Informed search algorithms

Informed search

- Uses problem-specific knowledge beyond the definition of the problem itself.
- Can find solutions more efficiently than

  an uninformed strategy

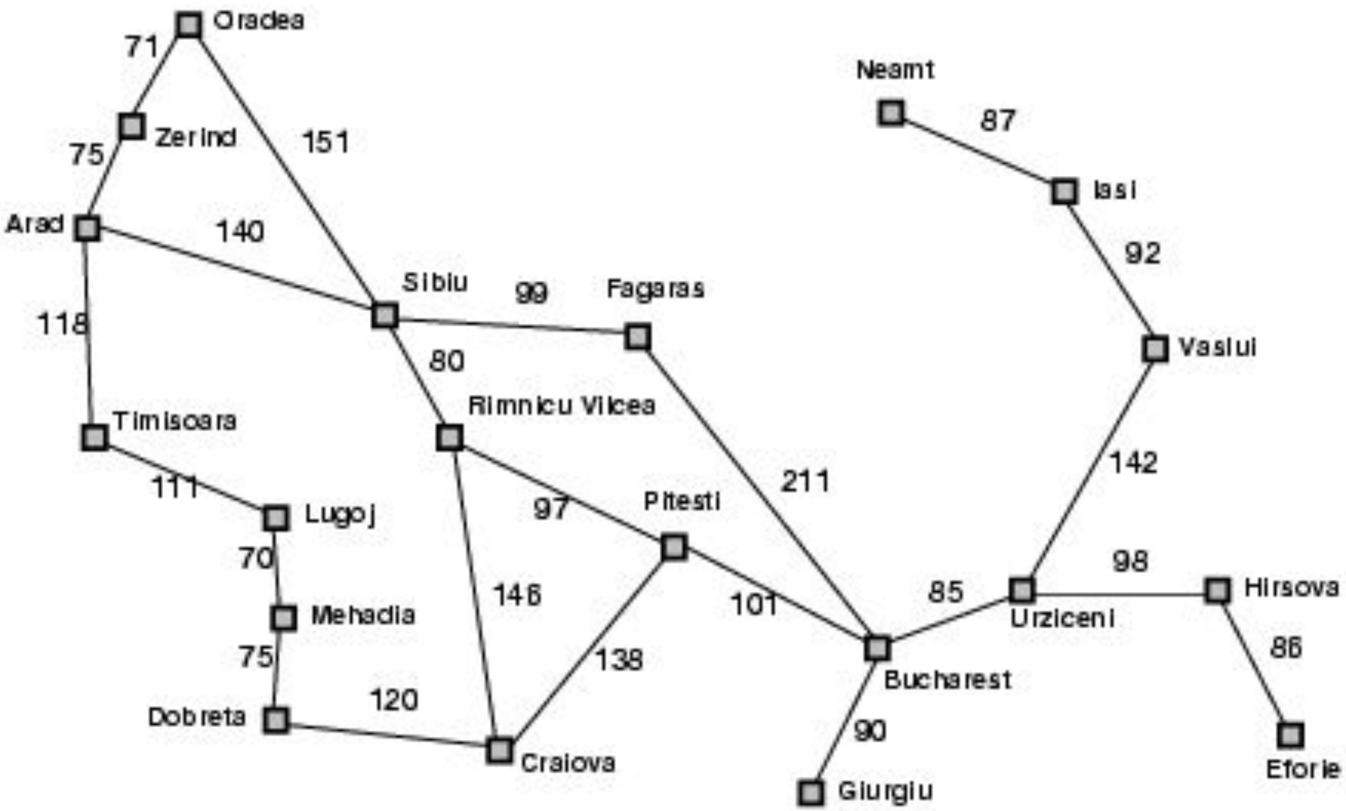
Best-first search

- Best-first search is an instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on an evaluation function, f(n).
- The evaluation function construed as a cost estimate, so the node with the lowest evaluation is expanded first.
- 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

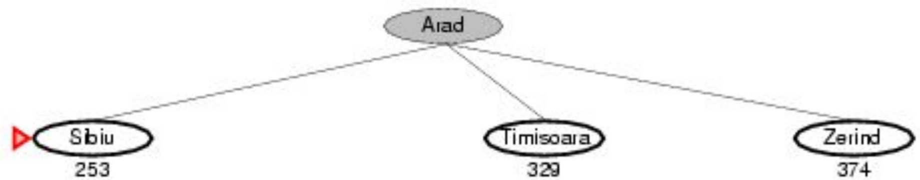
- Evaluation function $f(n) = h(n)$ (heuristic)

= estimate of cost from *n* to *goal*

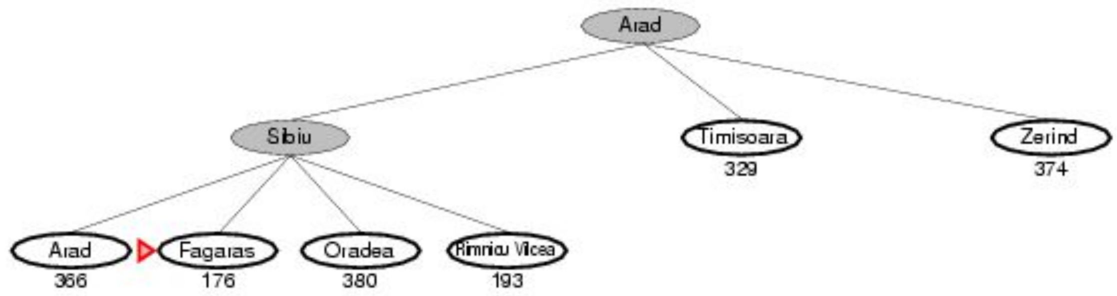
- e.g., $h_{SLD}(n)$ = straight-line distance from *n* to Bucharest
- Greedy best-first search expands the node that appears to be closest to goal

Greedy best-first search example

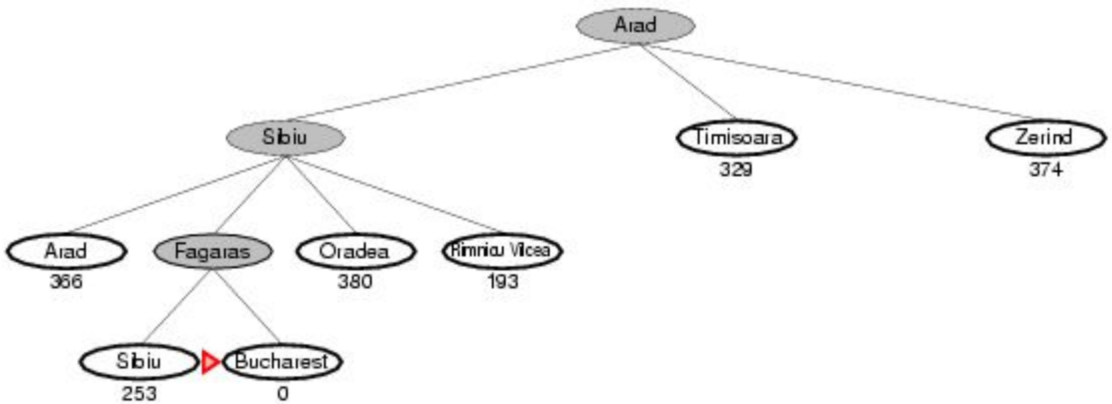
Greedy best-first search example



Greedy best-first search example



Greedy best-first search example



Properties of greedy best-first search

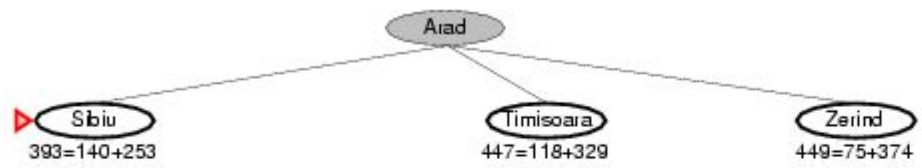
- Complete? No – can get stuck in loops, e.g., Iasi □ Neamt □ Iasi □ Neamt □
- Time? $O(b^m)$, but a good heuristic can give dramatic improvement
- Space? $O(b^m)$ -- keeps all nodes in memory
- Optimal? No

A^* search

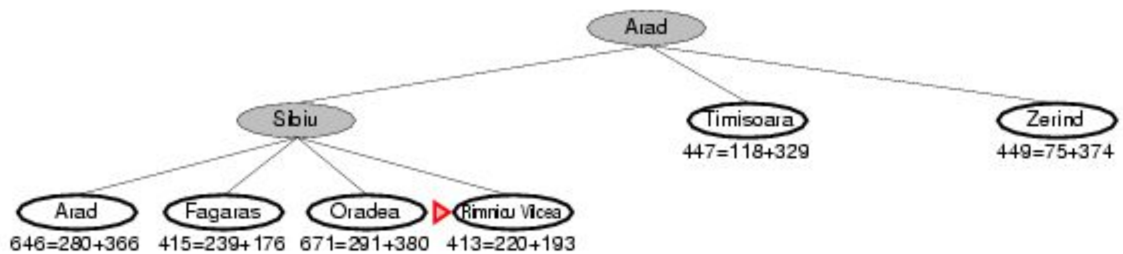
- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = cost so far to reach n
- $h(n)$ = estimated cost from n to goal
- $f(n)$ = estimated total cost of path through n to goal

A* search example

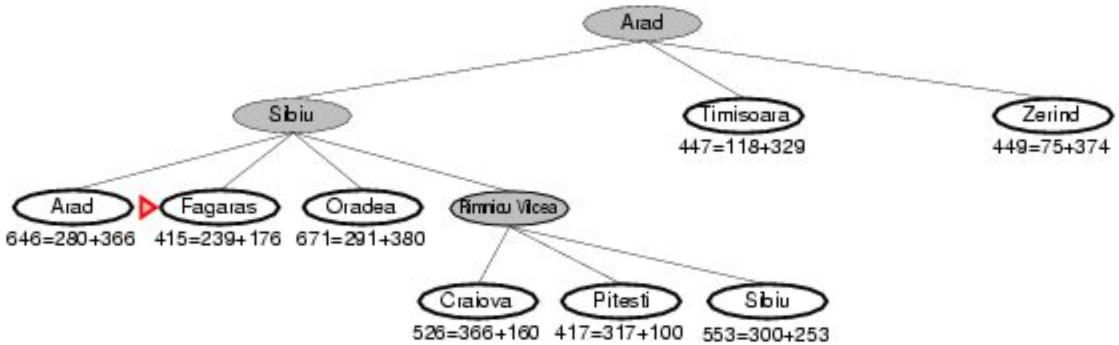
A^* search example



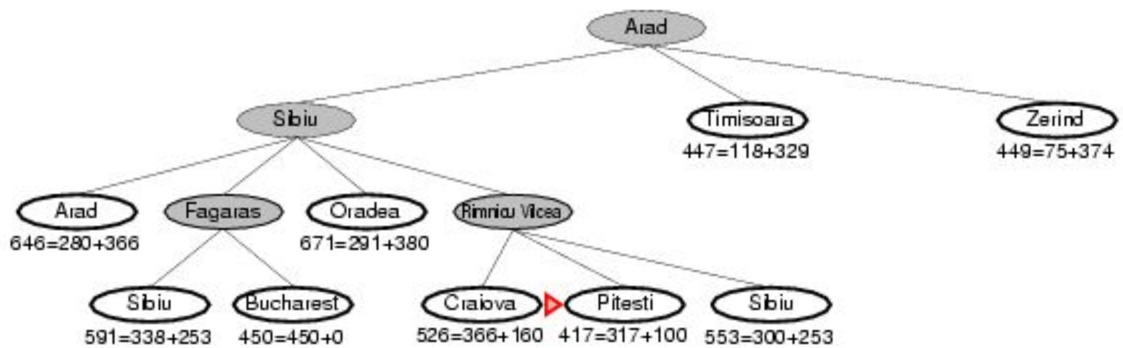
A* search example



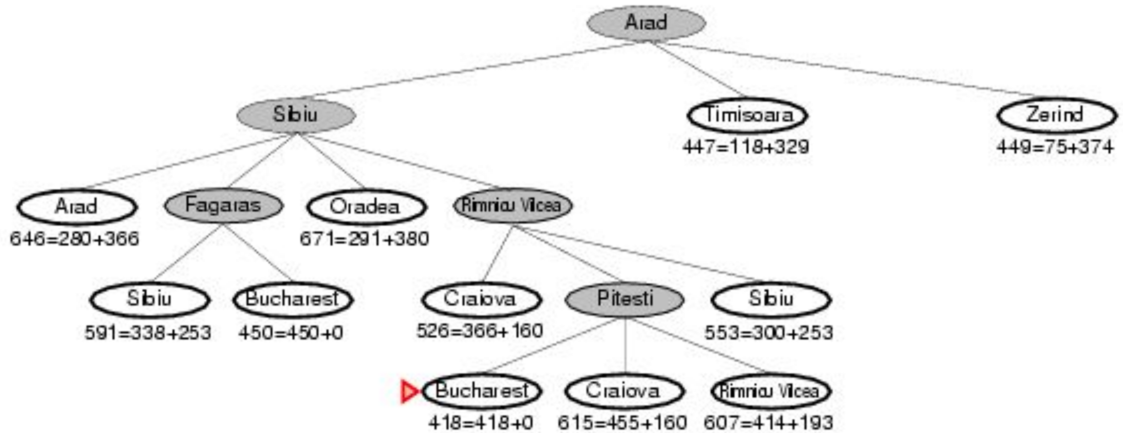
A* search example



A* search example



A* search example



Conditions for optimality: Admissibility and consistency

- An ADMISSIBLE HEURISTIC is one that never overestimates the cost to reach the goal.
- Because g(n) is the actual cost to reach n along the current path, and
- f(n) = g(n) + h(n), we have as an immediate consequence that f(n) never overestimates the true cost of a solution along the current path through n.
- Admissible heuristics are by nature optimistic because they think the cost of solving the problem is less than it actually is. Eg. straight line distance.

Admissible heuristics

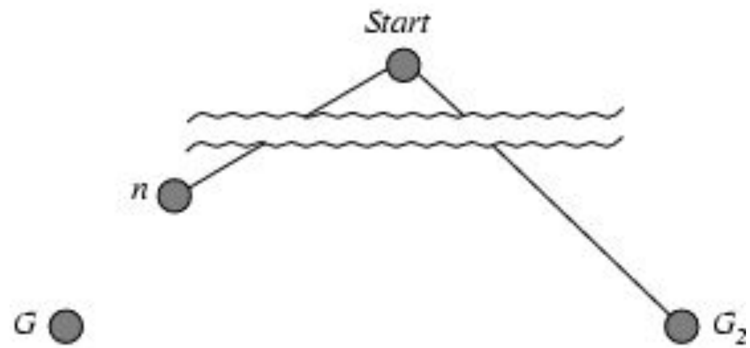
- A heuristic $h(n)$ is admissible if for every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n.

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

Optimality of A^* (proof)

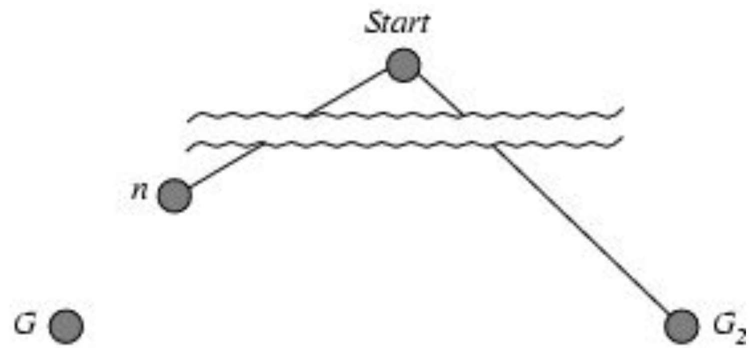
- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on a shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $g(G_2) > g(G)$ since G_2 is suboptimal
- $f(G) = g(G)$ since $h(G) = 0$
- $f(G_2) > f(G)$ from above

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*.



- $f(G_2) \quad > f(G) \quad$ from above
- $h(n) \quad \le h^*(n) \quad$ since h is admissible
- $g(n) + h(n) \quad \le g(n) + h^*(n)$
- $f(n) \quad \le f(G)$

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

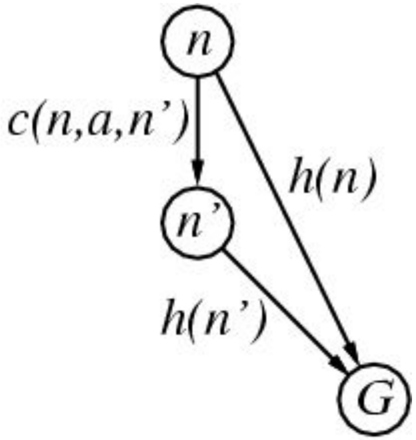
Consistency

- A second, slightly stronger condition called **CONSISTENCY** (or sometimes monotonicity) which is required only for applications of A^* to graph search.
- A heuristic h(n) is consistent if, for every node n and every successor n of n’ generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n’ plus the estimated cost of reaching the goal from n

Consistent heuristics

- A heuristic is consistent if for every node n, every successor n' of n generated by any action a,

$$h(n) \le c(n,a,n') + h(n')$$



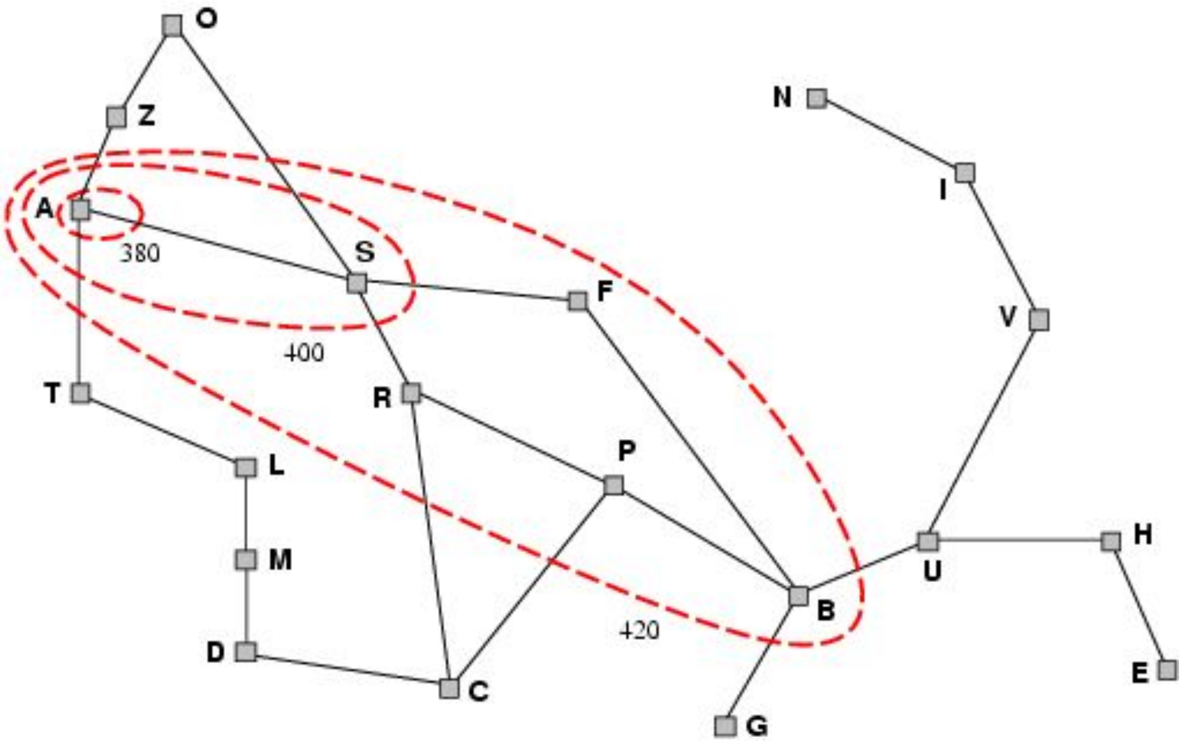
- If h is consistent, we have

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n,a,n') + h(n') \\ &\ge g(n) + h(n) \\ &= f(n) \end{aligned}$$

- i.e., $f(n)$ is non-decreasing along any path.
- If $h(n)$ is consistent, A* using `GRAPH-SEARCH` is optimal

Optimality of A*

- A* expands nodes in order of increasing *f* value
- Gradually adds "*f*-contours" of nodes
- Contour *i* has all nodes with $f=f_i$, where $f_i < f_{i+1}$

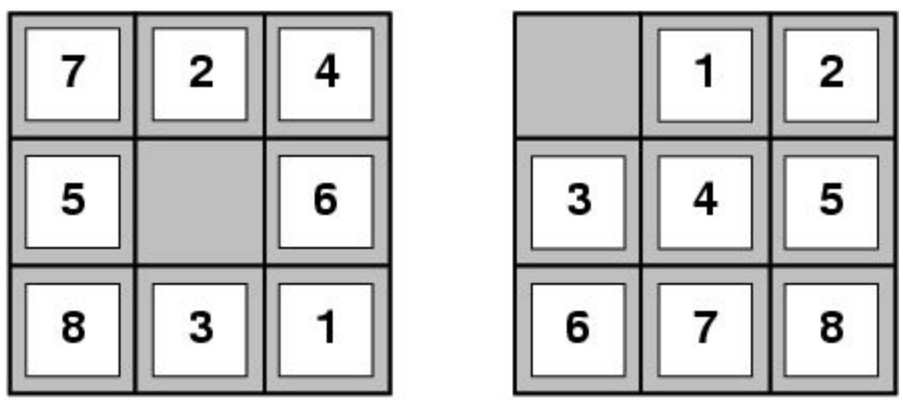


Properties of A*

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

Heuristic Functions

- The 8-puzzle is one of the earliest heuristic search problems.
- The objective of the puzzle is to slide the tiles horizontally or vertically into empty space until the configuration matches the goal state.
- Average solution cost is about 22 steps and average branching factor is 3 results in 3^{22} states.



Start State Goal State

Heuristic Functions

Heuristic Functions 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)

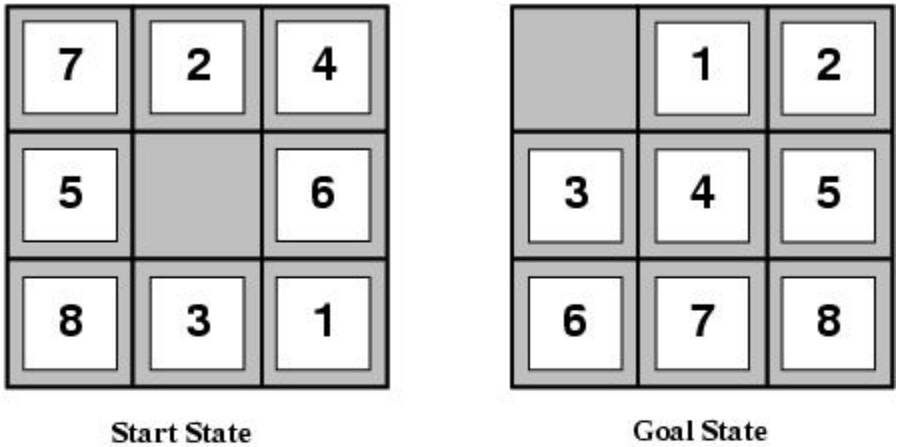
Both h_1 *and* h_2 *are admissible heuristics.*

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)



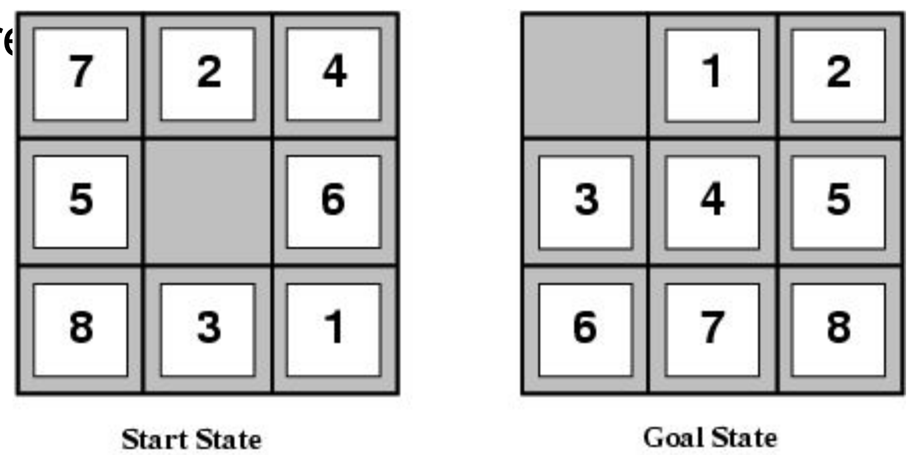
- $h_1(S) = ?$
- $h_2(S) = ?$

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 desire



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

Effect of Heuristic Accuracy on Performance

- The effective branching factor generated by any heuristic search gives the quality of heuristic.

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

Generating admissible heuristics from 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

Generating admissible heuristics from Subproblems

- Admissible heuristics can also be derived from solution cost of a subproblem of a given solution.