Informed search algorithms

Chapter 3,

Sections 3.5 to end

(Adapted from Stuart Russel, Dan Klein, and others. Thanks guys!)

Outline

- Best-first search
- A^* search (and variants)
- Heuristics

Review: Tree and Graph search

```
function Tree-Search( problem, frontier) returns a solution, or failure
    frontier ← Insert(Make-Node(Initial-State[problem]), frontier)
    loop do
     if frontier is empty then return failure
     node ← Remove-Front(frontier)
     if Goal-Test[problem] applied to State(node) succeeds return node
     frontier ← InsertAll(Expand(node, problem), frontier)
```

```
function Graph-Search( problem, frontier) returns a solution, or failure
    closed ← an empty set
    frontier ← Insert(Make-Node(Initial-State[problem]), frontier)
    loop do
        if frontier is empty then return failure
        node ← Remove-Front(frontier)
        if Goal-Test(problem, State[node]) then return node
        if State[node] is not in closed then
            add State[node] to closed
            frontier ← InsertAll(Expand(node, problem), frontier)
    end
```

A strategy is defined by picking the order of node expansion

Best-first search

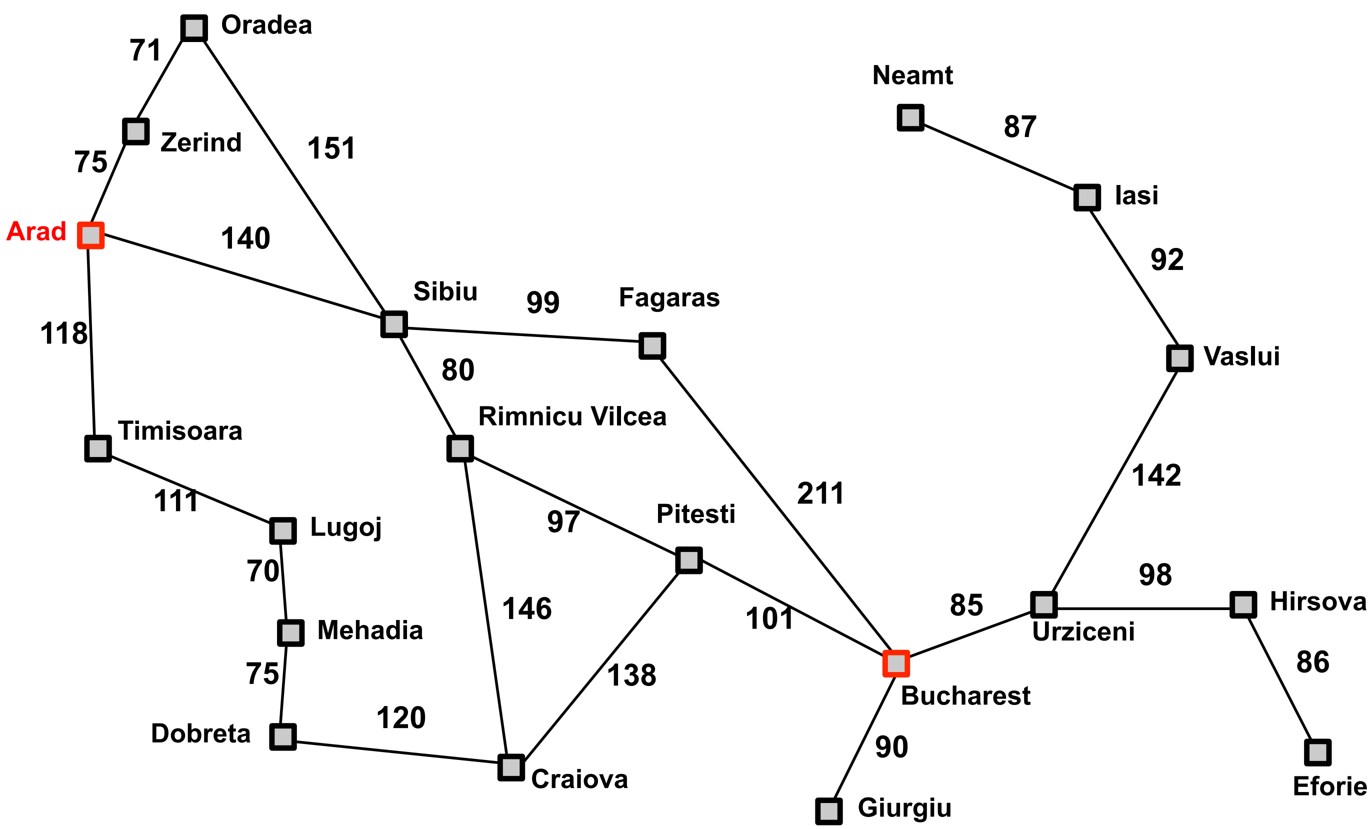
Plan: use an evaluation function for each node
– estimate of “desirability”

⇒ Expand most desirable unexpanded node

Implementation:
frontier is a queue sorted in decreasing order of desirability

Special cases:
- greedy search
- A* search

Example: Romania with step costs in km



Straight−line distance to Bucharest (SLD)

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
RimnicuVilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy search

Evaluation function $h(n)$ (heuristic)
= estimate of cost from n to the closest goal

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

Greedy search expands the node that appears to be closest to goal

Properties of greedy search

Complete??

Time??

Space??

Optimal??

A* Search

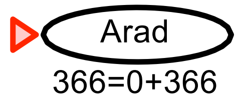
Idea:

- avoid expanding paths that are already expensive
- Work on paths that are “most promising”
-

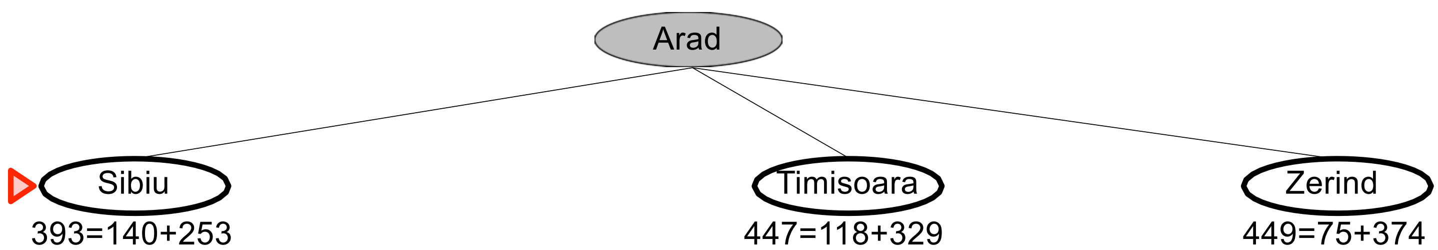
A* search uses an admissible heuristic

Theorem: A* search is optimal

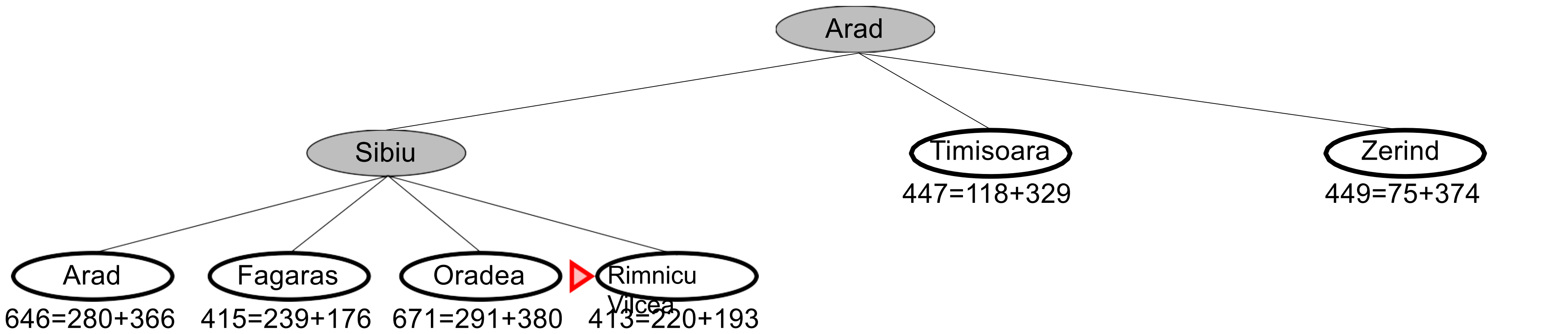
A* Search Example



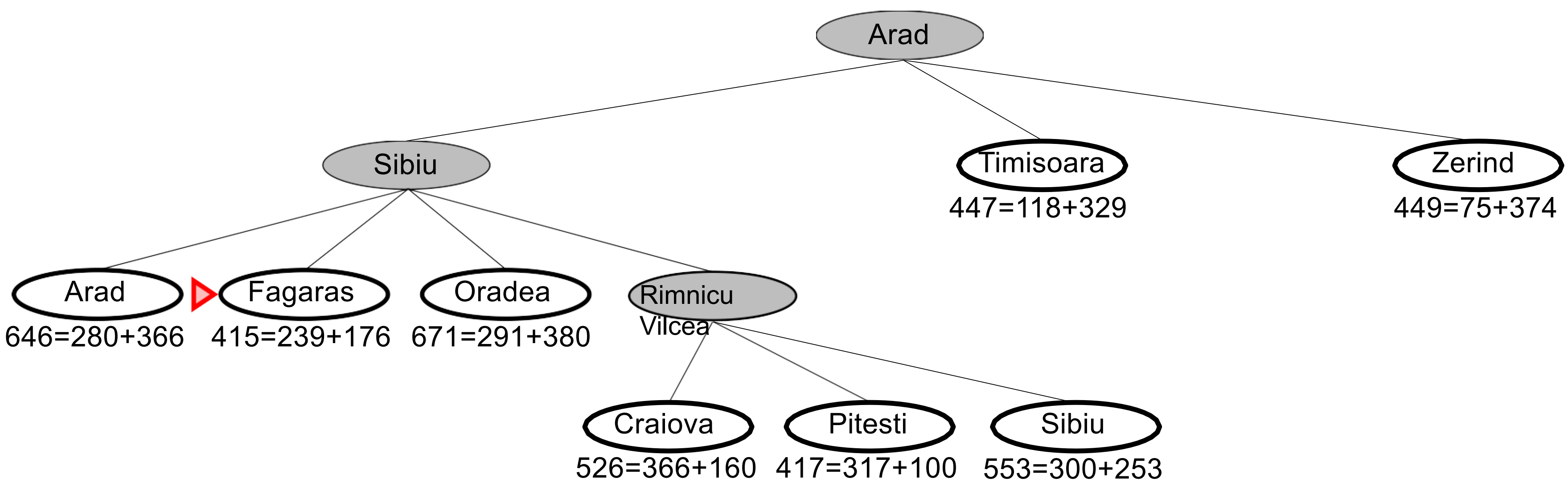
A* search example



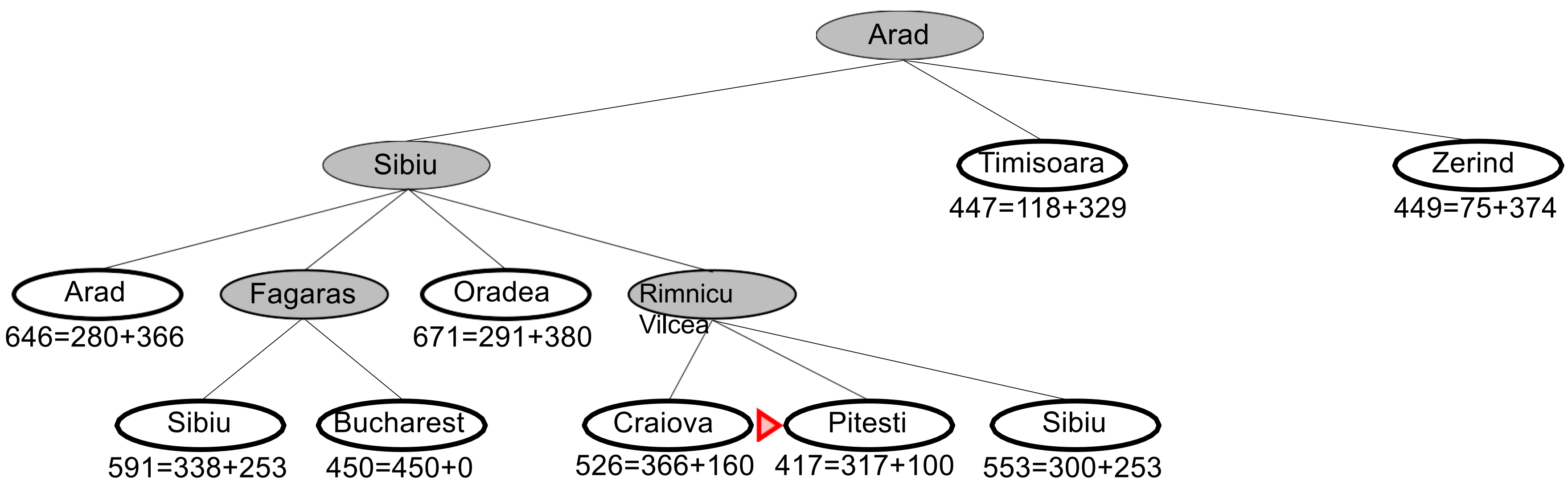
A* search example



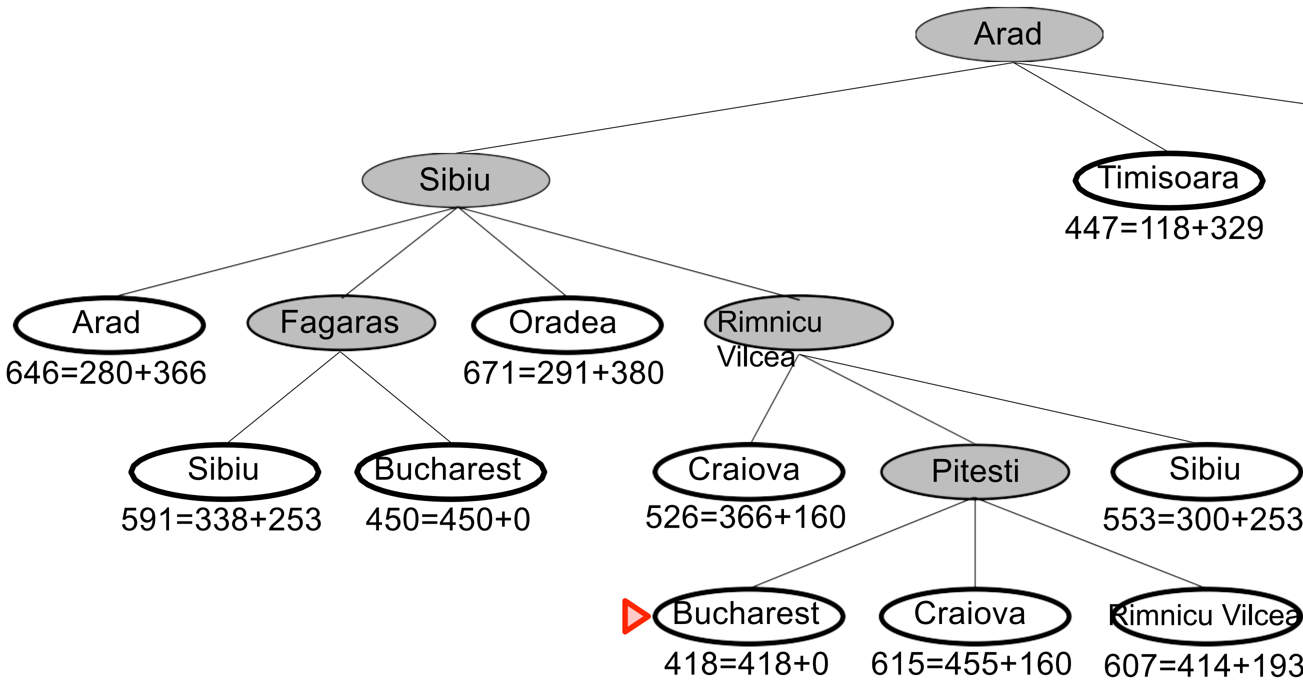
A^* search example



A* search example



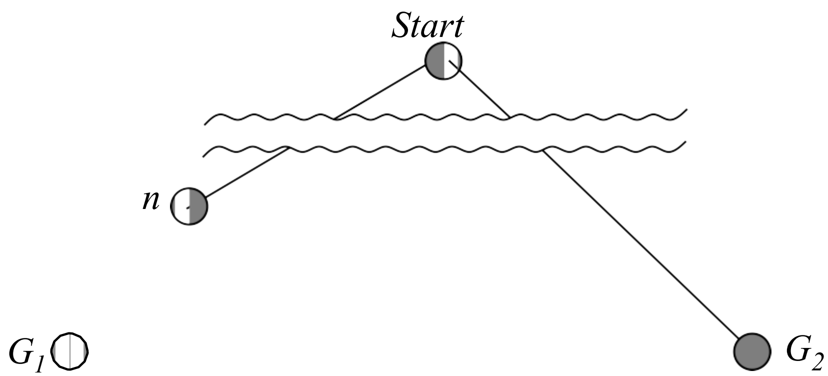
A* Search Example



Optimality of A*

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.

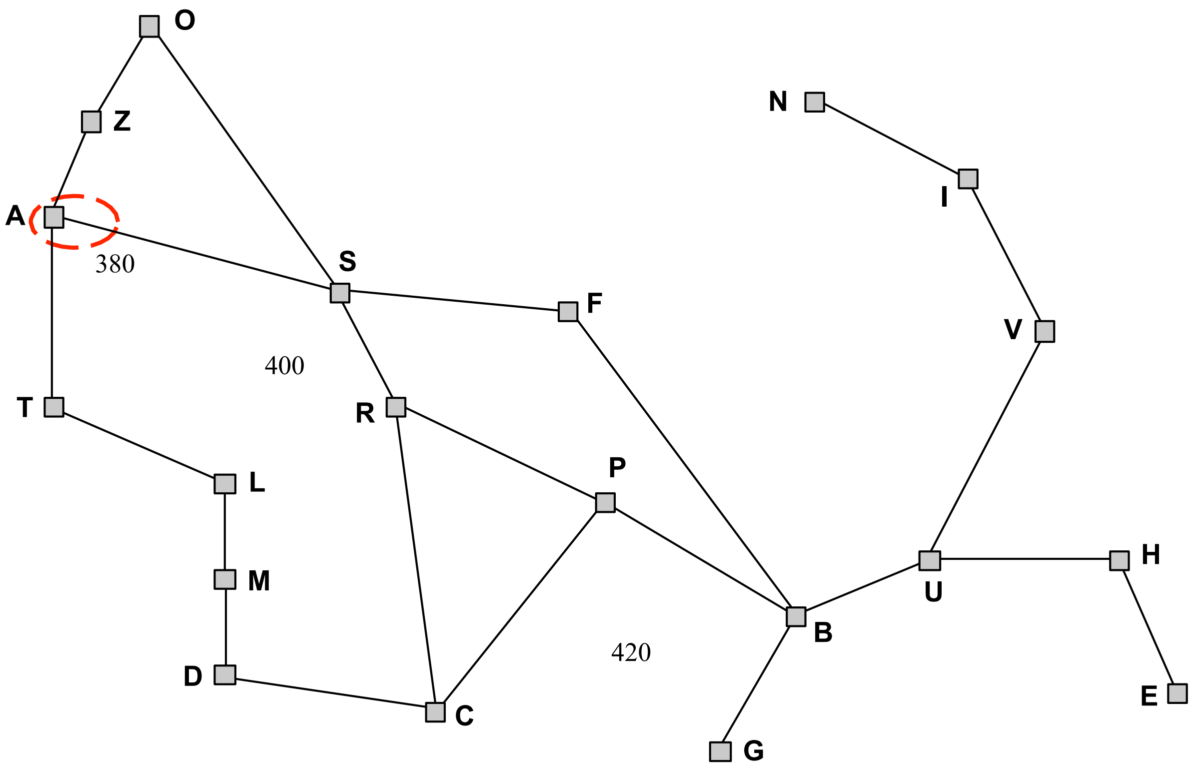


$$\begin{aligned} f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\ &> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\ &\geq f(n) && \text{since } h \text{ is admissible} \end{aligned}$$

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

Optimality of A*

Lemma: A* expands nodes in order of increasing *f*-value*



Properties of A*

Complete??

Time??

Space??

Optimal??

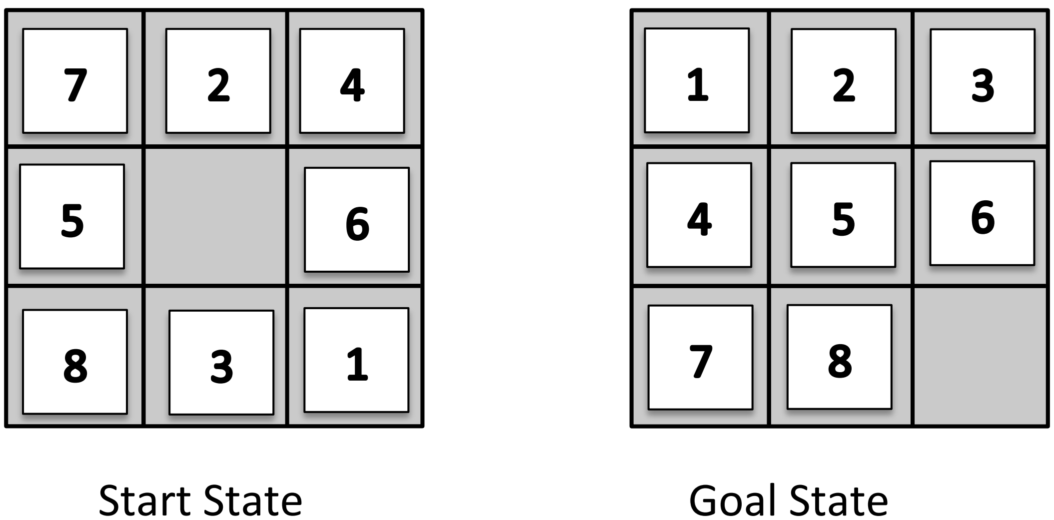
expands all nodes with $f(n) < C^*$

- 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)



$h_1(S)$ =?? 6

$h_2(S)$ =?? 4+0+3+3+1+0+2+1 = 14

Dominance

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:

$d = 14$ IDS = 3,473,941 nodes

- $A^*(h_1)$ = 539 nodes
- $A^*(h_2)$ = 113 nodes

$d = 24$ IDS $\approx$ 54,000,000,000 nodes

- $A^*(h_1)$ = 39,135 nodes
- $A^*(h_2)$ = 1,641 nodes

Given any admissible heuristics h_a, h_b,

$h(n) = \max(h_a(n), h_b(n))$

is also admissible and dominates h_a, h_b

Relaxed problems

Admissible heuristics can be derived from the exact solution cost of a relaxed version of the problem

E.g.:

- 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: Cost (optimal solution to relaxed prob) <=
Cost(actual problem)

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 and optimal
  - also optimally efficient (up to tie-breaks, for forward search)

- Admissible heuristics can be derived from exact solution of relaxed problems


STOP