21CS2213RA
AI for Data Science

Session -8

Contents: *Adversarial Search*

Adversarial Search

- Examine the problems that arise when we try to plan ahead in a world where other agents are planning against us.
- A good example is in board games.
- Adversarial games, while much studied in AI, are a small part of game theory in economics.

Typical AI assumptions

- Two agents whose actions alternate
- Utility values for each agent are the opposite of the other
  - creates the adversarial situation
- Fully observable environments
- In game theory terms: Zero-sum games of perfect information.
- We’ll relax these assumptions later.

Search versus Games

- Search – no adversary
  - Solution is (heuristic) method for finding goal
  - Heuristic techniques can find *optimal* solution
  - Evaluation function: estimate of cost from start to goal through given node
  - Examples: path planning, scheduling activities

- Games – adversary
  - Solution is **strategy** (strategy specifies move for every possible opponent reply).
  - **Optimality depends on opponent.**
  - Time limits force an *approximate* solution
  - Evaluation function: evaluate "goodness" of game position
  - Examples: chess, checkers, Othello, backgammon

Types of Games

	deterministic	Chance moves
Perfect information	Chess, checkers, go, othello	Backgammon, monopoly
Imperfect information (Initial Chance Moves)	Bridge, Skat	Poker, scrabble, blackjack

Game Setup

- Two players: MAX and MIN

- MAX moves first and they take turns until the game is over
  - Winner gets award, loser gets penalty.

- Games as search:
  - Initial state: e.g. board configuration of chess
  - Successor function: list of (move,state) pairs specifying legal moves.
  - Terminal test: Is the game finished?
  - Utility function: Gives numerical value of terminal states. E.g. win (+1), lose (-1) and draw (0) in tic-tac-toe or chess

Size of search trees

- b = branching factor

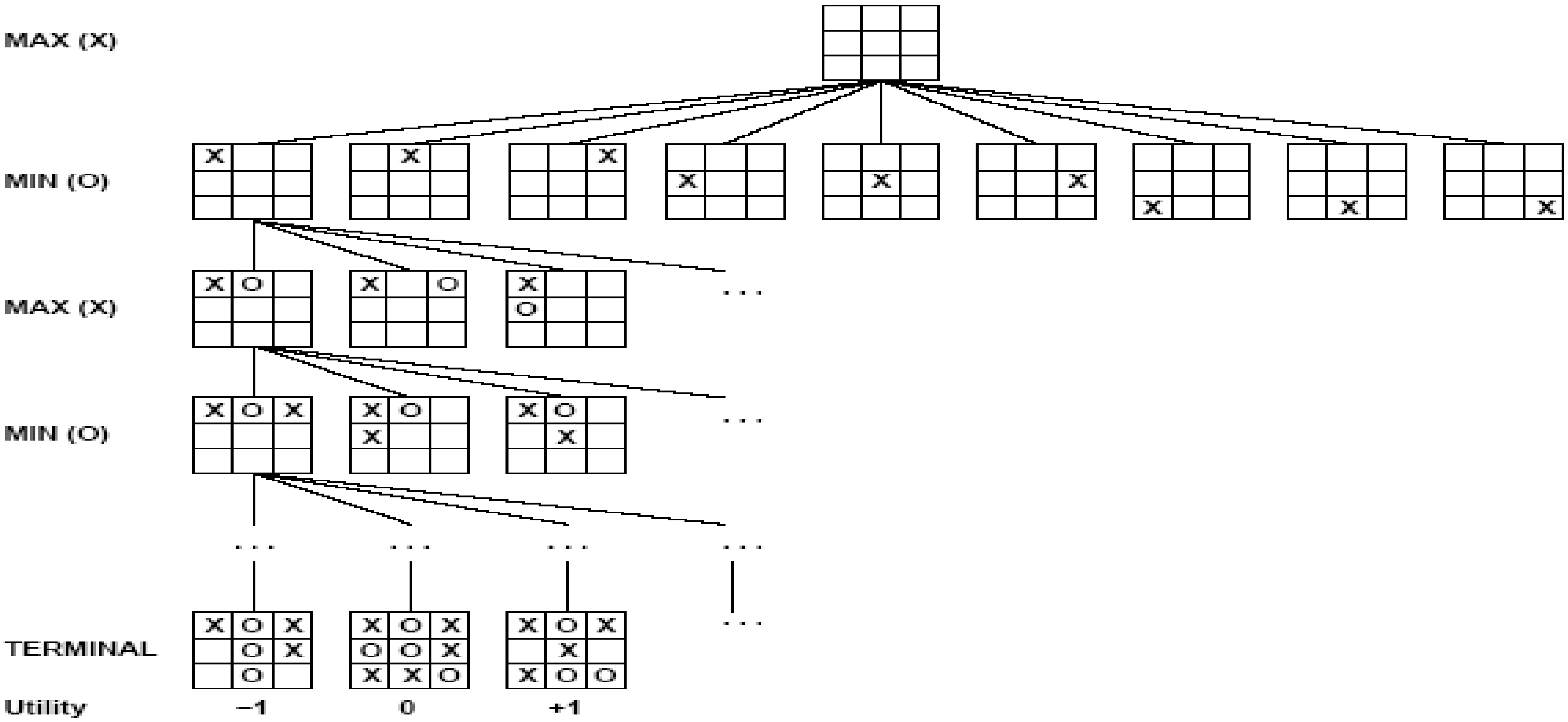
- m = number of moves by both players

- Search tree is $O(b^m)$

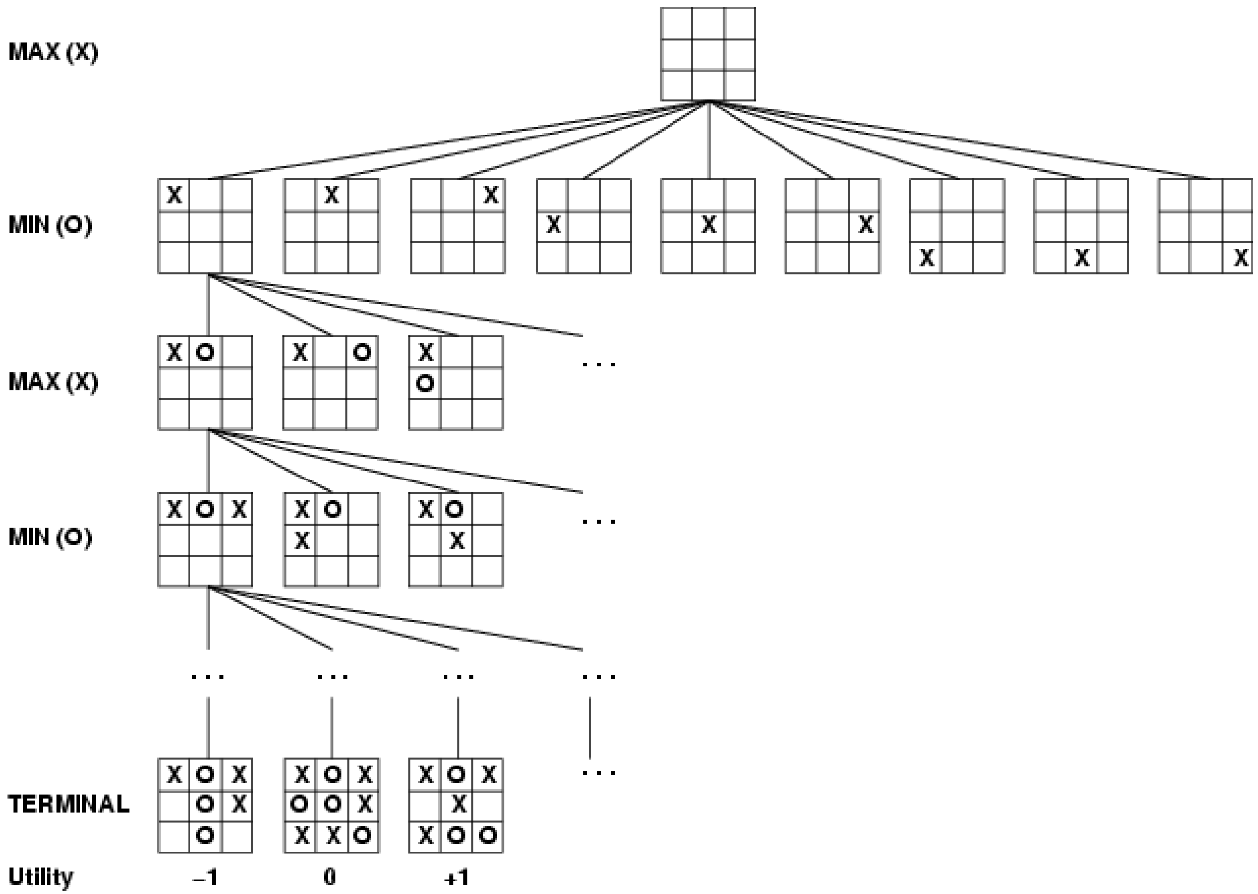
- Chess
  - b ~ 35
  - m ~100
    - search tree is ~ 10^{154} (!!)
    - completely impractical to search this

- Game-playing emphasizes being able to make optimal decisions in a finite amount of time
  - Somewhat realistic as a model of a real-world agent
  - Even if games themselves are artificial

Partial Game Tree for Tic-Tac-Toe



Game tree (2-player, deterministic, turns)



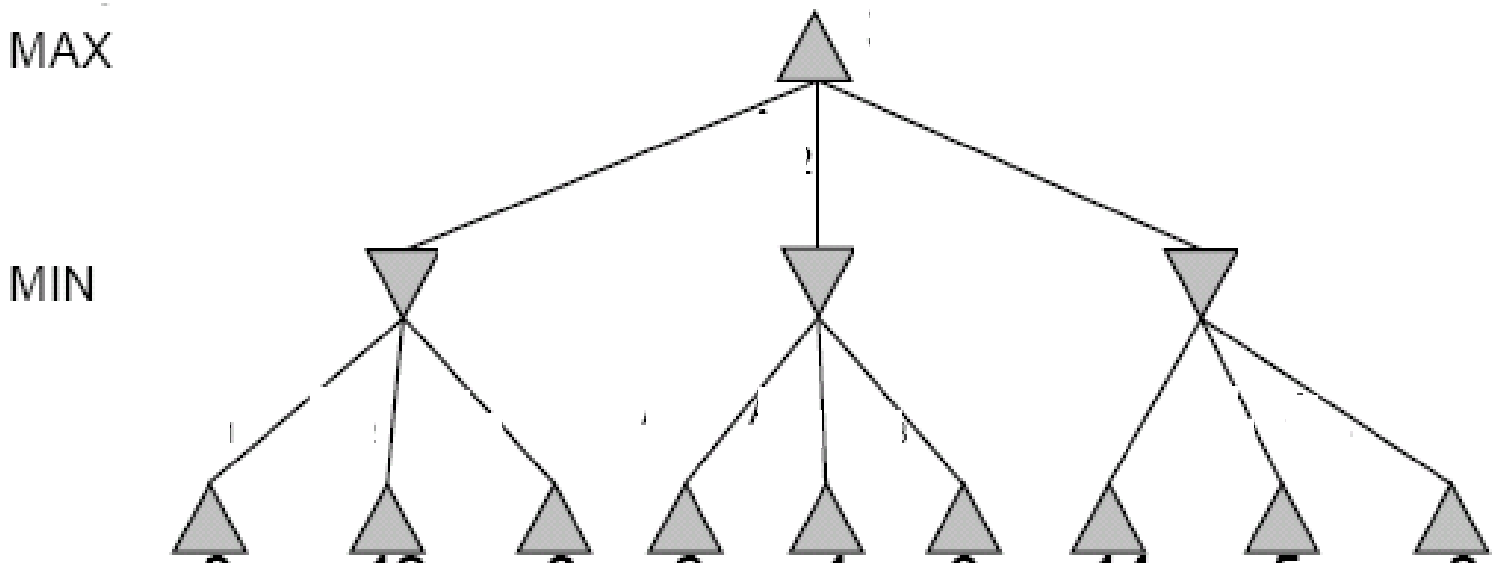
How do we search this tree to find the optimal move?

Minimax strategy:

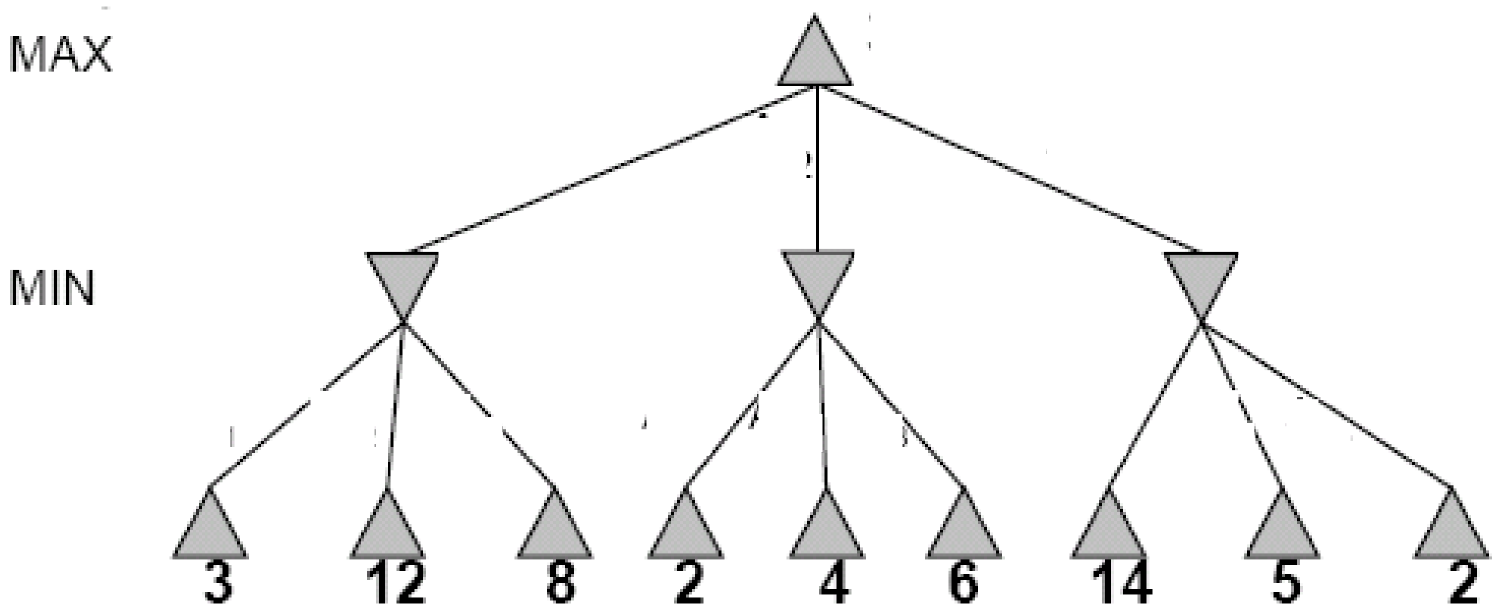
- Find the optimal *strategy* for MAX assuming an infallible MIN opponent
  - Need to compute this all the down the tree
  - Game Tree Search Demo

- Assumption: Both players play optimally!
- Given a game tree, the optimal strategy can be determined by using the **minimax value of each node.**
- Zermelo 1912.

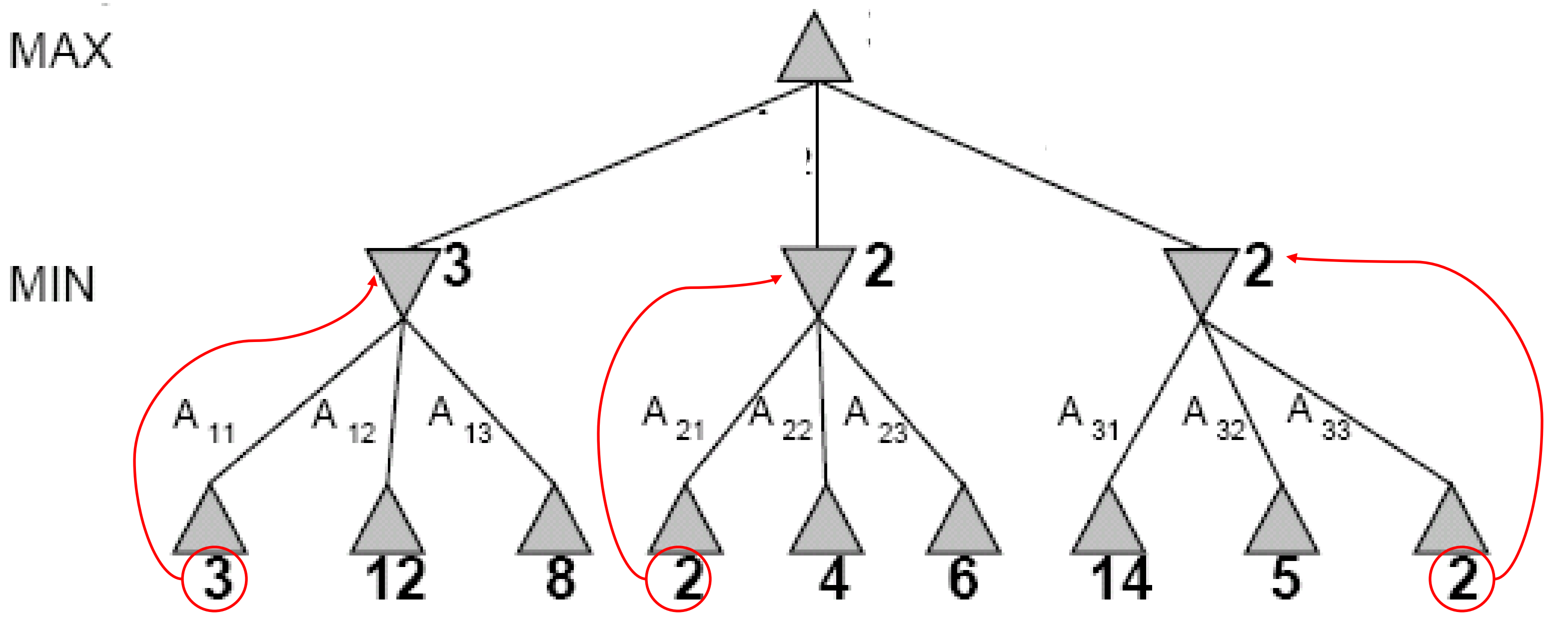
Two-Ply Game Tree



Two-Ply Game Tree

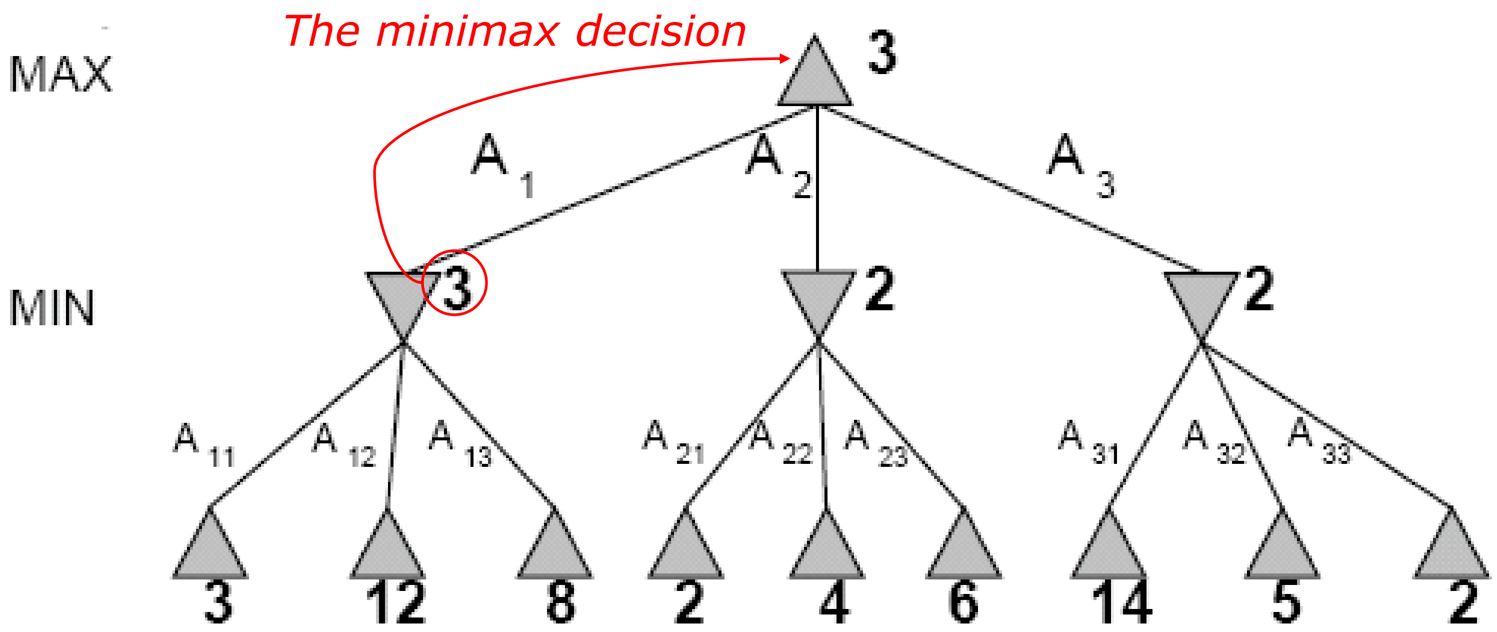


Two-Ply Game Tree



Two-Ply Game Tree

Minimax maximizes the utility for the worst-case outcome for max



Pseudocode for Minimax Algorithm

function MINIMAX-DECISION(*state*) **returns** *an action*
  inputs: *state*, current state in game
  $\boxed{v \leftarrow \text{MAX-VALUE}(state)}$
  return the *action* in SUCCESSORS(*state*) with value *v*

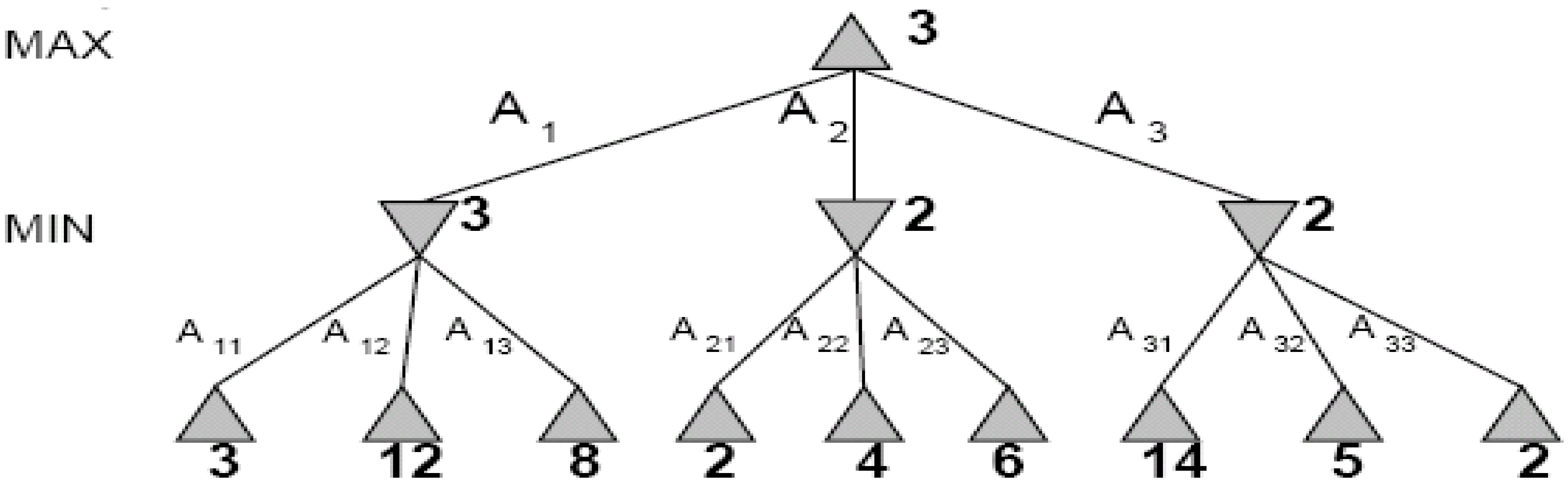
---

function MAX-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for *a,s* in SUCCESSORS(*state*) **do**
    $\boxed{v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s))}$
  return *v*

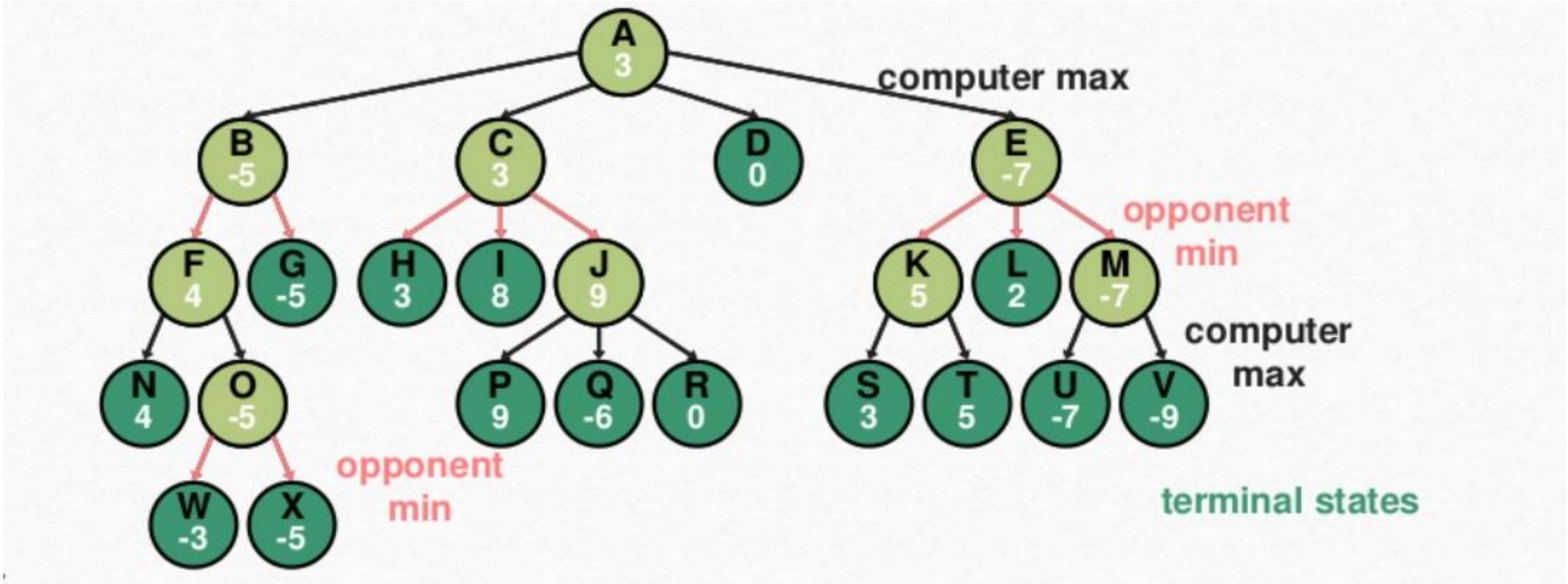
---

function MIN-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow \infty$
  for *a,s* in SUCCESSORS(*state*) **do**
    $\boxed{v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s))}$
  return *v*

Example-1

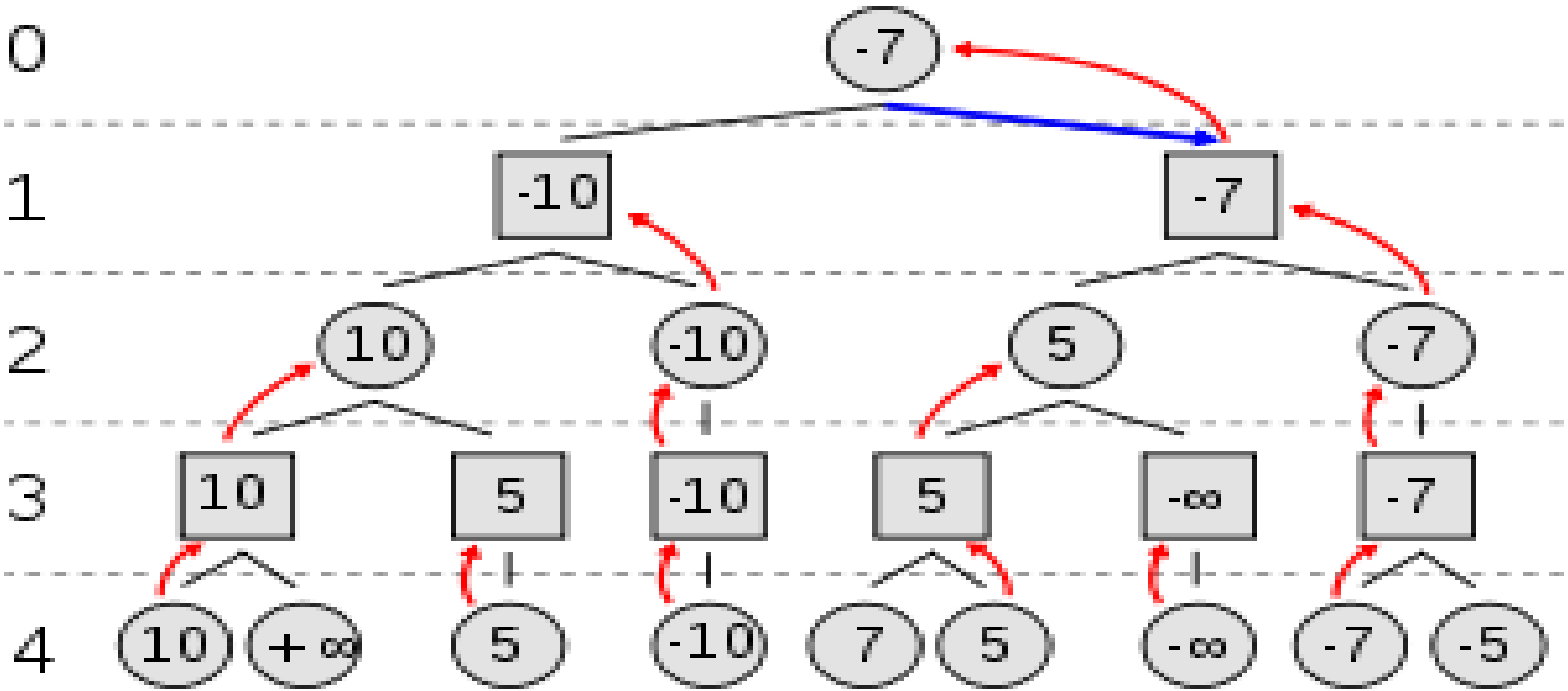

MAX
MIN
3
A 1
A 2
A 3
3
2
2
A 11
A 12
A 13
A 21
A 22
A 23
A 31
A 32
A 33
3
12
8
2
4
6
14
5
2

Example-2



A
3
computer max
B
-5
C
3
D
0
E
-7
opponent
min
F
4
G
-5
H
3
I
8
J
9
K
5
L
2
M
-7
computer
max
N
4
O
-5
P
9
Q
-6
R
0
S
3
T
5
U
-7
V
-9
W
-3
X
-5
opponent
min
terminal states

Example-3



Properties of Minimax

- **Complete?** Yes (if tree is finite)
- **Optimal?** Yes (against an optimal opponent)
- **Time complexity?** $O(b^m)$
- **Space complexity?** $O(bm)$ (depth-first exploration

End of Session

Thank you