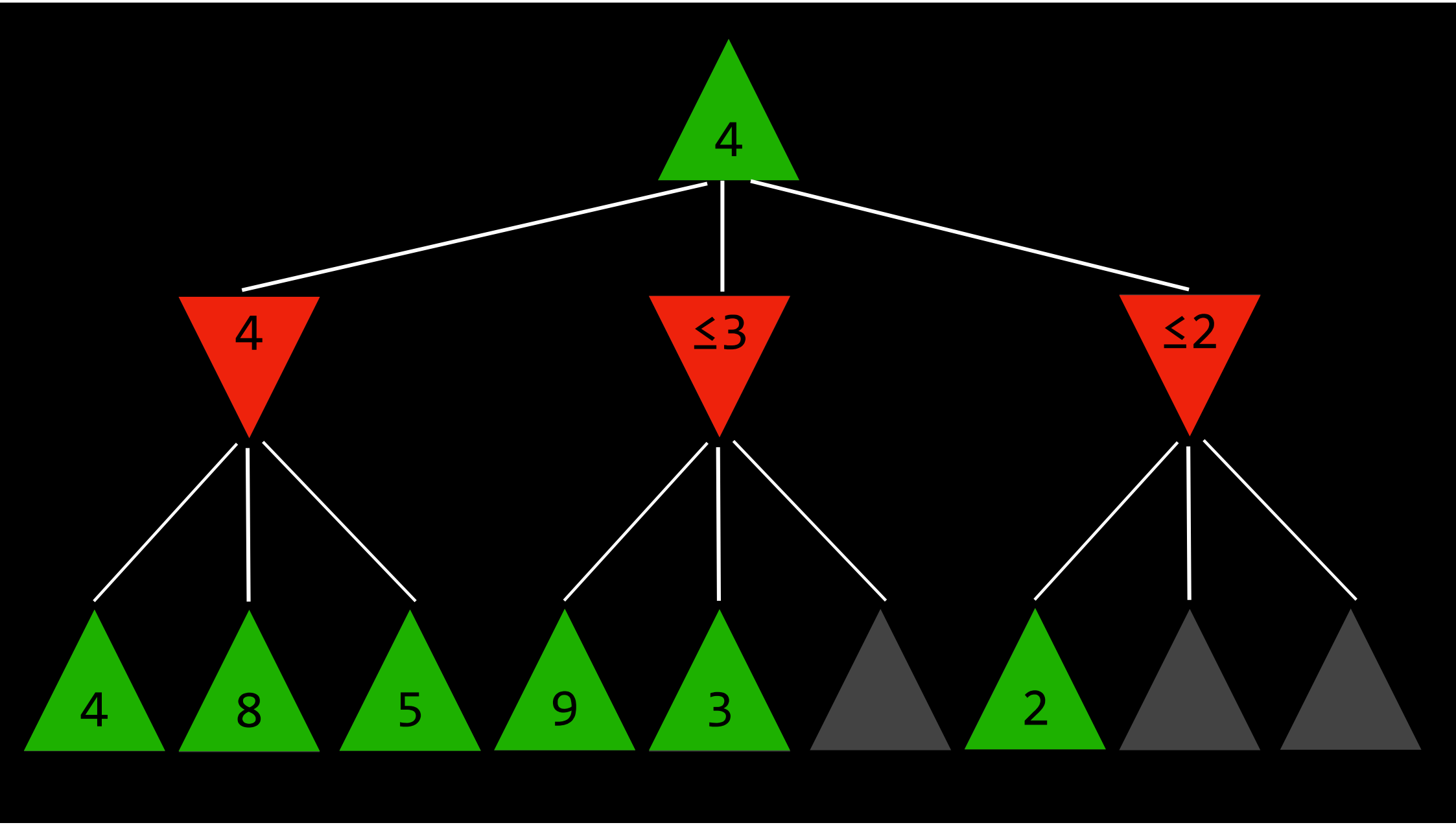
```
function MAX-VALUE(state):
    if TERMINAL(state):
        return UTILITY(state)
    v = -∞
    for action in ACTIONS(state):
        v = MAX(v, MIN-VALUE(RESULT(state, action)))
    return v
```

Minimax

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

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

255,168

total possible Tic-Tac-Toe games

288,000,000,000

total possible chess games
after four moves each

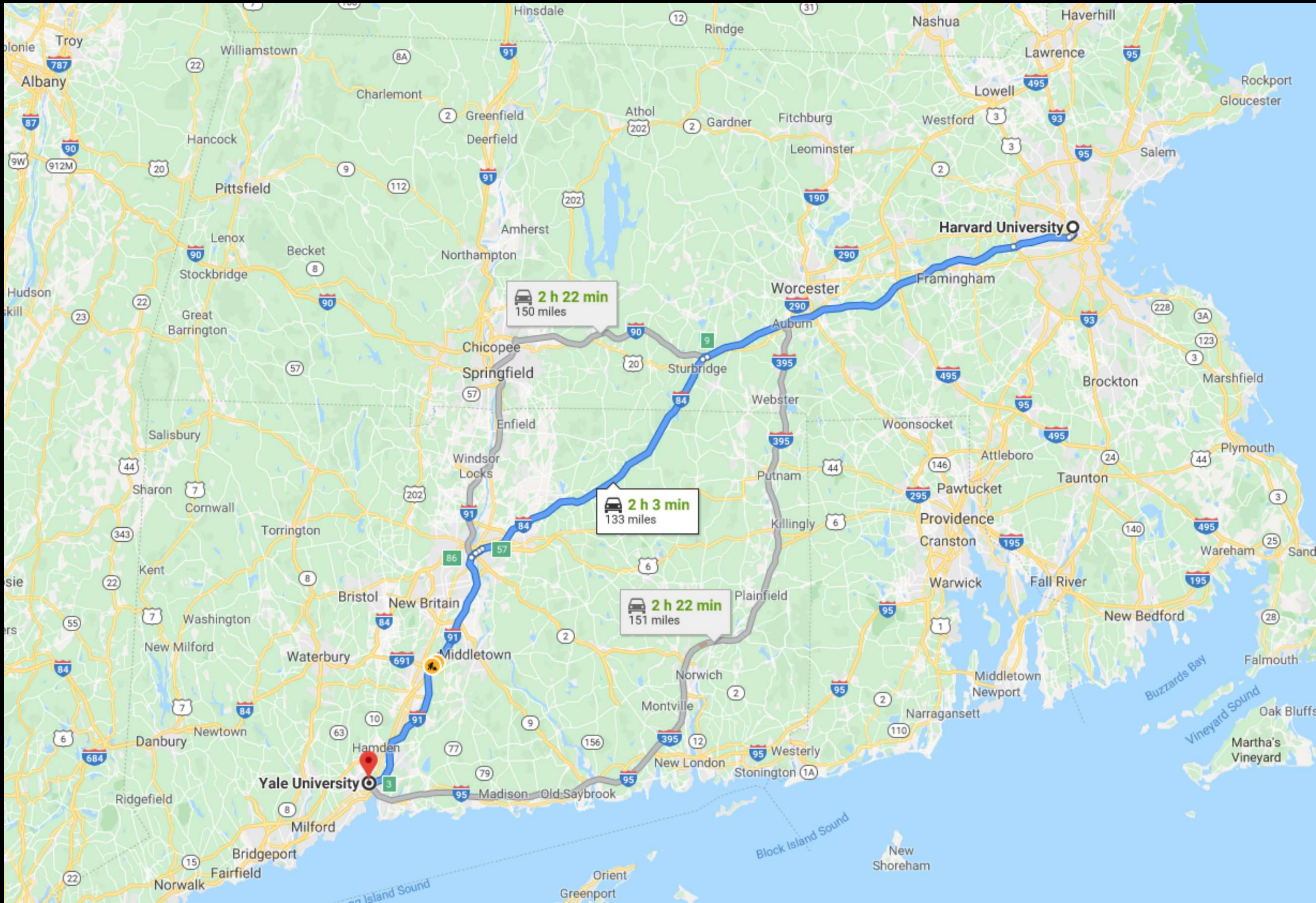
10^{29000}

total possible chess games
(lower bound)

Depth-Limited Minimax

evaluation function

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

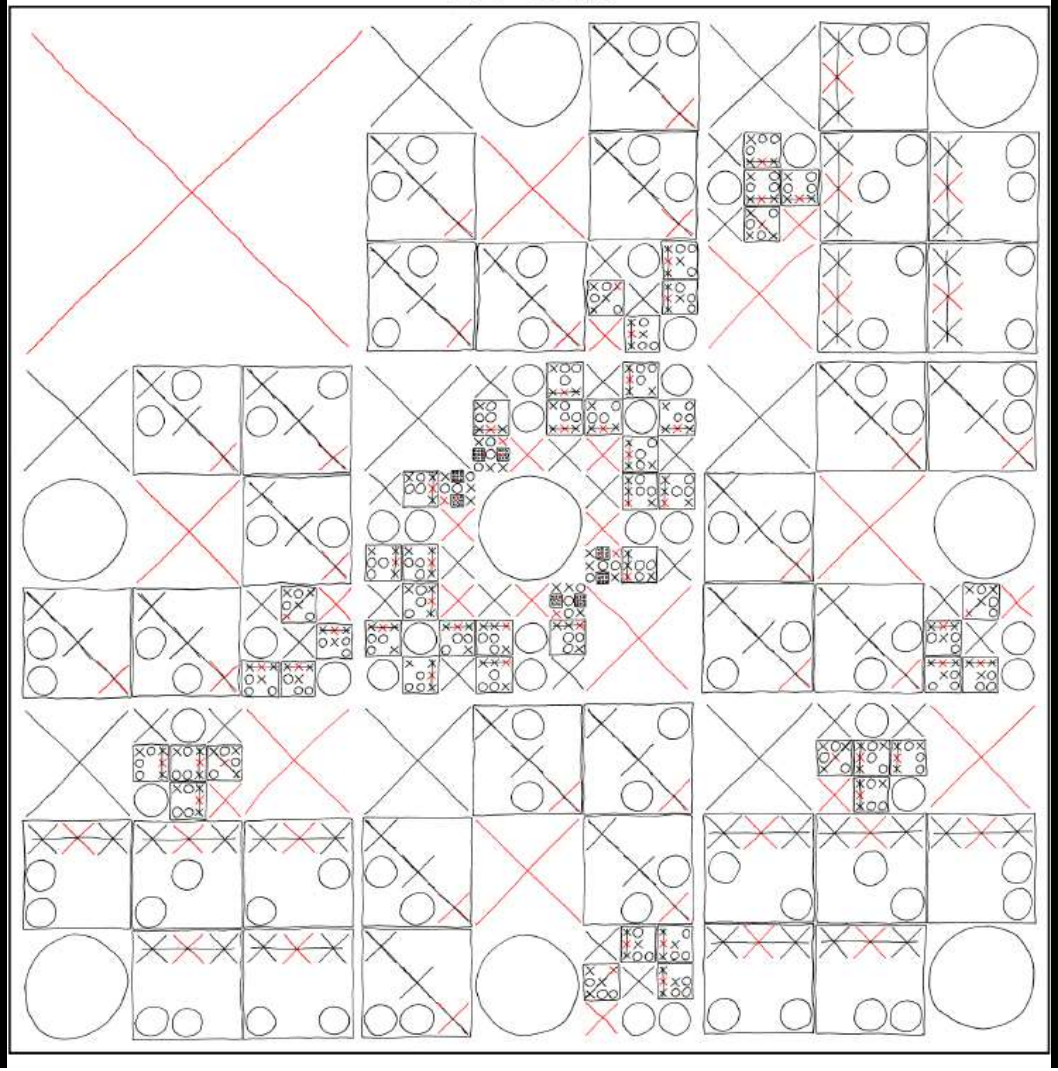


Harvard University
Yale University
2 h 22 min
150 miles
2 h 3 min
133 miles
2 h 22 min
151 miles
Troy
Albany
Williamstown
Hinsdale
Rindge
Nashua
Haverhill
Lawrence
Lowell
Rockport
Gloucester
Charlemont
Greenfield
Athol
Gardner
Fitchburg
Westford
Deerfield
Leominster
Salem
Hancock
Pittsfield
Amherst
Lenox
Becket
Northampton
Stockbridge
Framingham
Worcester
Hudson
Great Barrington
Auburn
Chicopee
Springfield
Sturbridge
Brockton
Marshfield
Webster
Enfield
Woonsocket
Salisbury
Windsor Locks
Attleboro
Plymouth
Sharon
Cornwall
Putnam
Pawtucket
Taunton
Torrington
Killingly
Providence
Cranston
Wareham
Kent
Warwick
Fall River
Bristol
New Britain
Plainfield
Washington
New Bedford
New Milford
Waterbury
Middletown
Falmouth
Norwich
Middletown
Newport
Montville
Narragansett
Buzzards Bay
Vineyard Sound
Oak Bluffs
Danbury
Newtown
Hamden
New London
Westerly
Stonington
Martha's Vineyard
Madison
Old Saybrook
Ridgefield
Milford
Bridgeport
Fairfield
Norwalk
Block Island Sound
New Shoreham
Orient
Greenport

COMPLETE MAP OF OPTIMAL TIC-TAC-TOE MOVES

YOUR MOVE IS GIVEN BY THE POSITION OF THE LARGEST RED SYMBOL ON THE GRID. WHEN YOUR OPPONENT PICKS A MOVE, ZOOM IN ON THE REGION OF THE GRID WHERE THEY WENT. REPEAT.

MAP FOR X:



https://xkcd.com/832/

Search

Introduction to

Artificial Intelligence

with Python