CSE 604

Artificial Intelligence

Chapter 3 (part 2): Heuristic Search

Adapted from slides available in Russell & Norvig's textbook webpage

Dr. Ahmedul Kabir

IIT
University of Dhaka

Outline

- Heuristics
- Best-first search
- Greedy best-first search
- A^* search
- More on heuristics

Definition of heuristics

- A heuristic technique (/hjuːˈrɪstɪk/; Ancient Greek: εὑρίσκω, "find" or "discover"), often called simply a heuristic, is any approach to problem solving, learning, or discovery that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals.

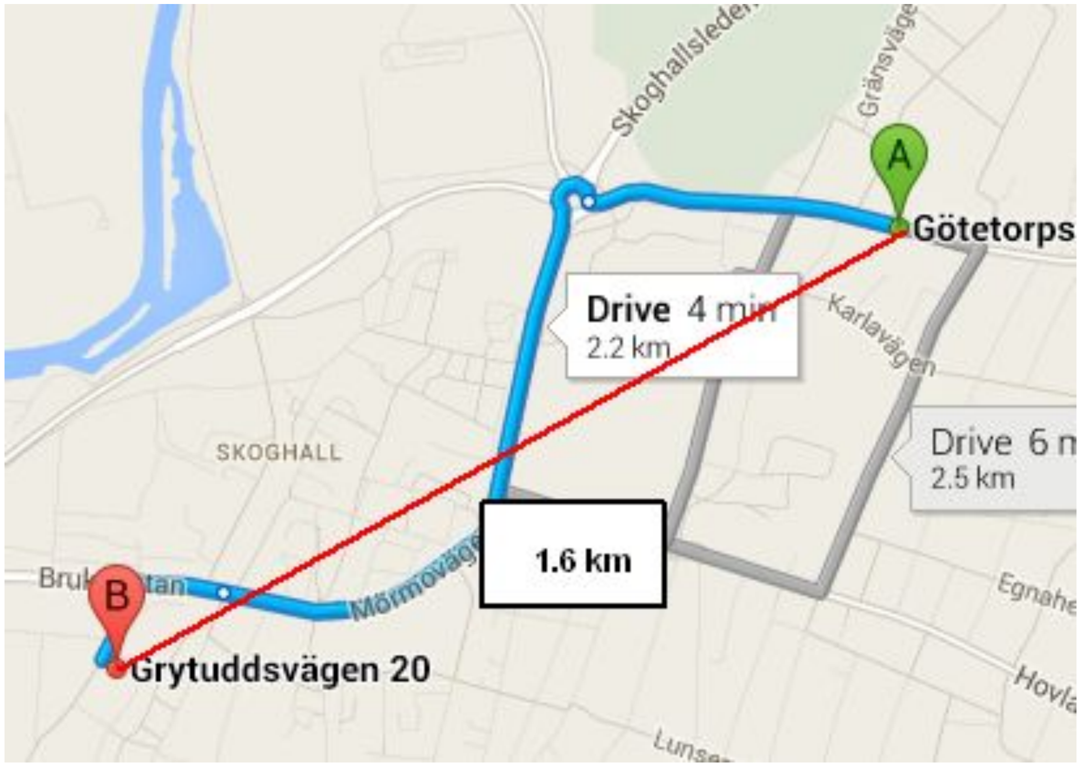
- Heuristics can be mental shortcuts that ease the cognitive load of making a decision.

- Examples of this method include using a rule of thumb, an educated guess, or common sense.

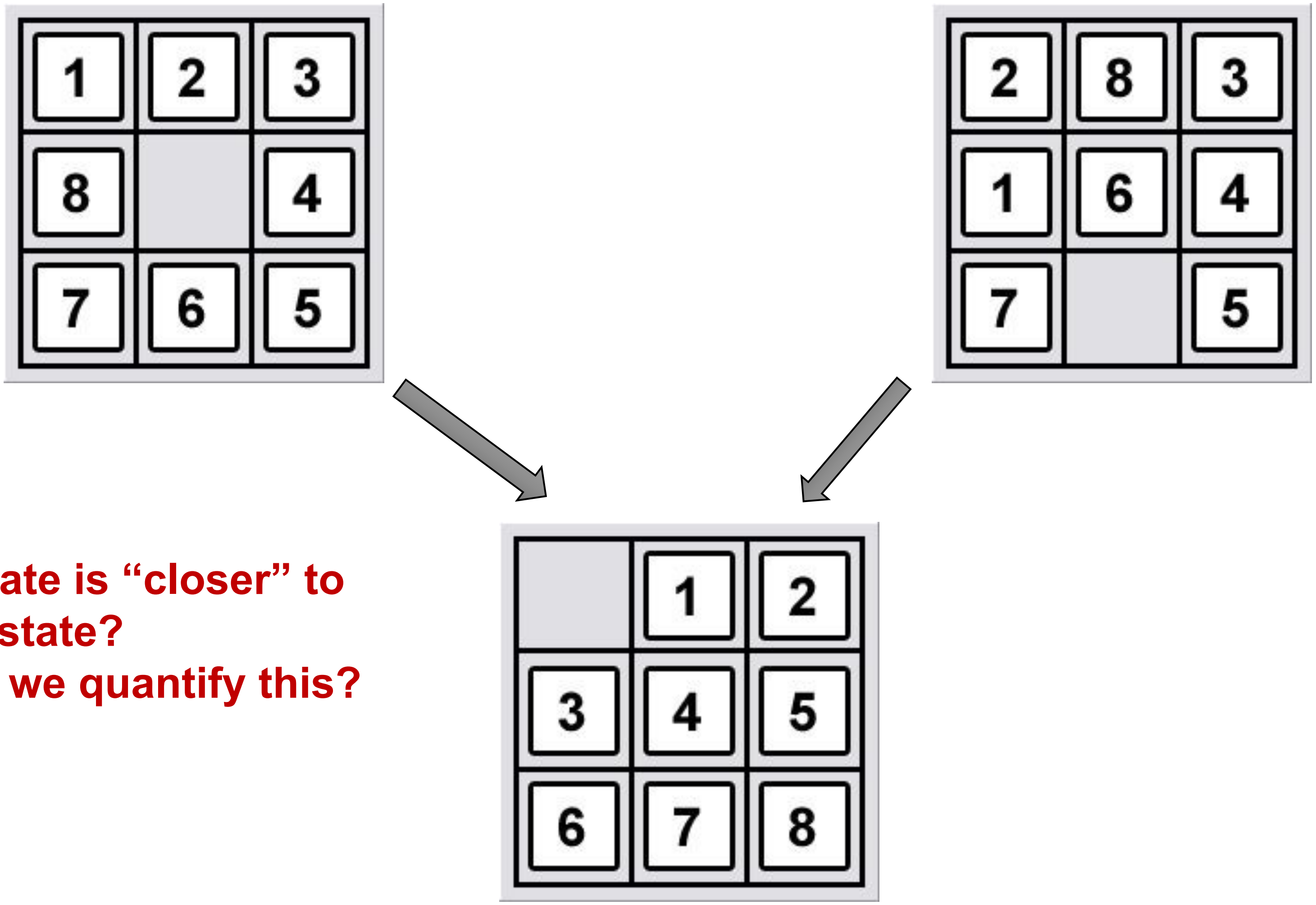
https://en.wikipedia.org/wiki/Heuristic

Example: Driving from A to B

- The straight line distance is a heuristic to estimate the driving distance



Example: 8-puzzle problem



Which state is "closer" to the goal state?
How can we quantify this?

Best-first search

- Idea: use an evaluation function *f(n)* for each node
  - estimate of "desirability“

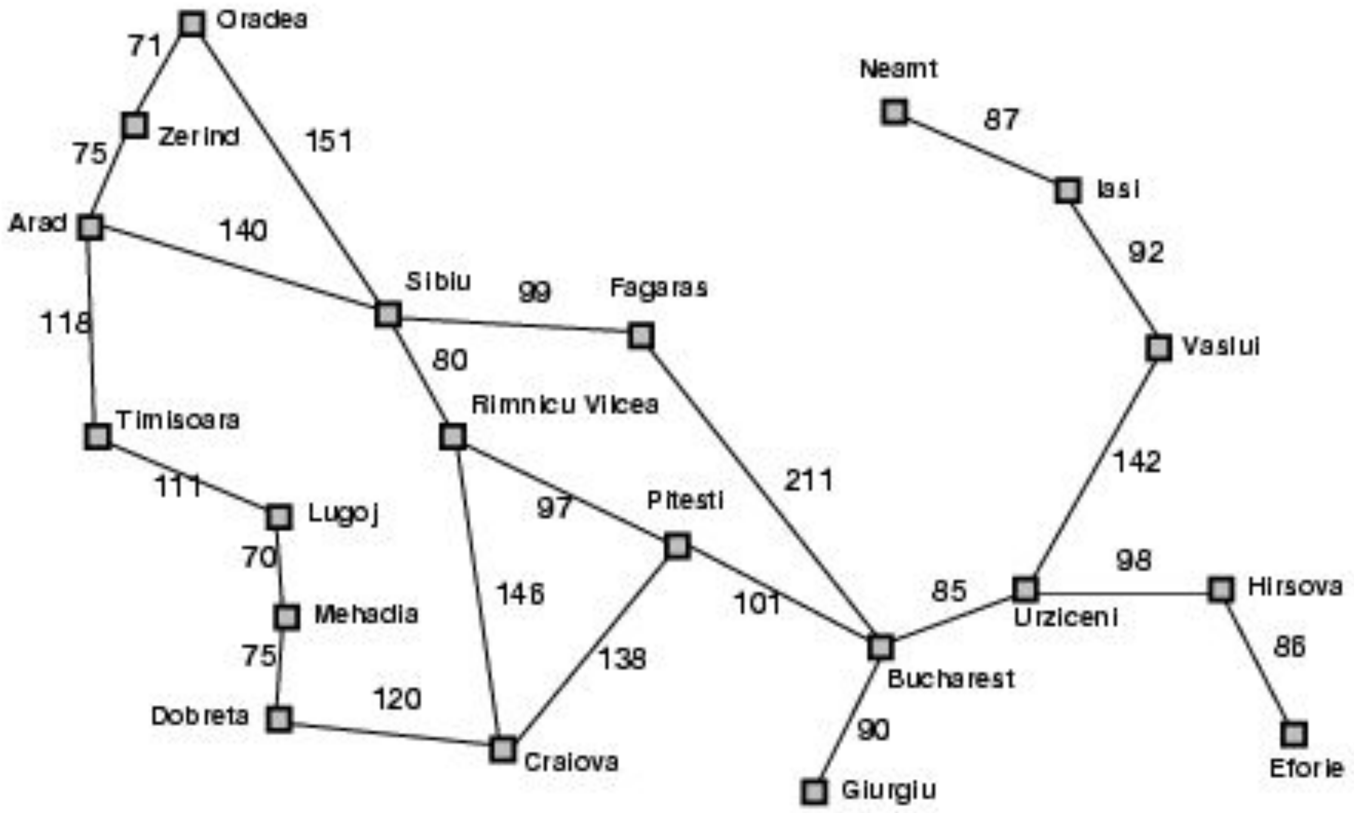
  □ Expand most desirable unexpanded node

- <u>Implementation</u>:

  Order the nodes in fringe in decreasing order of desirability

- Special cases:
  - Greedy best-first 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	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search

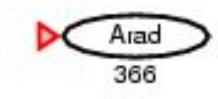
- Evaluation function $f(n) = h(n)$ (heuristic)

  = estimate of cost from n to *goal*

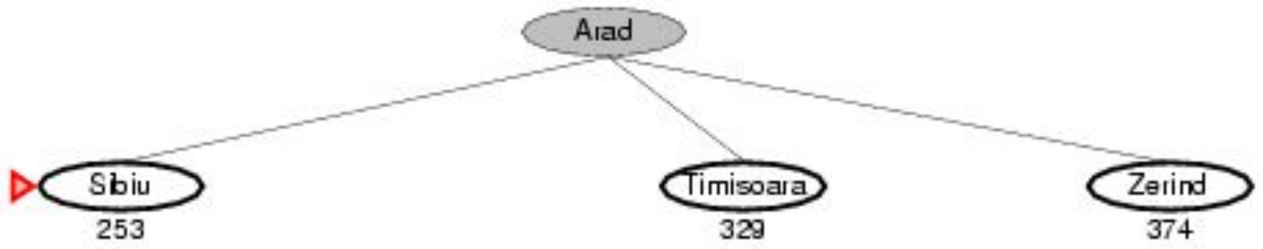
- e.g., $h_{SLD}(n)$ = straight-line distance from n to Bucharest

- Greedy best-first search expands the node that appears to be closest to goal

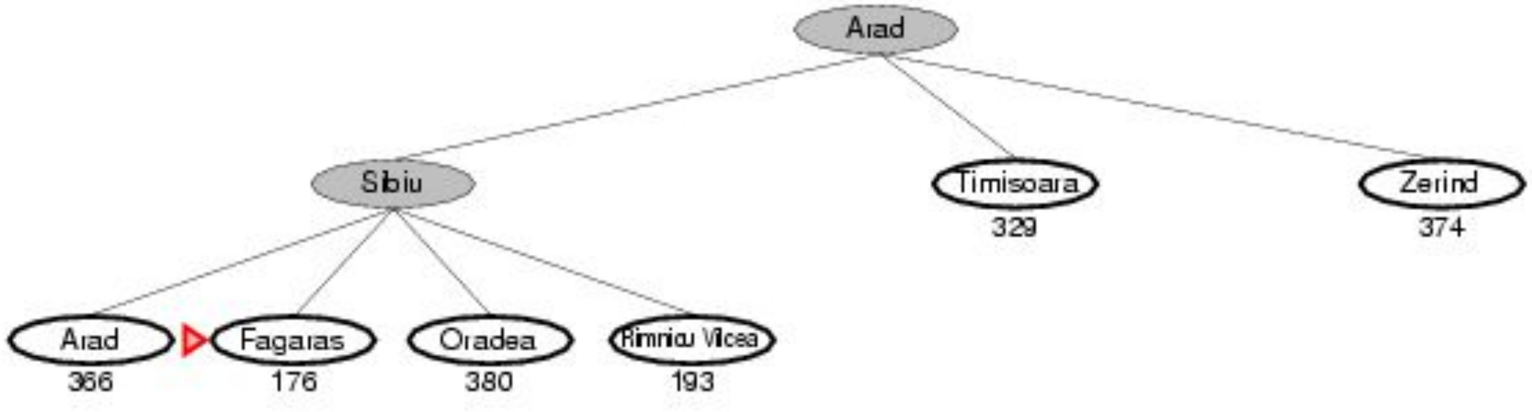
Greedy best-first search example



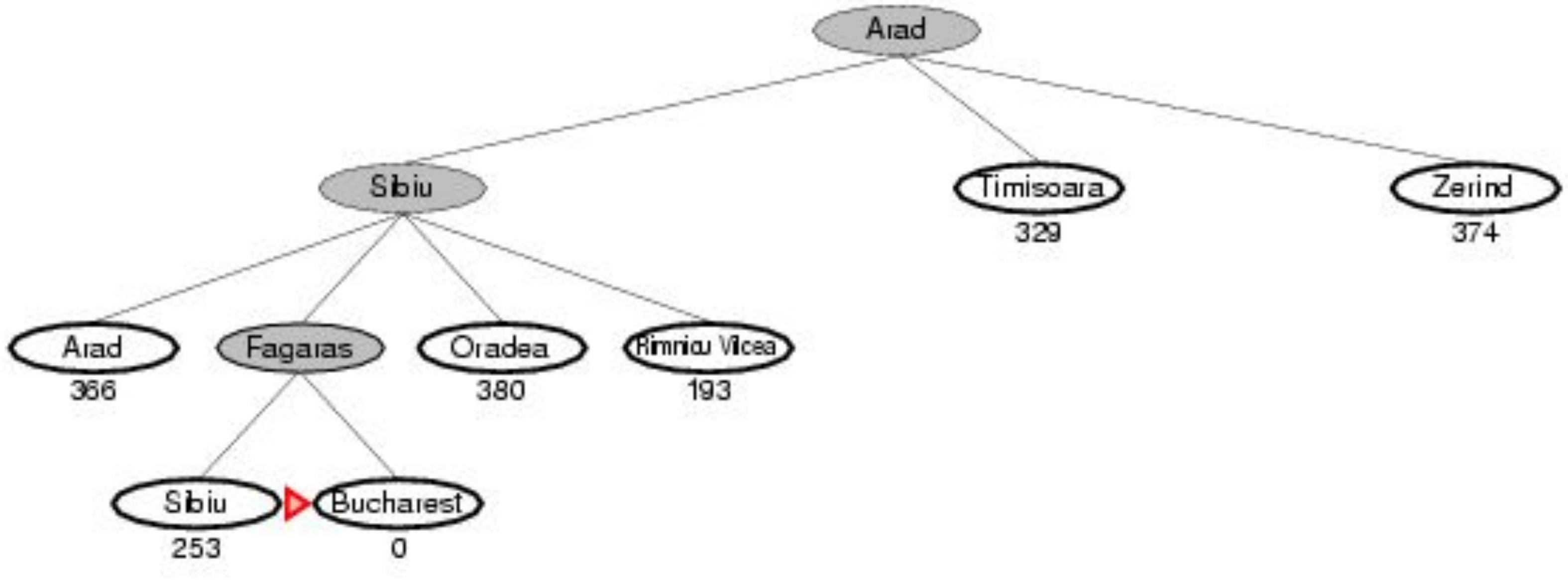
Greedy best-first search example



Greedy best-first search example



Greedy best-first search example



Properties of greedy best-first search

- Complete? No – can get stuck in loops, e.g., when going from Iasi to Fagars:

  Iasi   Neamt   Iasi   Neamt

- Time? $O(b^m)$, but a good heuristic can give dramatic improvement

- Space? $O(b^m)$ -- keeps all nodes in memory

- Optimal? No

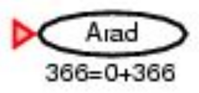
A^* search

- Idea: avoid expanding paths that are already expensive

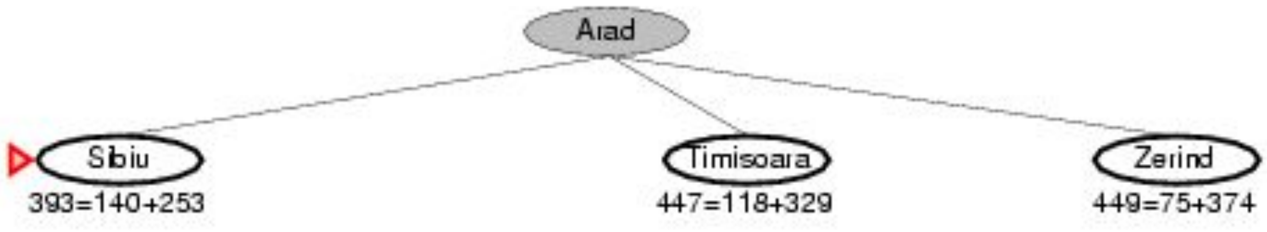
- Evaluation function $f(n) = g(n) + h(n)$

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

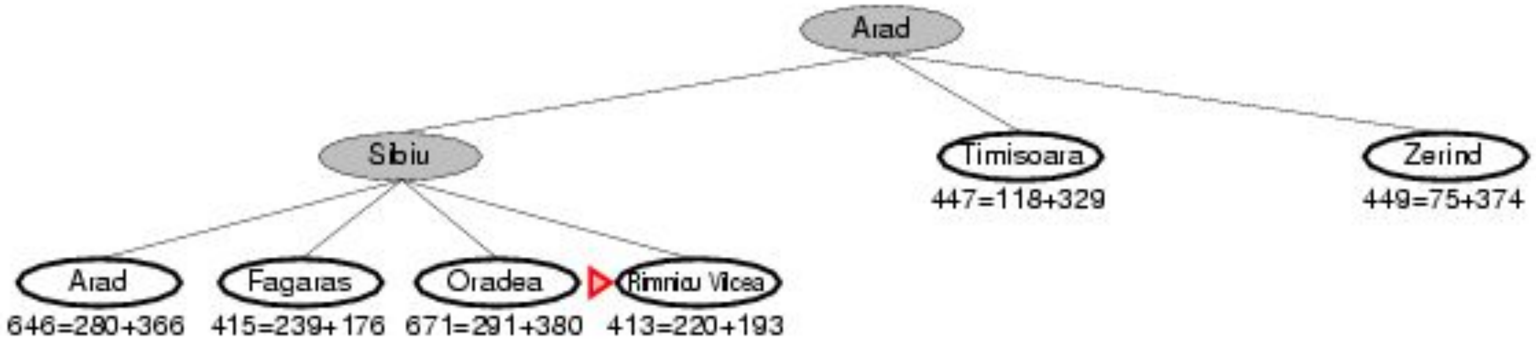
A* search example



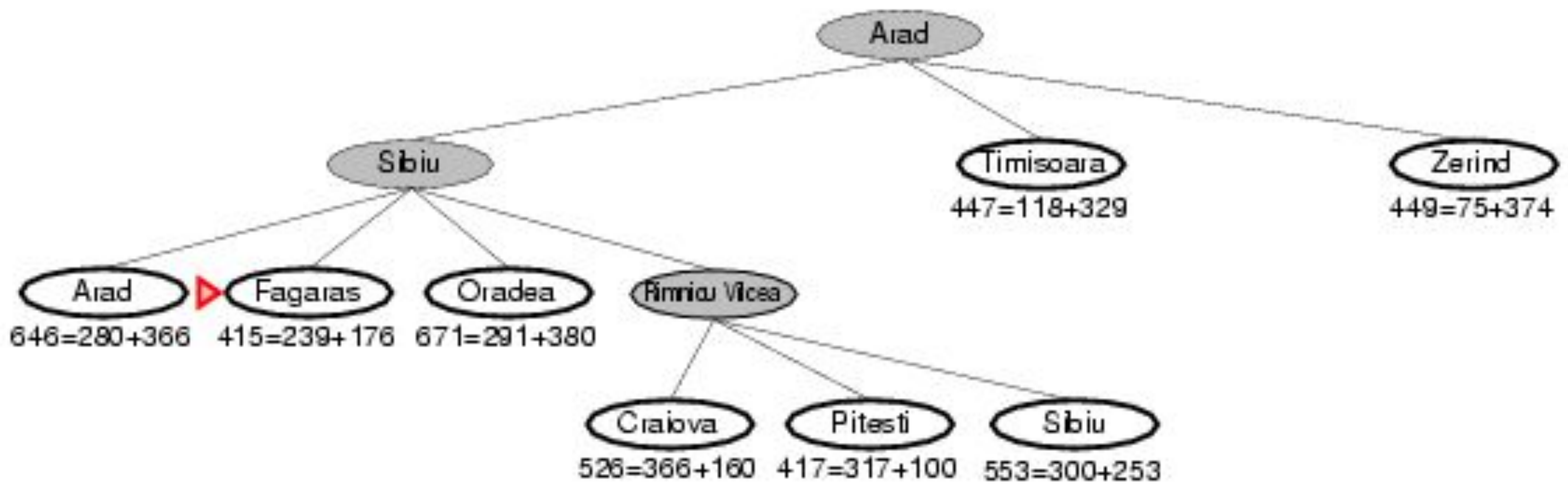
A* search example



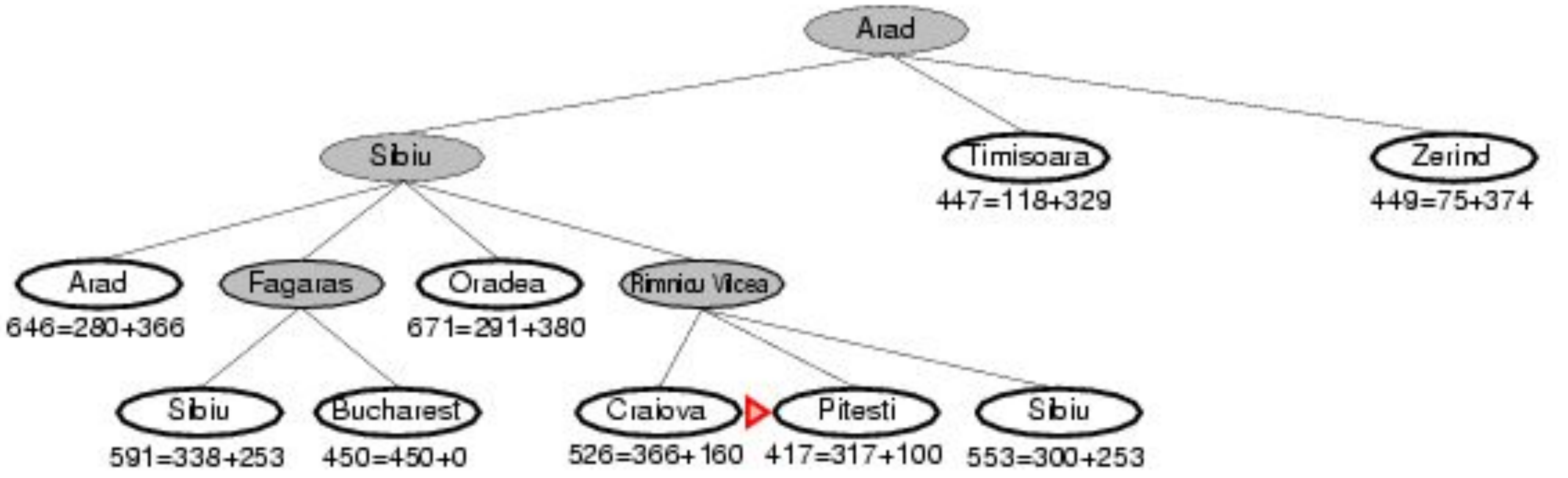
A^* search example



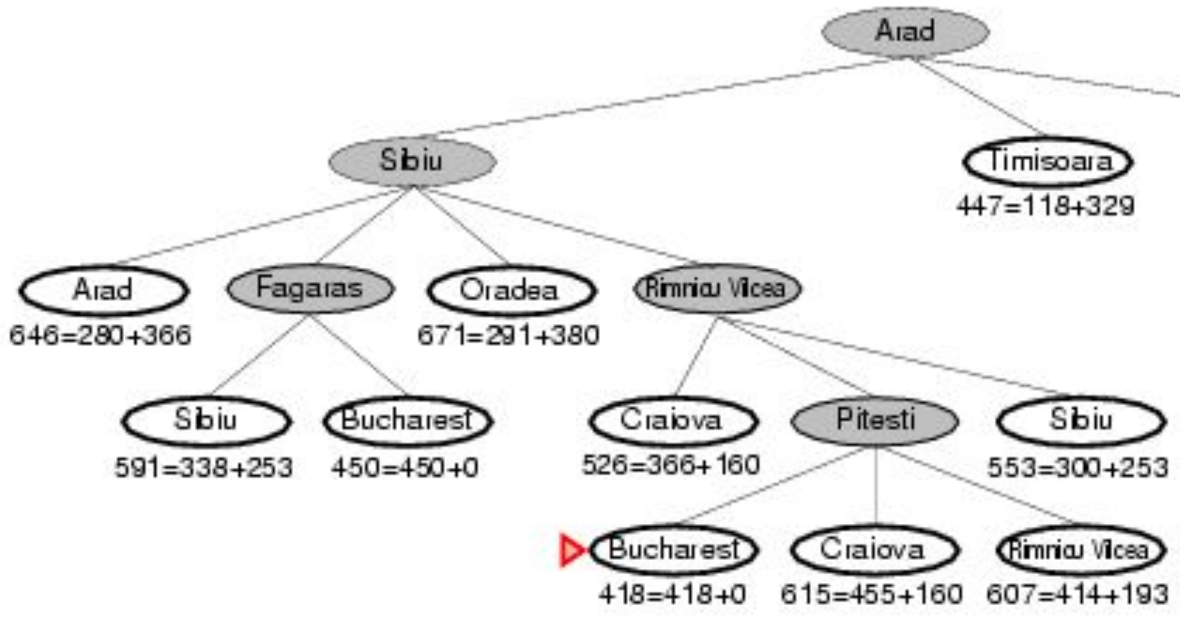
A* search example



A^* search example



A* search example



An **admissible heuristic** is one that **never overestimates** the true cost to reach the goal from any node. In other words, for every node n,

$h(n) <= h^*(n)$

Admissible = **optimistic**: it may underestimate, but **never** overestimate.

where
h(n) is your heuristic estimate, and
h*(n) is the actual least-cost from n to the goal

Admissible heuristics

- A heuristic $h(n)$ is admissible if for every node n,
  $h(n) \leq h^{*}(n)$, where $h^{*}(n)$ is the true cost to reach the goal state from n.

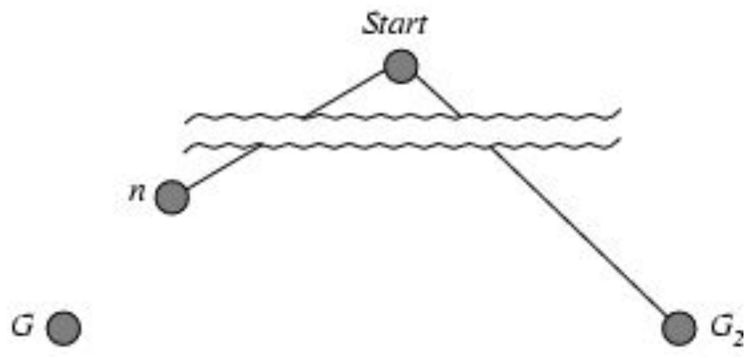
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic

- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

- Theorem: If $h(n)$ is admissible, A* using `TREE-SEARCH` is optimal

Optimality of A* (proof)

- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on a shortest path to an optimal goal G.



- We need to show: $f(n) < f(G_2)$
- $f(n) = g(n) + h(n)$

  $\leq g(n) + c(n, G)$ since h is admissible

  $= g(G)$

  $< g(G_2)$ since G_2 is suboptimal

  $= f(G_2)$ since $h(G_2) = 0$

Properties of A*

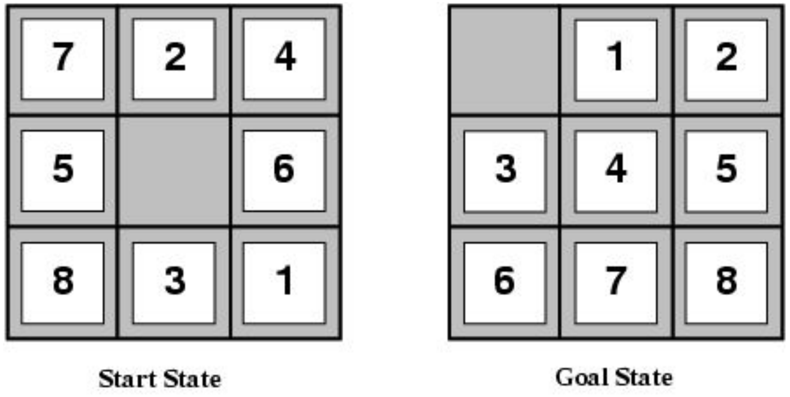
- Complete? Yes (unless there are infinitely many nodes with f $\leq$ *f(G)*)
- Time? Exponential
- Space? Keeps all nodes in memory
- Optimal? Yes

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)



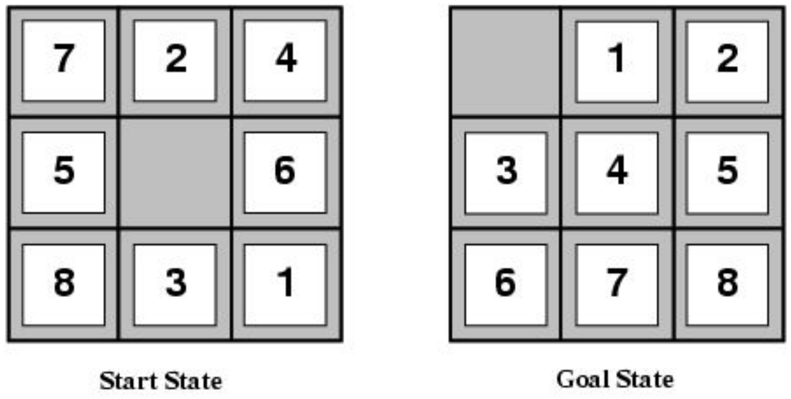
- $h_1(S)$ = ?
- $h_2(S)$ = ?

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)



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

Dominance

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

- Typical search costs (average number of nodes expanded):

- $d=12$ IDS = 3,644,035 nodes
  $A^*(h_1)$ = 227 nodes
  $A^*(h_2)$ = 73 nodes

- $d=24$ IDS = too many nodes
  $A^*(h_1)$ = 39,135 nodes
  $A^*(h_2)$ = 1,641 nodes

- Why is A* so much better?
  Because it reduces the effective branching factor

Relaxed problems

- A problem with fewer restrictions on the actions is called a relaxed problem
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original 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