Informed (Heuristic) Search

CMPSCI 383
September 20, 2011

Tip for doing well

Begin Assignments Early

Today's lecture

Informed (Heuristic) search methods

- Best-First Search
- Greedy Best-First Search
- A* search
- Heuristic functions

Review: Uninformed search strategies

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative deepening depth-first search

Review

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

---

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

Review

Note: step costs assumed non-negative

- Breadth-first search
  - Selects shallowest unexpanded node
  - FIFO queue
- Uniform-cost search
  - Selects node with lowest path cost
  - Priority queue by path cost
- Depth-first search
  - Selects deepest unexpanded node
  - LIFO queue (stack)
- Depth-limited search
  - Depth-first with nodes at depth limit treated as having no successors
- Iterative deepening depth-first search
  - Repeated depth-limited with increasing limit until goal found

Review: How do you evaluate a search strategy?

- **Completeness** — Does it always find a solution if one exists?
- **Optimality** — Does it find the *best* solution?
- **Time complexity**
- **Space complexity**

Informed (heuristic) search strategies

Use problem-specific knowledge beyond what is given in the problem definition

Best-First Search

- Idea: use an evaluation function *f(n)*
  - For each node, gives an 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

Heuristic function

- *heuristic function* $h(n)$ — estimated cost of the cheapest path form the state at node n to a goal state.
- "Heuristic"
  - "proceeding to a solution by trial and error or by rules that are only loosely defined." *
  - "a technique designed to solve a problem that ignores whether the solution can be proven to be correct, but which usually produces a good solution or solves a simpler problem that contains or intersects with the solution of the more complex problem." **

*Oxford American Dictionary; **Wikipedia

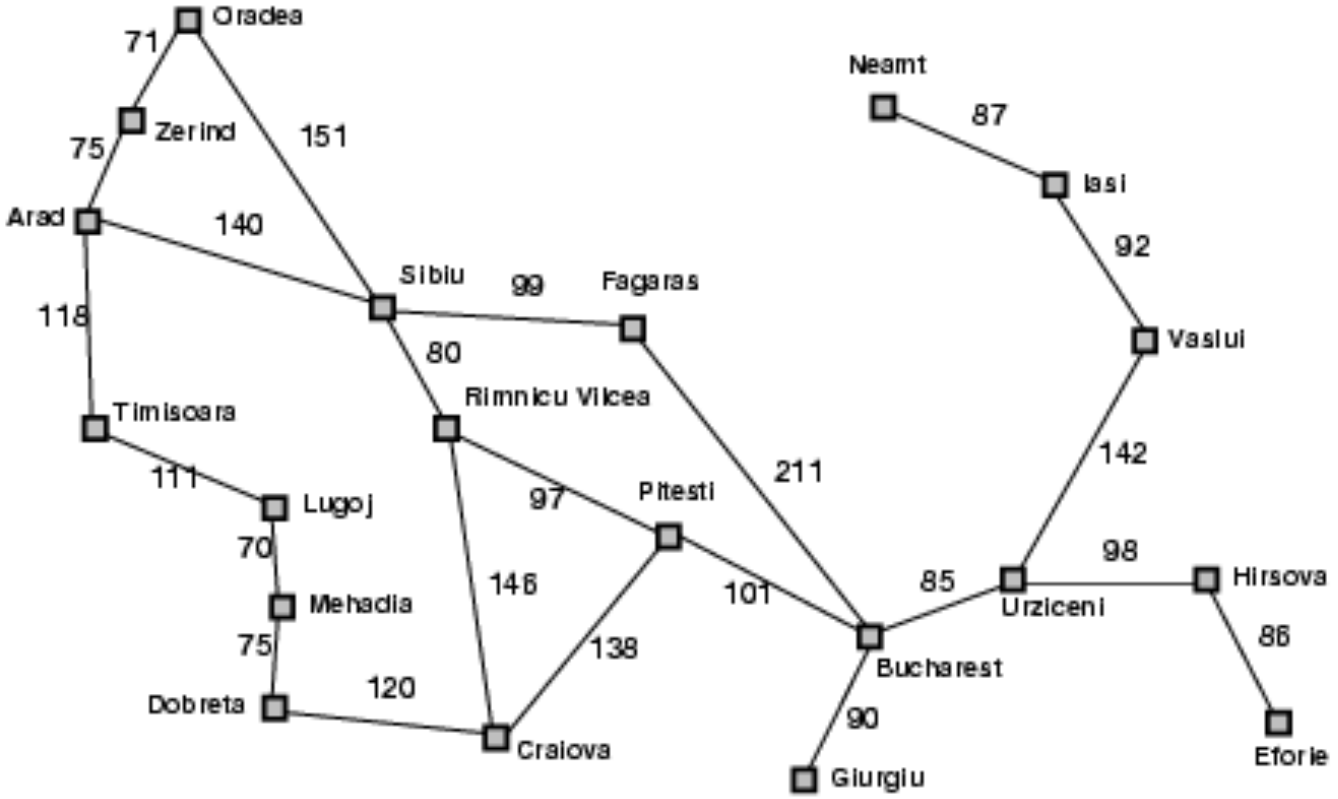
Greedy Best-First

- Evaluation function $f(n) = h(n)$ (heuristic)
  = estimate of cost from *n*'s state to *goal*

- e.g., $h_{SLD}(n)$ = straight-line distance from *n's state* to goal state in a navigation problem

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

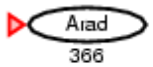
Rumania (with step costs in km)



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

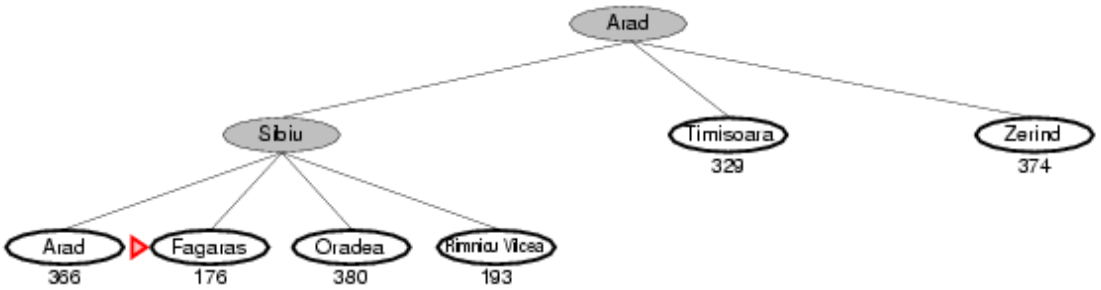
Greedy Best-First Example

Heuristic = h_{SLD}

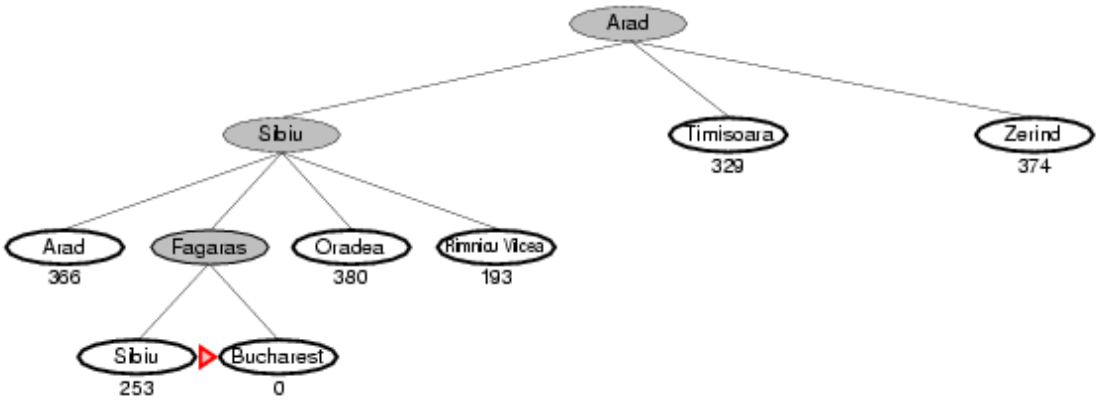


Greedy Best-First Example

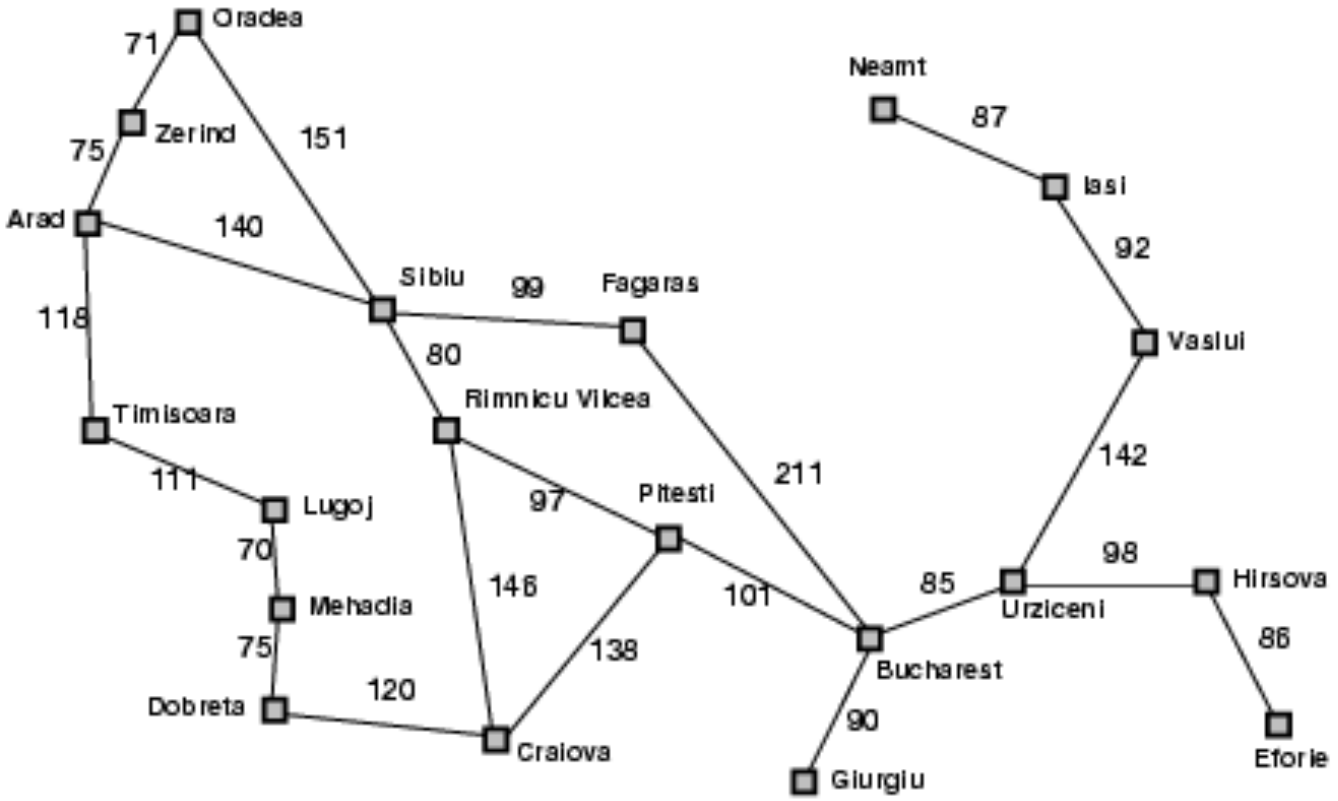
Greedy Best-First Example



Greedy Best-First Example



Rumania (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

Properties of Greedy Best-First Search

- Complete? No – can get stuck in loops, e.g.,
  Iasi → Neamt → Iasi → Neamt → …
  (But graph search version is complete in finite spaces.)

- Time? $O(b^m)$, but a good heuristic can give dramatic improvement (m = max depth of search space)

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

- Optimal? No

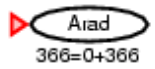
A* Search

- Main Idea: avoid expanding paths that are already expensive

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

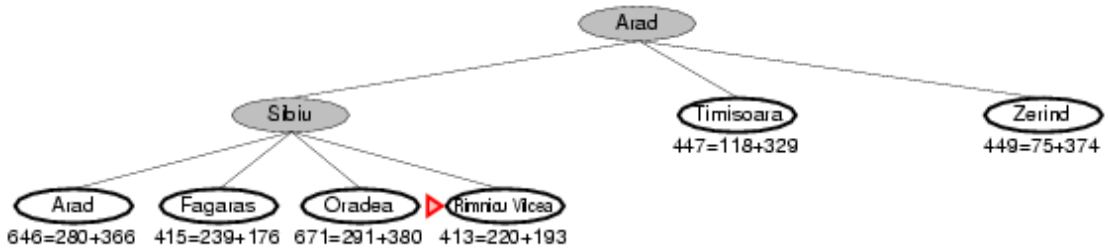
  - $g(n)$ = cost so far to reach n
  - $h(n)$ = estimated cost of cheapest path from n to goal
  - $f(n)$ = estimated total cost of cheapest path through n to goal

A* Search Example

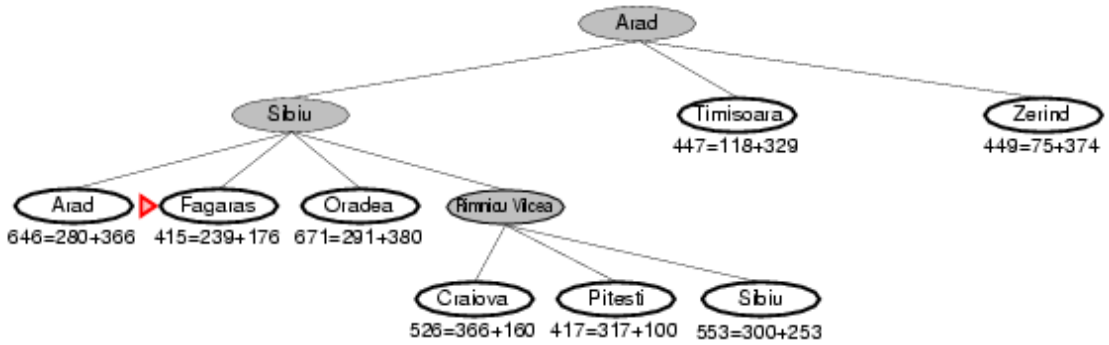


A* Search Example

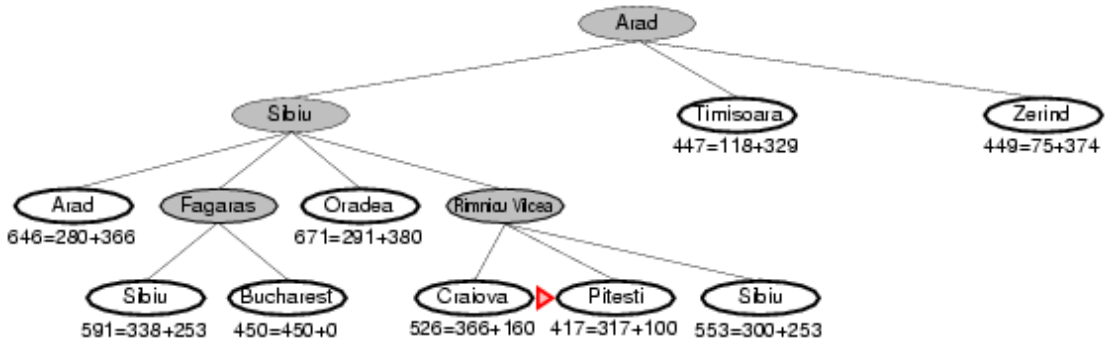
A* Search Example



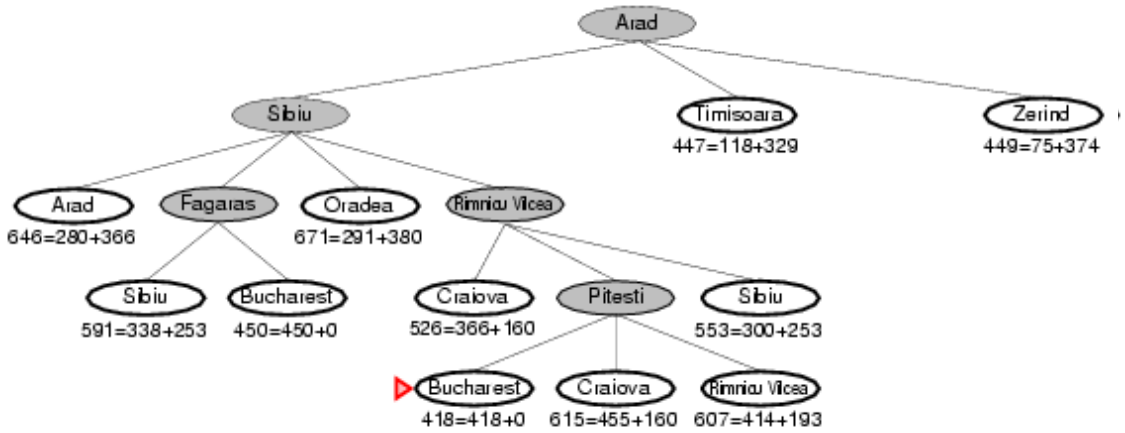
A* Search Example



A* Search Example



A* Search Example



Admissible Heuristics

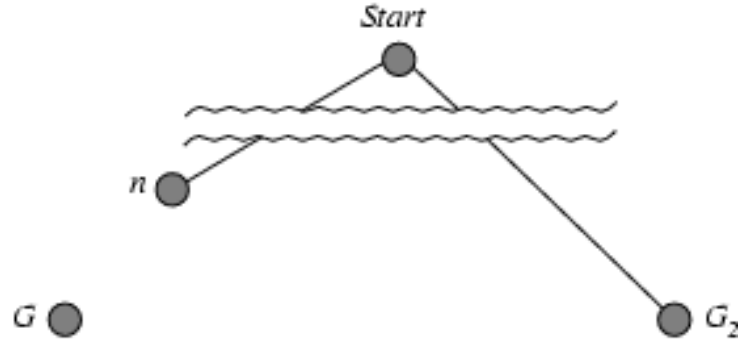
- A heuristic $h(n)$ is admissible if for every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from the state of node n.
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)
- Theorem: If $h(n)$ is admissible, A^* using TREE-SEARCH is optimal

Why Admissibility → Optimality

When A* terminates its search, it has found a path whose actual cost is lower than the estimated cost of any path through any frontier node. But since those estimates are optimistic, A* can safely ignore those nodes. In other words, A* will never overlook the possibility of a lower-cost path and so is optimal.

Optimality of A* (proof fragment)

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



- $f(n) = g(n) + h(n) \leq C^*$ since h is optimistic
- $f(G_2) = g(G_2) > C^*$ since $h(G_2) = 0$ and G_2 is suboptimal

Hence $f(G_2) > f(n)$, and A* will never expand G_2 using TREE-SEARCH

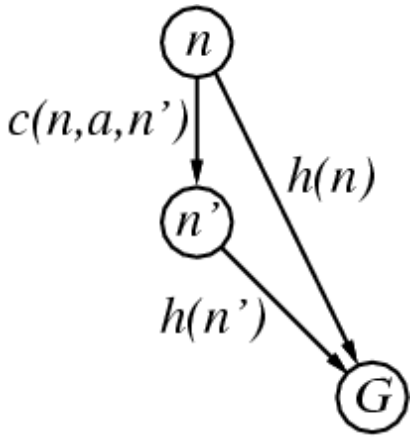
Consistent Heuristics

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

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

- If h is consistent, we have

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



- i.e., $f(n)$ is non-decreasing along any path.

Consistent Heuristics

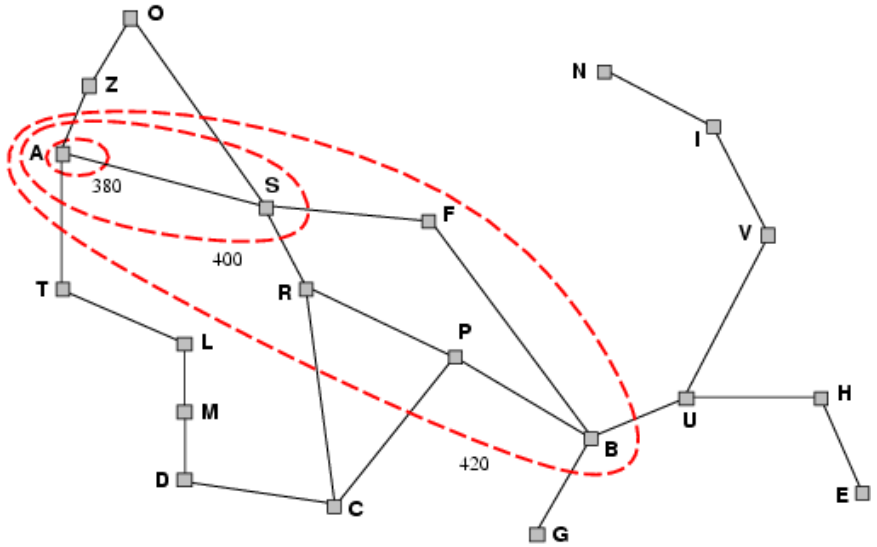
- Theorem: If *h(n)* is consistent, when A* selects a node for expansion, the least cost path to it has already been found.

- Theorem: If *h(n)* is consistent, A* using `GRAPH-SEARCH` is optimal

- Also: consistency ➔ admissibility (but not conversely)

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 $f_i < f_{i+1}$



- So the first goal node expanded must be optimal because f will be the true cost and all later nodes will cost more.

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 if $h(n)$ is admissible and consistent

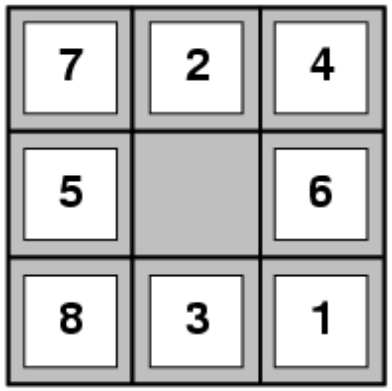
- A* is optimally efficient for any given heuristic function
  - Aside from ties in f, A* expands every node necessary to prove that we've found the shortest path, and no other nodes.

Admissible Heuristics

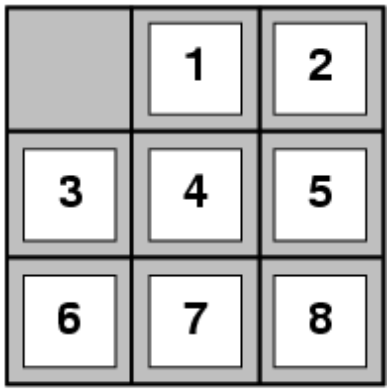
E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



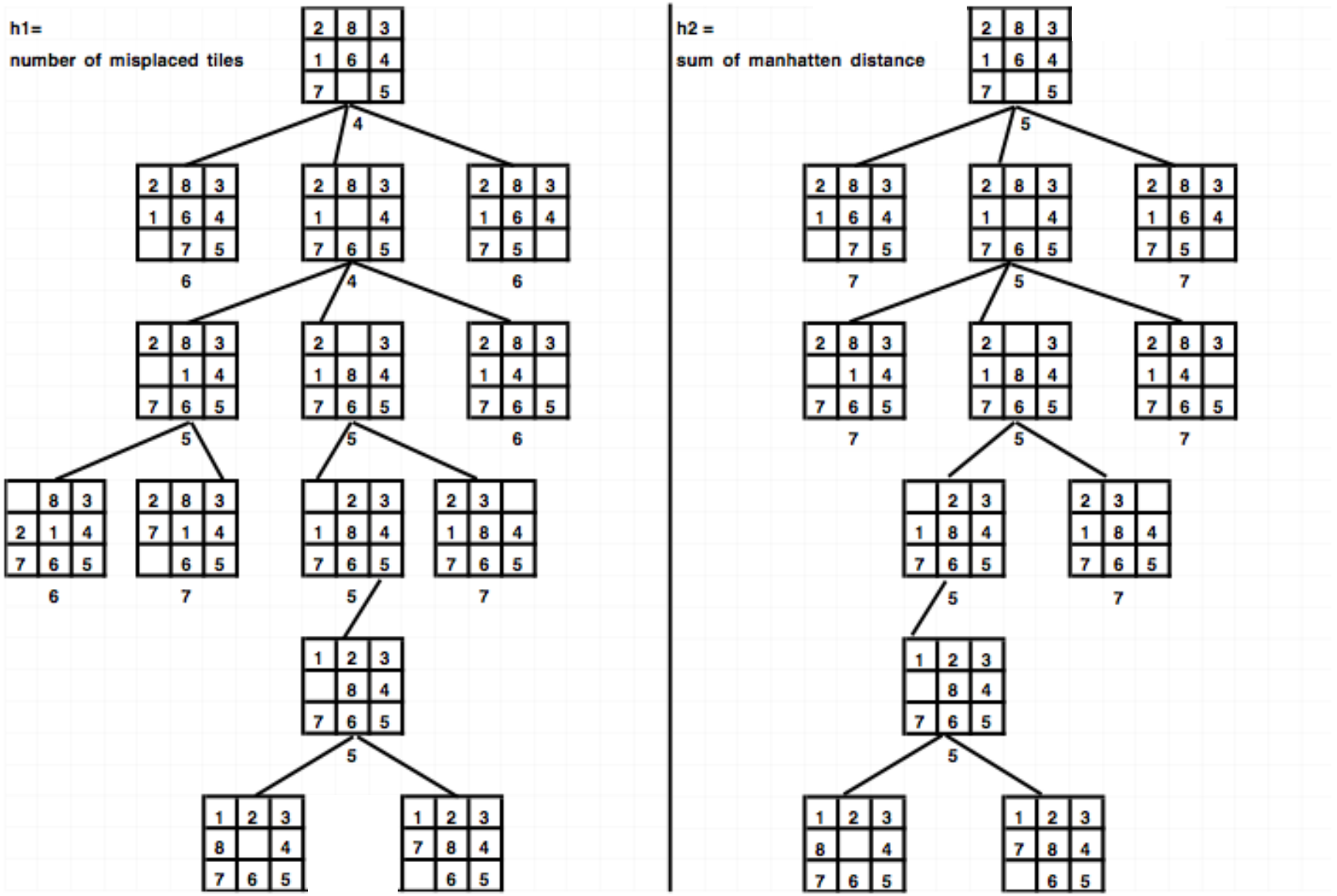
Start State



Goal State

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

Relative benefit of heuristic functions



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

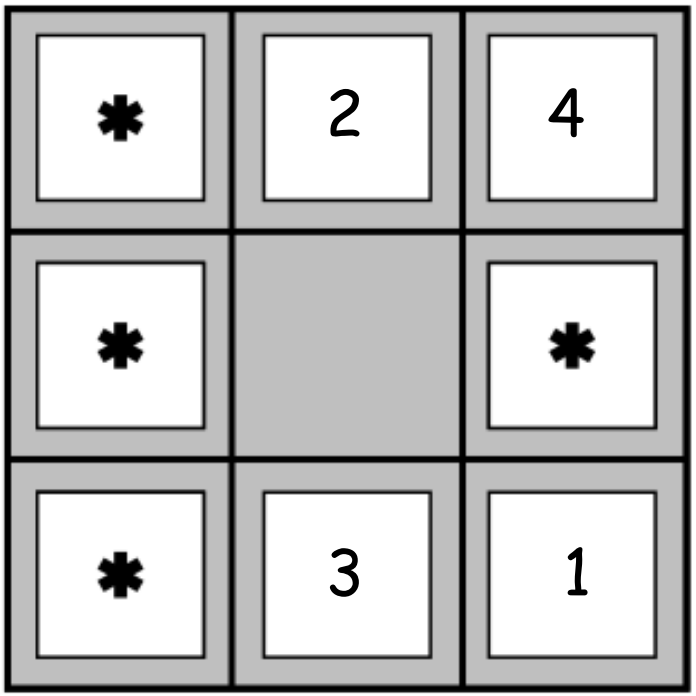
- Typical search costs (average number of nodes expanded):

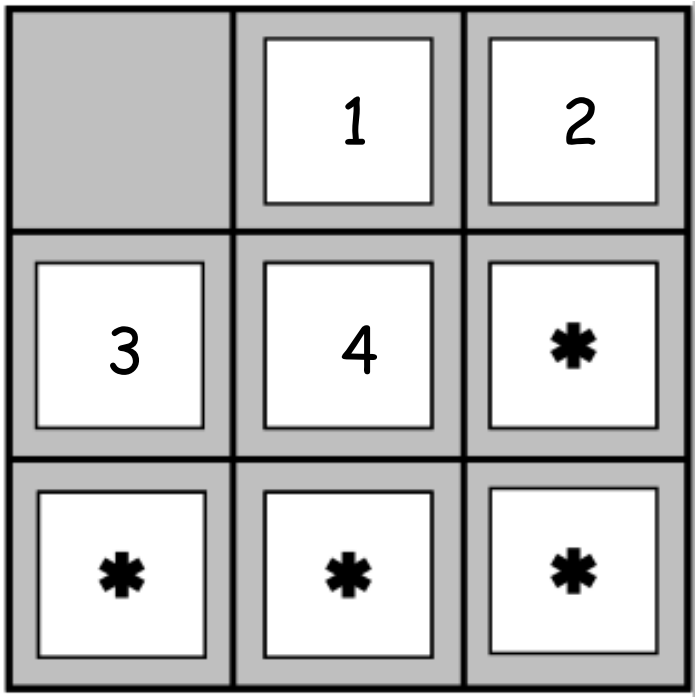
- *d=12* IDS = 3,644,035 nodes
  $A^*(h_1)$ = 227 nodes
  $A^*(h_2)$ = 73 nodes
- *d=24* IDS = too many nodes
  $A^*(h_1)$ = 39,135 nodes
  $A^*(h_2)$ = 1,641 nodes

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

Subproblems





Relative benefit of heuristic functions

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Heuristic search terminology

- State space: set of states that are all reachable from the initial state
- Evaluation function: provides a value for each node in the search tree.
- Greedy best-first search: selects the successor which has the highest value of the evaluation function
- Heuristic function: *estimates* the cost of the lowest-cost path from a state at a node to the goal state
- Admissible: holds for a heuristic if it *never overestimates* the distance to the goal state.
- Consistent: If this condition holds, then the values of the evaluation function along any path are *non-decreasing*.

A* Intuition

When A* terminates its search, it has found a path whose actual cost is lower than the estimated cost of any path through any frontier node. But since those estimates are optimistic, A* can safely ignore those nodes. In other words, A* will never overlook the possibility of a lower-cost path and so is optimal.

Next Class

- Local Search
- 4.1 - 4.2
- Problem set 1 due Sept. 29