CS W4701
Artificial Intelligence

Fall 2013

Chapter 3 Part 4:

Informed Search

Jonathan Voris

(based on slides by Sal Stolfo)

Announcements

- Midterm: Thursday **October 24** 2:40-3:55 PM in **Pupin 301**
- Final: Thursday **December 5th** 2:40-3:55 PM in **Pupin 301**

Summary of Uninformed Search 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

Outline

- Best-first search
  - Greedy best-first search
  - A^* search
- Heuristics

Recap: Tree Search

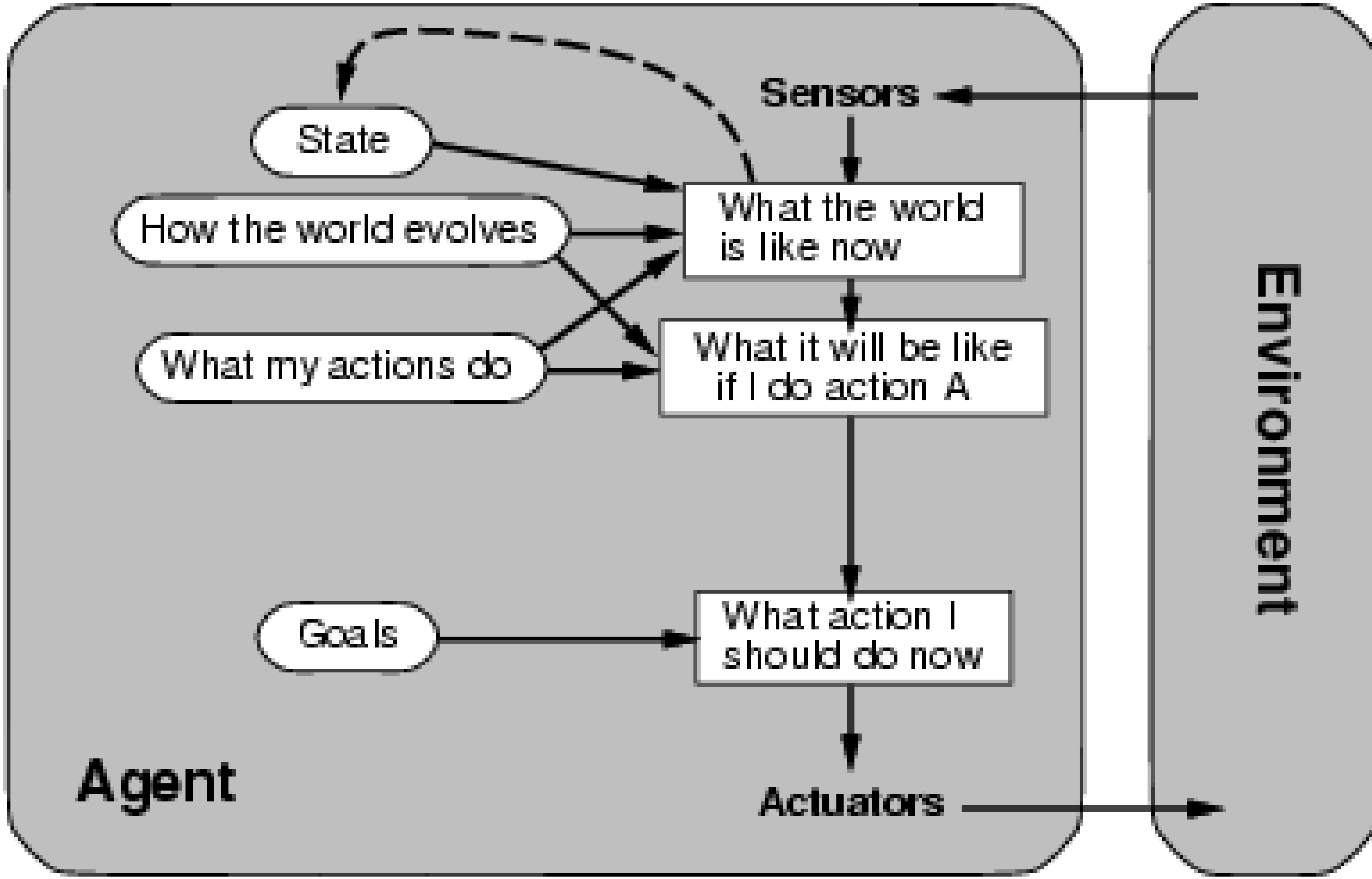
- Core concept:
  - Exploration of state space by generating successors of already-explored states (a.k.a. **expanding** states)

```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
```

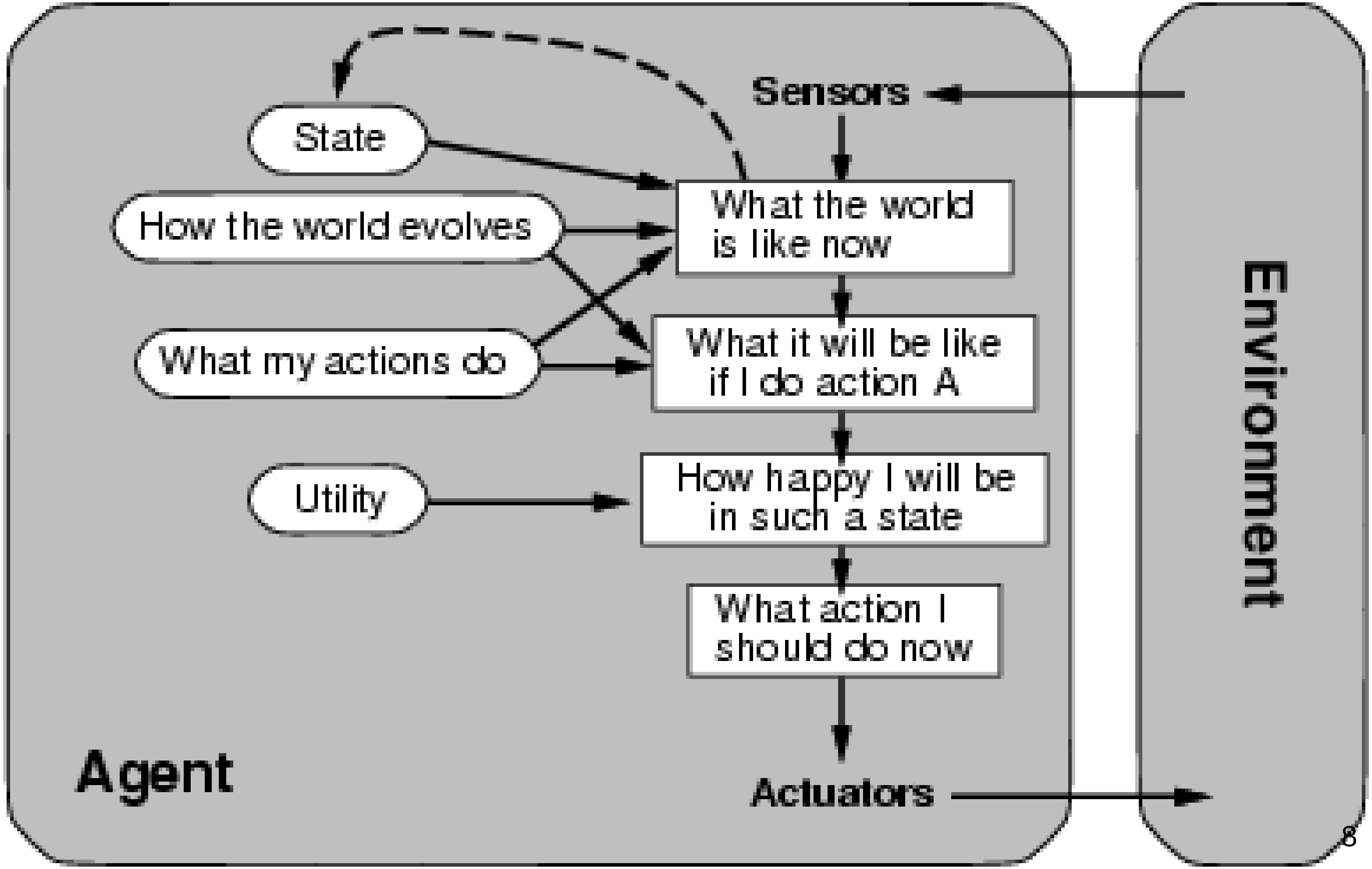
Smarter Search

- Uninformed search parameter selection: crude application of domain knowledge
  - Node ordering
  - Search strategy
- But still limited to:
  - Expand successors
  - Reached goal?
- What if we had a way to assess relative state quality?

Goal-based Agents



Utility-based Agents



Best-first Search

- Recall uniform-cost search
  - Expand node with lowest **path cost** function value g(n)
- New idea: use an **evaluation function** *f(n)* for each node
  - Estimate of "desirability“
  - Expand most desirable unexpanded node
- Implementation: Order the nodes in fringe in decreasing order of desirability
- Special cases:
  - Greedy best-first search
  - A^* search

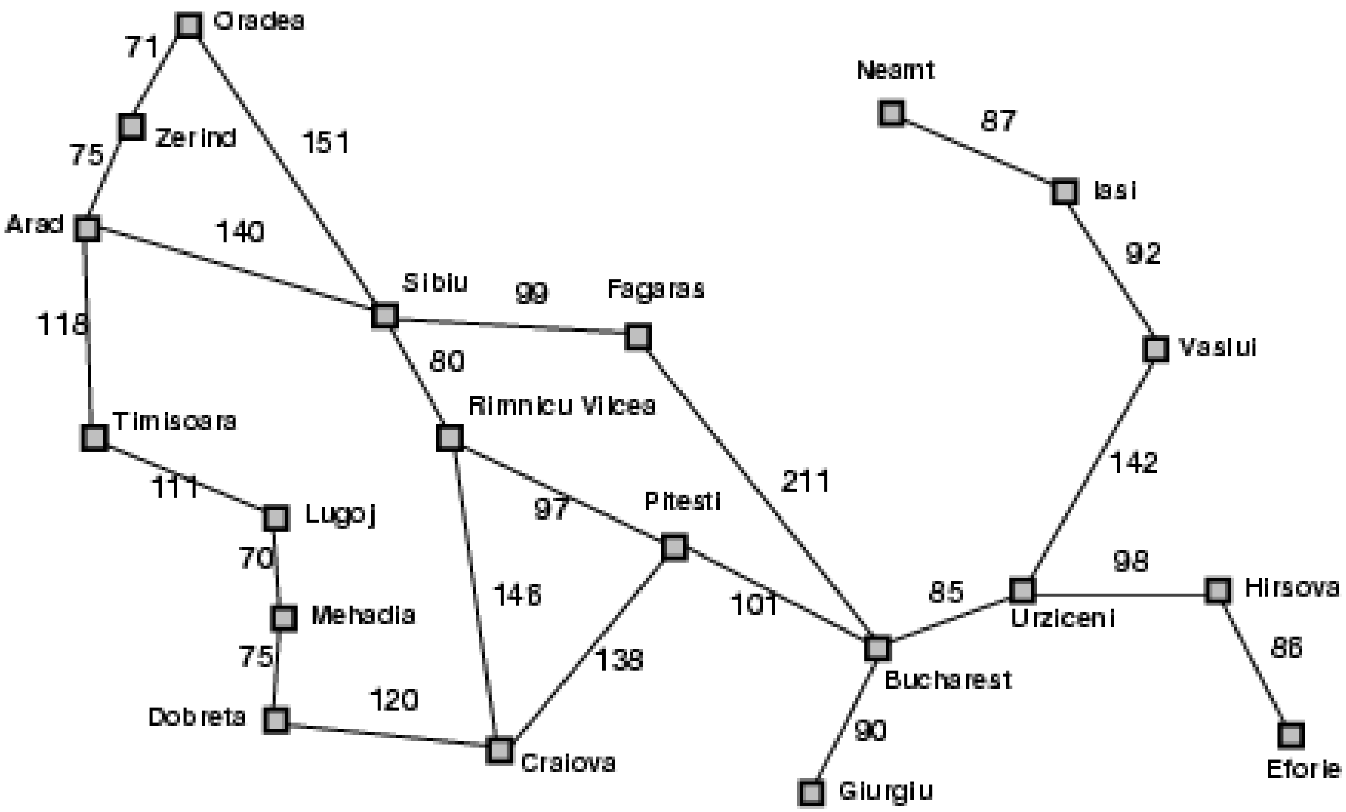
Heuristics

- f(n) presents a chicken and egg problem
  - Need to know which state is closest to goal
  - If we know that, what is the point of the agent?
- Instead, utilize domain specific knowledge to **estimate** preferable states
- Known as a **heuristic**
  - Greek word *heuriskein*: “To discover”
  - Learning aid
  - Feedback that facilitates self learning
- h(goal) = 0 always

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

Romania with Step Costs in km

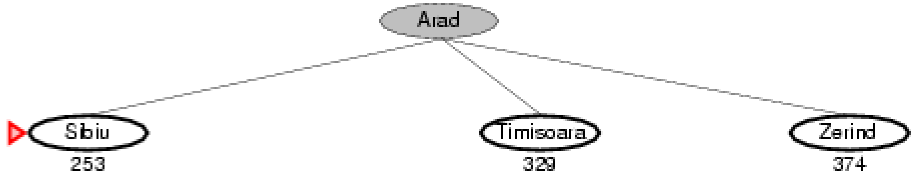


Straight–line distance to Bucharest

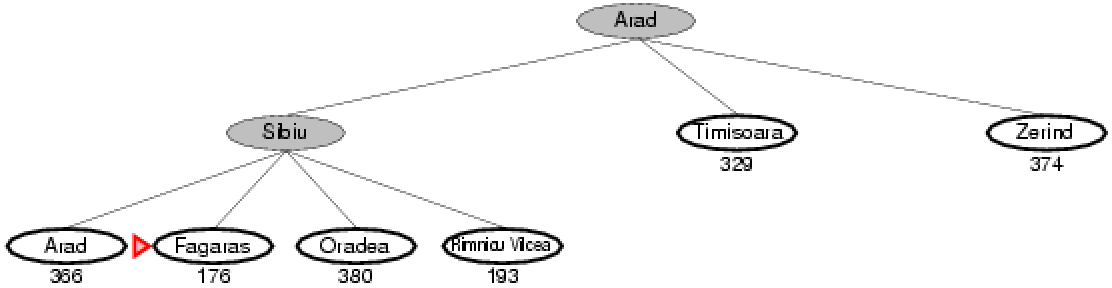
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best-first Search Example

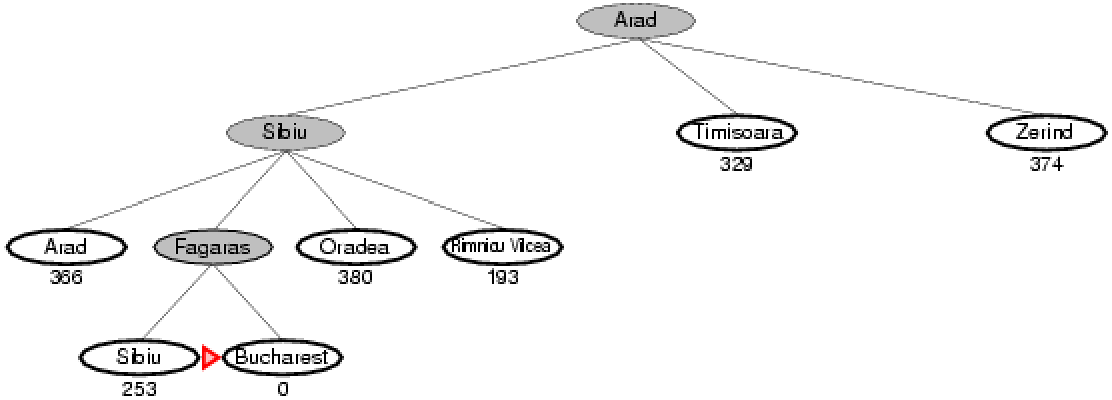
Greedy Best-first Search Example



Greedy Best-first Search Example



Greedy Best-first Search Example



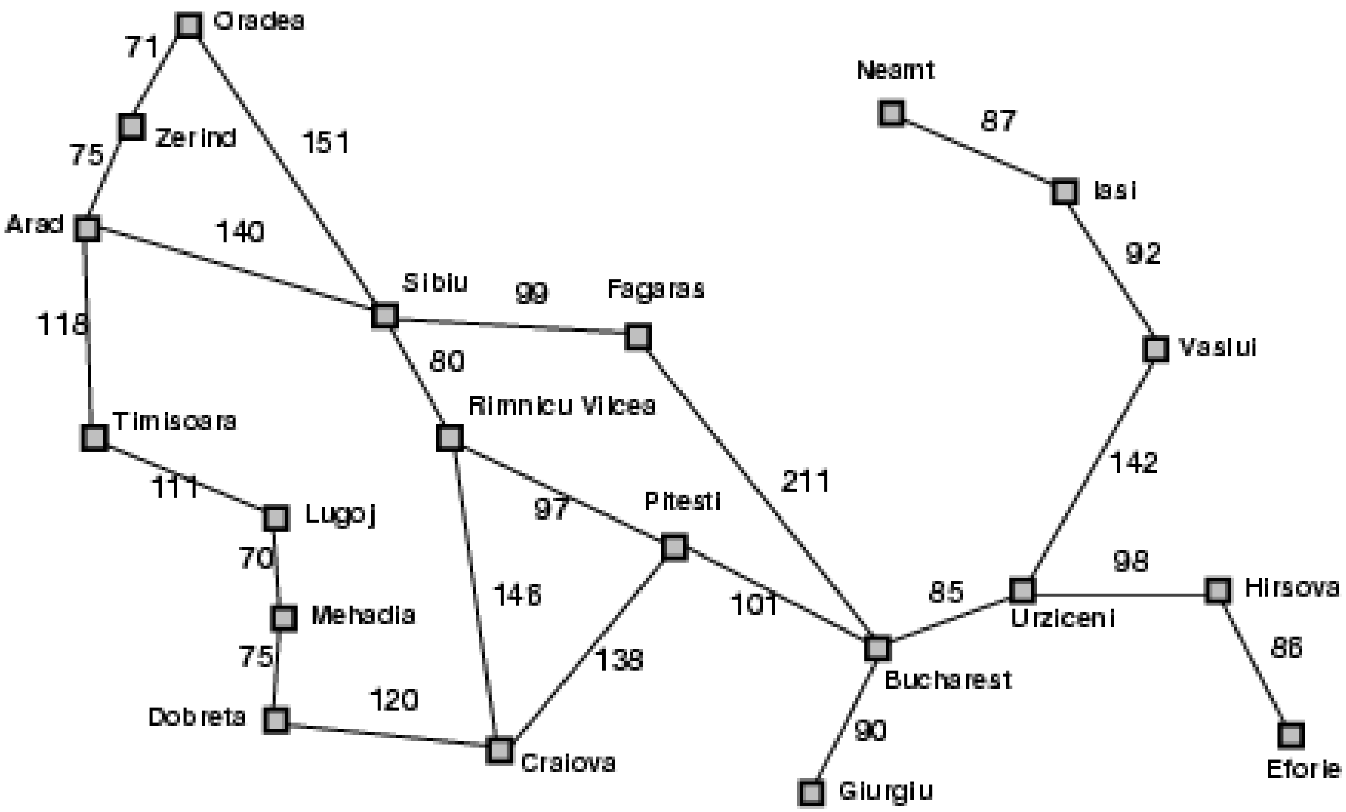
Properties of Greedy Best-first Search

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

A^* Search

- Idea: Avoid expanding paths that are already expensive
- Evaluation function
  - f(n) = estimated cost of cheapest path through n
  - $f(n) = g(n) + h(n)$
- $g(n)$ = 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

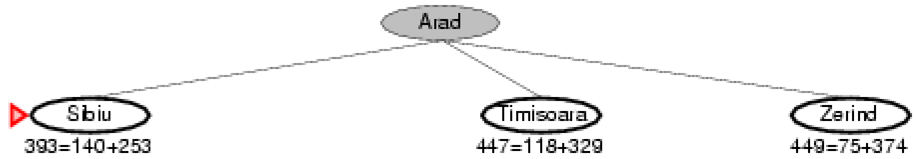
Romania with Step Costs in km



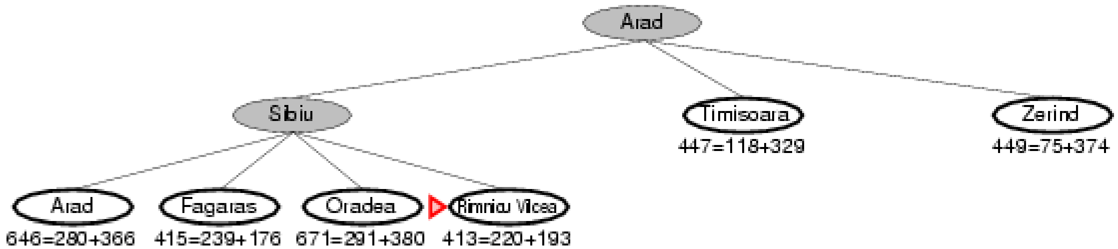
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A^* Search Example

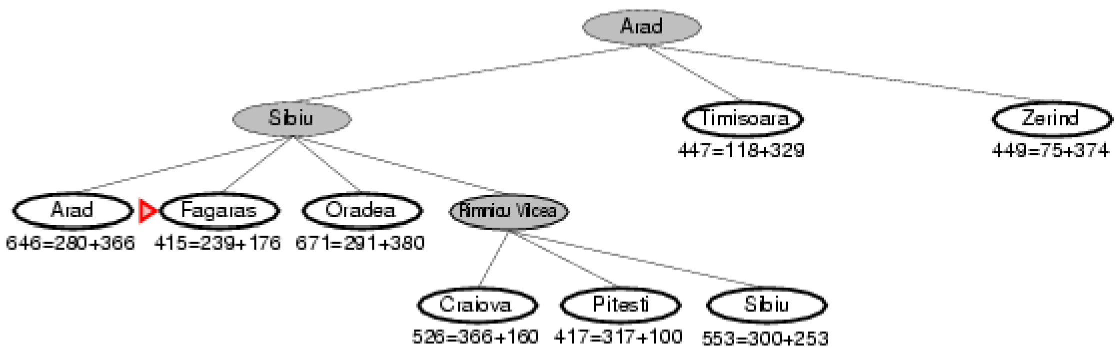
A* Search Example



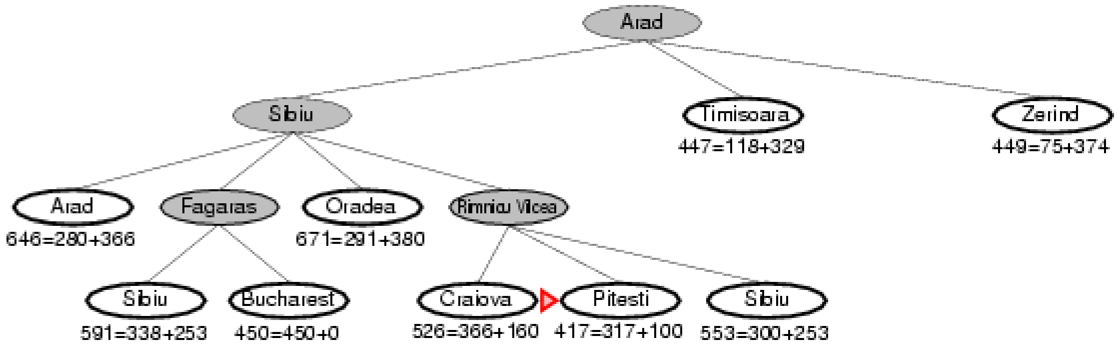
A* Search Example



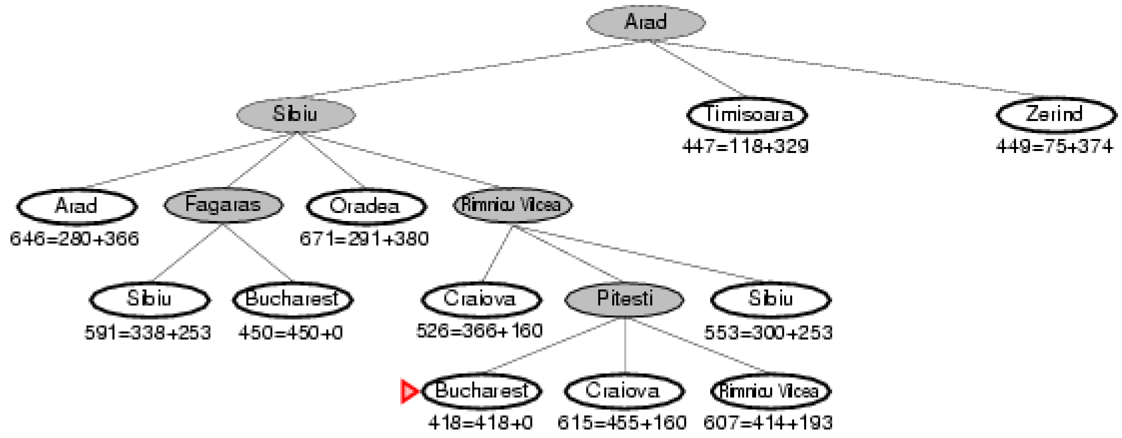
A^* Search Example



A* Search Example



A* Search Example

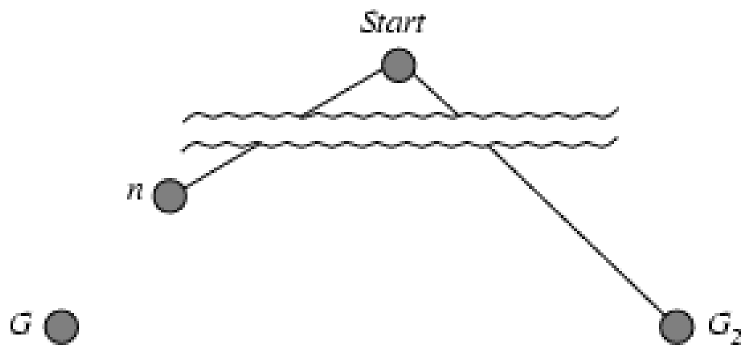


Admissible Heuristics

- 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**
  - f(n) won't overestimate then either
- Example: $h_{SLD}(n)$
  - Never overestimates the actual road distance
- <u>Theorem</u>: If $h(n)$ is admissible, A^* using `TREE-SEARCH` is optimal

Proof: Optimality of A^*

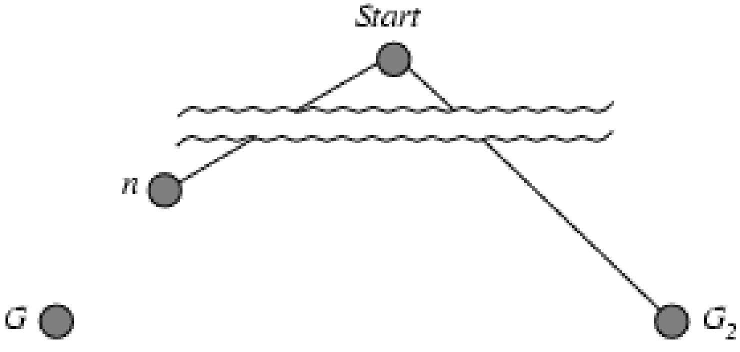
- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let *n* be an unexpanded node in the fringe such that *n* is on a shortest path to an optimal goal *G*.



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

Proof: Optimality of A*

- 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*.



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

So $f(G_2) > f(G) \geq f(n)$ and hence $f(G_2) > f(n)$

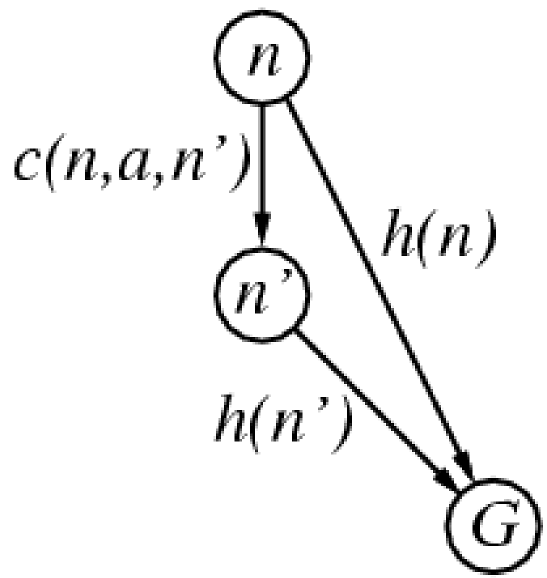
Therefore A* will never select G_2 for expansion.

Consistent Heuristics

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

  $h(n) \leq c(n,a,n') + h(n')$
- Consistency means f(n) should not

  decrease along path



Consistent Heuristics

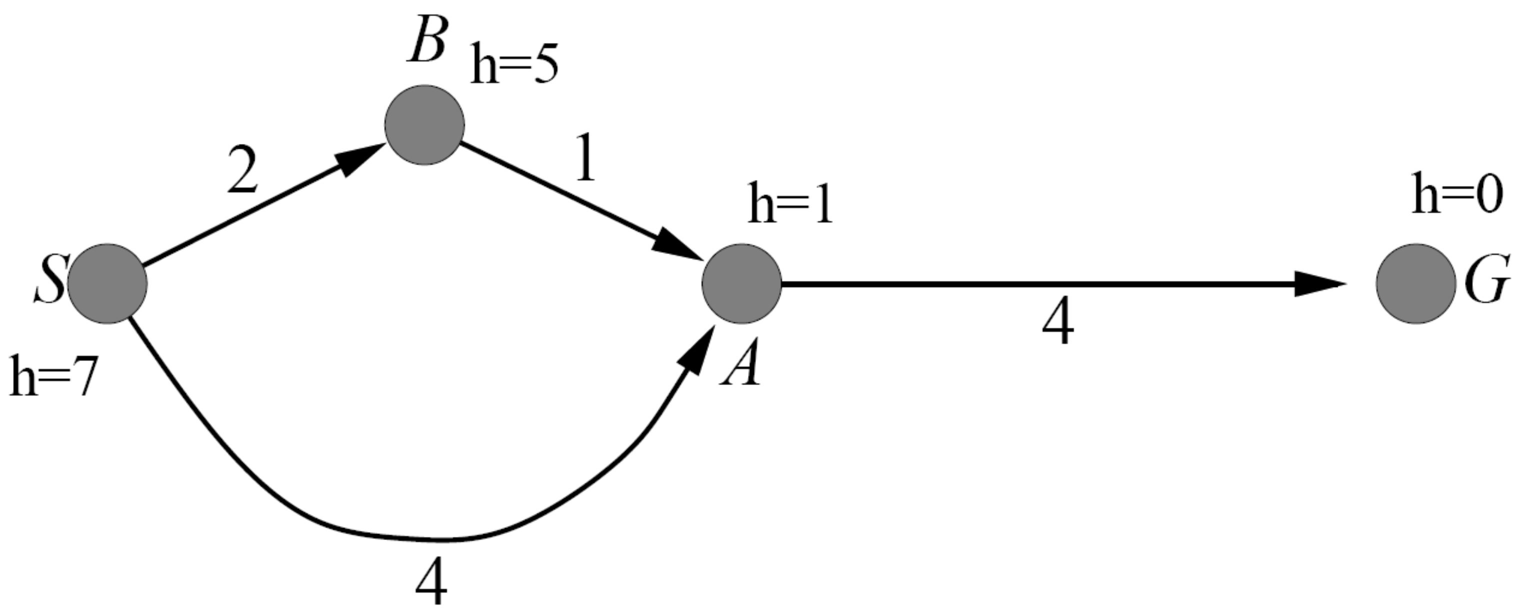
- **Theorem**: If *h(n)* is consistent, A* using `GRAPH-SEARCH` is optimal
- Assume h(n) is consistent
- f(n) along path is non-decreasing
- Mathematically speaking:

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

- Whenever n is expanded, we've found the best path to n
  - Otherwise we would've followed the better path first
- Thus all nodes expanded in non-decreasing order of f(n)
- f(goal) = g(goal)
  - Because f(goal) = g(goal) + h(goal) and h(goal) = 0
- First goal node expanded must be therefore be least expensive goal

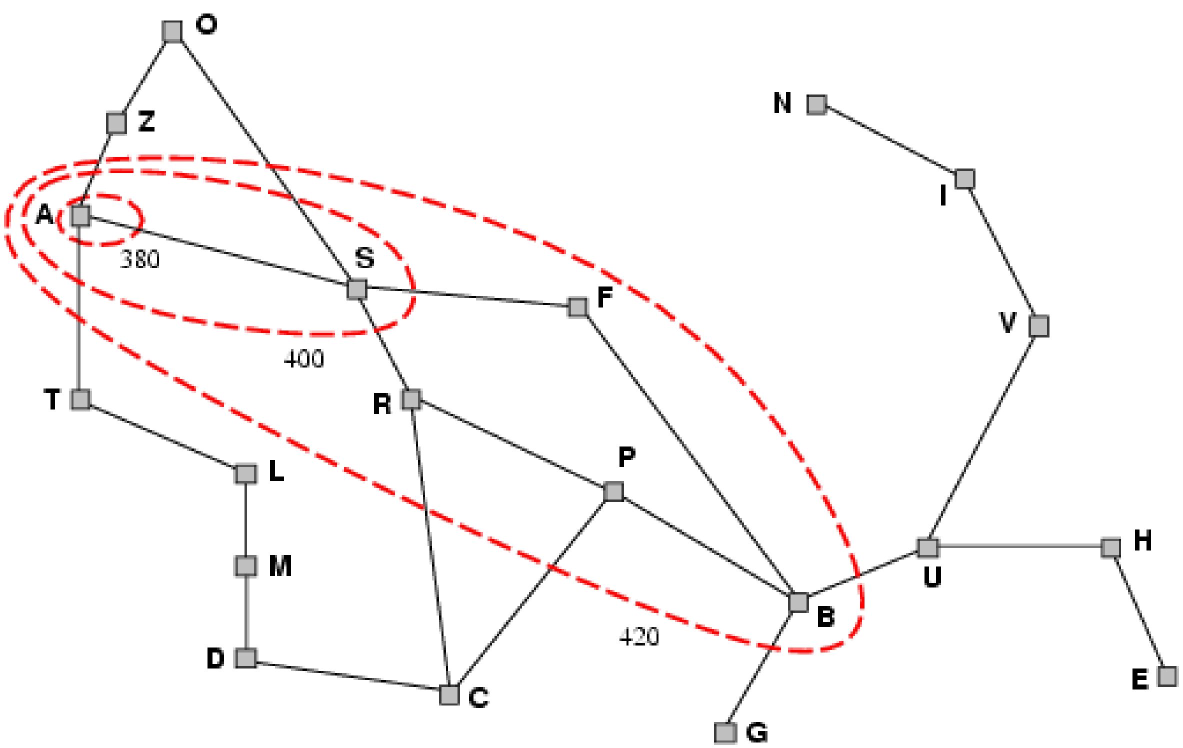
Admissibility and Consistency

- Consistency is stricter than admissibility
  - All consistent heuristics are admissible
  - Not all admissible heuristics are consistent



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*

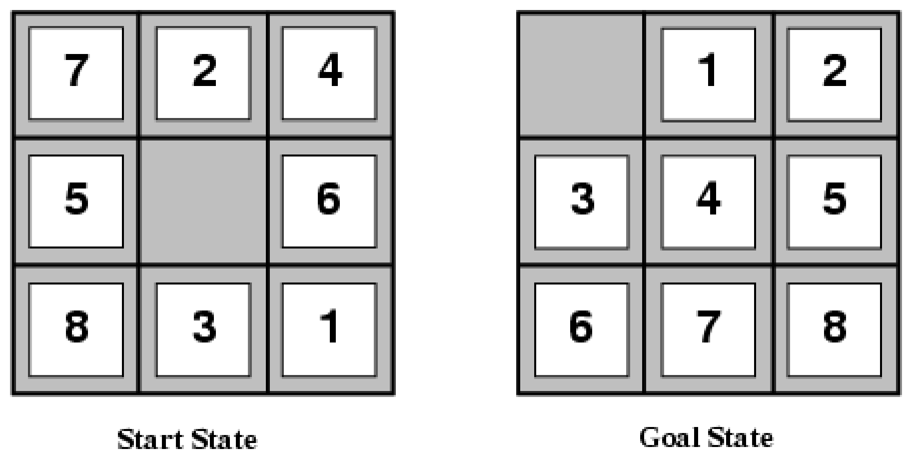
- Complete?
  - Yes (unless there are infinitely many nodes with f(n) ≤ optimal solution cost C*)
- Time?
  - Exponential
  - $O(b^{(h^*-h)})$
- Space?
  - Keeps all nodes in memory
- Optimal?
  - Yes
  - Also optimally efficient
    - Expanding fewer nodes may miss optimal solution

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)



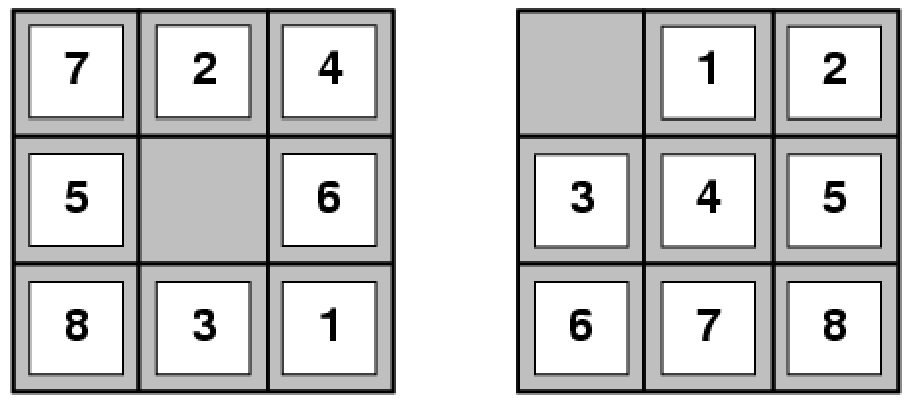
- $h_1(S) = ?$
- $h_2(S) = ?$

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)



Start State Goal State

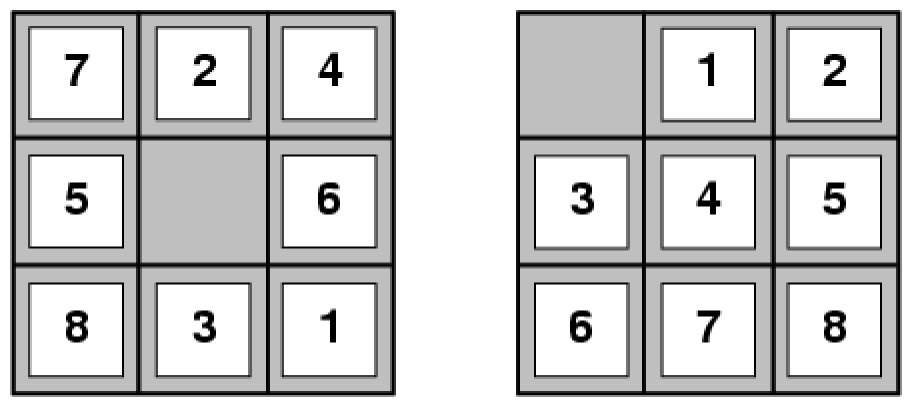
- $h_1(S) = 8$
- $h_2(S) = ?$

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

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

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) over 100 8-puzzle instances:
- $d=12$
  - IDS = 364,404 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

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

Summary

- Heuristic functions estimate costs of shortest paths
- Good heuristics can dramatically reduce search cost
- Greedy best-first search expands lowest ***h***
  - incomplete and not always optimal
- A* search expands lowest ***g + h***
  - Complete
  - Optimal
  - Also optimally efficient (up to tie-breaks, for forward search)
    - Can’t explore fewer nodes due to risk of missing optimal solution
- Admissible heuristics can be derived from exact solution of relaxed problems