CMU 15-381

Search

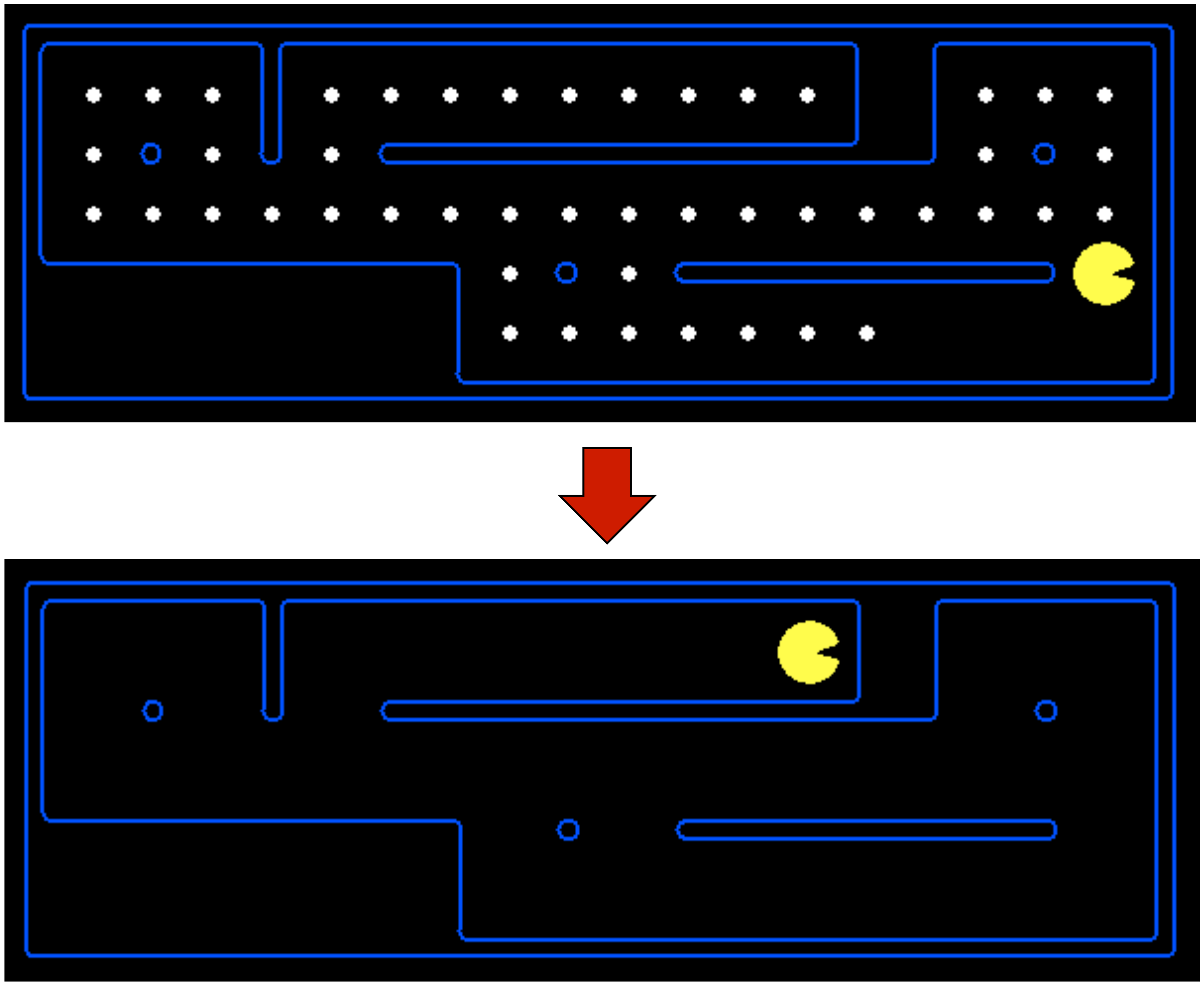
Teachers:
Drew Bagnell
Emma Brunskill (THIS TIME)

With thanks to Ariel Procaccia and other prior instructions for slides

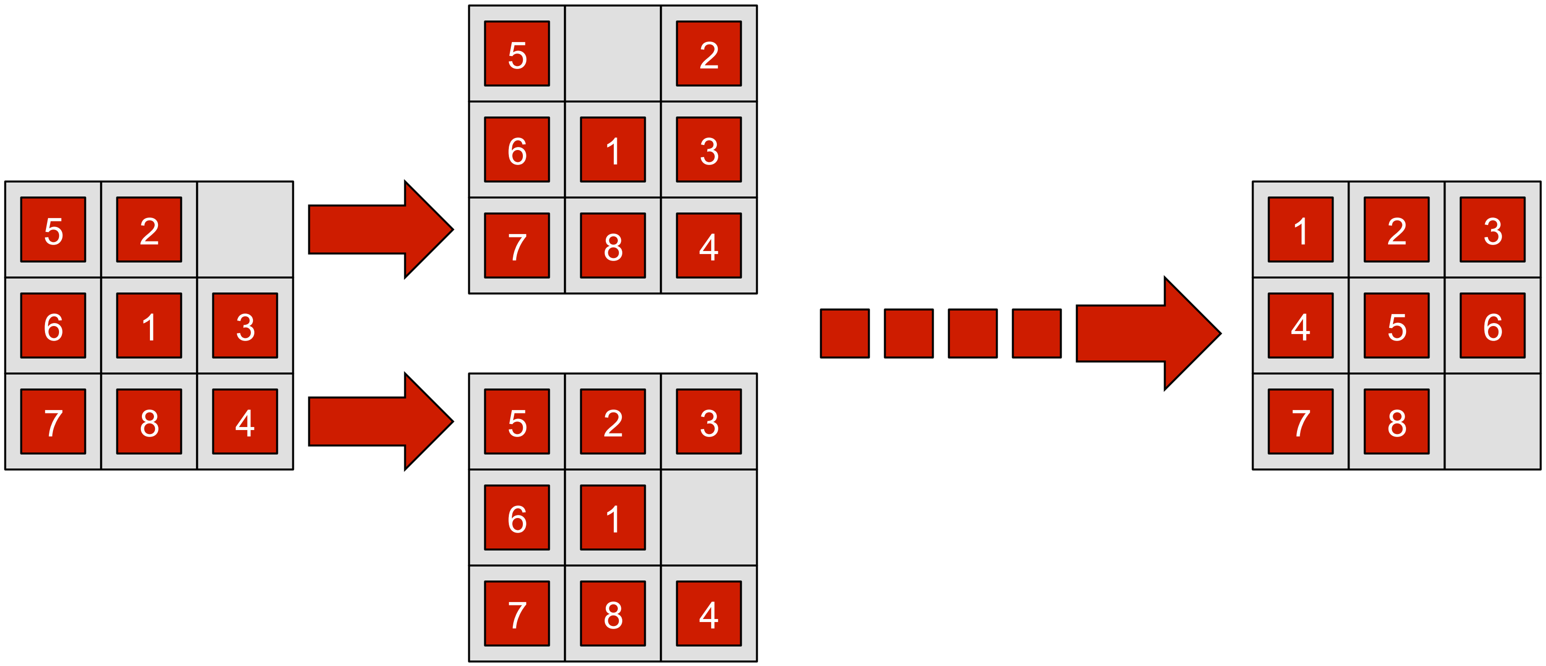
PATH OPTIMIZATION

- Last time:
  - Finding optimal configuration / solution (single point)
- Today
  - Finding minimal cost path
  - Trajectory of states

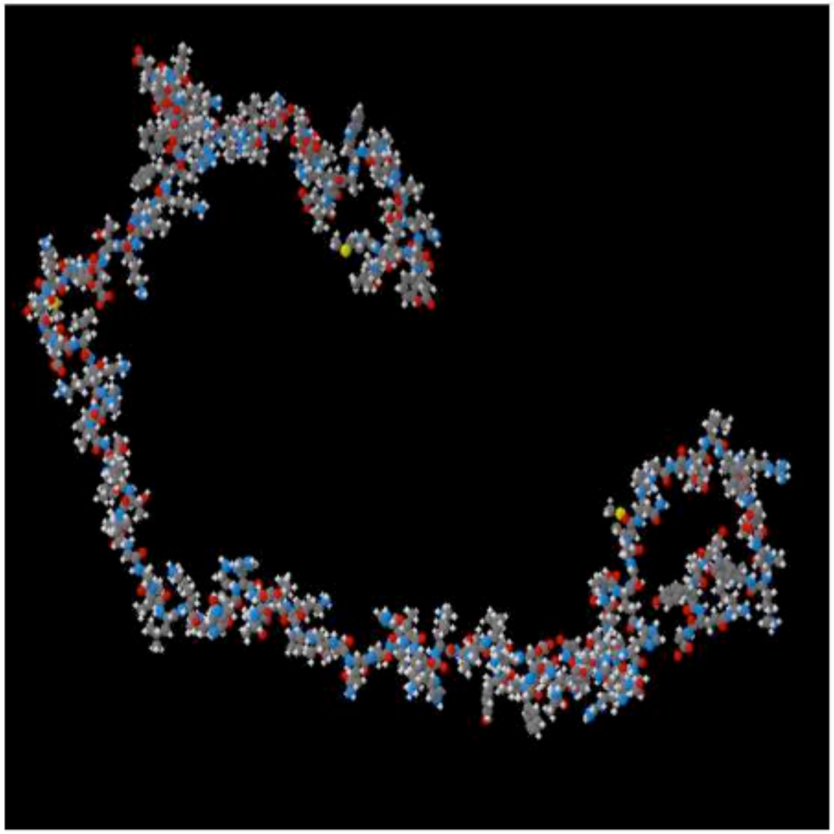
Search problems

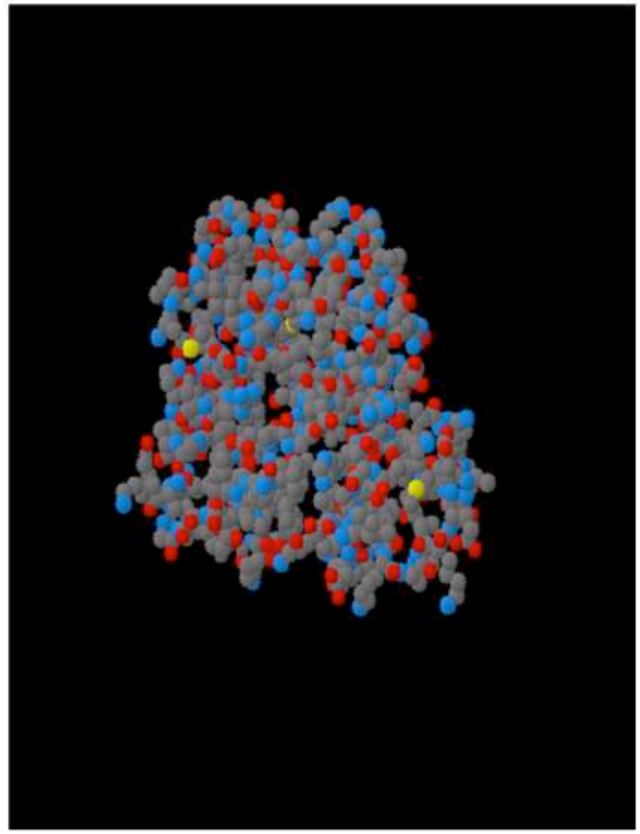


Search problems

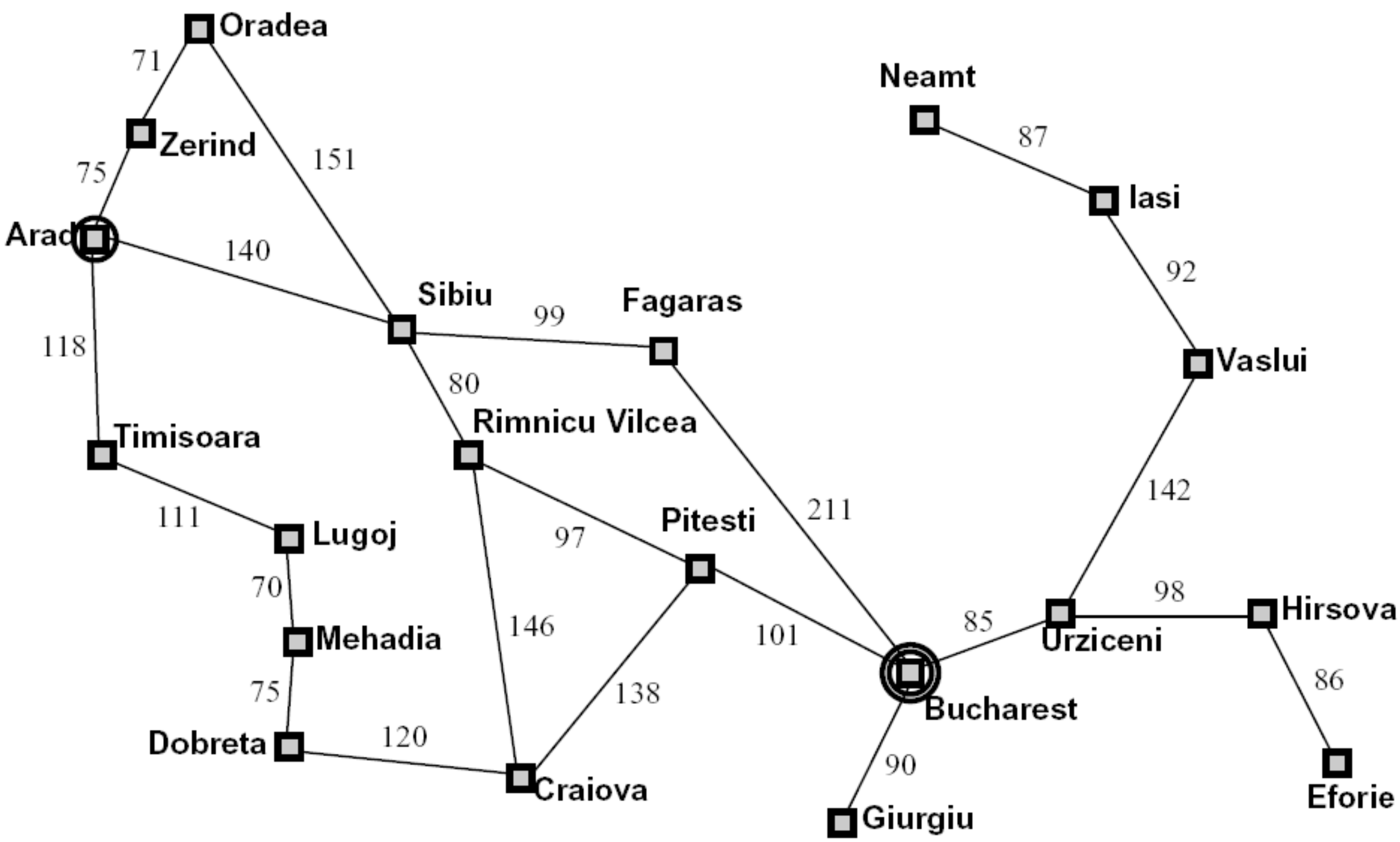


Search problems





Search problems



Overview

1. **Definition & evaluating search algorithms**
2. Types of search
3. A*: best informed search algorithm

Problem Definition

- Initial state
- Transition model & actions
- Successor states
- Step cost: cost of taking an action a in state s to reach state s'
- Goal states (or goal test)

Problem Definition

- Initial state
- Transition model & actions
- Successor states
- Step cost: cost of taking an action a in state s to reach state s’
- Goal states (or goal test)
- Objective: efficiently find minimal cost path from initial state to a goal state

Evaluating Search Approaches

Evaluating Search Approaches

- Completeness
  - Guaranteed to find soln if exists?
- Optimality
  - Find optimal solution?
- Time complexity
- Space complexity

Overview

1. Definition & evaluating search algorithms
2. Types of search
3. A*: best informed search algorithm

Overview

1. Definition & evaluating search algorithms
2. **Types of search**
3. A*: best informed search algorithm

Best-first search

- Find a path from initial state to goal state
- Relies on an evaluation function
- Nodes with best evaluation value are explored first
- Different evaluation functions induce different algorithms

Uninformed Search

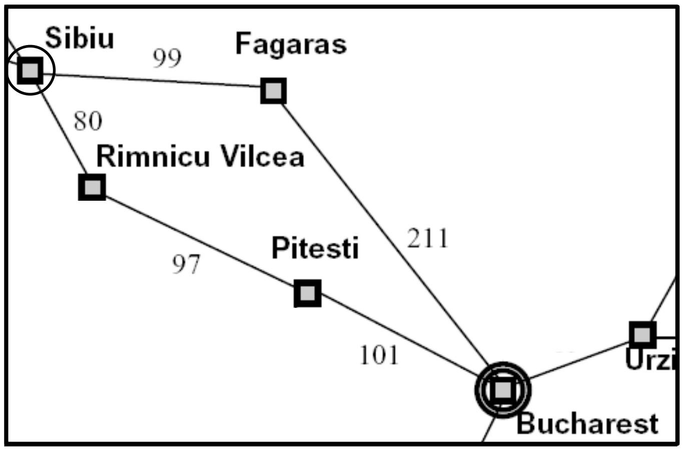
- Only use information from problem definition
- Examples
  - Depth first search
  - Breadth first search
  - Uniform cost search

Depth First Search

- Best first search with an evaluation function d(n)
- d(n) is number of nodes from start state
- If view as a search tree, starting with initial state, always expand deepest node in search tree that has successors
- Stop when hit goal state

Uniform cost search

- Best first search with evaluation function g(n)
- g(n) = work done so far
- Like Dijkstra's algorithm, but with goal



Uniform cost search: Click!

- Best first search with evaluation function g(n)
- g(n) = work done so far
- Is Uniform cost search:
- A) Not complete, not optimal
- B) Complete, not optimal
- C) Optimal but not complete
- D) Optimal and complete

Informed Search

- Use additional information about cost to reach a goal from node n: h(n)
- Known as a heuristic function
- Problem specific

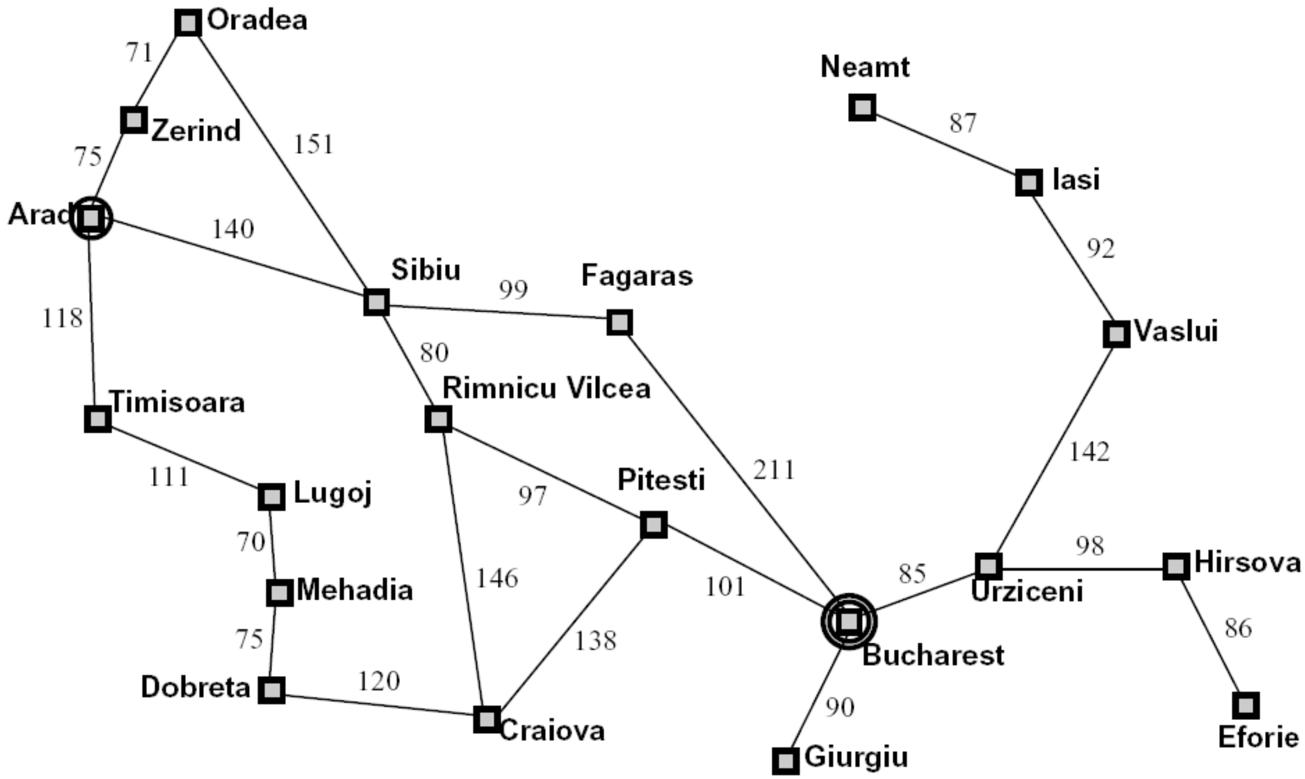
Straight-line Distance to Bucharest

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Greedy search

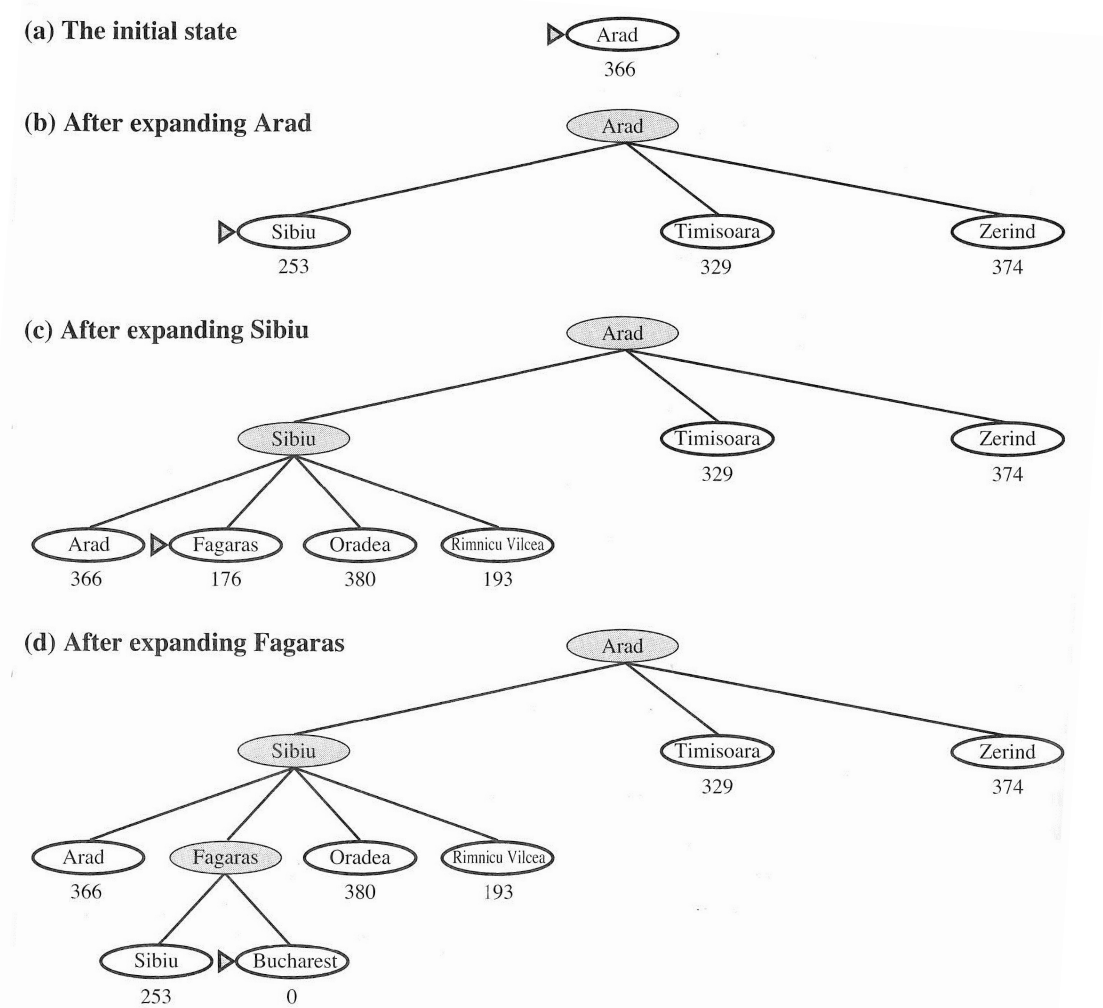
- Best-first search with evaluation function h(n)
- h(n) = estimated cost from node n to a goal state

City	Aerial dist
Arad	366
Sibiu	253
Rimnicu Vilcea	193
Fagaras	176
Pitesti	100



Greedy search: example

City	Aerial dist
Arad	366
Sibiu	253
Rimnicu Vilcea	193
Fagaras	176
Pitesti	100

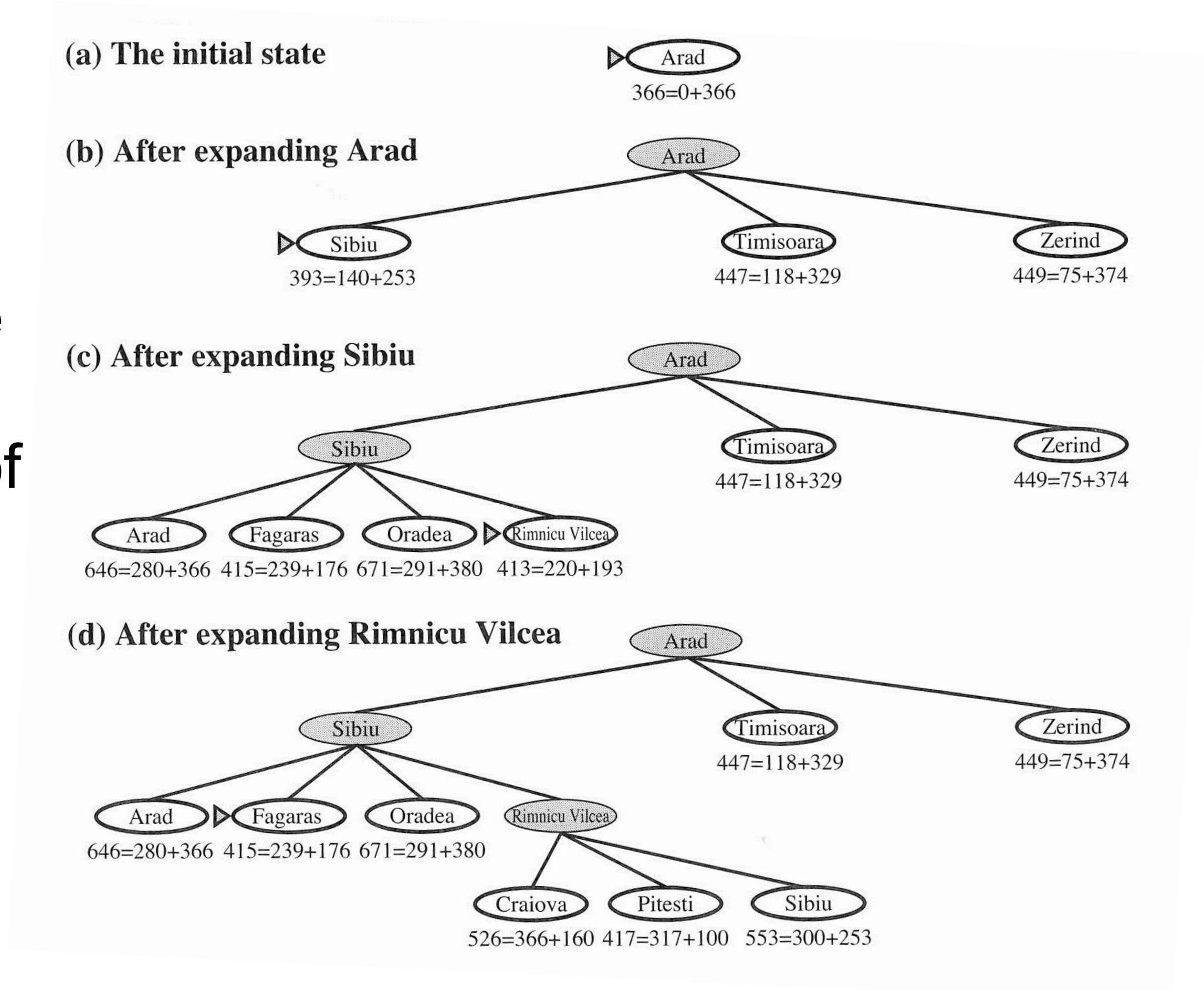


Overview

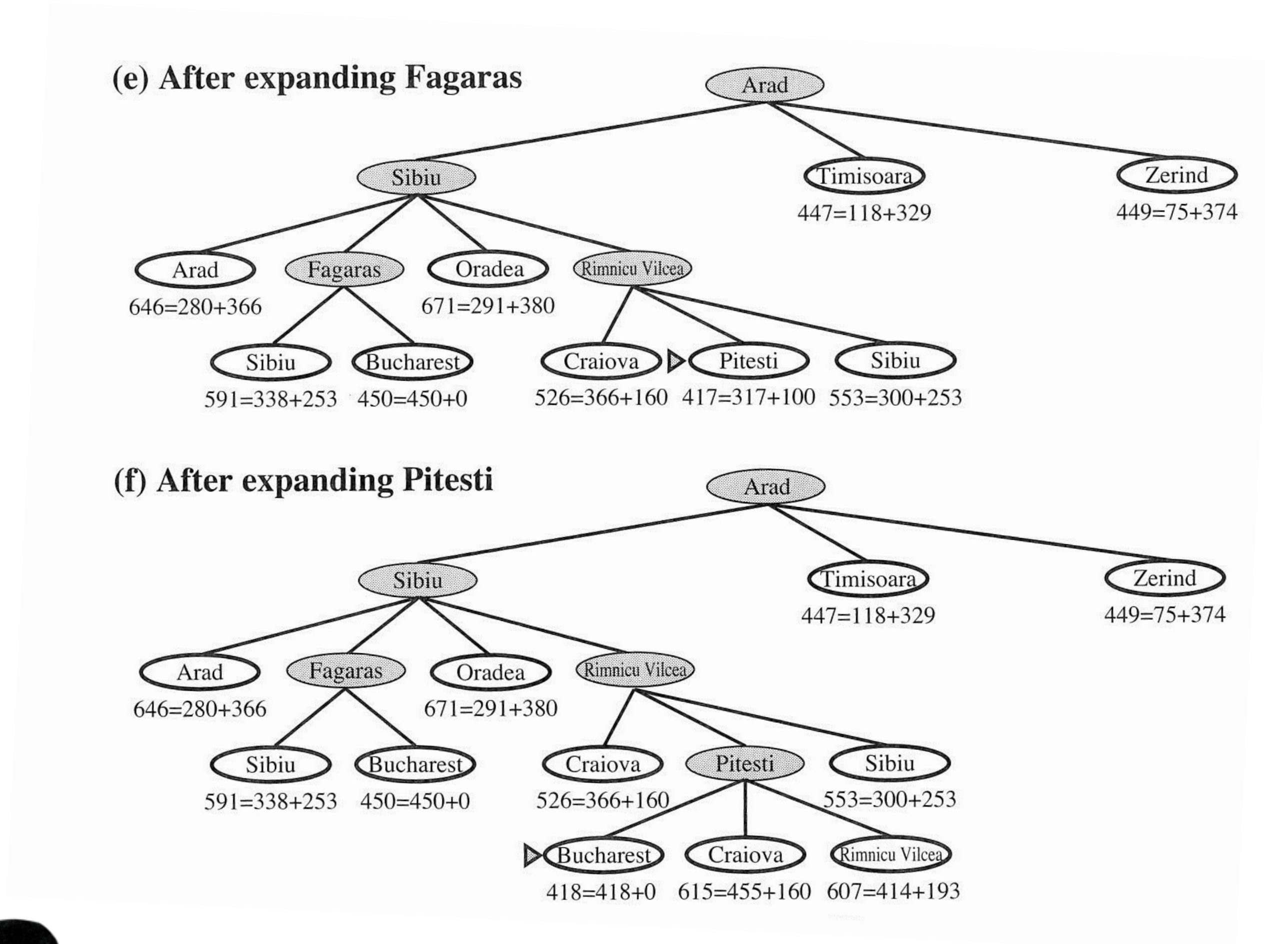
1. Definition & evaluating search algorithms
2. Types of search
3. **A*: best informed search algorithm**

A* Search

- Best-first search with
  f(n) = g(n)+h(n)
- g(n) = work done so far,
  h(n) = estimate of remaining work
- Let's click!

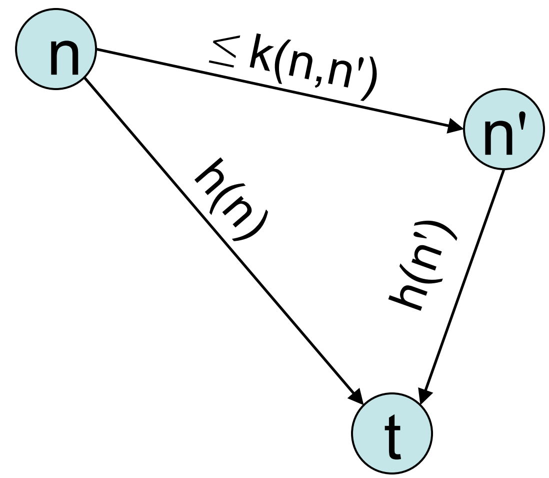


A* Search


(e) After expanding Fagaras
Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253
(f) After expanding Pitesti
Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
Sibiu
553=300+253
Bucharest
418=418+0
Craiova
615=455+160
Rimnicu Vilcea
607=414+193

Consistent heuristics

- k(n,n’) = cost of cheapest path between nodes n and n’
- h is **consistent** if for all n,n’, $h(n) \leq k(n,n') + h(n')$
- Line distance heuristic is consistent by the triangle inequality



Optimality of A*

- **Theorem:** If h is consistent, A* returns the min cost solution + never has to recalculate costs
- Proof:
  - Assume $h(n) \le k(n,n') + h(n')$
  - Values of $f(n)$ on a path are nondecreasing: if n' is the successor of n then
    $f(n) = g(n)+h(n) \le g(n)+k(n,n')+h(n') = g(n')+h(n') = f(n')$
  - When A* selects n for expansion, the optimal path to n has been found: otherwise there is a frontier node n' on optimal path to n that should be expanded first
  - $\Rightarrow$ Nodes expanded in nondecreasing $f(n)$
  - First goal state that is expanded must be optimal QED

ADMISSIBILITY

- $h^*(n)$ = cost of cheapest path from n to a goal
- h is **admissible** if for all nodes n, $h(n) \leq h^*(n)$

ADMISSIBILITY

- $h^*(n)$ = cost of cheapest path from n to a goal
- h is **admissible** if for all nodes n, $h(n) \leq h^*(n)$
- Consistency vs. admissibility: let's click!

A. All admissible h are consistent (but not vica versa)

B. All consistent h are admissible (but not vica versa)

C. All consistent h are admissible and vica versa

D. No relationship

ADMISSIBILITY

- $h^*(n)$ = cost of cheapest path from n to a goal
- h is **admissible** if for all nodes n, $h(n) \leq h^*(n)$
- Consistency vs. admissibility: let’s click!
- Informally: A* with admissible h is optimal when allowed to revisit nodes
- Heuristic for finding heuristic:
  - Look for admissible heuristic, it is likely consistent!

Optimality over other algs

- Any alg that is optimal given consistent heuristics will expand all nodes surely expanded by A* [Dechter and Pearl, Thm 8 on page 522]
- This is not true if the heuristic is merely admissible [Dechter an Pearl, pages 524-525]

R. Dechter and J. Pearl. Best-first search and the optimality of A*. Journal of the ACM 32:506-536, 1985 (link on course website)

More About Heuristics

- What’s the best possible heuristic?
- What is a simple heuristic that’s always admissible for any problem?

8-puzzle heuristics

- h_1: #tiles in wrong position
- h_2: sum of Manhattan distances of tiles from goal
- Both are admissible
- h_2 dominates h_1, i.e., $h_1(n) \le h_2(n)$ for all n

5	2	
6	1	3
7	8	4

Example state

1	2	3
4	5	6
7	8	

Goal state

A* and Heuristics

- Given two different consistent heuristics, will A* expand the same nodes? Click!

- A) Yes
- B) No

THE IMPORTANCE OF A GOOD HEURISTIC

- The following table gives the search cost of A* with the two heuristics, averaged over random puzzles, for various solution lengths

Length	$A^*(h_1)$	$A^*(h_2)$
16	1301	211
18	3056	363
20	7276	676
22	18094	1219
24	39135	1641

Summary

1. Know how to define a search problem
2. Types of search algorithms
3. Metrics for comparing search approaches
4. Be able to run A* and know under what types of heuristics it is the best possible search algorithm
5. Be able to create and select among heuristics