Chapter 3 Solving Problems by Searching
3.5 –3.6 Informed (heuristic) search strategies

CS4811 - Artificial Intelligence

Nilufer Onder
Department of Computer Science
Michigan Technological University

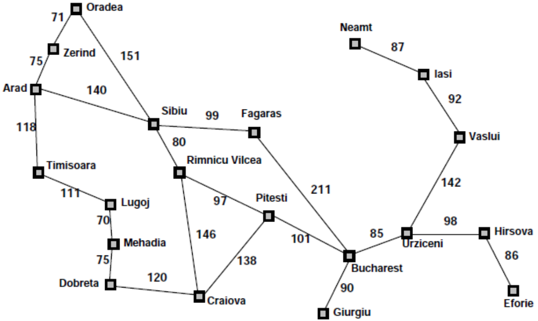
Outline

- Best-first search
  - Greedy search
  - A^* search
- Heuristics
- (Iterative deepening A^* search)

Best-first search

- Remember that the *frontier* contains the unexpanded nodes
- Idea: use an *evaluation function* for each node
  (the evaluation function is an estimate of "desirability")
- Expand the most desirable unexpanded node
- Implementation:
  Frontier is a queue sorted in decreasing order of desirability
- Special cases:
  - Greedy search
  - A* search

Romania with step costs in km



Straight–line distance
to Bucharest

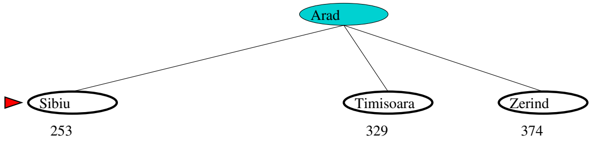
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy search

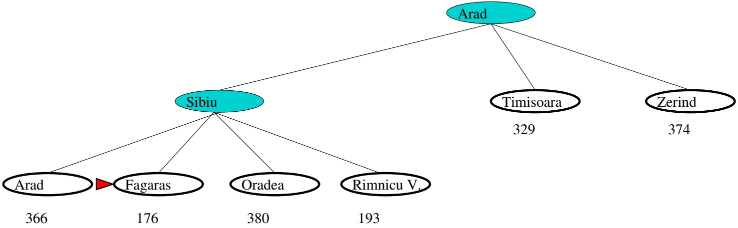
- Evaluation function
  $h(n)$ = estimate of cost from n to the closest goal
  h is the *heuristic* function
- E.g., $h_{\mathrm{SLD}}(n)$ = straight-line distance from n to Bucharest
- Greedy search expands the node that appears to be closest to the goal

Greedy search example

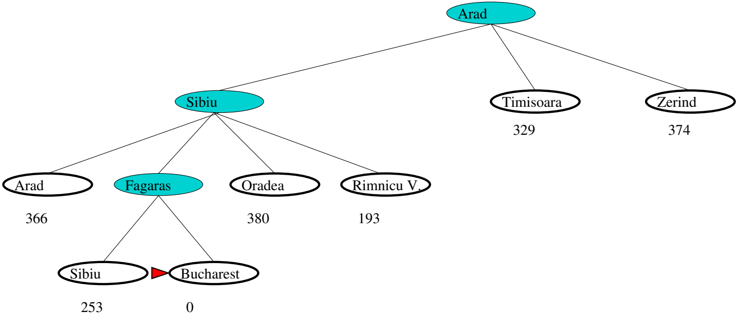
After expanding Arad



After expanding Sibiu



After expanding Fagaras



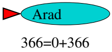
Properties of greedy search

- *Complete:* No — can get stuck in loops, e.g.,
  Iasi $\rightarrow$ Neamt $\rightarrow$ Iasi $\rightarrow$ Neamt $\rightarrow$
  Complete in finite space with repeated-state checking
- *Time:* $O(b^m)$, but a good heuristic can give dramatic improvement
- *Space:* $O(b^m)$ (keeps every node in memory)
- *Optimal:* No

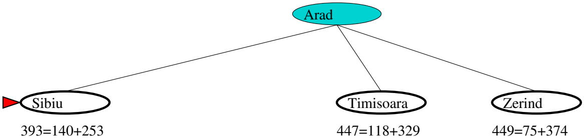
A* search

- Idea: avoid expanding paths that are already expensive
- *Evaluation function* $f(n) = g(n) + h(n)$
  - $g(n)$ = cost so far to reach n
  - $h(n)$ = estimated cost to goal from n
  - $f(n)$ = estimated total cost of path through n to goal
- A* search uses an *admissible* heuristic
  - if h is an admissible heuristic then $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from n.
  - Also require $h(n) \geq 0$, so $h(G) = 0$ for any goal G.
  - An admissible heuristic is allowed to underestimate, but can never overestimate cost.
  - E.g., $h_{\mathrm{SLD}}(n)$ never overestimates the actual road distance.

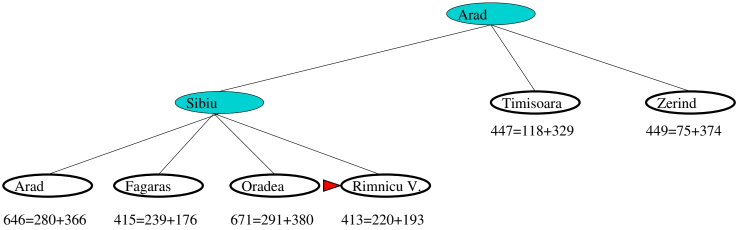
A* search example



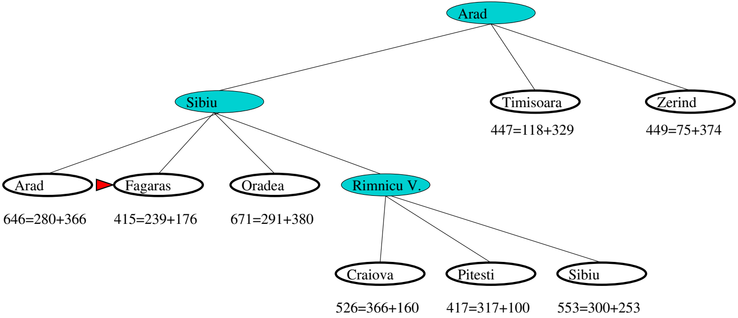
After expanding Arad



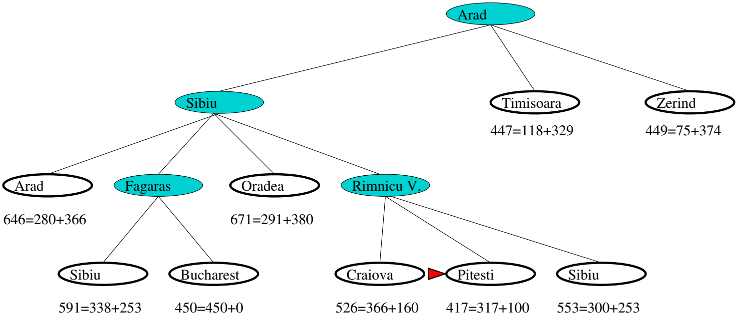
After expanding Sibiu



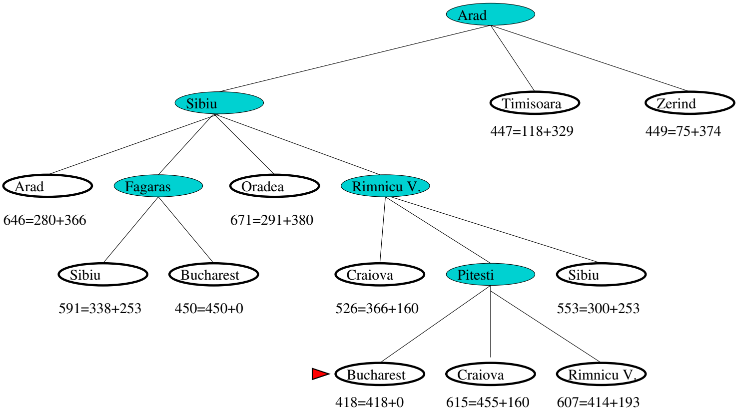
After expanding Rimnicu Vilcea



After expanding Fagaras



After expanding Pitesti

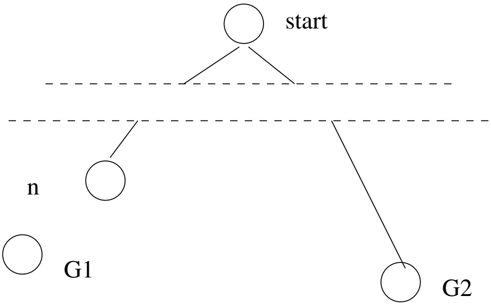


Optimality of A*

Theorem: A* search is optimal.

Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G_1.

Proof for the optimality of A*



$$\begin{aligned} f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\ &> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\ &\geq f(n) && \text{since } h \text{ is admissible} \end{aligned}$$

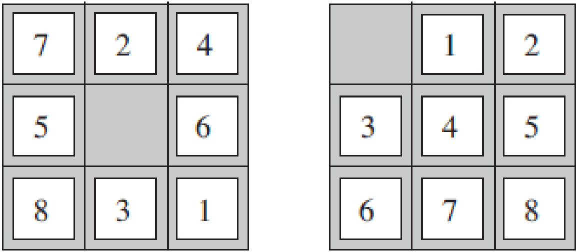
Since $f(G_2) > f(n)$, A* will never select G_2 for expansion

Properties of A*

- *Complete:* Yes, unless there are infinitely many nodes with $f \leq f(G)$
- *Time:* Exponential in (relative error in h $\times$ length of solution)
- *Space:* Keeps all nodes in memory
- *Optimal:* Yes—cannot expand f_{i+1} until f_i is finished
  - A* expands all nodes with $f(n) < C^*$
  - A* expands some nodes with $f(n) = C^*$
  - A* expands no nodes with $f(n) > C^*$

Admissible heuristics

E.g., for the 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)

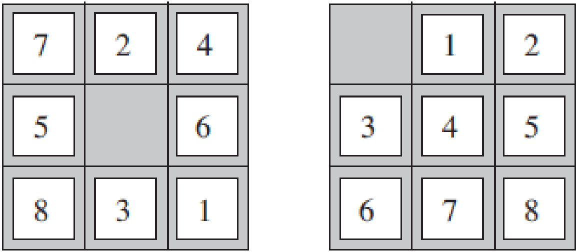


Start State Goal State

$h_1(S) = ??$
$h_2(S) = ??$

Admissible heuristics

E.g., for the 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

$h_1(S) = 8$
$h_2(S) = 3+1+2+2+3+2+2+3 = 18$

Dominance

A "better" heuristic is one that minimizes the *effective branching factor, b^**.

If $h_2(n) \geq h_1(n)$ for all n (both admissible)
then h_2 *dominates* h_1 and is better for search

Typical search costs:

$d = 12$	IDS $=$ 3,644,035 nodes	$b^* = 2.78$
	$\mathrm{A}^*(h_1) =$ 539 nodes	$b^* = 1.42$
	$\mathrm{A}^*(h_2) =$ 113 nodes	$b^* = 1.24$
$d = 24$	IDS $\approx$ 54,000,000,000 nodes	
	$\mathrm{A}^*(h_1) =$ 39,135 nodes	$b^* = 1.48$
	$\mathrm{A}^*(h_2) =$ 1,641 nodes	$b^* = 1.26$

Relaxed problems

- Admissible heuristics can be derived from the exact solution cost of a *relaxed* version of the problem
- If the rules of the 8-puzzle are relaxed so that a tile can move "anywhere", then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to "any adjacent square", then $h_2(n)$ gives the shortest solution
- Key point: the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

Iterative Deepening A* (IDA*)

- Idea: perform iterations of DFS. The cutoff is defined based on the f-cost rather than the depth of a node.
- Each iteration expands all nodes inside the contour for the current f-cost, peeping over the contour to find out where the contour lies.

Summary

- Heuristic search algorithms
- Finding good heuristics for a specific problem is an area of research
- Think about the time to compute the heuristic

Sources for the slides

- AIMA textbook (3^{rd} edition)
- AIMA slides (http://aima.cs.berkeley.edu/)