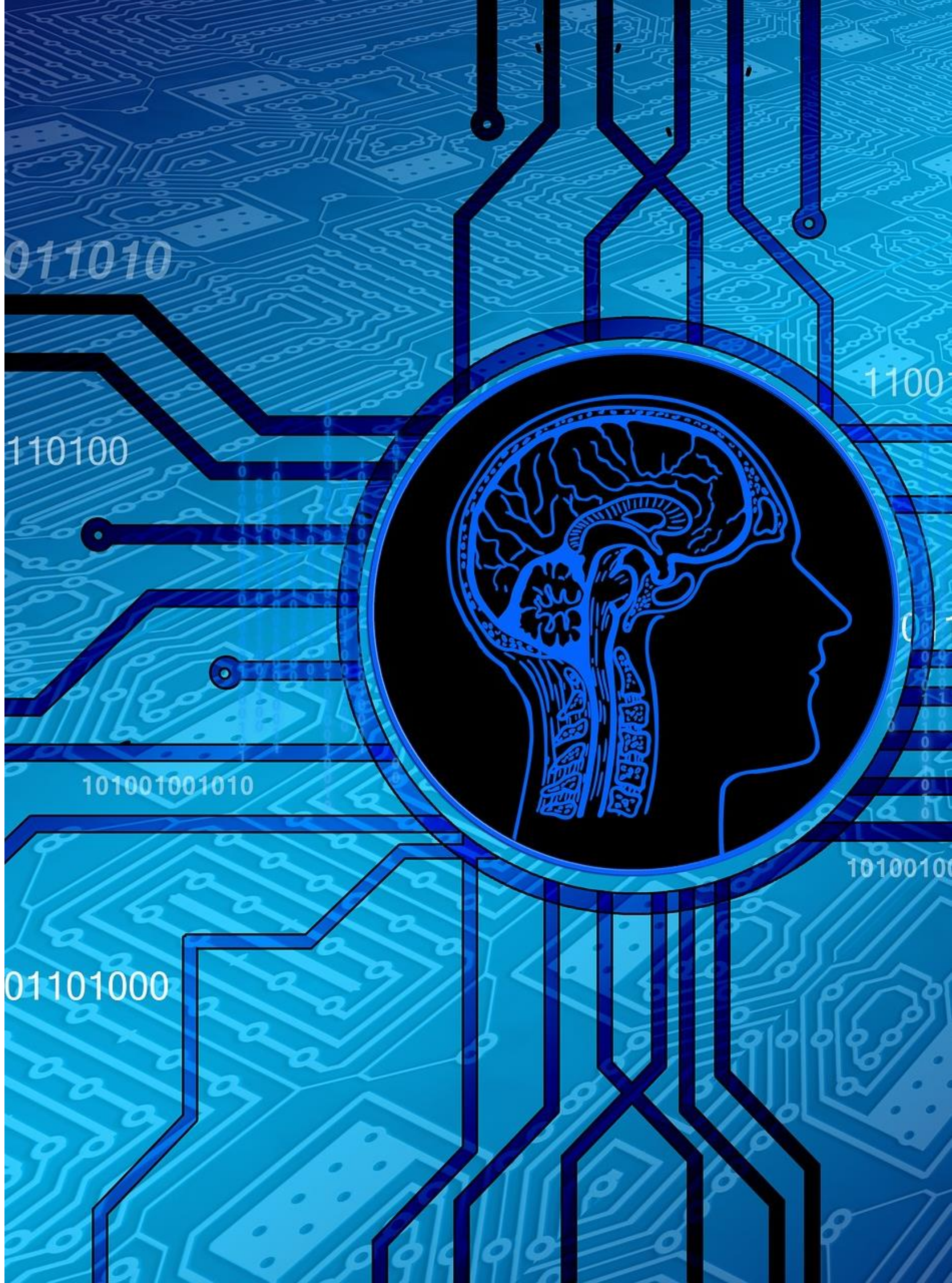
Revision

Informatics 2D: Reasoning and Agents



Intelligent Agents and their Environments

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents
- Learning agents

- Properties of environments
  - Partially vs. fully observable
  - Deterministic vs. stochastic
  - Episodic vs. sequential
  - Static vs. dynamic
  - Discrete vs. continuous
  - Single vs. multi-agent

Problem Solving by Searching

- Problem formulation usually requires abstracting away real-world details to define a state space that can feasibly be explored.

- Variety of uninformed search strategies:
  - breadth-first, depth-first, iterative deepening

- Iterative deepening search uses only linear space and not much more time than other uninformed algorithms.

Evaluating search strategies

completeness: does it always find a solution if one exists?

time complexity: number of nodes generated / expanded

space complexity: maximum number of nodes in memory

optimality: does it always find a least-cost solution?

Time and space complexity are measured in terms of:

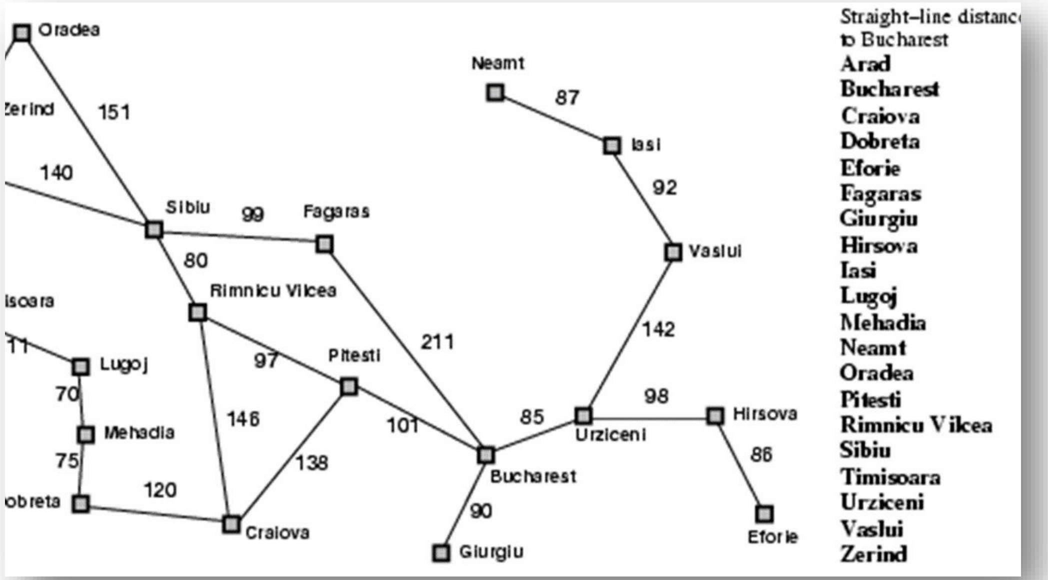
- ***b***: maximum branching factor of the search tree
- ***d***: depth of the least-cost solution
- ***m***: maximum depth of the state space (may be ∞)

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Summary of algorithms

Informed Search

➢ Smart search based on heuristic scores
- Best-first search
- Greedy best-first search
- A* search
- Admissible heuristics and optimality.



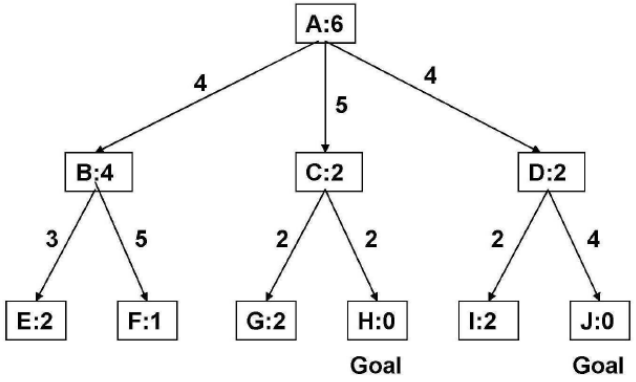
A* search

- Evaluation function $f(n) = g(n) + h(n)$
  - $g(n)$ = cost so far to reach n
  - $h(n)$ = estimated cost from n to goal
  - $f(n)$ = estimated total cost of path through n to goal

- Avoid expanding paths that are already expensive

Example

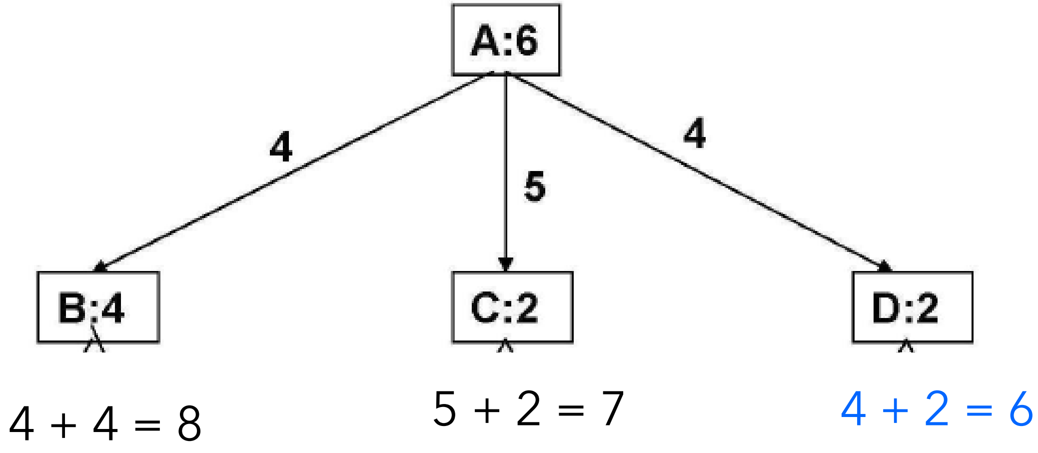
3. Consider the following search tree in which the nodes represent states and the arcs represent the moves connecting these states. Each node is labelled by a letter. The numbers on the arcs represent the *true* cost of the associated move. The numbers on the nodes represent the *estimated* cost of reaching the goal state from that node.



In which order would the A^* algorithm explore this search tree?

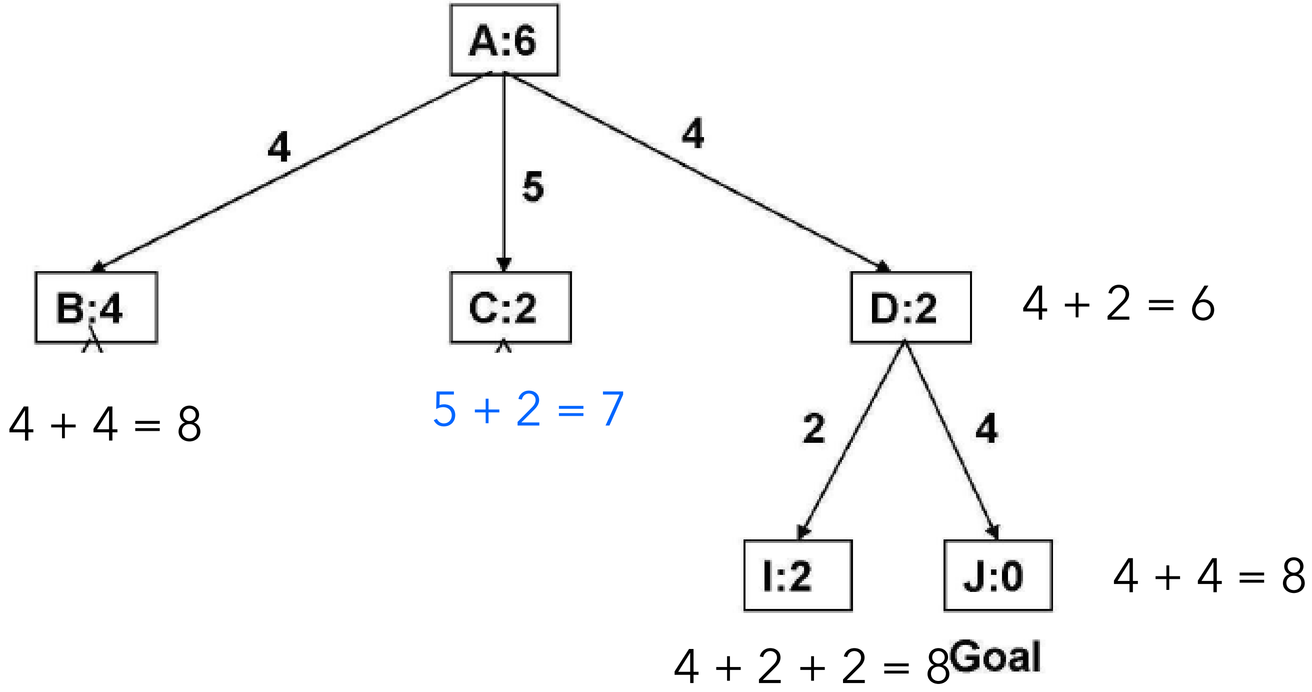
(a) A, B, C, D, I, J, G, H.
(b) A, B, C, D, I, J.
(c) A, C, H.
(d) A, D, I, J.
(e) A, C, G, H.

Example



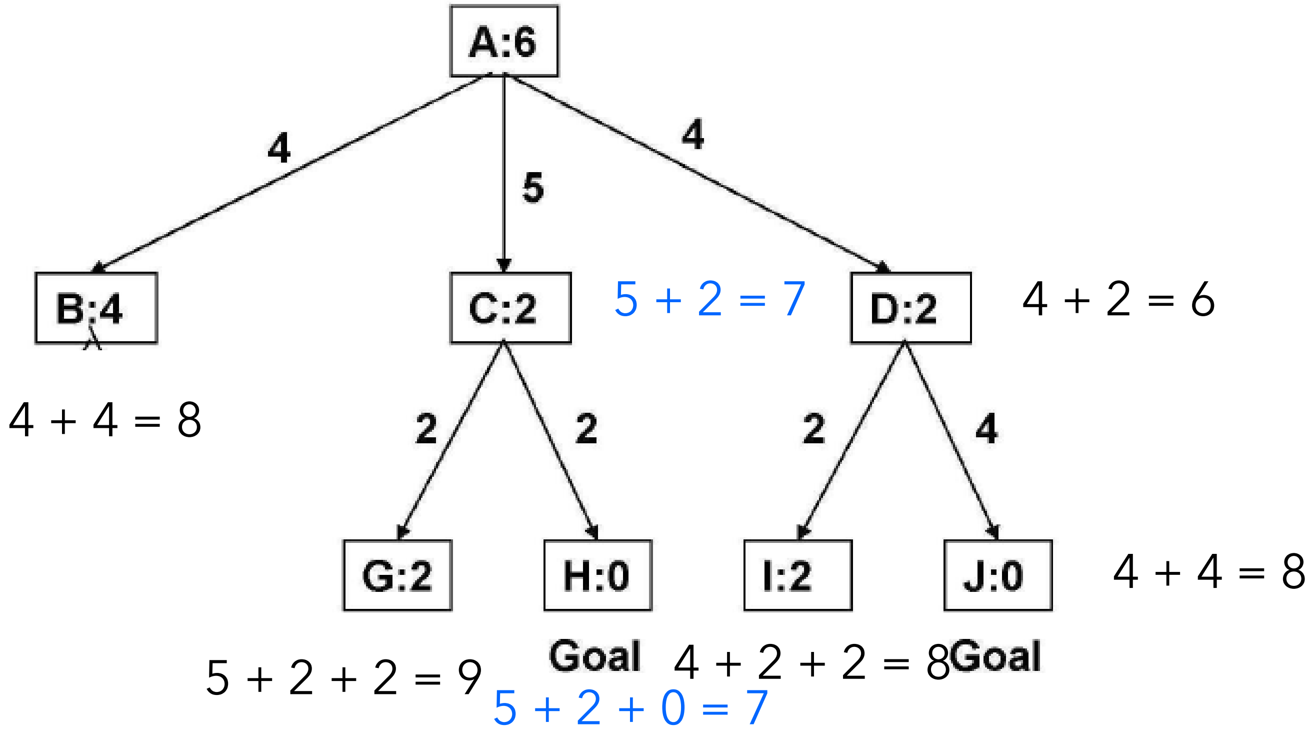
A
B C D

Example



A
B C D
I J

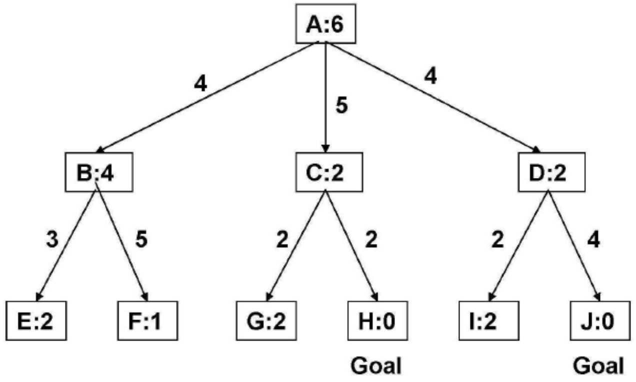
Example



A
B C D
I J
G H

We're done as we've expanded a node containing a goal state

3. Consider the following search tree in which the nodes represent states and the arcs represent the moves connecting these states. Each node is labelled by a letter. The numbers on the arcs represent the *true* cost of the associated move. The numbers on the nodes represent the *estimated* cost of reaching the goal state from that node.



In which order would the A^* algorithm explore this search tree?

(a) A, B, C, D, I, J, G, H.
(b) A, B, C, D, I, J.
(c) A, C, H.
(d) A, D, I, J.
(e) A, C, G, H.

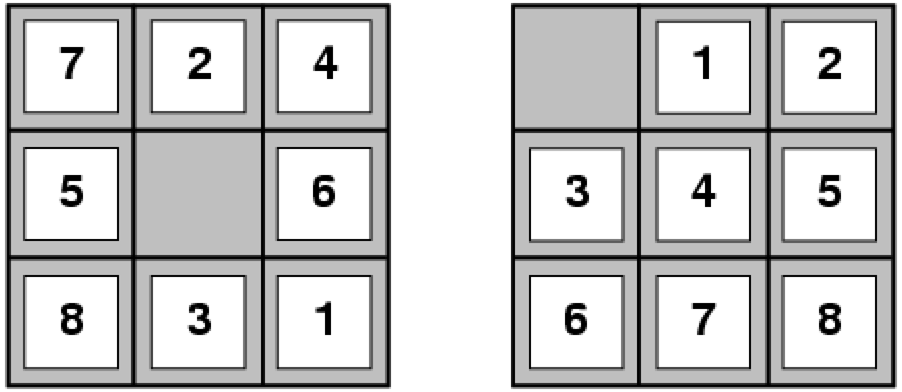
Example

Admissible heuristics

Example: 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



Start State

Goal State

$h_1(S) = ?$

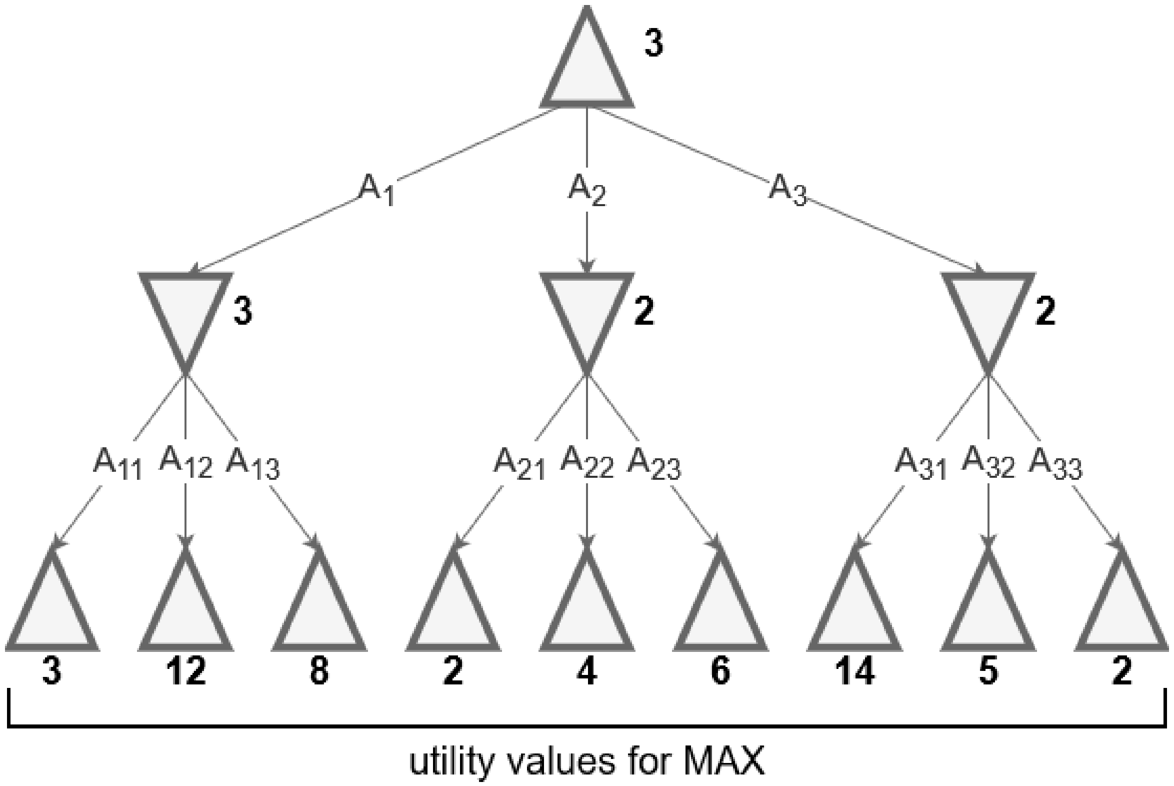
$h_2(S) = ?$

Smart Searching Using Constraints

- Constraint Satisfaction Problems (CSPs):
  - states defined by values of a fixed set of variables
  - goal test defined by constraints on variable values
- Backtracking = depth-first search with one variable assigned per node.
- Variable ordering and value selection heuristics help significantly.
- Forward checking prevents assignments that guarantee later failure.
- Constraint propagation (e.g., arc consistency) does additional work to constrain values and detect inconsistencies.

Adversarial Search

- Minimax assumes that both players play optimally
- Informally: Each agent is making its decision for the next move based on the assumption that the other agent is playing as well as it can.



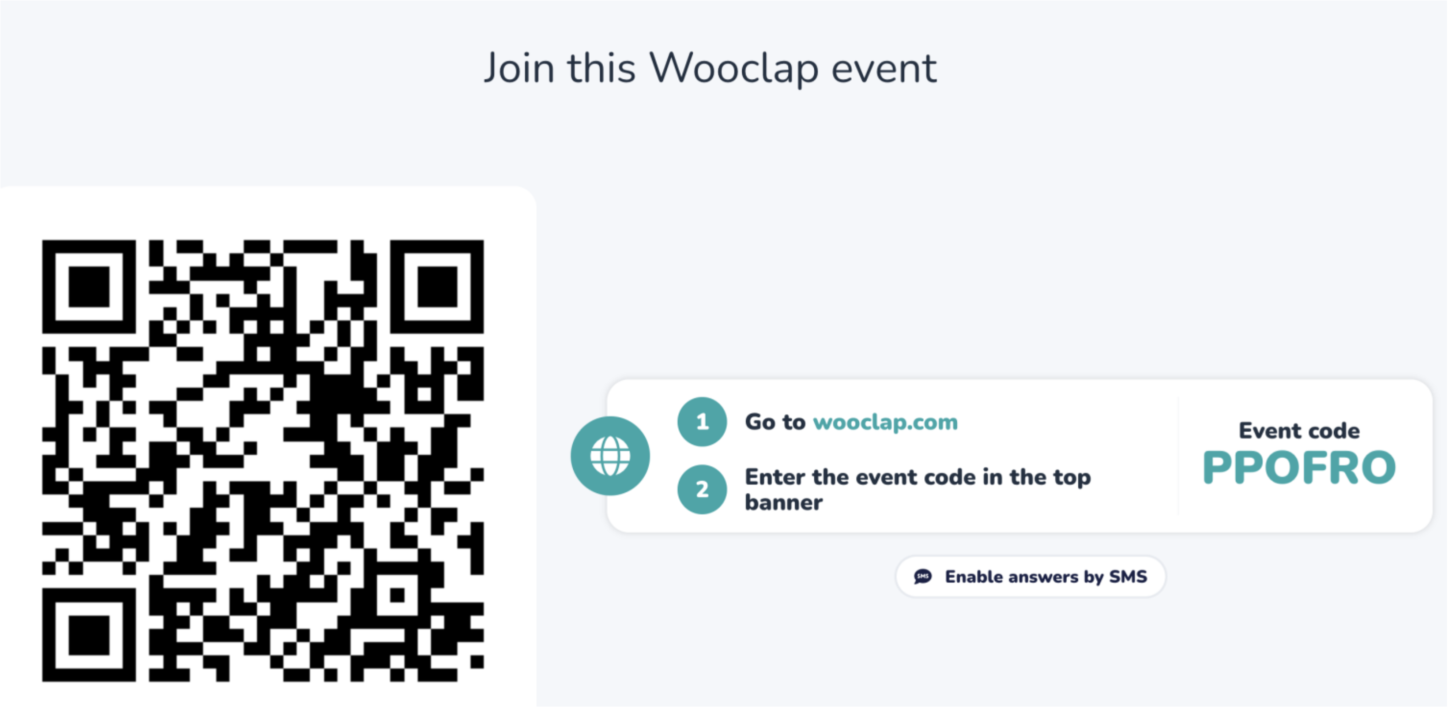
Adversarial Search (Contd)

- α-β Pruning and its properties
- Reasoning about relevant computations only enables search space to be pruned.
- How to deal with deep trees: need for evaluation functions.

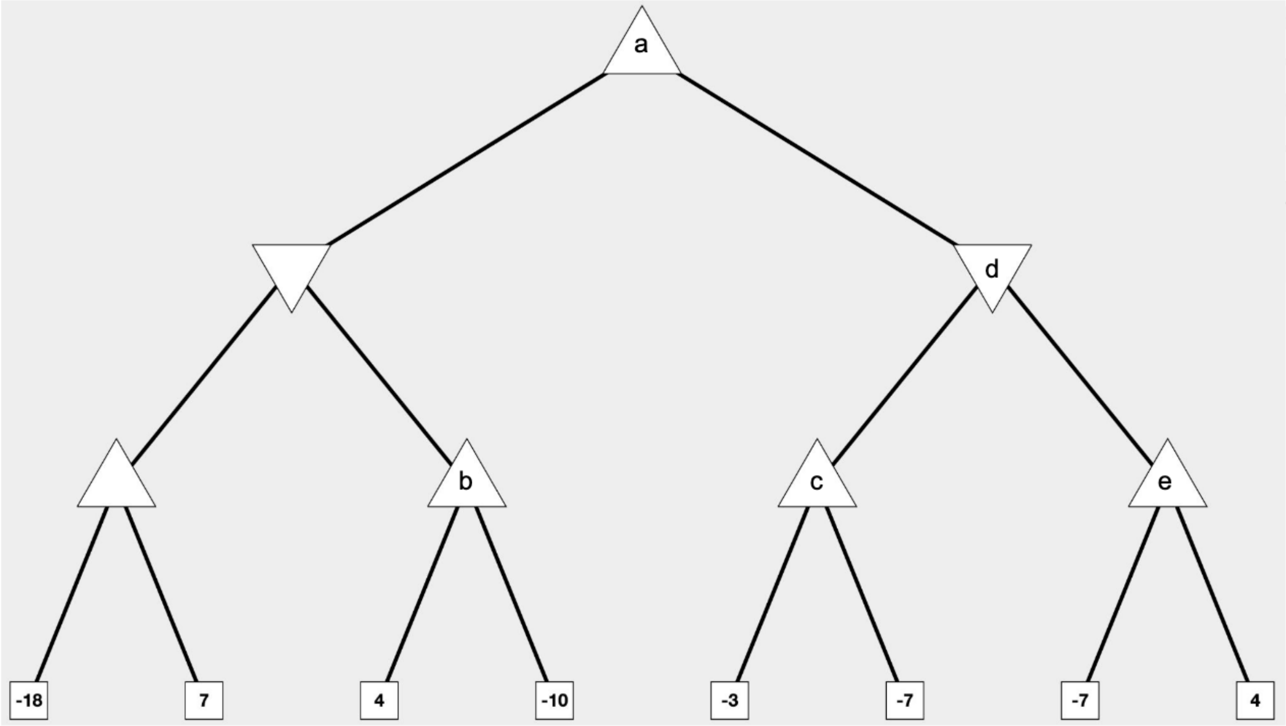
α-β Pruning (Example)

https://people.cs.pitt.edu/~litman/courses/cs2710/lectures/pruningReview.pdf

Wooclap Time!



Consider the following search tree for a two-person game, which is searched depth-first, left-to-right. Some nodes are named by a letter. Nodes with shape △ are where Max is due to play; nodes with shape ▽ are where Min is due to play. The numbers below the leaves of the tree are the results of the evaluation function applied to that leaf. Answer Questions 6 and 7 by considering this game tree.



Thank you!