INFORMED SEARCH AND HEURISTICS

Summary of algorithms

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(bl)$	$O(bd)$
Optimal?	Yes	Yes	No	No	Yes

C^* is the path cost of the optimal solution

ϵ is the minimum step cost

C^*/ϵ then is the maximum depth of the solution

Today

- Reading
  - AIMA Chapter 3
- Goals
  - Informed search algorithms
  - Heuristics
  - Creating heuristics for problems

Informed search

- Use information beyond the problem to guide the search process to promising regions
- Define an evaluation function f(n) for each node n
  - estimates "desirability" of node
  - choose most desirable node from frontier (priority queue)
- Choices for f(n)
  - g(n) = distance from start node
  - h(n) = *estimate* of distance to goal node (heuristic function)

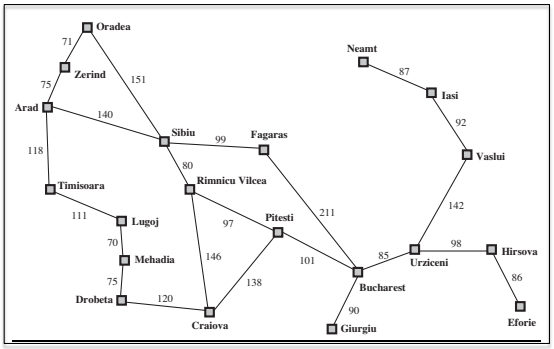
Informed search

- Recall for uninformed search
  - BFS: FIFO queue
  - DFS: LIFO queue
- For informed search
  - UCS: priority queue with f(n) = g(n)
  - Greedy Best-First: priority queue with f(n) = h(n)
  - A*: priority queue with f(n) = g(n) + h(n)

g(n) = distance from start h(n) = estimate to goal

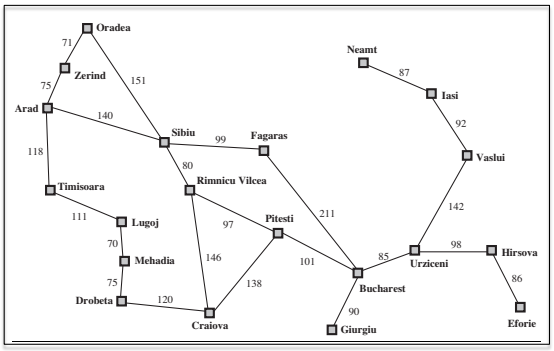
Heuristic functions

- An heuristic function is an *estimate* of cost from n to the goal



Heuristic functions

- An heuristic function is an *estimate* of cost from n to the goal
- Example: straight-line distance



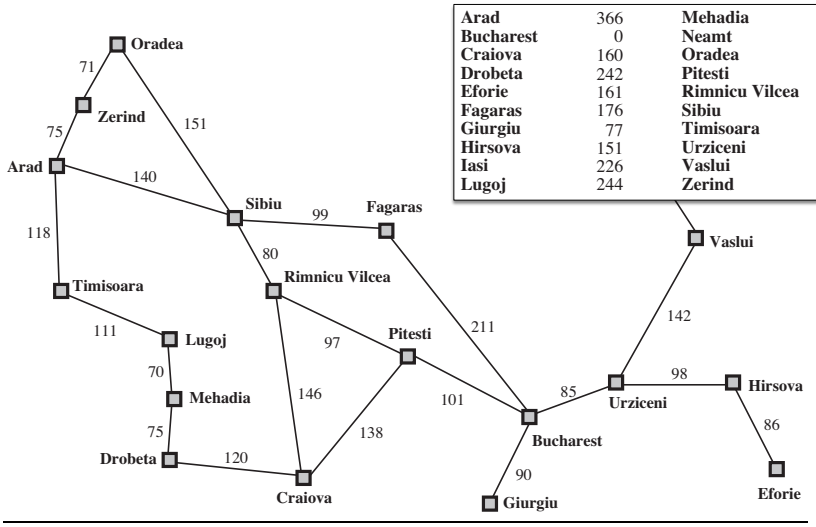
h(n) = straight-line distance

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

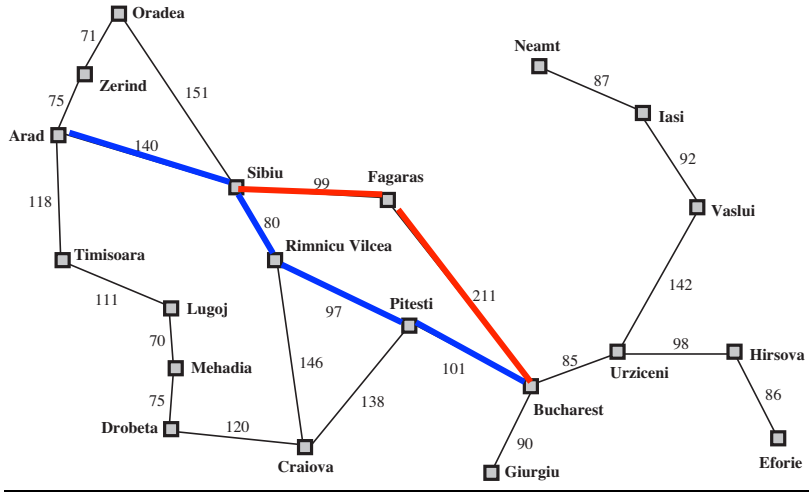
Greedy best-first search

- Define f(n) = h(n)
- Expand the node that seems closest to the goal
- What could possibly go wrong?

Greedy best-first search example



Greedy best-first search example

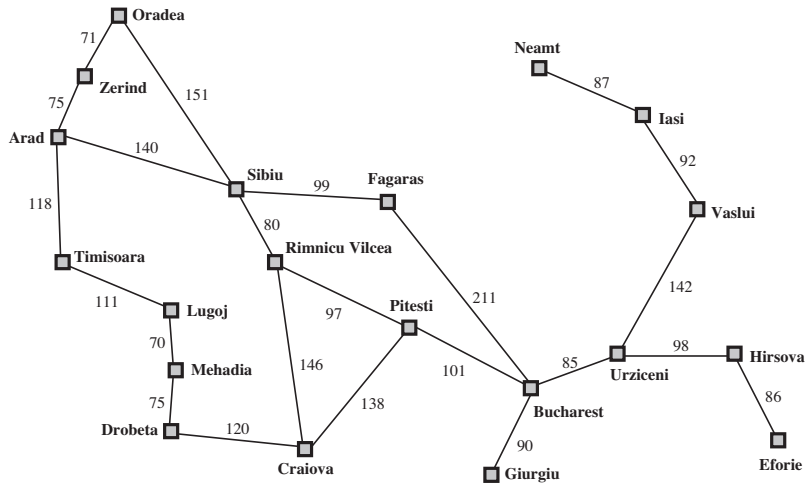


A* search

- Greedy best-first search considers only cost to goal
  - Can lead confidently to the (wrong) solution
- Idea: avoid expanding paths that are *already* expensive
- Evaluation function $f(n) = g(n) + h(n)$

g(n) = distance from start h(n) = estimate to goal

A* search example



A* search: conditions for optimality

- A heuristic h(n) is admissable if it never *overestimates* the cost to the goal

  $0 \leq h(n) \leq h^*(n)$ $h^*(n)$ is true cost to goal

- A heuristic h(n) is consistent if

  $h(n) \leq c(n, a, n') + h(n')$ n' is a successor

- Tree-search version of A* is optimal if h(n) is admissable
- Graph-search version of A* is optimal if h(n) is consistent

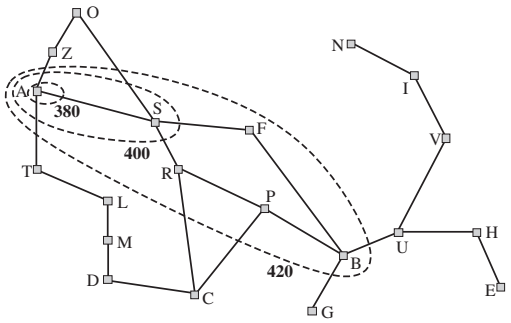
Optimality of A* search

1. If h(n) is consistent, then the values of f(n) along any path are non-decreasing (direct proof)

2. Whenever A* selects a node n for expansion, the optimal path to that node has been found (proof by contradiction)

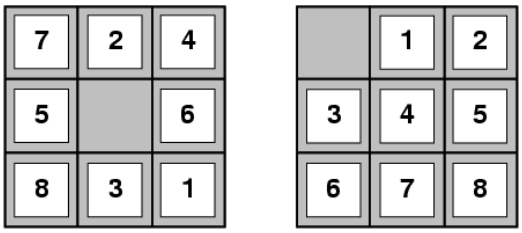
Properties of A* search

- A* expands
  - all nodes with f(n) < C*
  - some nodes with f(n) = C*
  - no nodes with f(n) > C*
- Optimally efficient
- Complete if finite number of nodes with f(n) ≤ C*



Creating admissable heuristic functions

- How do we construct a heuristic function that doesn't overestimate the cost to the goal?



- What are some ideas for heuristic functions?

Creating admissable heuristic functions

- Two-well used heuristics:
  - h_1 = number of misplaced tiles
  - h_2 = sum of the distances of the tiles from goal positions (Manhattan distance)

Goal

	Tiles out of place	Sum of distances out of place
2 8 3 1 6 4 7 5	5	6
2 8 3 1 4 7 6 5	3	4
2 8 3 1 6 4 7 5	5	6

Why are these admissable?

Creating admissable heuristic functions

- Often admissable heuristics are solutions to relaxed problems with fewer restrictions
- A tile can move from square A to square B if
  - A is horizontally or vertically adjacent to B
  - B is blank
- Pick up tiles and place in the correct spot
  - Induces h_1 heuristic, i.e. number tiles out of place

	Number nodes expanded for solution depth d		
	d = 4	d = 8	d= 12
IDS	112	6384	3.6 million
$A^*(h_1)$	13	39	227

Creating admissable heuristic functions

- Often admissable heuristics are solutions to relaxed problems with fewer restrictions
- A tile can move from square A to square B if
  - ~~A is horizontally or vertically adjacent to B~~
  - ~~B is blank~~

  ← relax the rules
- Induces h_1 heuristic, i.e. number of tiles out of place
  - Allows you to pick up the tiles and place in the correct spot

	Number nodes expanded for solution depth d		
	d = 4	**d = 8**	**d= 12**
IDS	112	6384	3.6 million
$A^*(h_1)$	13	39	227

Creating admissable heuristic functions

- Often admissable heuristics are solutions to relaxed problems with fewer restrictions
- A tile can move from square A to square B if
  - A is horizontally or vertically adjacent to B
  - ~~B is blank~~

  ← relax the rules
- Induces h_2 heuristic, i.e. sum of distances to goal position
  - Allows you to move a tile to an adjacent square

	Number nodes expanded for solution depth d		
	d = 4	**d = 8**	**d= 12**
IDS	112	6384	3.6 million
$A^*(h_1)$	13	39	227
$A^*(h_2)$	12	25	73

Creating admissable heuristic functions

expands fewer nodes

	Number nodes expanded for solution depth d		
	d = 4	**d = 8**	**d= 12**
IDS	112	6384	3.6 million
$A^*(h_1)$	13	39	227
$A^*(h_2)$	12	25	73

Creating admissable heuristic functions

- Some heuristics are better than others
  - If $h_1(n) \leq h_2(n) \leq h^*(n)$ then h2 dominates h1
  - Manhattan distance dominates tiles out of place
  - A-star search using h_2 will never expand more nodes than A-star search using h_1
  - Can combine admissable heuristics using max

Informed search summary

- Uniform-cost search considers only the cost from the start node
- Greedy best-first search considers only the (estimate of the) cost to the goal node
  - Confidently heads straight to the (wrong) solution
- A^* search considers both cost from start and estimate to goal
  - A^* is optimal with admissable/consistent heuristic
- A good heuristic is the key
  - Consider solutions to relaxed problems