Artificial Intelligence

Lecture 05: Adversarial Searching

AIMA C5

Lecture Summary

- Adversarial Search

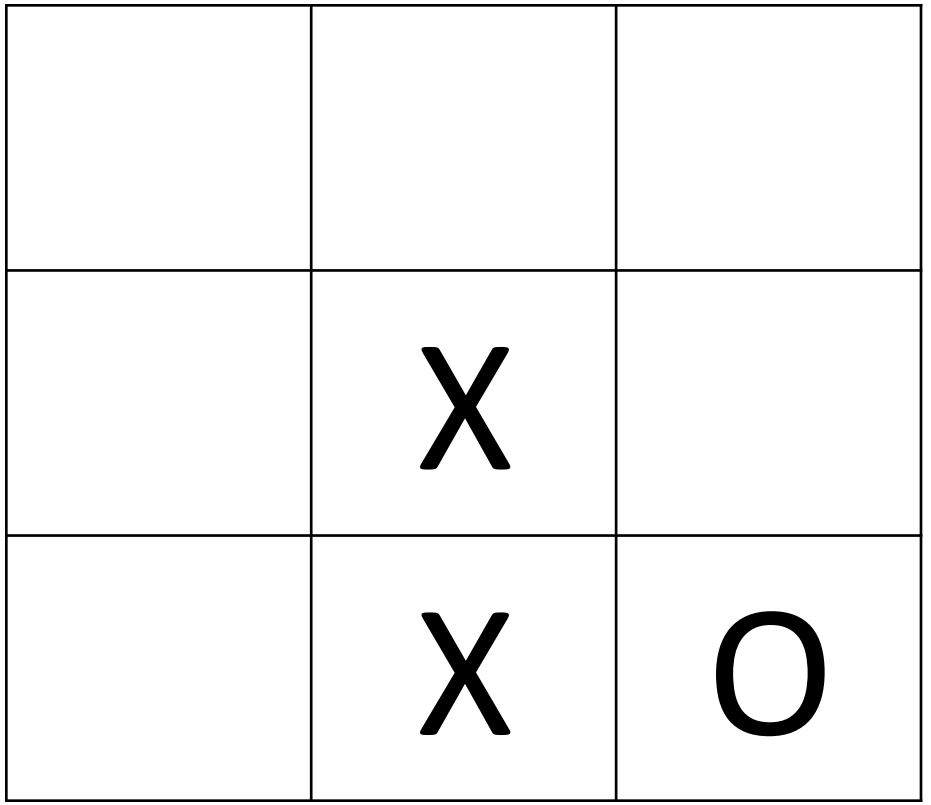
Game Examples

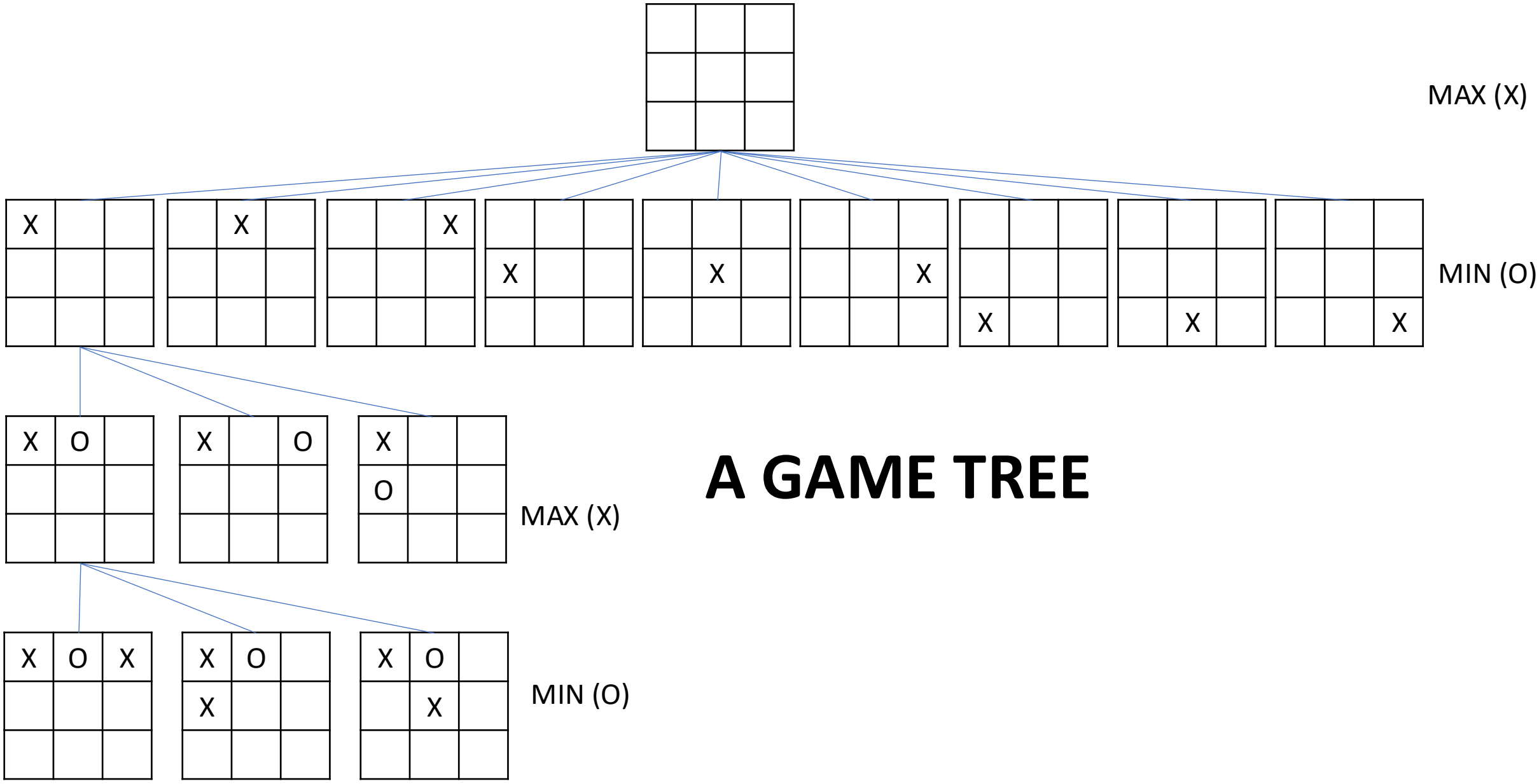
	Deterministic	**Chance**
Perfect information	chess, checkers, go, othello, tic-tac-toe	backgammon, monopoly
Imperfect information	battleships, blind tic-tac-toe	bridge, poker, scrabble, nuclear war

Tic Tac Toe

Two player, perfect information, deterministic

- Each player takes a turn placing an X or a O (one player is X, the other player is O)
- The winner is the player who can get three of their symbols in a row, horizontally, vertically or diagonally





A GAME TREE

Some nomenclature (naming things in a certain way)

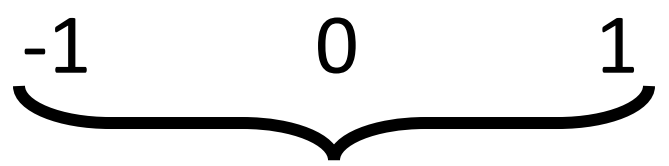
Why MAX and MIN?

- We have a polarised definition of victory (we are in this case X)
- So… O is trying to minimise our ‘wins’

X	O	X
X	O	X
	O	

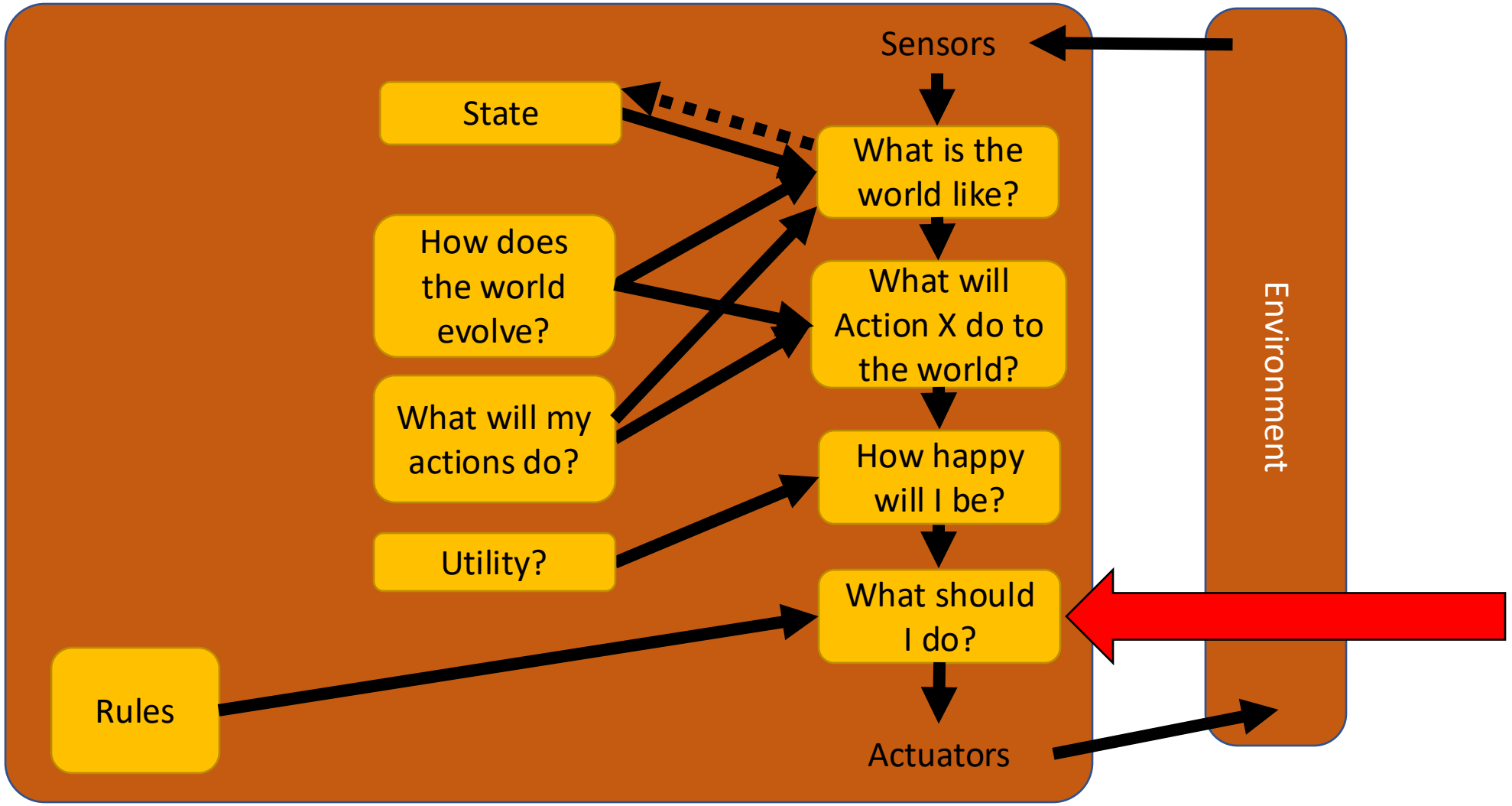
X	O	X
X	X	O
O	X	O

O	O	
	X	O
X	X	X



Utility values

Utility-based Agent



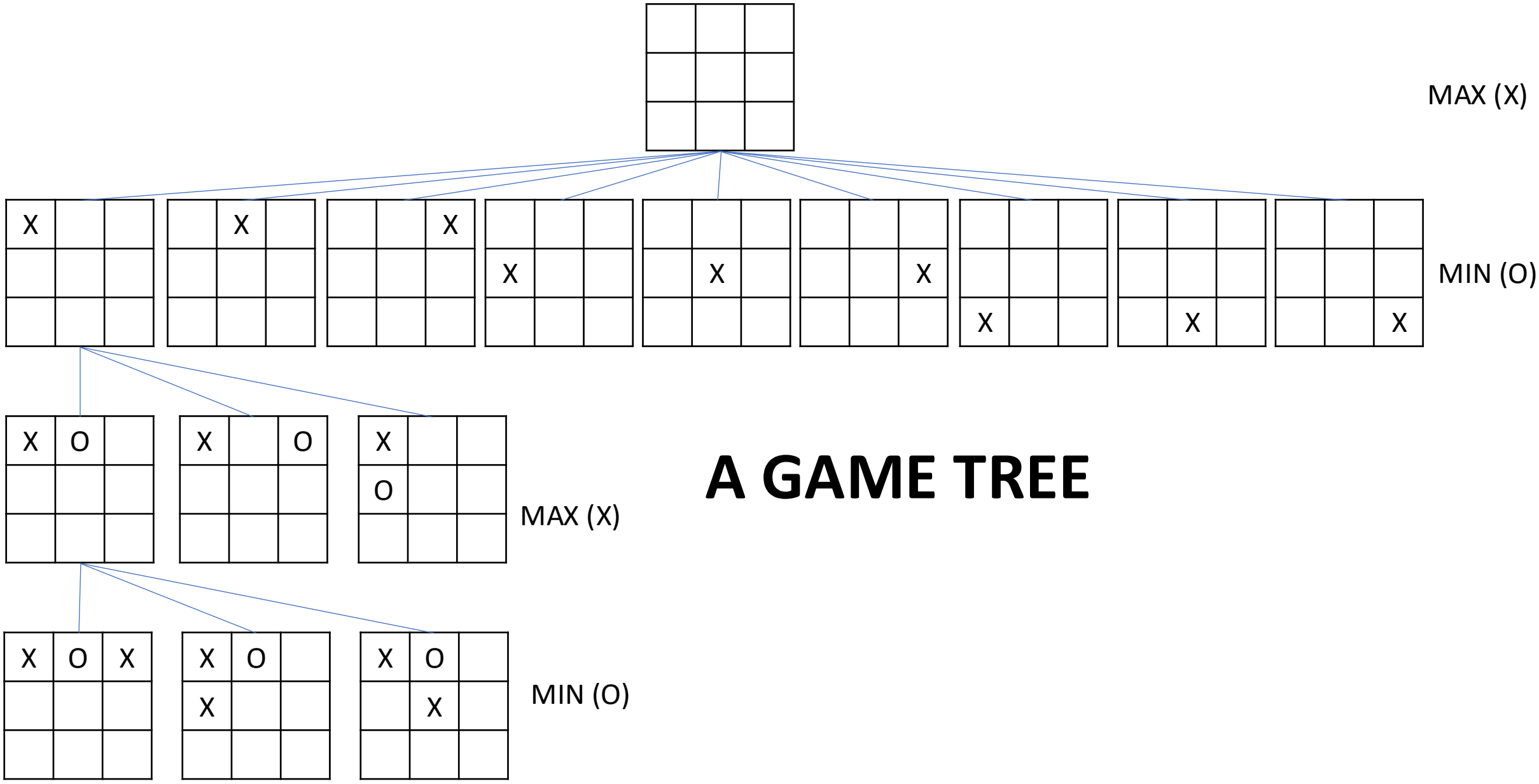
What to do?

Search?

- Pretty easy to find a path to a win?

Cost?

- I guess there are ‘quicker’ wins… but this isn’t really the goal



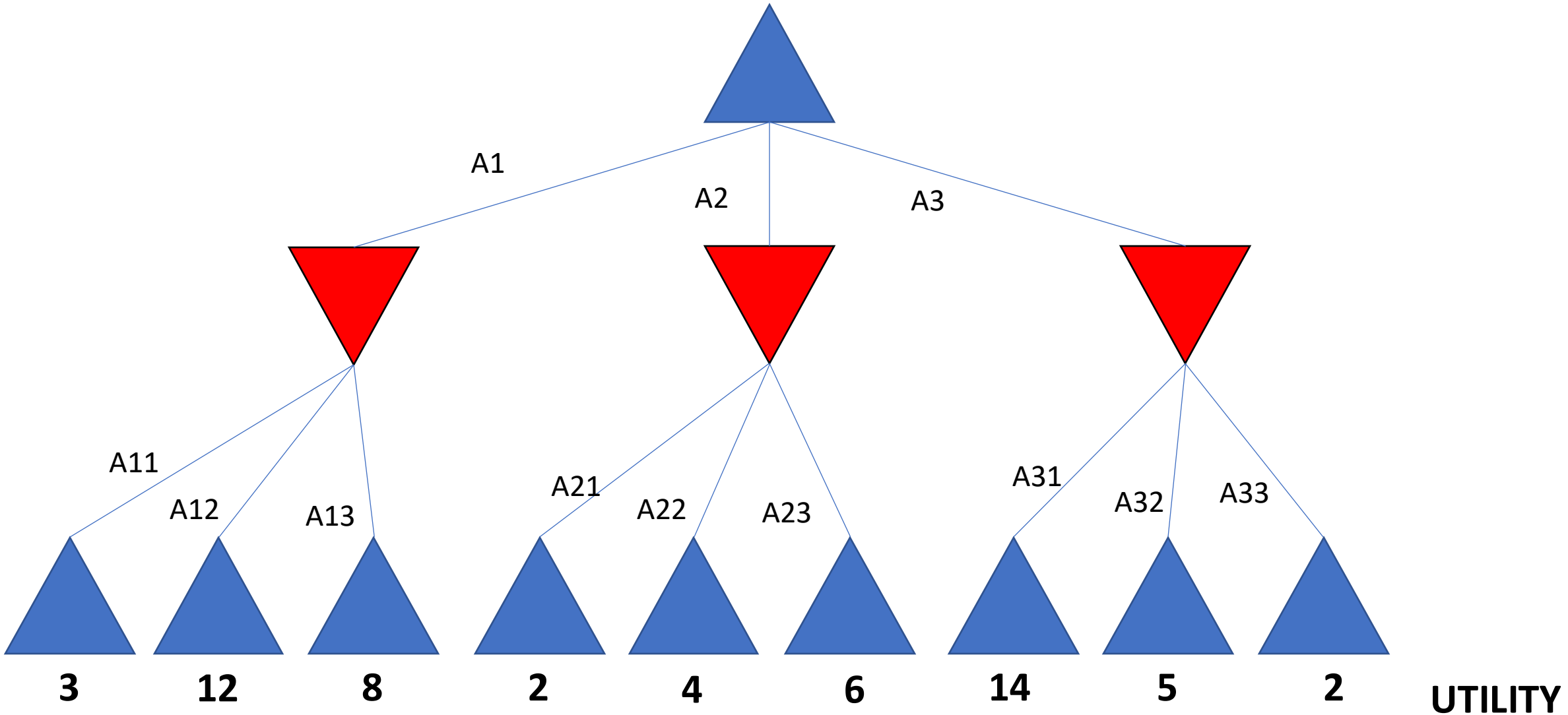
A GAME TREE

What is an adversary? (enemy)

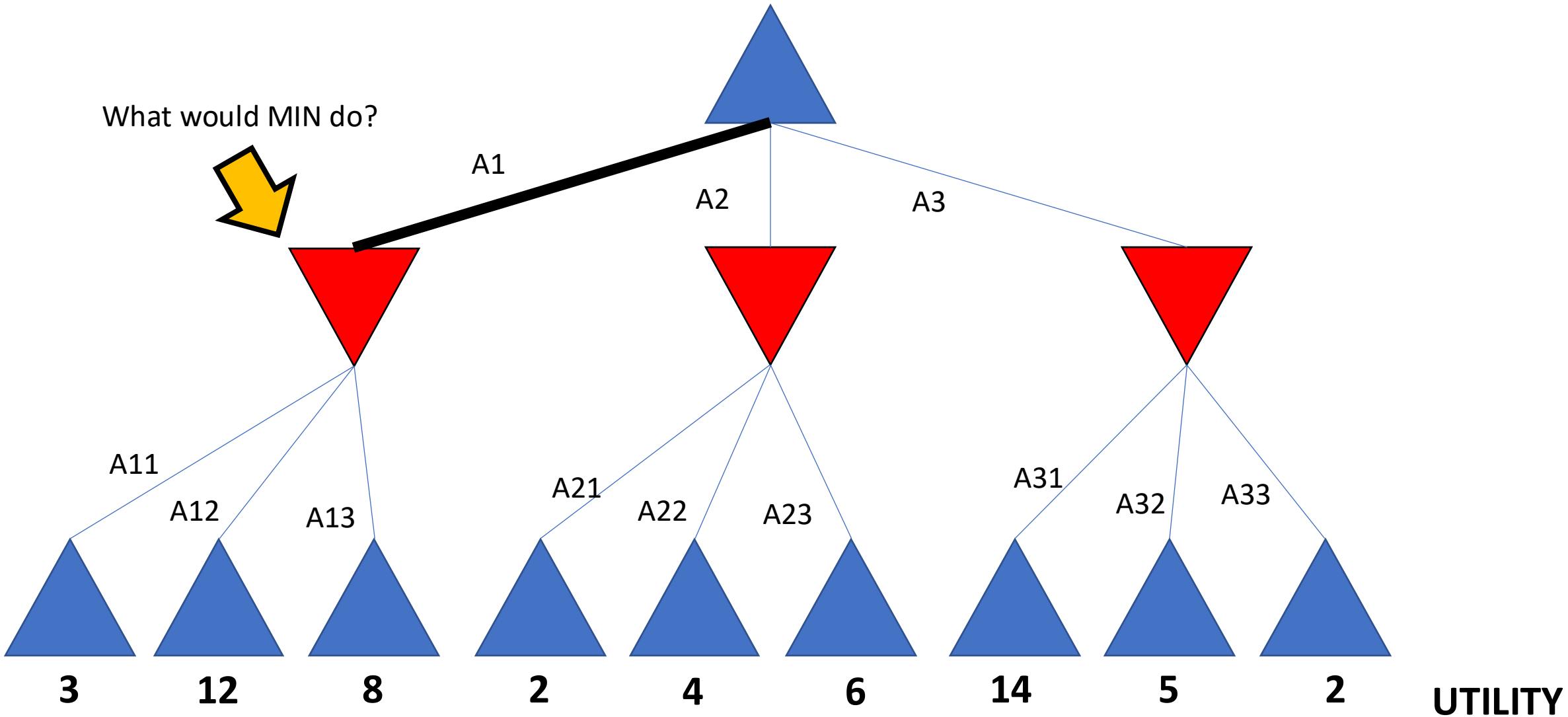
???

- They are trying to make me lose?
- How?

Example Game



Minimax


What would MIN do?
A1
A2
A3
A11
A12
A13
A21
A22
A23
A31
A32
A33
3
12
8
2
4
6
14
5
2
UTILITY

Algorithmic Minimax

function max()

- Call min() // expand the tree
- Find the best!
- return **Utility**

function min()

- Call max() // expand the tree
- Find the worst!
- return **Utility**

Properties of minimax

Complete?

- If tree is finite
  - Tic-tac-toe Yes
  - Chess No

Optimal?

- Assuming an optimal opponent… yes
- Otherwise… definitely not…

Complexity (Time)?

- $O(b^m)$
- Tic-tac-toe… b = 9 (even though it declines)

Complexity (Time)

- DFS, O(bm)

It is pretty… brute-force

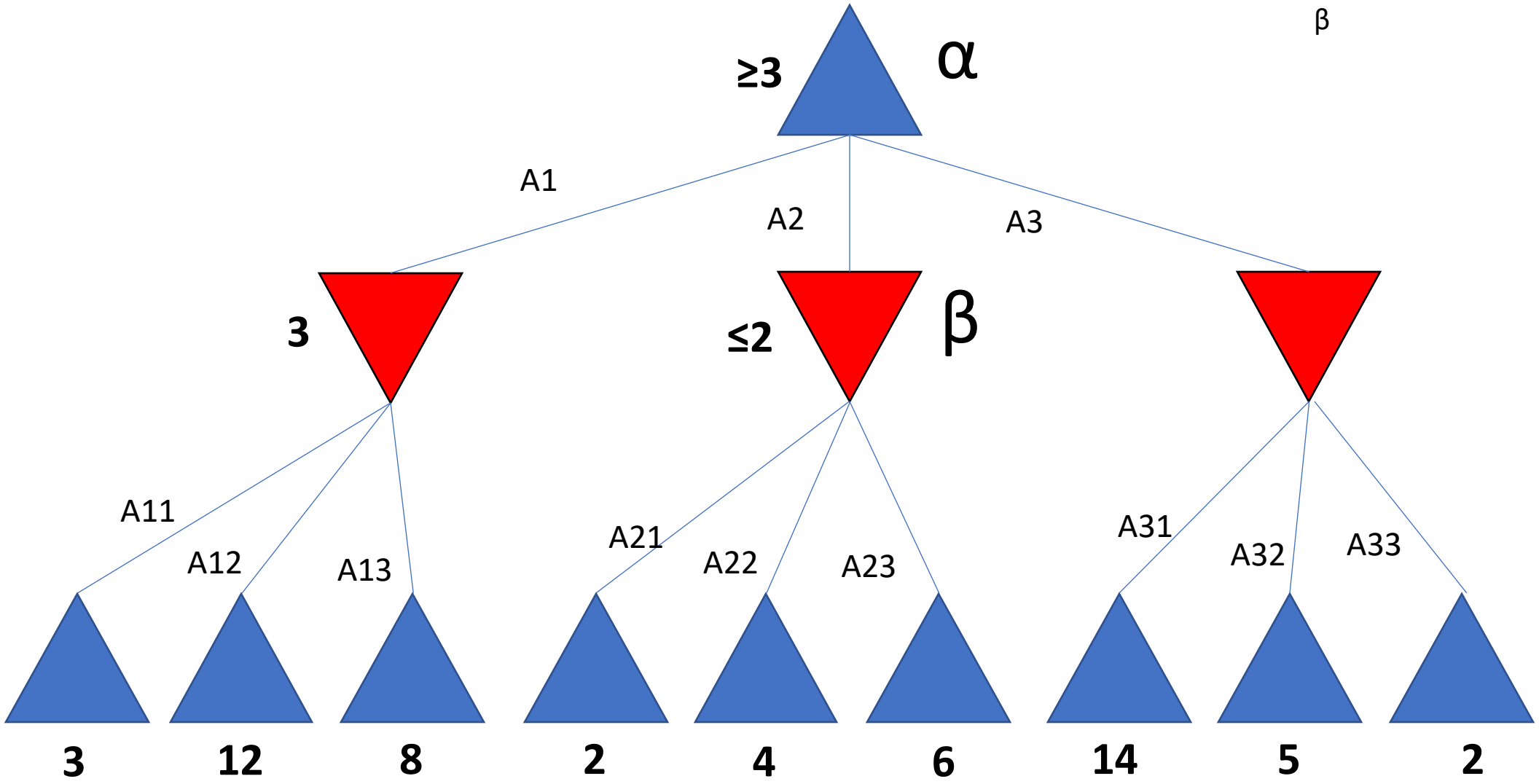
α-β Pruning

Deep Blue (chess-bot) - sorta

The idea… if we start searching down the game… and we realise we can do better than the decision our opponent would make… we can stop exploring anything to do with that decision.

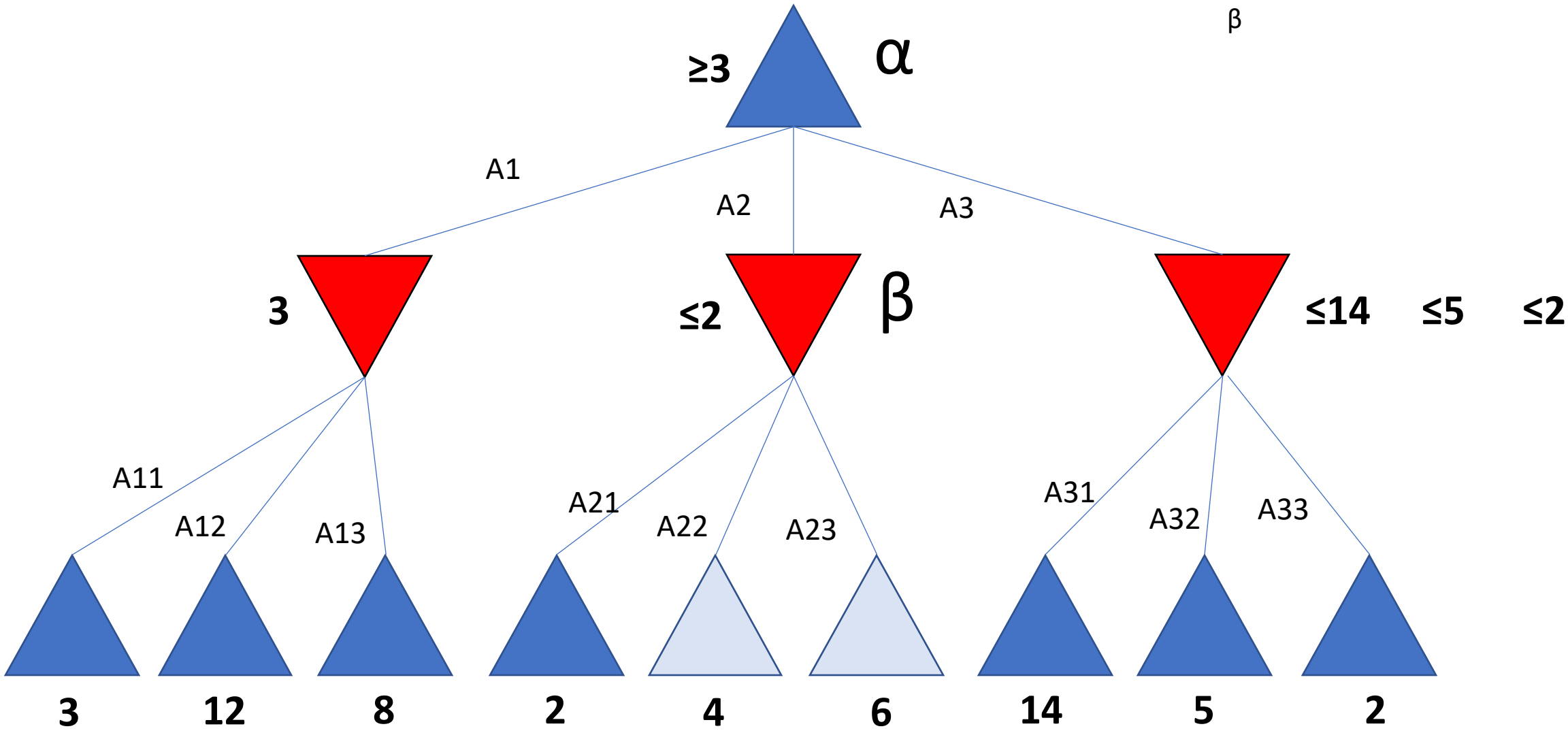
α-β Pruning

if (α > β):
// Stop Searching leaves of β



α-β Pruning

if (α > β):
// Stop Searching leaves of β



Properties of α-β Pruning

Pruning **does not** affect the final result

Lucky *move orders* make pruning more effective

Complexity (Time)?

- Perfect ordering
- $O(b^{m/2})$

But…

$O(b^{m/2})$ is still exponential…

What next?

Resource Limits

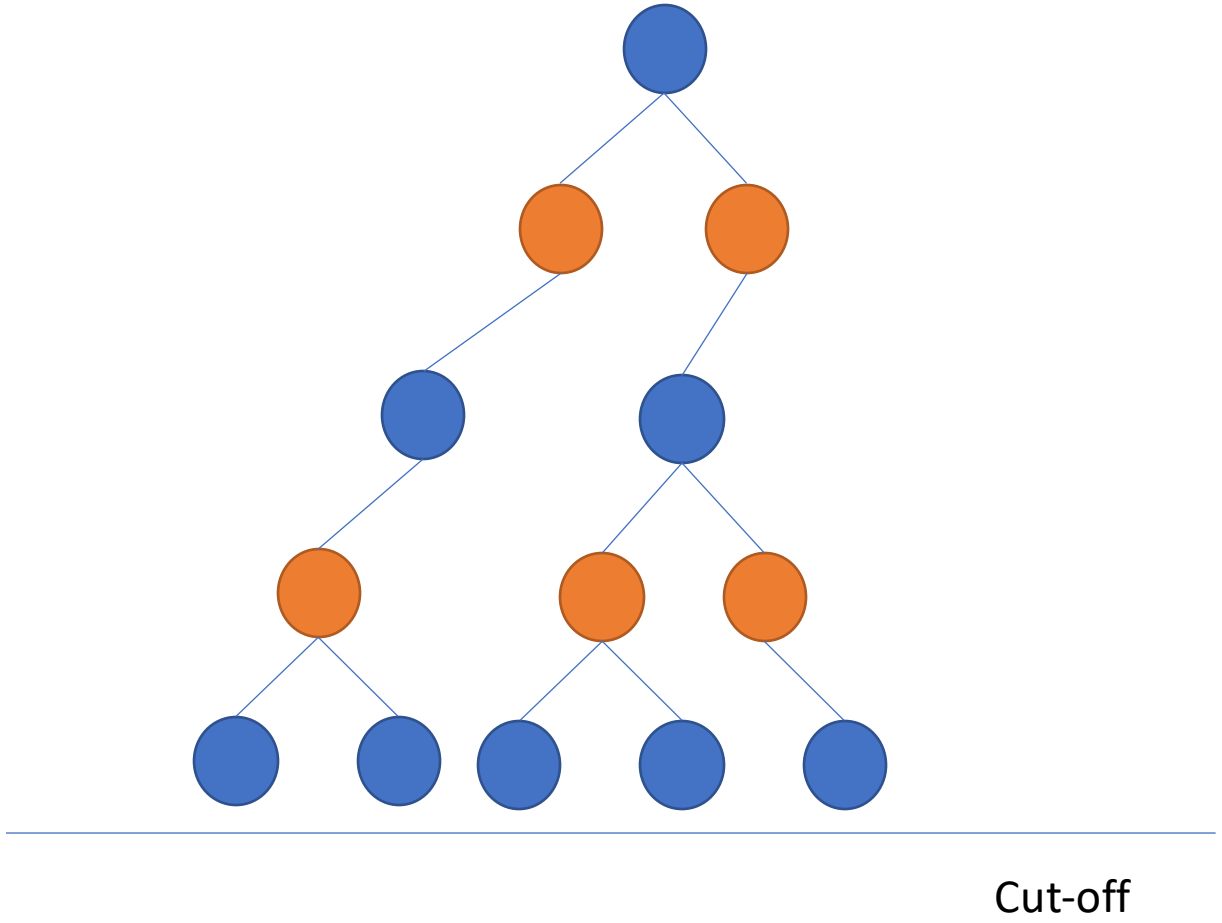
Simple Approaches

Cut-off test

- Apply a depth limit

Eval function instead of **Utility**

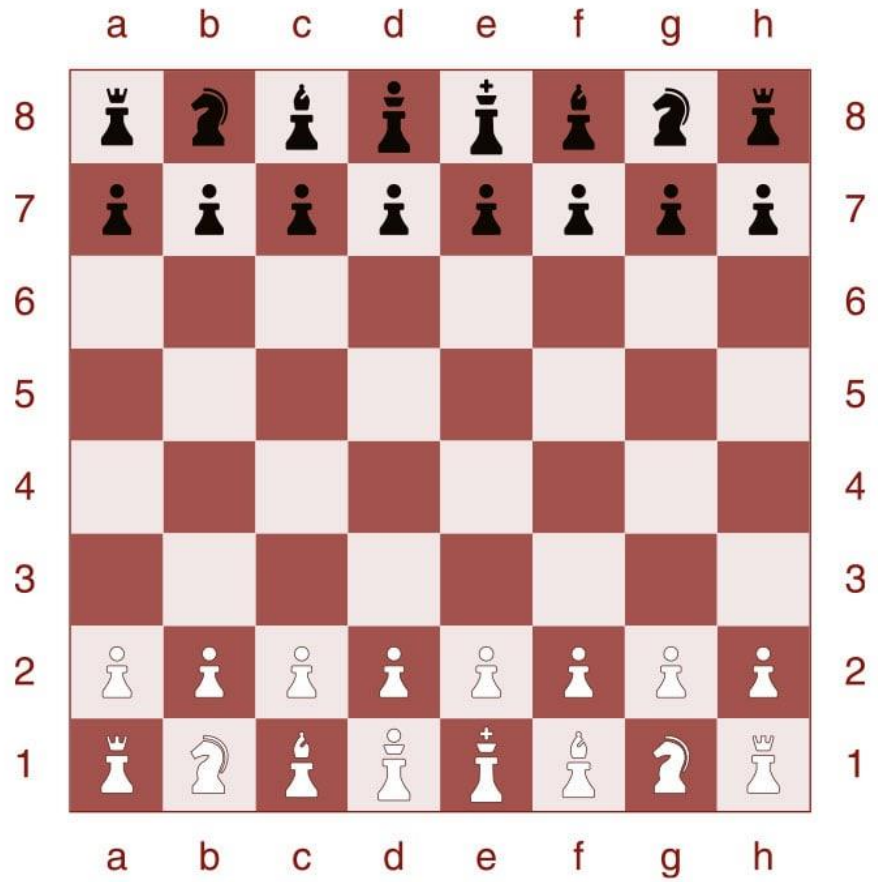
- Some kind of evaluation that estimates



Evaluation?

What could we evaluate?

- Value of each piece
- Q > R > B > N > P



Utilities are Ordinal

Having Utilities of:

- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Is the same as:

- 10, 20, 30, 40, 50, 60, 70, 80, 90, 10000000000

Some factoids

Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board:

443, 748, 401, 247 positions

Brute force much?

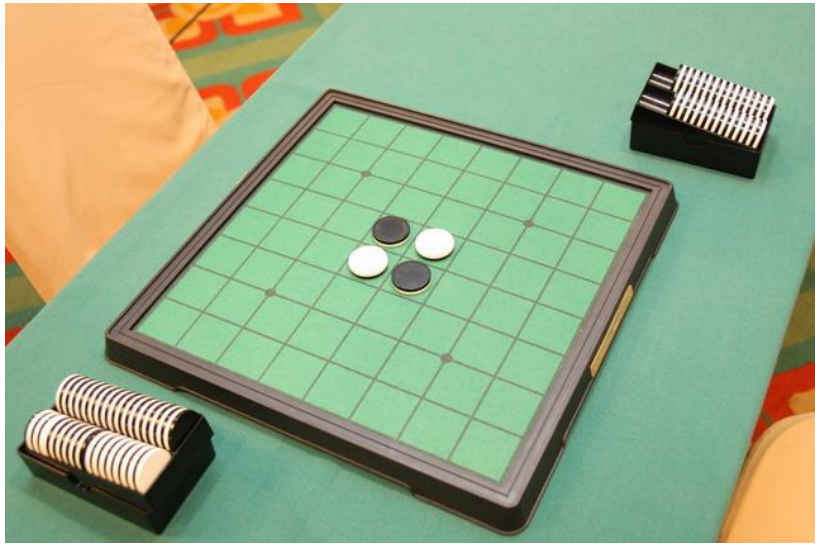
Chess: Deep Blue beat Kasparov in 1997 (6-match game)

- Lots of operations (2 x 10^8/s)
- Sophisticated evaluation
- Followed some lines up to 40 ply

More factoids

Othello:

- Humans refuse to play computers…



In 1997, Logistello defeated the reigning human world champion Takeshi Murakami by a score of 6–0. After that match, no top human players agreed to a rematch—hence the common claim that "humans refuse to play computers" at Othello. --ChatGPT-4o

Go:

- AlphaGo beat world champion Lee Sedol 4v1
- AlphaZero beats AlphaGo
- All Deep Learning models
  - Neural network + Monte Carlo Tree Search

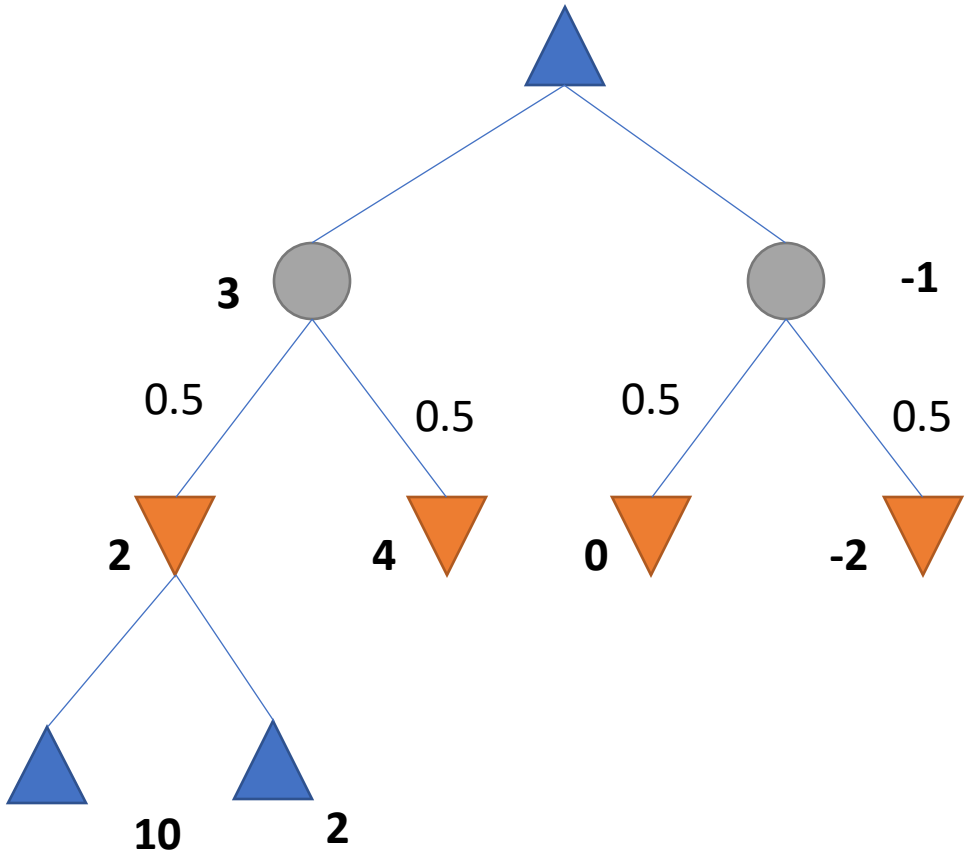
Non-deterministic games

Backgammon:

- Roll dice to move counters around the board (to an end zone)
- If you land on an enemy you ‘capture’ and they have to restart
- Can’t capture two-or-more counters
- Doubles let you move twice as many counters

Non-deterministic games

Two dice adds randomness



We model randomness like a player, then do a weighted sum to determine which outcome we want…

Expectiminimax

Non-deterministic games: Backgammon

Dice roll (2D6) => 21 possible results (1, 6 is the same as 6,1)

Backgammon ~20 legal moves

For depth = 4:

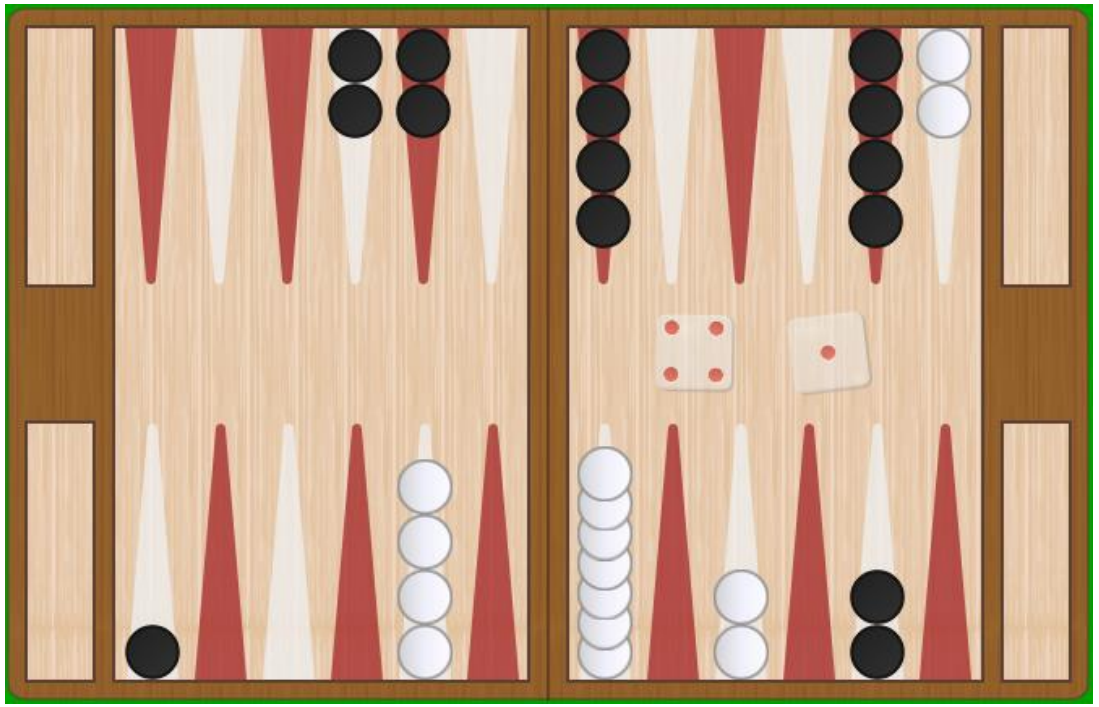
$20 * (21 * 20)^3 \approx 1.2 \times 10^9$

Also… each time you search down, the probability of that result happening (what if 6, 6, 6, 6, 6, 6, 6?). Look-ahead becomes more and more wrong.

α-β pruning becomes less effective

TD-Gammon uses depth-2 search + very good EVAL ≈ world champion (1998, lost by a mere margin of 8 points)

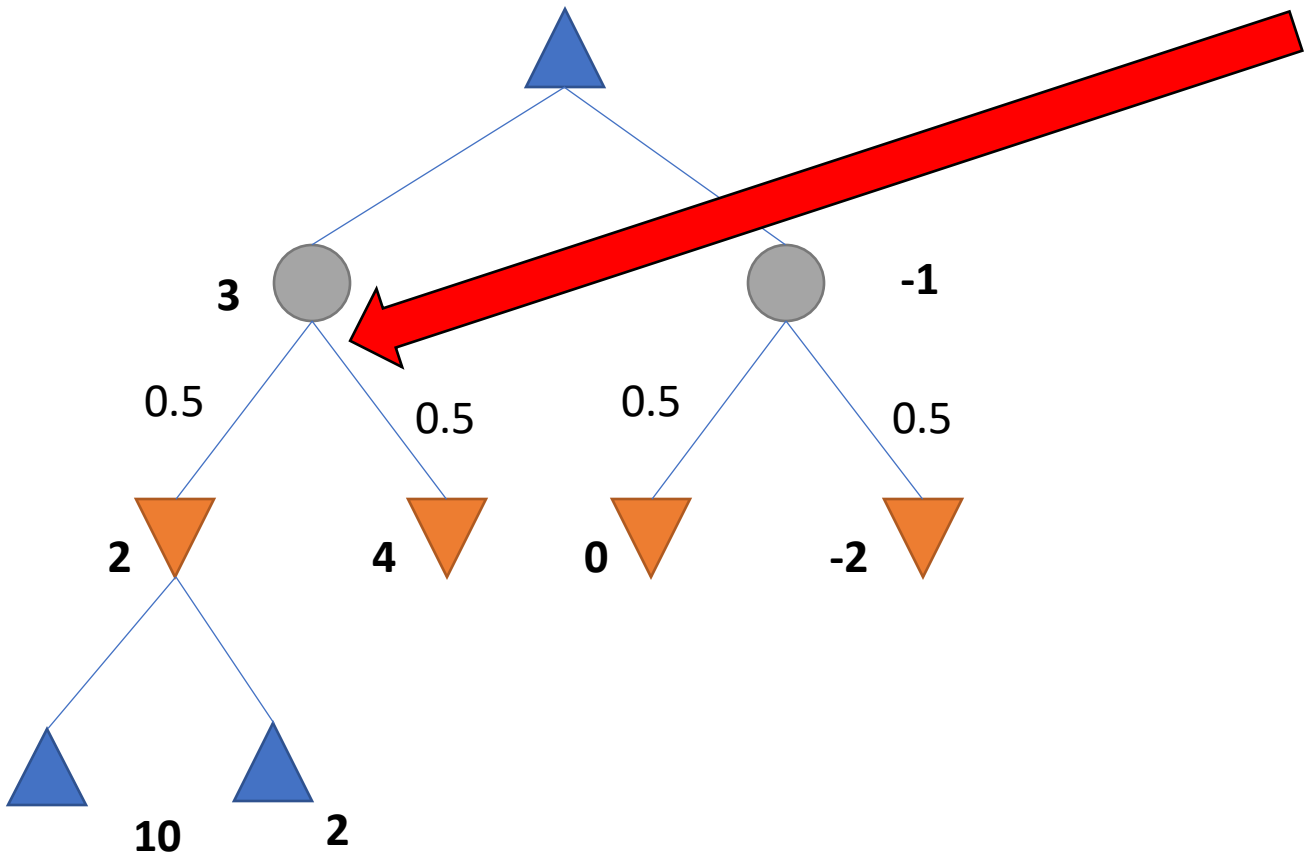
Non-deterministic games: Backgammon



What if I know which games are good or bad… because I've done a lot of… documenting each game state?

This board is… a score of 19? Maybe?

Non-deterministic games: Mathy stuff…



We're doing multiplication here

My exact weights become more important…

Is 0.8 * **2** > or < 0.2 * **4**?

Do I need to adjust my 2 and 4 so that they better represent the 'payoff'.

Questions

What are your questions?