Search

CSE 473
Artificial Intelligence

Oren Etzioni

What is Search?

Search is a class of techniques for systematically finding or constructing solutions to problems.

Example technique: generate-and-test.

Example problem: Combination lock.

1. Generate a possible solution.
2. Test the solution.
3. If solution found THEN done ELSE return to step 1.

Why is search interesting?

Many (all?) AI problems can be formulated as search problems!

Examples:

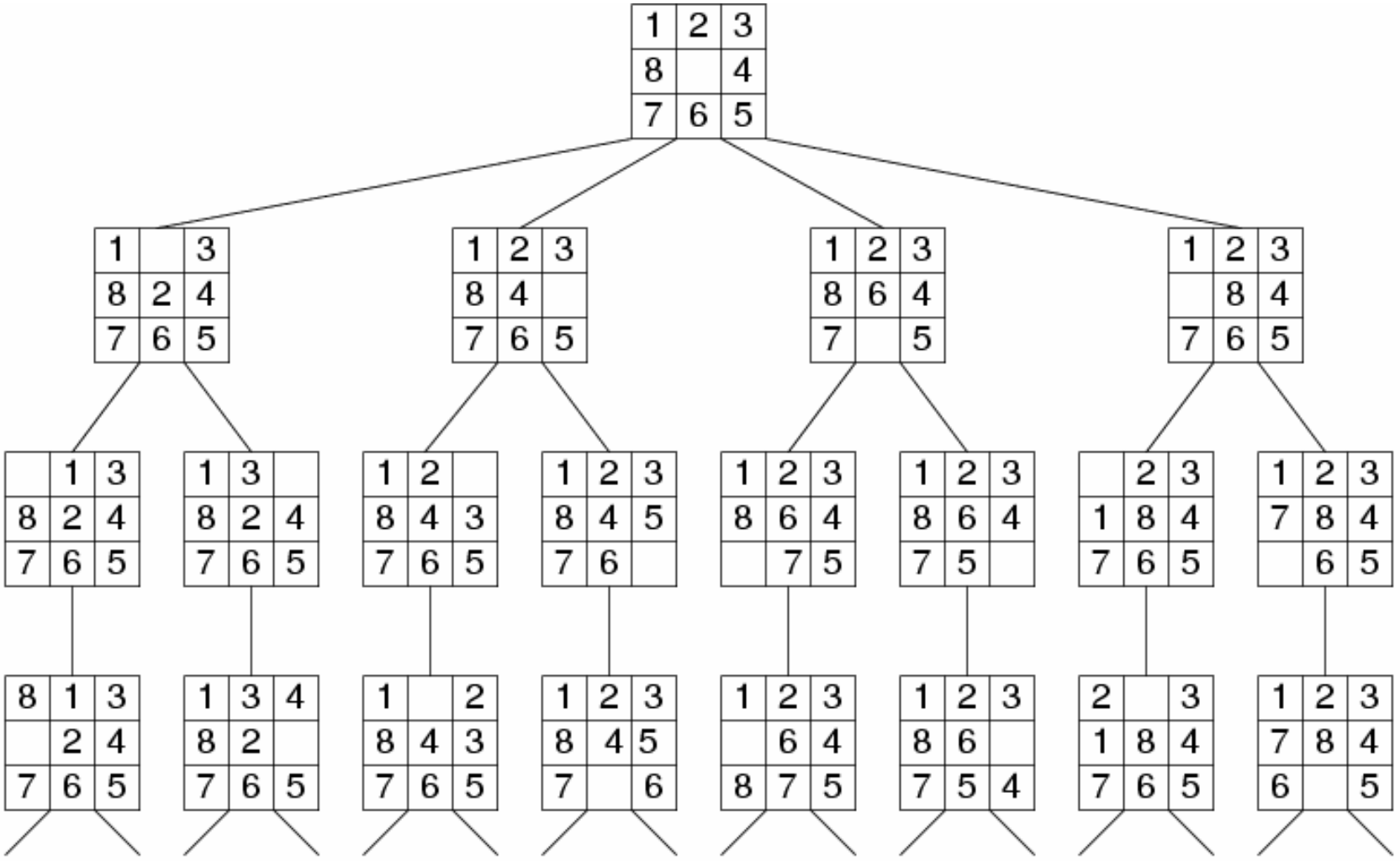
- Path planning
- Games
- Natural Language Processing
- Machine learning
- Genetic algorithms

Weak Methods

"In the knowledge lies the power…"
[Feigenbaum]

But what if no knowledge????

Search Tree Example: Fragment of 8-Puzzle Problem Space



Search thru a Problem Space / State Space

Input:

- Set of states
- Operators [and costs]
- Start state
- Goal state [test]

Output:

- Path: start ⇒ a state satisfying goal test
- [May require shortest path]

Example: Route Planning

Input:

- Set of states
- Operators [and costs]
- Start state
- Goal state (test)

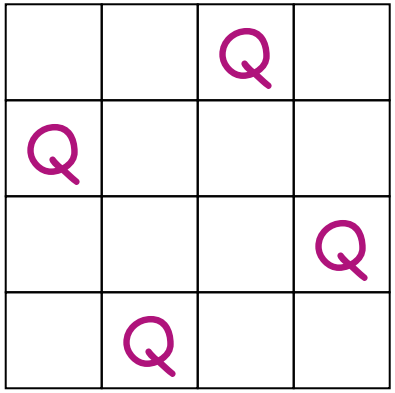
Output:

Example: N Queens

Input:

- Set of states
- Operators [and costs]
- Start state
- Goal state (test)

Output



Classifying Search

GUESSING ("Tree Search")

- Guess how to extend a partial solution to a problem.
- Generates a tree of (partial) solutions.
- The leafs of the tree are either "failures" or represent complete solutions

SIMPLIFYING ("Inference")

- Infer new, stronger constraints by combining one or more constraints (without any "guessing")

Example: X+2Y = 3

X+Y =1

therefore Y = 2

WANDERING ("Markov chain")

- Perform a (biased) random walk through the space of (partial or total) solutions

Guessing – State Space Search

1. BFS
2. DFS
3. Iterative Deepening
4. Bidirectional
5. Best-first search
6. A*
7. Game tree
8. Davis-Putnam (logic)
9. Cutset conditioning (probability)

Simplification – Constraint Propagation

1. Forward Checking
2. Path Consistency (Waltz labeling, temporal algebra)
3. Resolution
4. "Bucket Algorithm"

Wandering – Randomized Search

1. Hillclimbing
2. Simulated annealing
3. Walksat
4. Monte-Carlo Methods

Constraint Satisfaction

Search Strategies v2

Blind Search

- Depth first search
- Breadth first search
- Iterative deepening search
- Iterative broadening search

Informed Search

Constraint Satisfaction

Adversary Search

Depth First Search

Maintain stack of nodes to visit

Evaluation

- Complete?

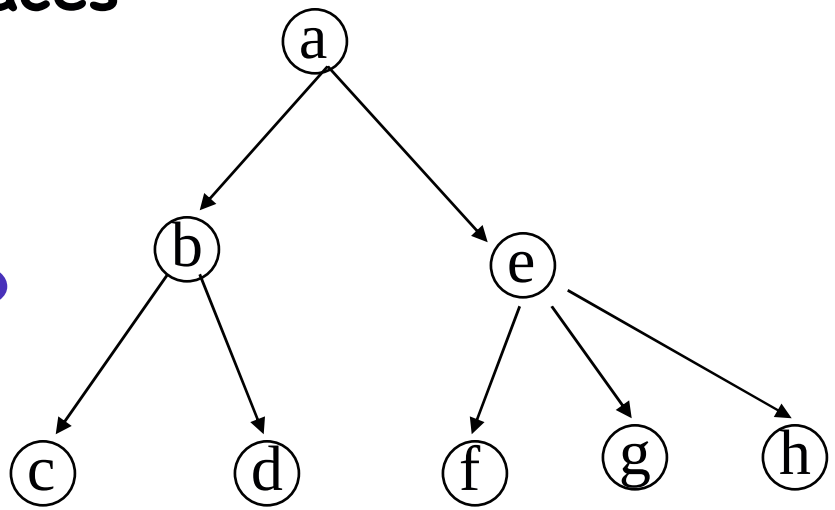
  Not for infinite spaces

- Time Complexity?

  O(b^d)

- Space Complexity?

  O(d)

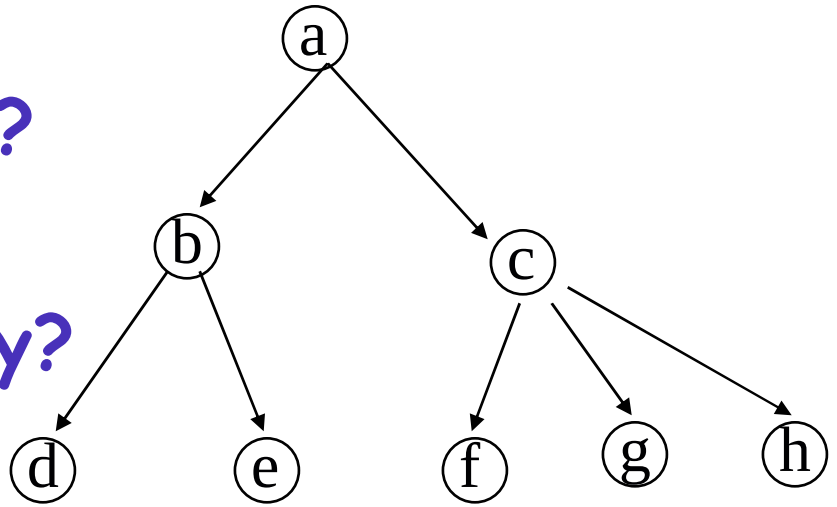


Breadth First Search

Maintain queue of nodes to visit

Evaluation

- Complete?
  Yes
- Time Complexity?
  O(b^d)
- Space Complexity?
  O(b^d)



Memory a Limitation?

Suppose:

- 2 GHz CPU
- 1 GB main memory
- 100 instructions / expansion
- 5 bytes / node

- 200,000 expansions / sec
- Memory filled in 100 sec … < 2 minutes

Iterative Deepening Search

DFS with limit; incrementally grow limit

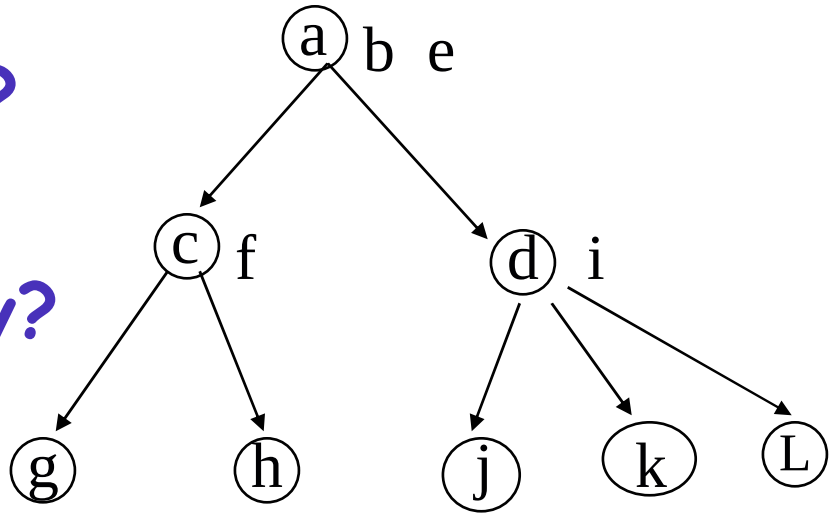
Evaluation

- Complete?

  Yes
- Time Complexity?

  O(b^d)
- Space Complexity?

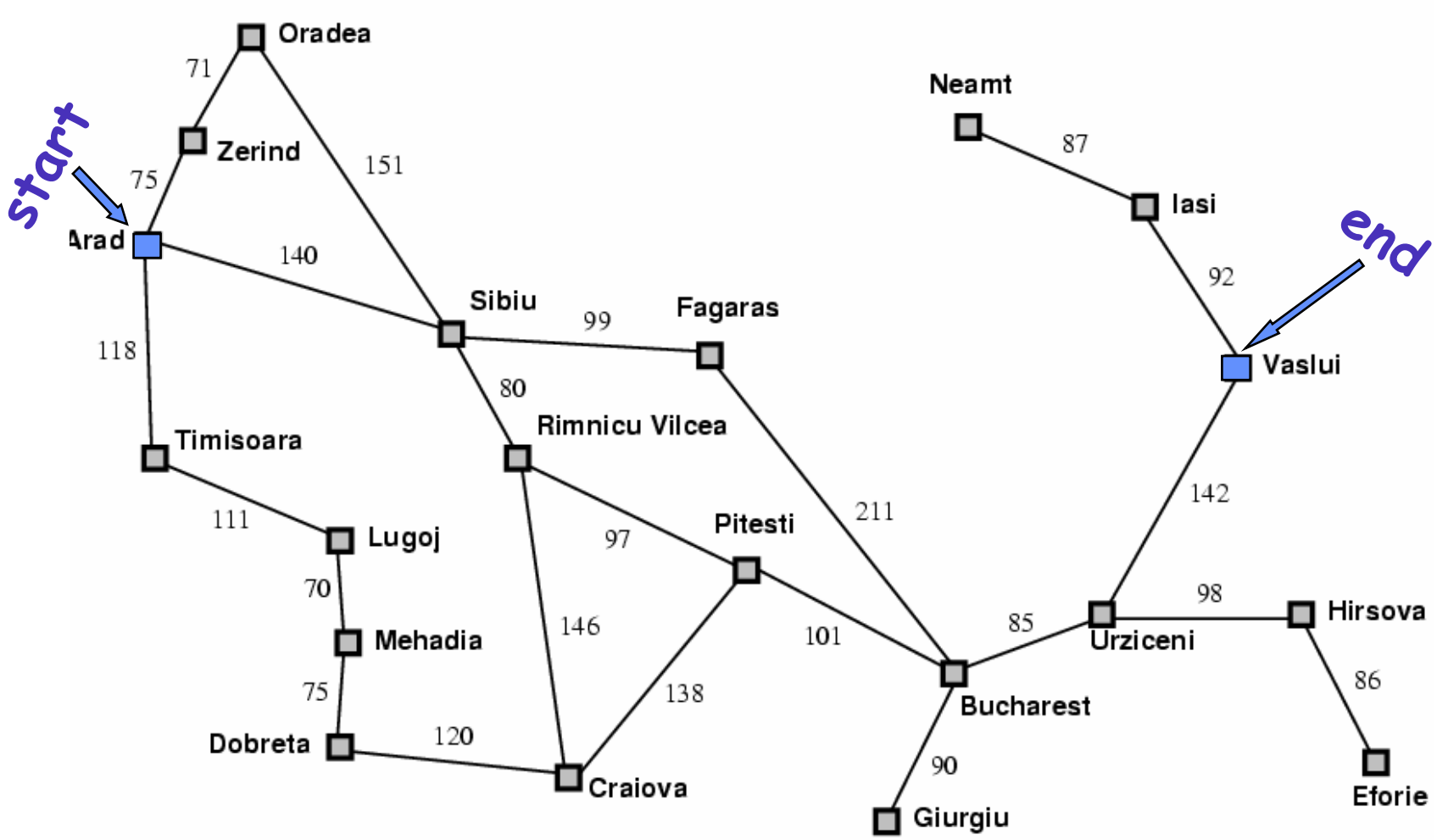
  O(d)



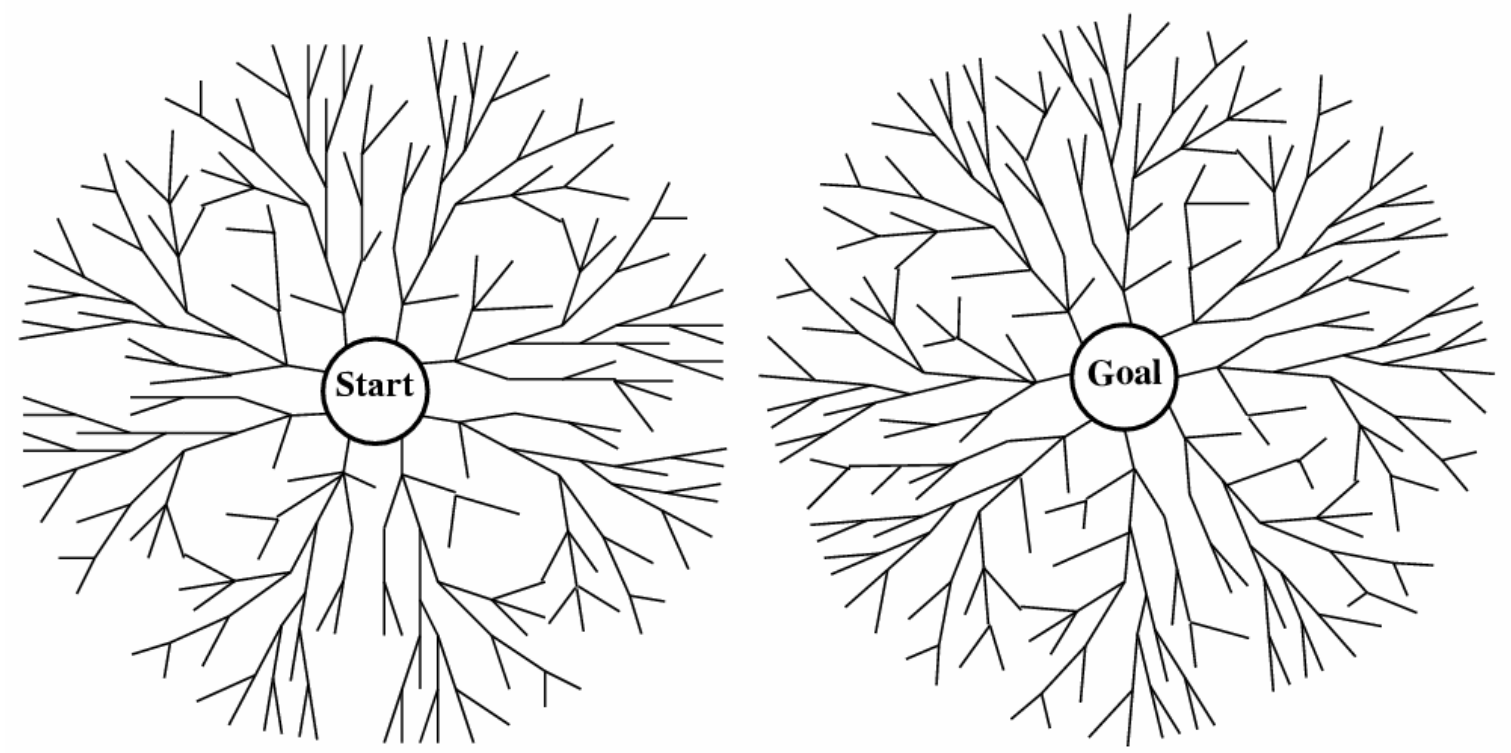
Cost of Iterative Deepening

b	ratio ID to DFS
2	3
3	2
5	1.5
10	1.2
25	1.08
100	1.02

Forwards *vs.* Backwards



vs. Bidirectional



Problem

All these methods are slow (blind)

Solution → add guidance ("heurstic estimate")
→ "informed search"