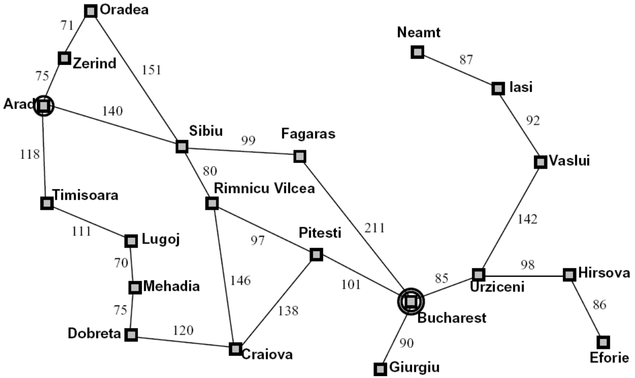
Problem Solving with AI Techniques
Informed Search

Paul Weng

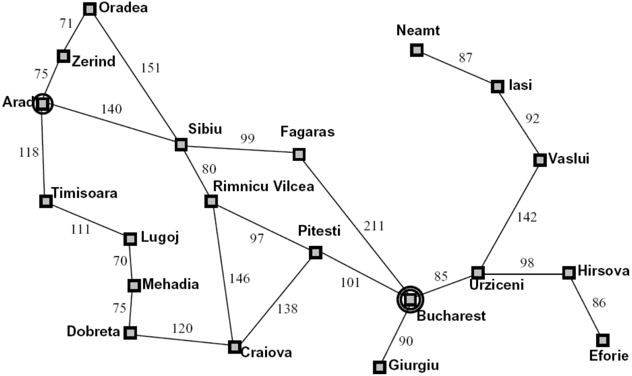
UM-SJTU Joint Institute

VE593, Fall 2018

Examples


Oradea
Zerind
Arad
Sibiu
Fagaras
Timisoara
Rimnicu Vilcea
Lugoj
Pitesti
Mehadia
Dobreta
Craiova
Neamt
Iasi
Vaslui
Urziceni
Hirsova
Bucharest
Giurgiu
Eforie
71
75
151
140
118
99
80
211
111
97
70
146
101
75
138
120
87
92
142
98
85
86
90

Examples



$$
\begin{array}{cccccc}
 & & S & E & N & D \\
+ & & M & O & R & E \\
\hline
= & M & O & N & E & Y
\end{array}
$$

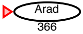
Heuristic Search

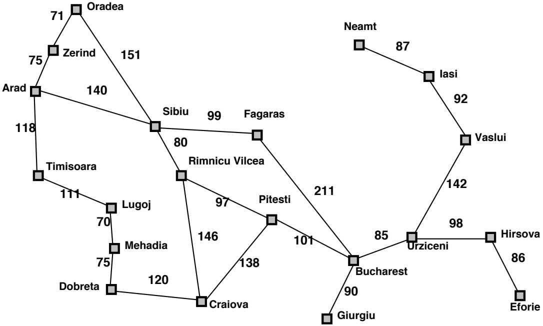
- Heuristic method: technique that can provide a feasible solution, but doesn't generally have any optimality guarantee
- Takes advantage of extra knowledge about goal/problem
- Extra knowledge in the form of a heuristic function h
  - Common sense rules intended to increase probability of solving
  - "Rules of thumb"
  - h estimates cost to reach goal
    e.g., h_{SLD} = straight-line distance from n to Bucharest
- Guides search by heuristic function h

Principle of Greedy Best-First Search

- Strategy: expand node that appear the closest to goal (as measured by h)
- Implementation: same as uniform-cost search using h instead of path costs (i.e., $g(n)$ = cost-so-far to reach a node n)

Example of Greedy Best-First Search

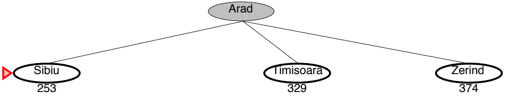


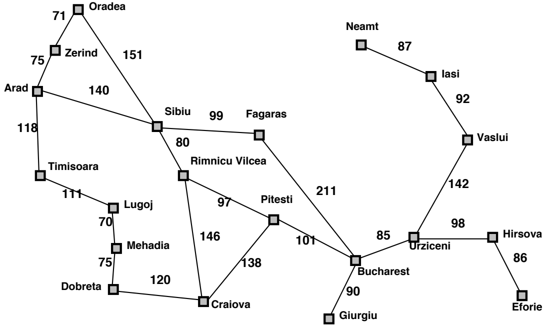


Straight–line distance to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example of Greedy Best-First Search

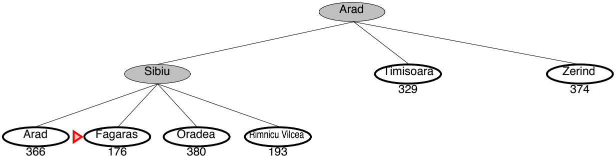


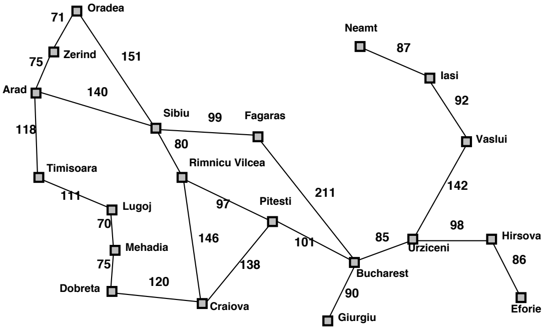


Straight–line distance
to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example of Greedy Best-First Search

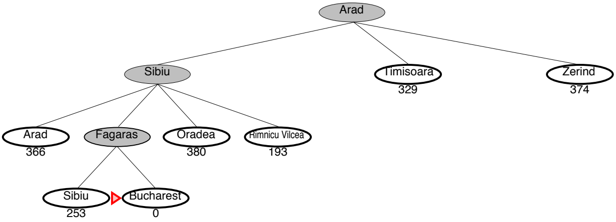


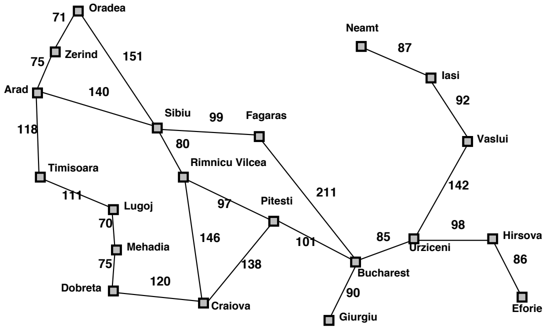


Straight–line distance to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example of Greedy Best-First Search

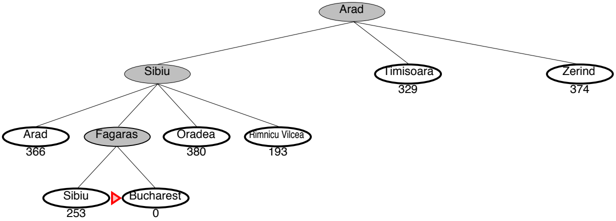


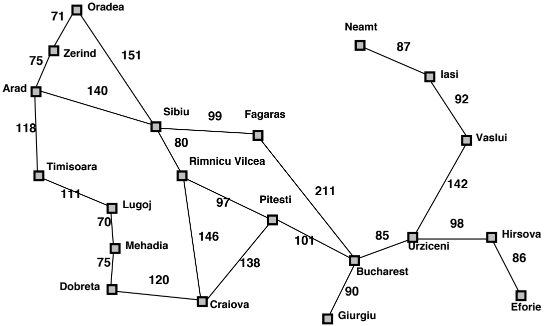


Straight–line distance to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example of Greedy Best-First Search





Straight–line distance
to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

But, going via Rimnicu Vilcea is **shorter**!!

Properties of Greedy Best-First Search

- Complete? No, can get stuck in loops, e.g., with Oradea as goal
  Iasi $\rightarrow$ Neamt $\rightarrow$ Iasi $\rightarrow$ Neamt $\rightarrow$
- Time? $\mathcal{O}(b^m)$, but good heuristic can give dramatic improvement
- Space? $\mathcal{O}(b^m)$, keeps all nodes in memory
- Optimal? No

- Greedy best-first search doesn't care about the "past" (the cost-so-far)

Principle of A* Search

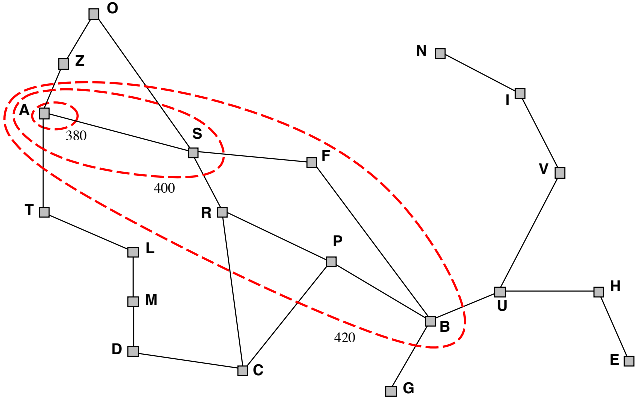
- Greedy best-first search doesn't exploit the information of $g(n)$
- Idea: avoid expanding paths that are already expensive
- Strategy: expand node that appears to be promising wrt

$$f(n) = g(n) + h(n) \quad \text{where}$$

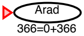
  - $g(n) =$ cost-so-far to reach n
  - $h(n) =$ estimated cost-to-go from n
  - $f(n) =$ estimated total cost of path through n to goal
- Implementation: same as uniform-cost search using $f(n)$

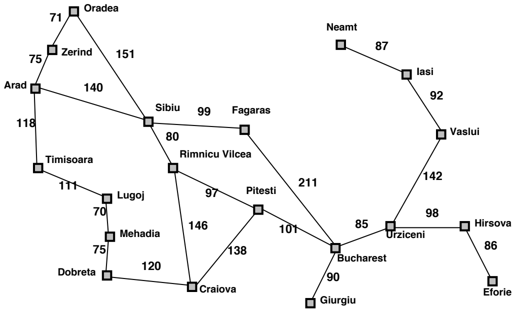
Intuition of A*

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



Example of A* Search

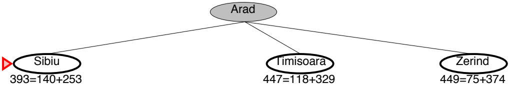


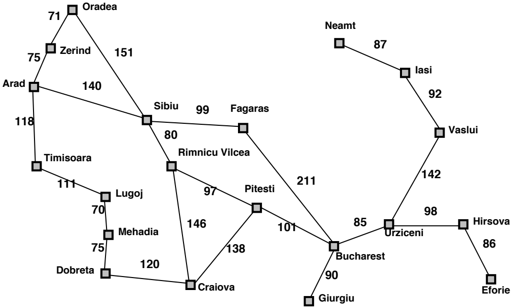


Straight–line distance
to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example of A* Search

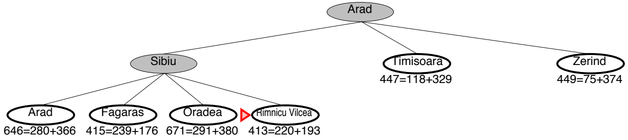


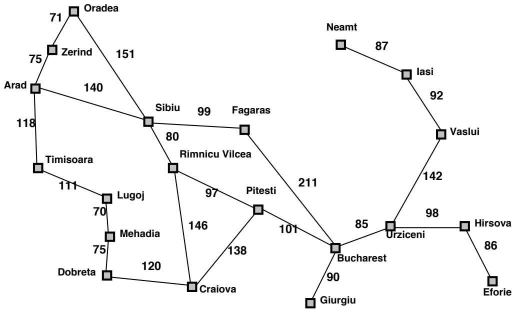


Straight–line distance
to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example of A* Search

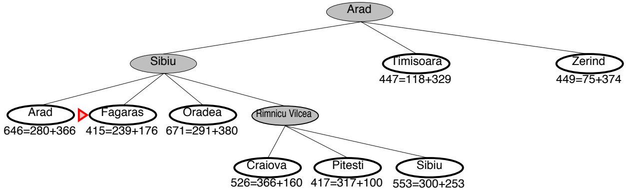


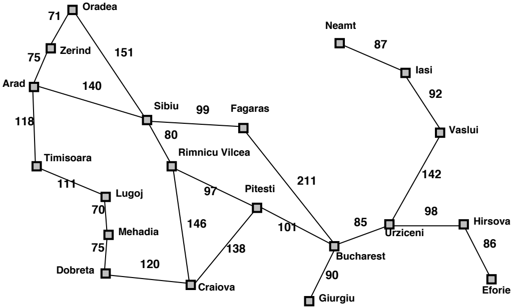


Straight–line distance to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example of A* Search

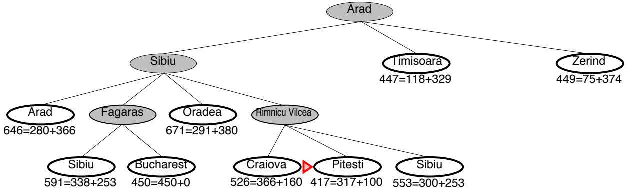


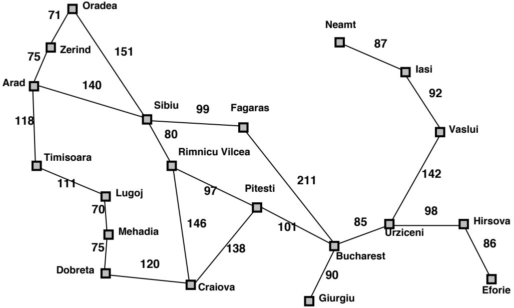


Straight–line distance
to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example of A* Search

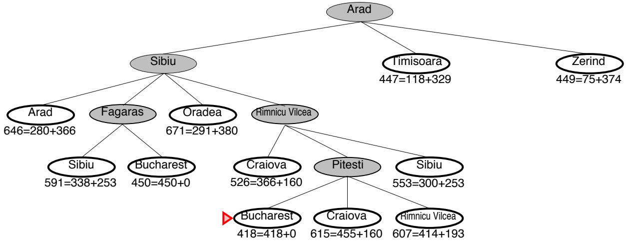


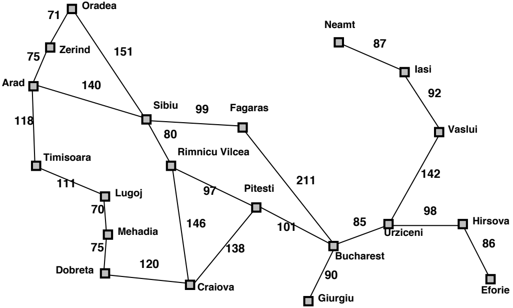


Straight–line distance to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example of A* Search

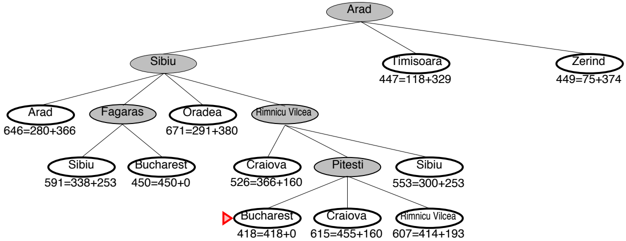


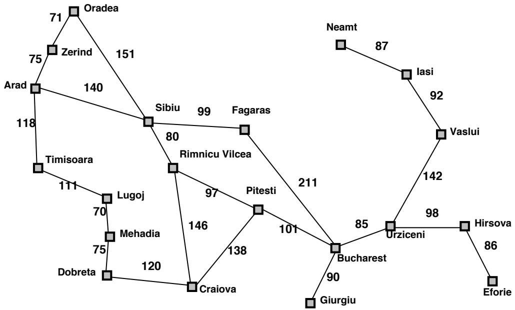


Straight–line distance
to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example of A* Search





Straight–line distance to Bucharest

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
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

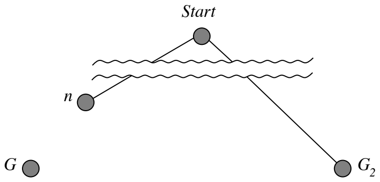
But, is A* always optimal?

Admissible Heuristics

- Heuristic $h(n) \geq 0$ is admissible if for every node n, $h(n) \leq h^*(n)$ where $h^*(n)$ is true cost to reach goal from n
- Admissible heuristic never overestimates cost to reach goal, i.e., it is optimistic
- Example: $h_{SLD}(n)$ never overestimates actual road distance. Why?
- Theorem: If $h(n)$ is admissible, A* using `TREE-SEARCH` is optimal

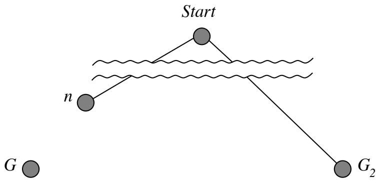
Proof of Optimality of A*

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 the shortest path to an optimal goal G.



Proof of Optimality of A*

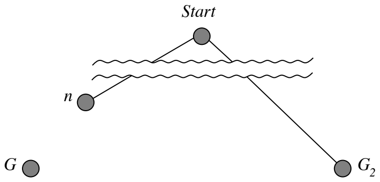
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$

Proof of Optimality of A*

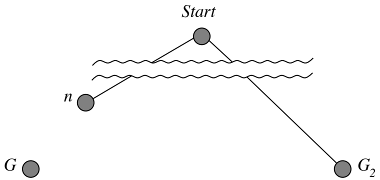
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



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

Proof of Optimality of A*

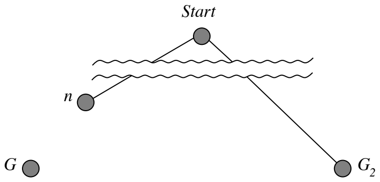
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $g(G_2) > g(G)$

Proof of Optimality of A*

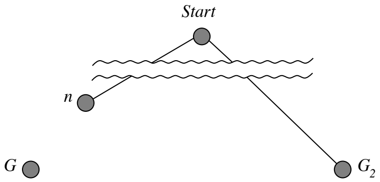
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



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

Proof of Optimality of A*

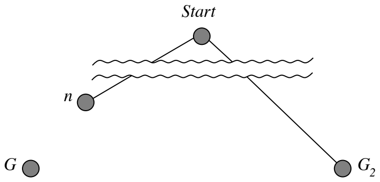
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



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

Proof of Optimality of A*

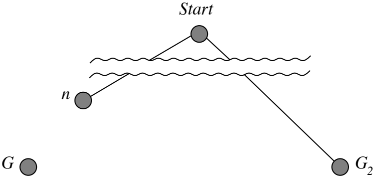
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



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

Proof of Optimality of A*

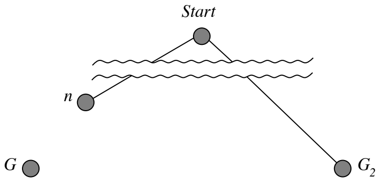
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



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

Proof of Optimality of A*

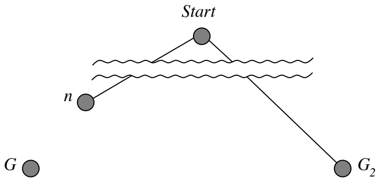
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$ 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

Proof of Optimality of A*

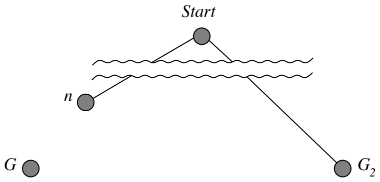
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$ 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
- $h(n) \leq h^*(n)$

Proof of Optimality of A*

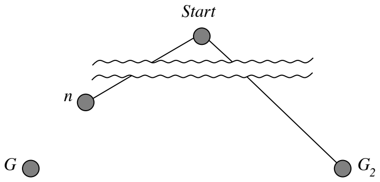
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$ 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
- $h(n) \leq h^*(n)$ since h is admissible

Proof of Optimality of A*

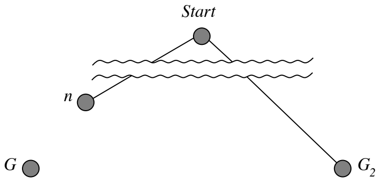
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$ 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
- $h(n) \leq h^*(n)$ since h is admissible
- $g(n) + h(n) \leq g(n) + h^*(n)$

Proof of Optimality of A*

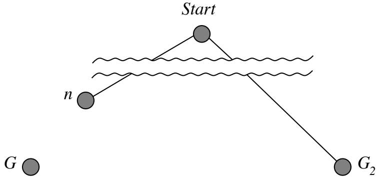
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$ 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
- $h(n) \leq h^*(n)$ since h is admissible
- $g(n) + h(n) \leq g(n) + h^*(n)$ by adding $g(n)$ both sides

Proof of Optimality of A*

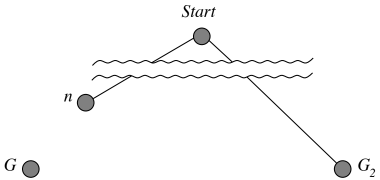
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$ 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
- $h(n) \leq h^*(n)$ since h is admissible
- $g(n) + h(n) \leq g(n) + h^*(n)$ by adding $g(n)$ both sides
- $f(n) \leq f(G)$

Proof of Optimality of A*

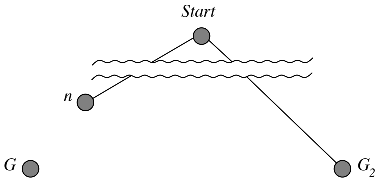
Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on the shortest path to an optimal goal G.



- $f(G_2) = g(G_2)$ 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
- $h(n) \leq h^*(n)$ since h is admissible
- $g(n) + h(n) \leq g(n) + h^*(n)$ by adding $g(n)$ both sides
- $f(n) \leq f(G)$ by definition

Proof of 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 the 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
- $h(n) \leq h^*(n)$ since h is admissible
- $g(n) + h(n) \leq g(n) + h^*(n)$ by adding $g(n)$ both sides
- $f(n) \leq f(G)$ by definition

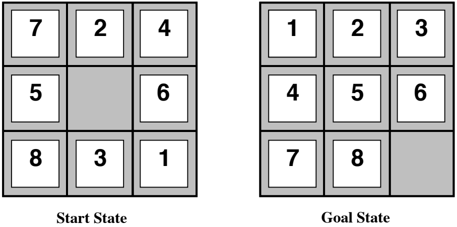
Hence $f(G_2) > f(n)$ and A* would never select G_2 for expansion.

Properties of A* Search

- Complete? Yes, unless there are infinitely many nodes n with $f(n) \leq f(G) = c^*$
- Time? Exponential in [relative error in $h \times$ length of soln]
- Space? Exponential, keeps all nodes in memory
- Optimal? Yes

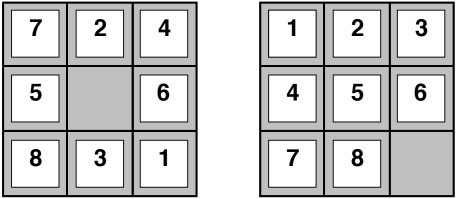
- A* expands all nodes with $f(n) \leq c*$
- A* expands some nodes with $f(n) = c*$
- A* expands no nodes with $f(n) > c*$

Example for the 8-puzzle



- $h_1(n) = \#$ of misplaced tiles

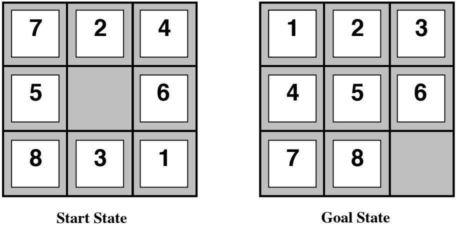
Example for the 8-puzzle



Start State **Goal State**

- $h_1(n) = \#$ of misplaced tiles
- $h_2(n) =$ total Manhattan distance, i.e., $\#$ of squares from desired location of each tile

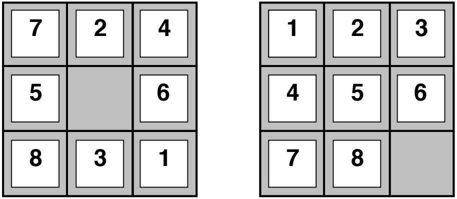
Example for the 8-puzzle



- $h_1(n) = \#$ of misplaced tiles
- $h_2(n) =$ total Manhattan distance, i.e., $\#$ of squares from desired location of each tile

- $h_1(\text{start}) =$

Example for the 8-puzzle

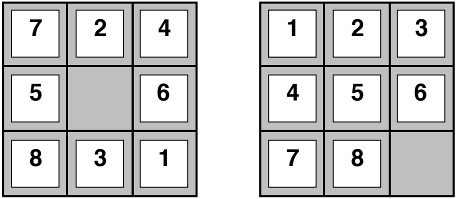


Start State **Goal State**

- $h_1(n) = \#$ of misplaced tiles
- $h_2(n) =$ total Manhattan distance, i.e., $\#$ of squares from desired location of each tile

- $h_1(\text{start}) = 6$

Example for the 8-puzzle

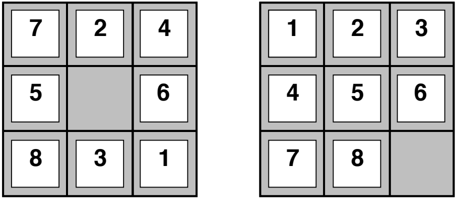


Start State Goal State

- $h_1(n) = \#$ of misplaced tiles
- $h_2(n) =$ total Manhattan distance, i.e., # of squares from desired location of each tile

- $h_1(\text{start}) = 6$
- $h_2(\text{start}) =$

Example for the 8-puzzle



Start State **Goal State**

- $h_1(n) = \#$ of misplaced tiles
- $h_2(n) =$ total Manhattan distance, i.e., # of squares from desired location of each tile

- $h_1(\text{start}) = 6$
- $h_2(\text{start}) = 4+0+3+3+1+0+2+1$
  $=14$

Dominance Relation Between Heuristics

- h_2 dominates h_1 if $\forall n, h_2(n) \geq h_1(n)$ (both admissible)
- h_2 better for search
  e.g., in previous example, h_2 dominates h_1
- Typical search costs (average # 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

- Given any two admissible heuristics h_a and h_b,

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

  is also admissible and dominates h_a, h_b

Relaxed Problems

- Problem with fewer restrictions on actions is called relaxed problem
- Cost of an optimal solution to relaxed problem = admissible heuristic for original problem
- h_1 optimal cost if rules of 8-puzzle relaxed so that a tile can move anywhere
- h_2 optimal cost if rules of 8-puzzle relaxed so that a tile can move to any adjacent square
- Key point: the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

1. A Priori Knowledge

2. Heuristic Search
   - Motivation
   - Greedy Best-First Search
   - A* Search
   - How to Define Good Heuristics?
   - Variants of A*

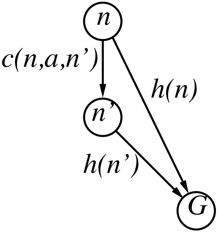
Consistent Heuristics

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

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

- 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 (hence, monotone)
- Theorem: If $h(n)$ is consistent, A* using `GRAPH-SEARCH` is optimal
Keeps all checked nodes in memory to avoid repeated states.

Principle of Simple Memory-Bounded A* (SMA*)

- As with BFS, A* has exponential space complexity
- Strategy:
  - Expand as usual until a memory bound is reach
  - Then, whenever adding a node, remove the *worst* node n' from tree
  - Worst means: n' with highest $f(n')$
  - To not lose information, *backup* the measured step-cost $cost(\tilde{n}, a, n')$ to improve the heuristic $h(\tilde{n})$ of its parent
- Properties: complete and optimal if the depth of the optimal path is within the memory bound