Informed search

CE417: Introduction to Artificial Intelligence
Sharif University of Technology
Spring 2016

Soleymani

“Artificial Intelligence: A Modern Approach”, Chapter 3

Outline

- Best-first search
- Greedy best-first search
- A^* search
- Finding heuristics

Informed search

- When exhaustive search is impractical, heuristic methods are used to speed up the process of finding a satisfactory (or optimal) solution.

Best-first search

- Idea: use an evaluation function $f(n)$ for each node and expand the most desirable unexpanded node
  - More general than "$g(n)$ = cost so far to reach n"
  - Evaluation function provides an upper bound on the desirability (lower bound on the cost) that can be obtained through expanding a node

- Implementation: priority queue with decreasing order of desirability (search strategy is determined based on evaluation function)

- Special cases:
  - Greedy best-first search
  - A* search
  - Uniform-cost search

Heuristic Function

- Incorporating problem-specific knowledge in search
  - Information more than problem definition
  - In order to come to an optimal solution as rapidly as possible

- Heuristic function can be used as a component of $f(n)$

- $h(n)$: estimated cost of cheapest path from n to a goal
  - Depends only on n (not path from root to n)
  - If n is a goal state then $h(n)$=0
  - $h(n) \geq 0$

- Examples of heuristic functions include using a rule-of-thumb, an educated guess, or an intuitive judgment

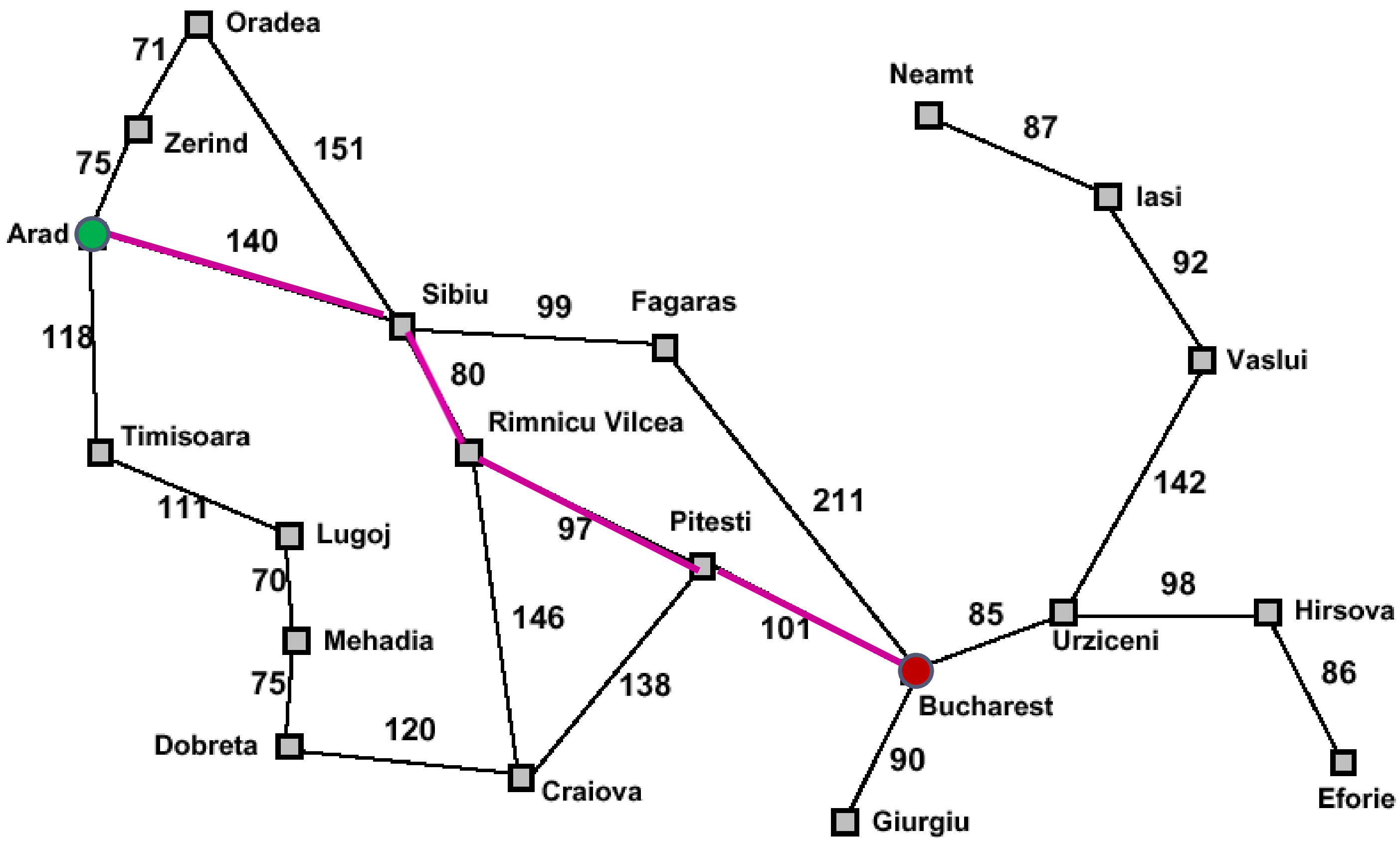
Greedy best-first search

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

Romania with step costs in km

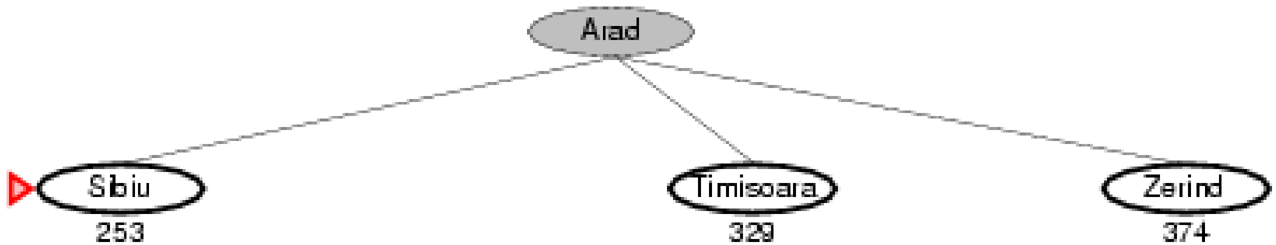


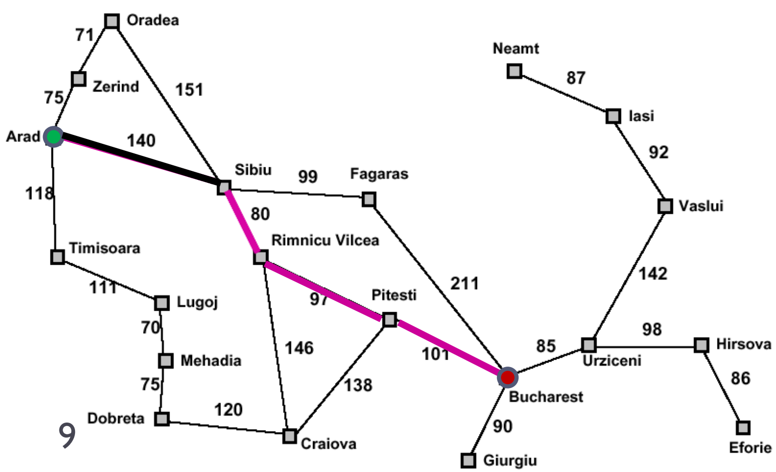
Straight–line distance to Bucharest

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

Greedy best-first search example

Greedy best-first search example

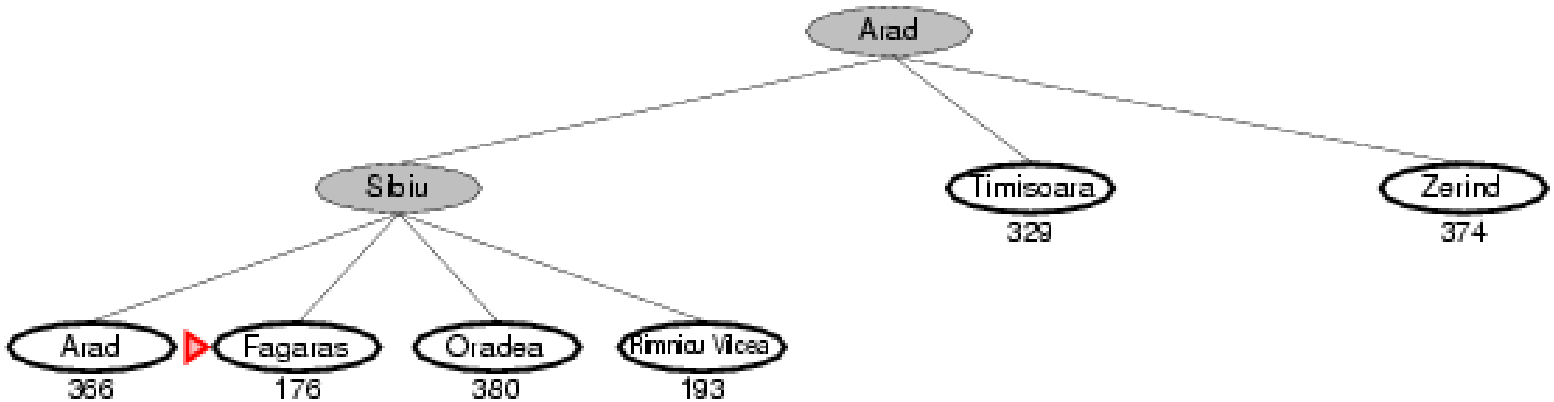


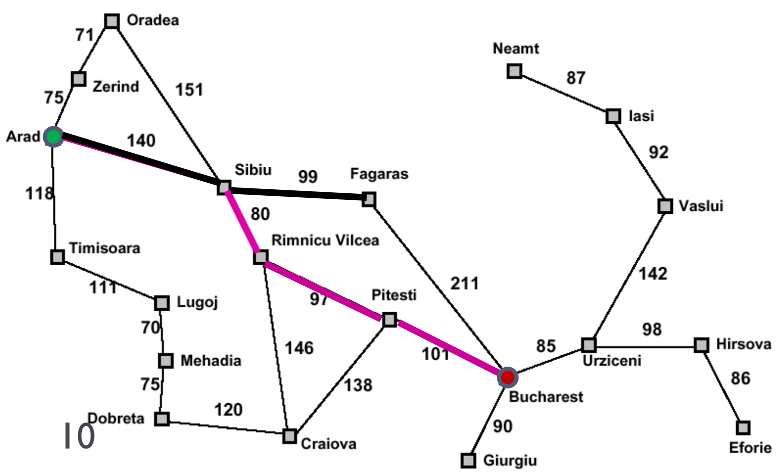


Straight−line distance
to Bucharest

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

Greedy best-first search example

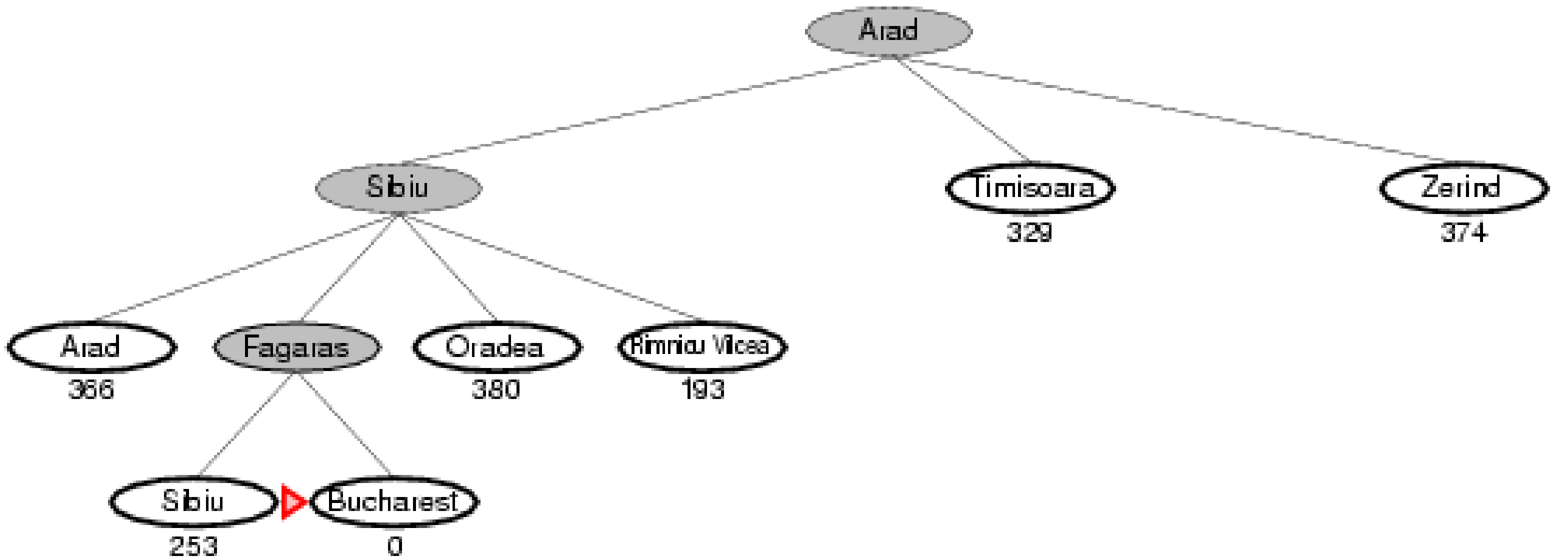


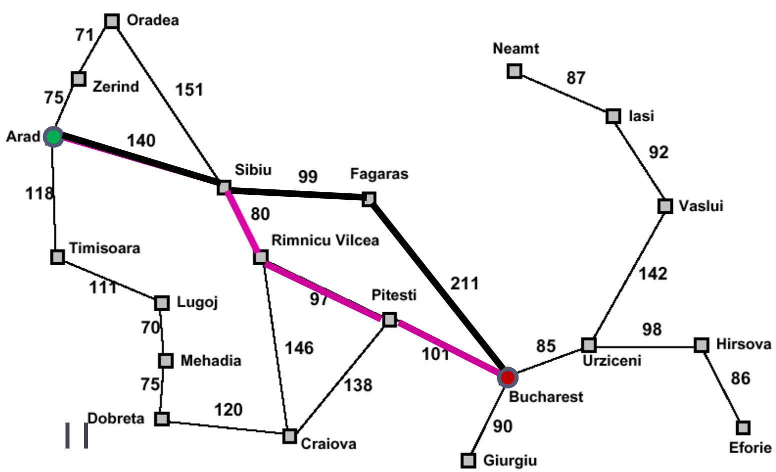


Straight–line distance to Bucharest

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

Greedy best-first search example





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

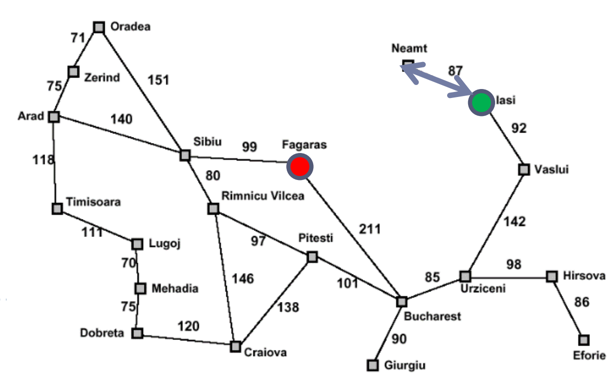
Properties of greedy best-first search

- Complete? No
  - Similar to DFS, only graph search version is complete in finite spaces
  - Infinite loops, e.g., (Iasi to Fagaras) 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

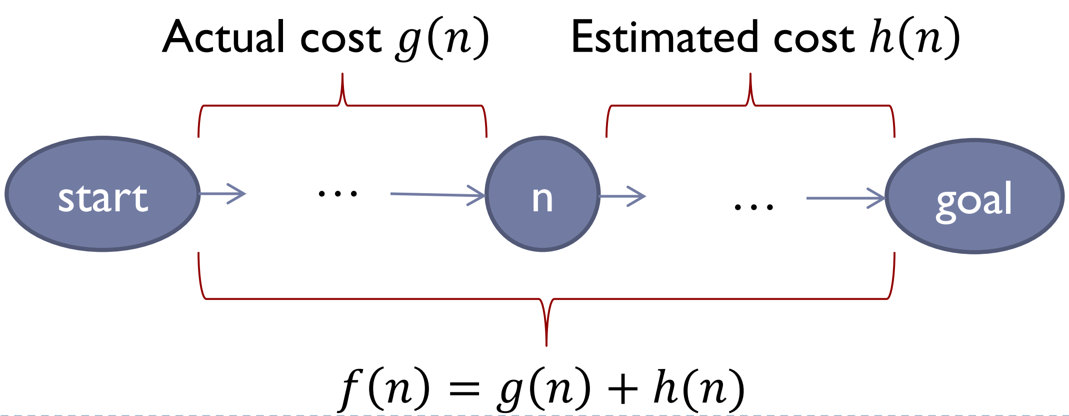
- Optimal? No



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

A* search

- Idea: minimizing the total estimated solution cost
- Evaluation function $f(n) = g(n) + h(n)$
  - $g(n)$ = cost so far to reach n
  - $h(n)$ = estimated cost of the cheapest path from n to goal
  - So, $f(n)$ = estimated total cost of path through n to goal

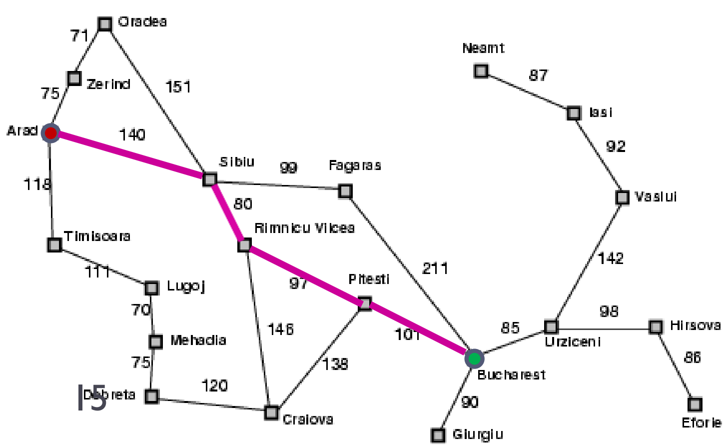


A* search

- Combines advantages of uniform-cost and greedy searches

- A* can be complete and optimal when $h(n)$ has some properties

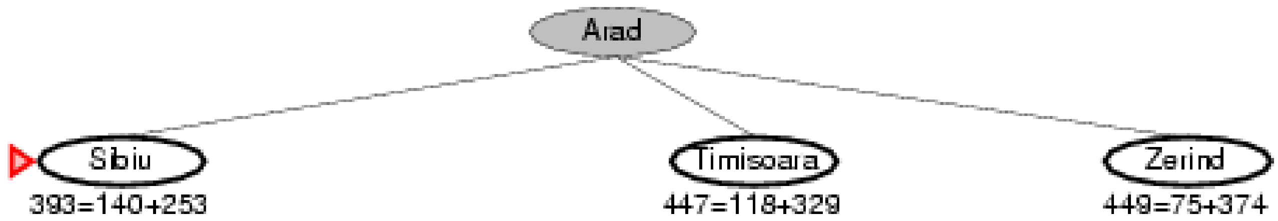
A* search: example

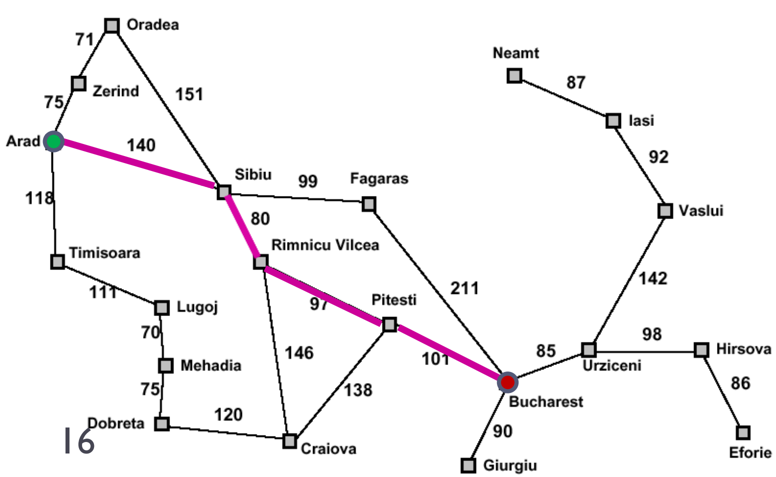


Straight–line distance to Bucharest

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

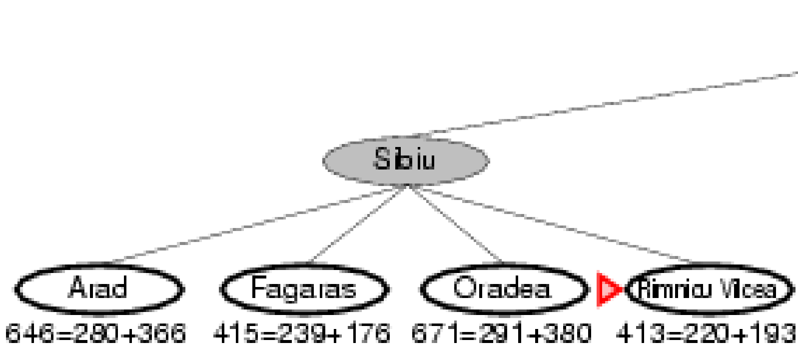


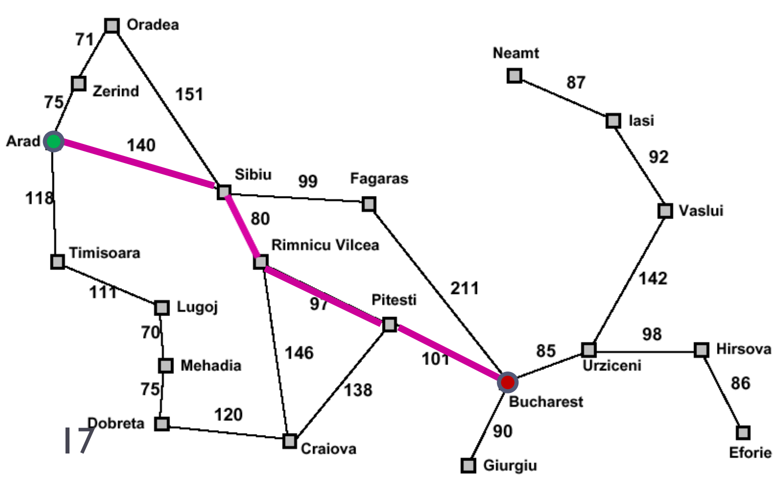


Straight–line distance to Bucharest

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

A* search: example

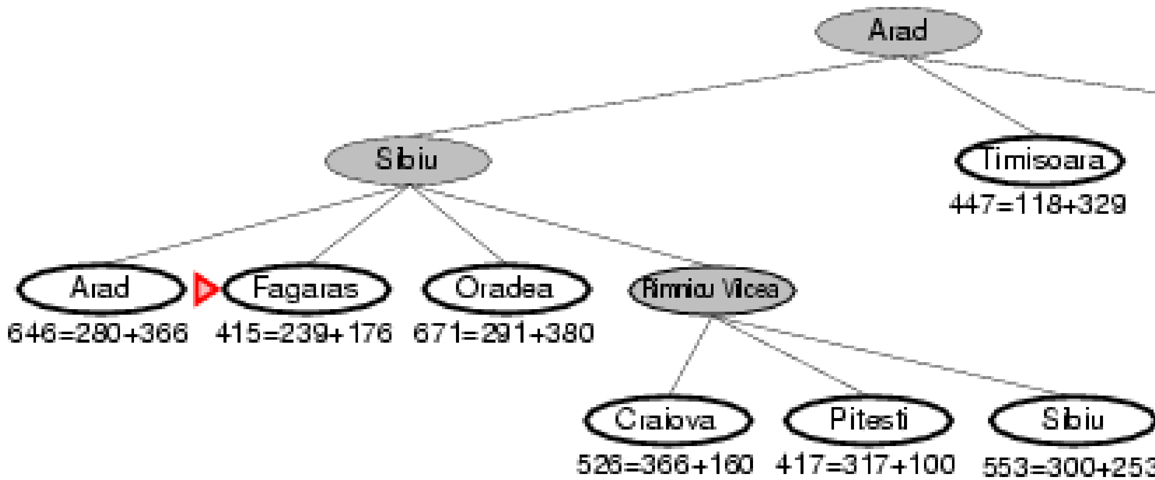


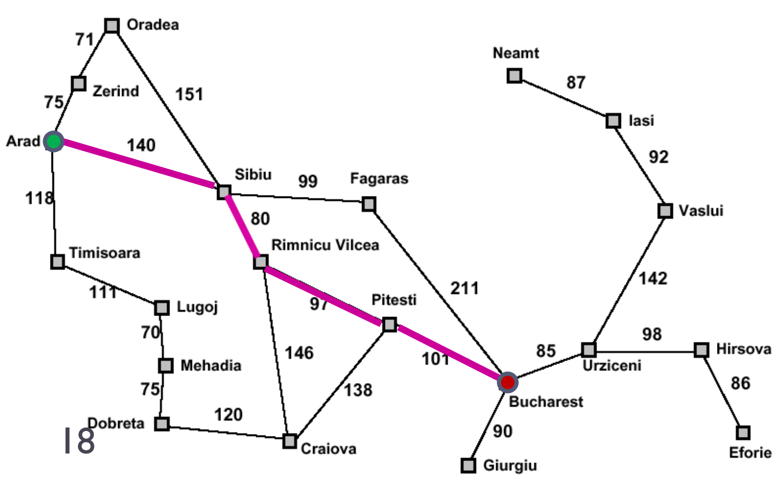


Straight–line distance
to Bucharest

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

A* search: example

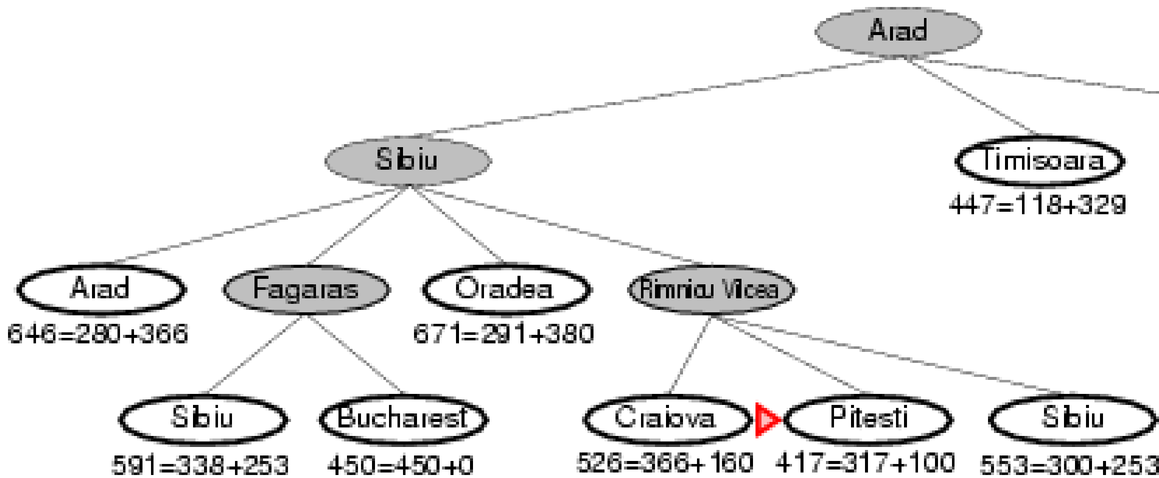


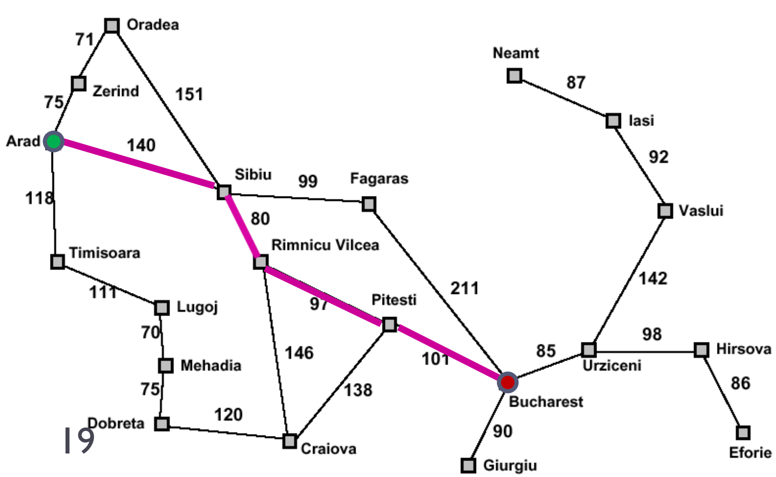


Straight−line distance
to Bucharest

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

A* search: example

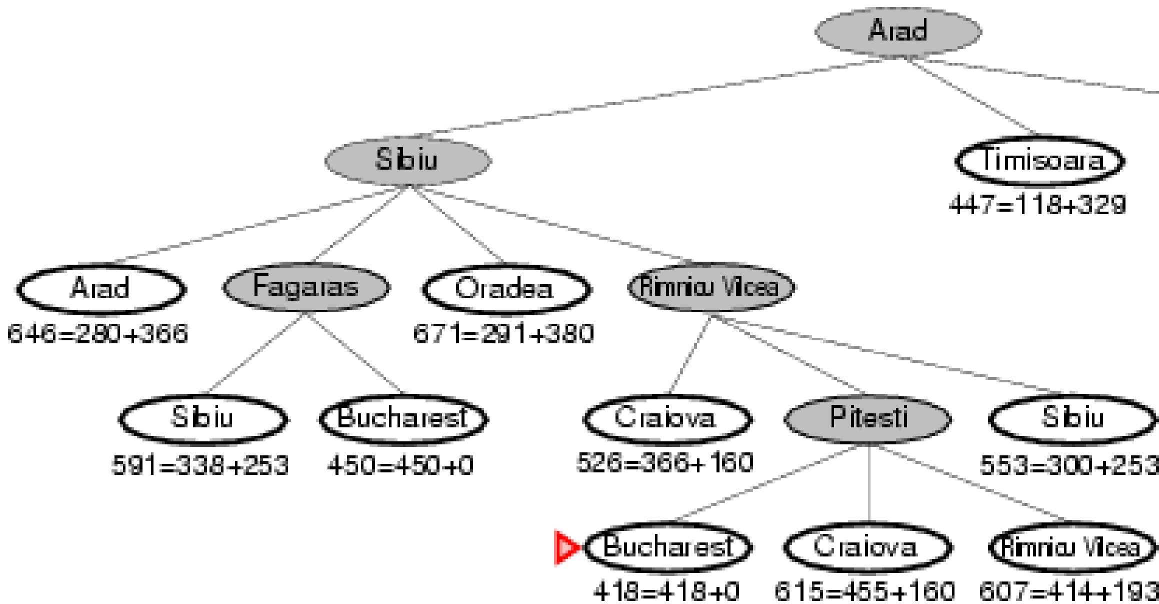


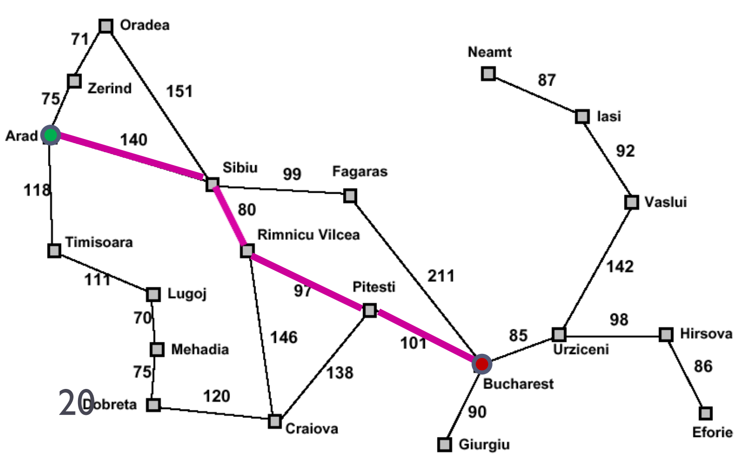


Straight–line distance to Bucharest

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

A* search: example





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

Conditions for optimality of A*

- Admissibility: $h(n)$ be a lower bound on the cost to reach goal
  - Condition for optimality of TREE−SEARCH version of A*
- Consistency (monotonicity): $h(n) \leq c(n, a, n') + h(n')$
  - Condition for optimality of GRAPH−SEARCH version of A*

Admissible heuristics

- Admissible heuristic $h(n)$ never overestimates the cost to reach the goal (optimistic)
  - $h(n)$ is a lower bound on path cost from n to goal

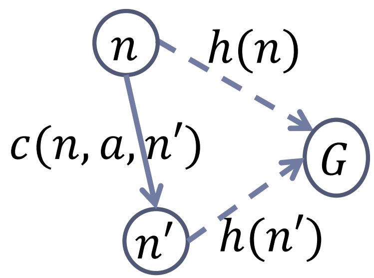
$$\forall n, h(n) \leq h^*(n)$$

  where $h^*(n)$ is the real cost to reach the goal state from n

    - Example: $h_{SLD}(n) \leq$ the actual road distance

Consistent heuristics

- Triangle inequality



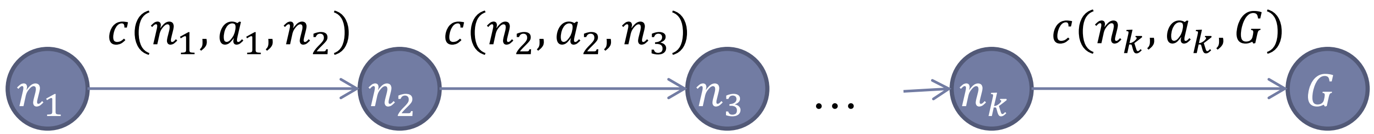
for every node n and every successor n' generated by any action a

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

$c(n,a,n')$: cost of generating n' by applying action to n

Consistency vs. admissibility

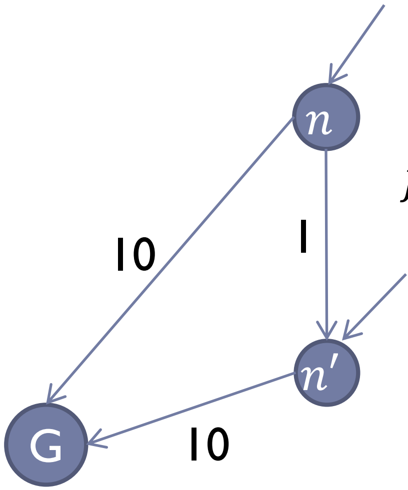
- Consistency ⇒ Admissblity
  - All consistent heuristic functions are admissible
  - Nonetheless, most admissible heuristics are also consistent



$$
\begin{aligned}
h(n_1) &\le c(n_1, a_1, n_2) + h(n_2) \\
&\le c(n_1, a_1, n_2) + c(n_2, a_2, n_3) + h(n_3) \\
&\dots \\
&\le \textstyle\sum_{i=1}^{k} c(n_i, a_i, n_{i+1}) + 0
\end{aligned}
$$

$\Rightarrow h(n_1) \le$ cost of (every) path from n_1 to goal
$\le$ cost of optimal path from n_1 to goal

Admissible but not consistent: Example



$g(n) = 5$
$h(n) = 9$
$f(n) = 14$

$g(n') = 6$
$h(n') = 6$
$f(n') = 12$

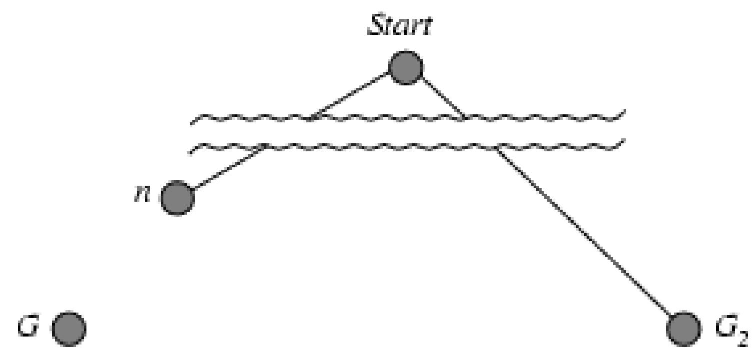
$c(n, a, n') = 1$
$h(n) = 9$
$h(n') = 6$
$\Longrightarrow h(n) \nleq h(n') + c(n, a, n')$

- f (for admissible heuristic) may decrease along a path
- Is there any way to make h consistent?

$$\bar{h}(n') = \max(h(n'), \bar{h}(n) - c(n, a, n'))$$

Optimality of A* (admissible heuristics)

- **Theorem**: If $h(n)$ is admissible, A* using `TREE-SEARCH` is optimal
- Assumptions: G_2 is a suboptimal goal in the frontier, n is an unexpanded node in the frontier and it is on a shortest path to an optimal goal G.



I. $h(G_2) = 0 \Rightarrow f(G_2) = g(G_2)$

II. $h(G) = 0 \Rightarrow f(G) = g(G)$

III. G_2 is suboptimal $\Rightarrow g(G_2) > g(G)$

IV. I, II, III $\Rightarrow f(G_2) > f(G)$

V. h is admissible $\Rightarrow h(n) \le h^*(n)$

$$\Rightarrow g(n) + h(n) \le g(n) + h^*(n)$$

$$\Rightarrow f(n) \le f(G) \overset{IV}{\Rightarrow} f(n) < f(G_2)$$

A* will never select G_2 for expansion

Optimality of A* (consistent heuristics)

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

Lemma1: if $h(n)$ is consistent then $f(n)$ values are non-decreasing along any path

Proof: Let n' be a successor of n

I. $f(n') = g(n') + h(n')$
II. $g(n') = g(n) + c(n, a, n')$
III. $I, II \Rightarrow f(n') = g(n) + c(n, a, n') + h(n')$
IV. $h(n)$ is consistent $\Rightarrow h(n) \leq c(n, a, n') + h(n')$
V. $III, IV \Rightarrow f(n') \geq g(n) + h(n) = f(n)$

Optimality of A* (consistent heuristics)

⇒The sequence of nodes expanded by A* is in non-decreasing order of $f(n)$

Since $h = 0$ for goal nodes (f is the true cost for goal nodes), the first selected goal node for expansion provides an optimal solution. Indeed, we cannot reach a goal node with lower value of f because of the non-decreasing order of f.

We can also show that:

If A* selects a node n for expansion, the optimal solution to that node has been found.

Proof by contradiction: Another frontier node n' must exist on the optimal path from initial node to n (using graph separation property). Moreover, based on Lemma 1, $f(n') \leq f(n)$ and thus n' would have been selected first.

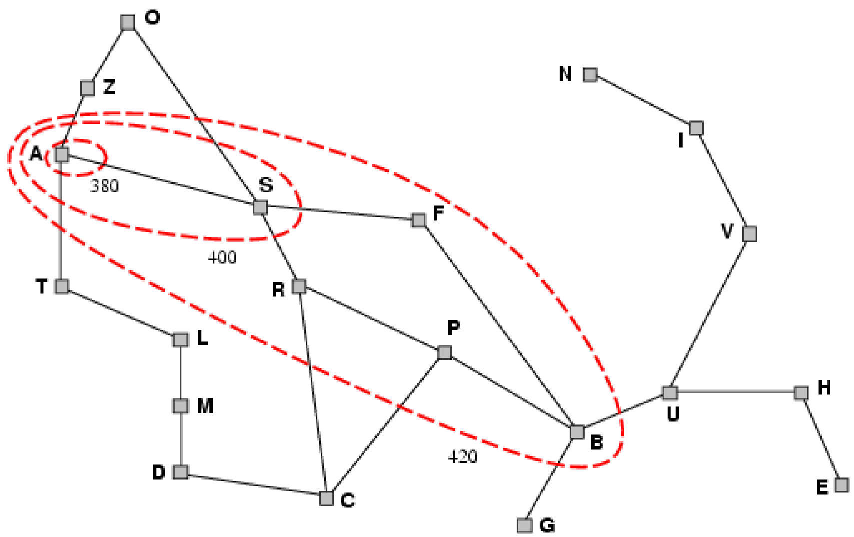
Admissible vs. consistent (tree vs. graph search)

- Consistent heuristic: When selecting a node for expansion, the path with the lowest cost to that node has been found

- When an admissible heuristic is not consistent, a node will need repeated expansion, every time a new best (so-far) cost is achieved for it.

Contours in the state space

- A^* (using `GRAPH-SEARCH`) expands nodes in order of increasing f value
- Gradually adds "f-contours" of nodes
  - Contour i has all nodes with $f = f_i$ where $f_i < f_{i+1}$

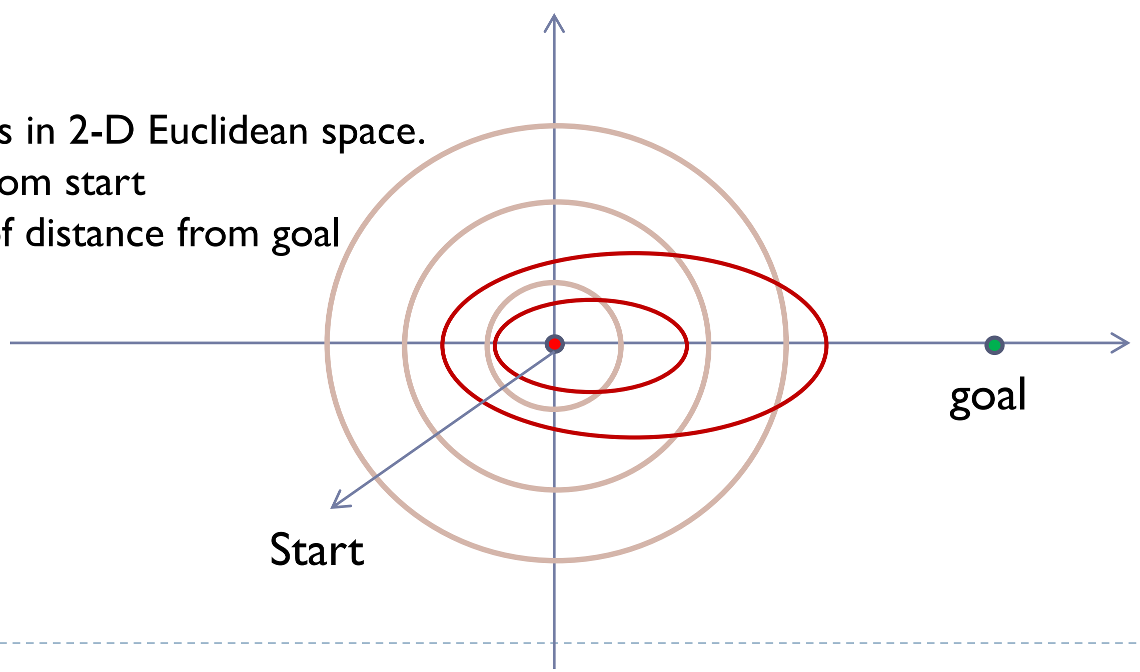


A* expands all nodes with f(n) < C*
A* expands some nodes with f(n) = C* (nodes on the goal contour)
A* expands no nodes with f(n) > C* $\Longrightarrow$ pruning

A^* search vs. uniform cost search

- Uniform-cost search (A^* using $h(n) = 0$) causes circular bands around initial state
- A^* causes irregular bands
  - More accurate heuristics stretched toward the goal (more narrowly focused around the optimal path)

States are points in 2-D Euclidean space.
g(n)=distance from start
h(n)=estimate of distance from goal

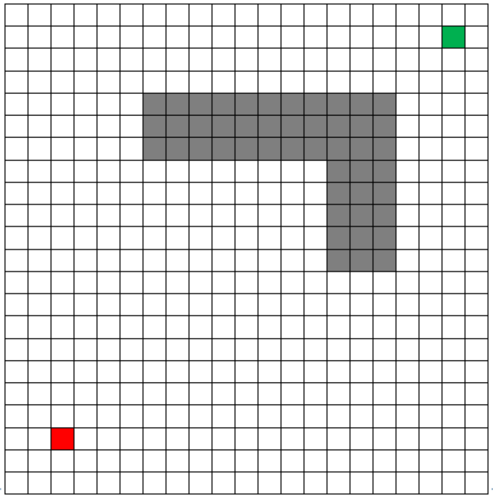


Properties of A*

- Complete?
  - Yes if nodes with $f \leq f(G) = C^*$ are finite
    - Step cost$\geq \varepsilon > 0$ and b is finite
- Time?
  - Exponential
    - But, with a smaller branching factor
      - b^{h^*-h} or when equal step costs $b^{d\times\frac{h^*-h}{h^*}}$
    - Polynomial when $|h(x) - h^*(x)| = O(log\ h^*(x))$
  - However, A* is optimally efficient for any given consistent heuristic
    - No optimal algorithm of this type is guaranteed to expand fewer nodes than A* (except to node with $f = C^*$)
- Space?
  - Keeps all leaf and/or explored nodes in memory
- Optimal?
  - Yes (expanding node in non-decreasing order of f)

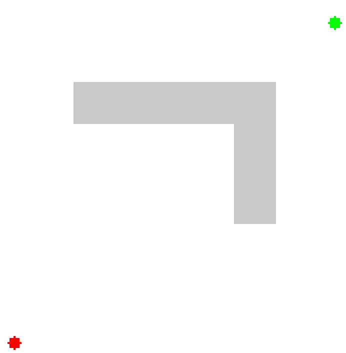
Robot navigation example

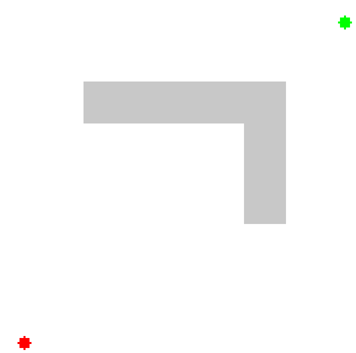
- Initial state? Red cell
- States? Cells on rectangular grid (except to obstacle)
- Actions? Move to one of 8 neighbors (if it is not obstacle)
- Goal test? Green cell
- Path cost? Action cost is the Euclidean length of movement



A* vs. UCS: Robot navigation example

- Heuristic: Euclidean distance to goal
- Expanded nodes: filled circles in red & green
  - Color indicating g value (red: lower, green: higher)
- Frontier: empty nodes with blue boundary
- Nodes falling inside the obstacle are discarded



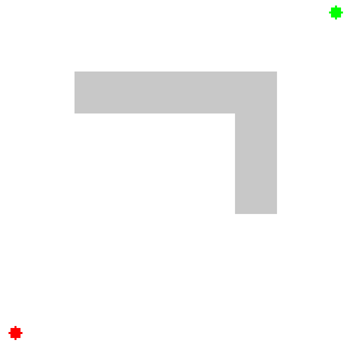


Adopted from: http://en.wikipedia.org/wiki/Talk%3AA*_search_algorithm

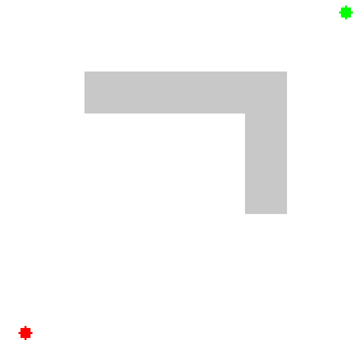
Robot navigation: Admissible heuristic

- Is Manhattan $d_M(x, y) = |x_1 - y_1| + |x_2 - y_2|$ distance an admissible heuristic for previous example?

A*: inadmissible heuristic



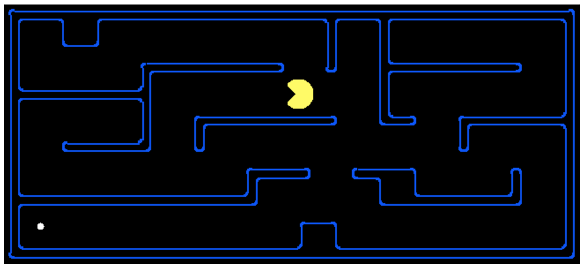
$h = 5 * h\_SLD$



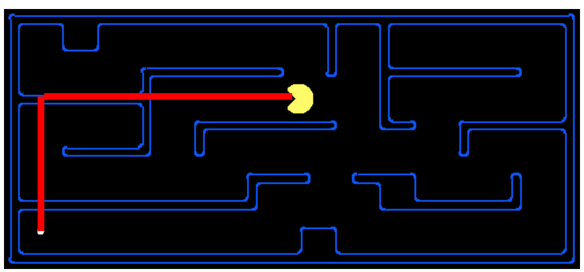
$h = h\_SLD$

Adopted from: http://en.wikipedia.org/wiki/Talk%3AA*_search_algorithm

A*, Greedy, UCS: Pacman

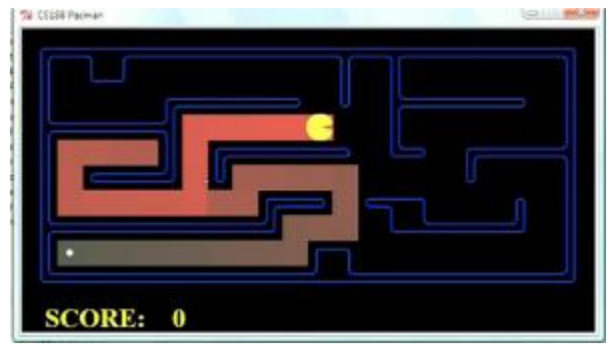


Heuristic: Manhattan distance



Color: expanded in which iteration (red: lower)

UCS



Greedy

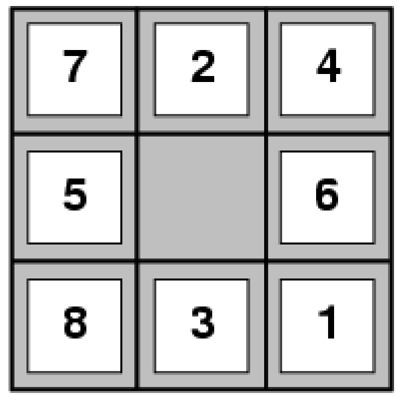
A*

Adapted from Dan Klein’s slides

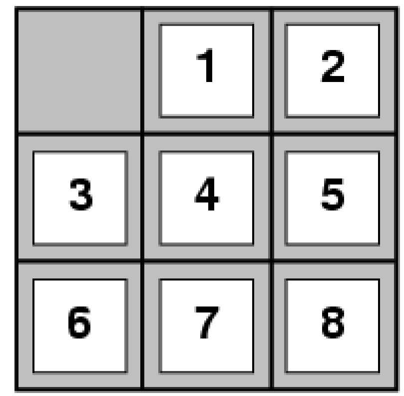
A* difficulties

- Space is the main problem of A*
- Overcoming space problem while retaining completeness and optimality
  - IDA*, RBFS, MA*, SMA*
- A* time complexity
  - Variants of A* trying to find suboptimal solutions quickly
  - More accurate but not strictly admissible heuristics

8-puzzle problem: state space



Start State

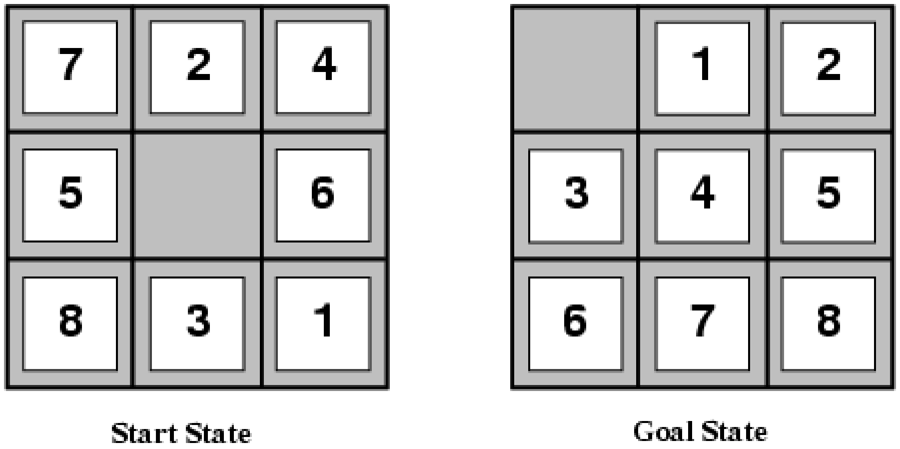


Goal State

- $b \approx 3$, average solution cost for random **8-puzzle** ≈ 22
- **Tree search**: $\mathrm{b} \approx 3, \mathrm{d} \approx 22 \Longrightarrow 3^{22} \approx 3.1 \times 10^{10}$states
- **Graph search**: $9!/2 \approx 181{,}440$ states for **8-puzzle**
  - 10^{13} for 15-puzzle

Admissible heuristics: 8-puzzle

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = sum of Manhattan distance of tiles from their target position
  - i.e., no. of squares from desired location of each tile



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

Effect of heuristic on accuracy

- N: number of generated nodes by A*
- d: solution depth

- Effective branching factor b^*: branching factor of a uniform tree of depth d containing $N + 1$ nodes.

$$N + 1 = 1 + b^* + (b^*)^2 + \cdots + (b^*)^d$$

- Well-defined heuristic: b^* is close to one

Comparison on 8-puzzle

Search Cost (N)

d	IDS	A*(h_1)	A*(h_2)
6	680	20	18
12	3644035	227	73
24	--	39135	1641

Effective branching factor (b^*)

d	IDS	A*(h_1)	A*(h_2)
6	2.87	1.34	1.30
12	2.78	1.42	1.24
24	--	1.48	1.26

Heuristic quality

If $\forall n, h_2(n) \geq h_1(n)$ (both admissible)

then h_2 dominates h_1 and it is better for search

- Surely expanded nodes: $f(n) < C^* \Rightarrow h(n) < C^* - g(n)$
  - If $h_2(n) \geq h_1(n)$ then every node expanded for h_2 will also be surely expanded with h_1 (h_1 may also causes some more node expansion)

More accurate heuristic

- Max of admissible heuristics is admissible (while it is a more accurate estimate)

$$h(n) = \max(h_1(n), h_2(n))$$

- How about using the actual cost as a heuristic?
  - $h(n) = h^*(n)$ for all n
    - Will go straight to the goal ?!
  - Trade of between accuracy and computation time

Generating heuristics

- Relaxed problems
  - Inventing admissible heuristics automatically
- Sub-problems (pattern databases)
- Learning heuristics from experience

Relaxed problem

- Relaxed problem: Problem with fewer restrictions on the actions

- Optimal solution to the relaxed problem may be computed easily (without search)

- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
  - The optimal solution is the shortest path in the super-graph of the state-space.

Relaxed problem: 8-puzzle

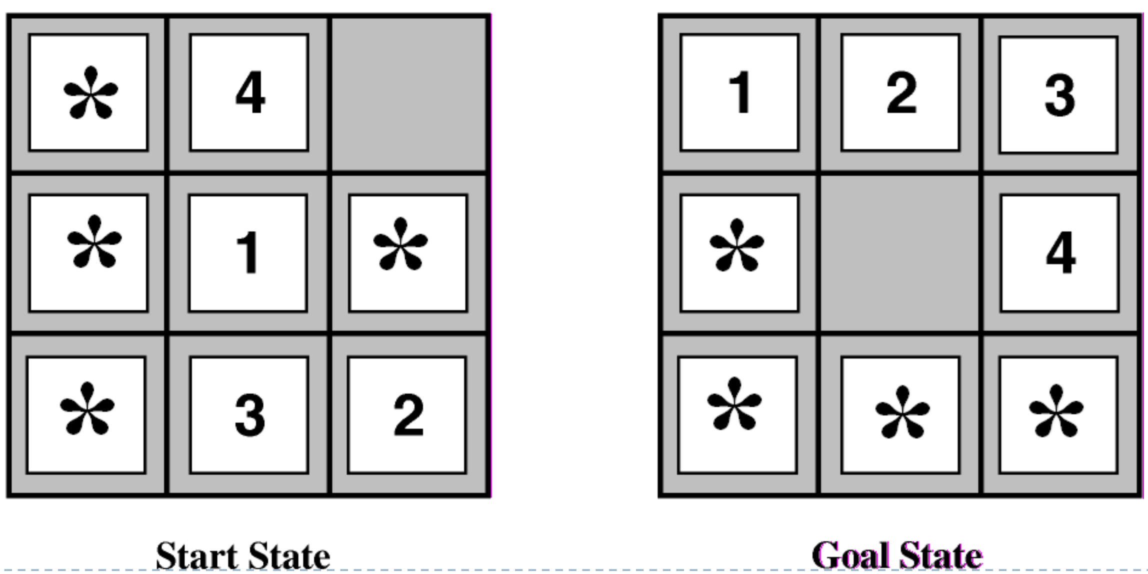
- 8-Puzzle: move a tile from square A to B if <u>A is adjacent (left, right, above, below) to B and B is blank</u>
  - Relaxed problems
    1) can move from A to B if A is adjacent to B (ignore whether or not position is blank)
    2) can move from A to B if B is blank (ignore adjacency)
    3) can move from A to B (ignore both conditions)

- Admissible heuristics for original problem ($h_1(n)$ and $h_2(n)$) are optimal path costs for relaxed problems
  - First case: <u>a tile can move to any adjacent square</u> $\Rightarrow$ $h_2(n)$
  - Third case: <u>a tile can move anywhere</u> $\Rightarrow$ $h_1(n)$

Sub-problem heuristic

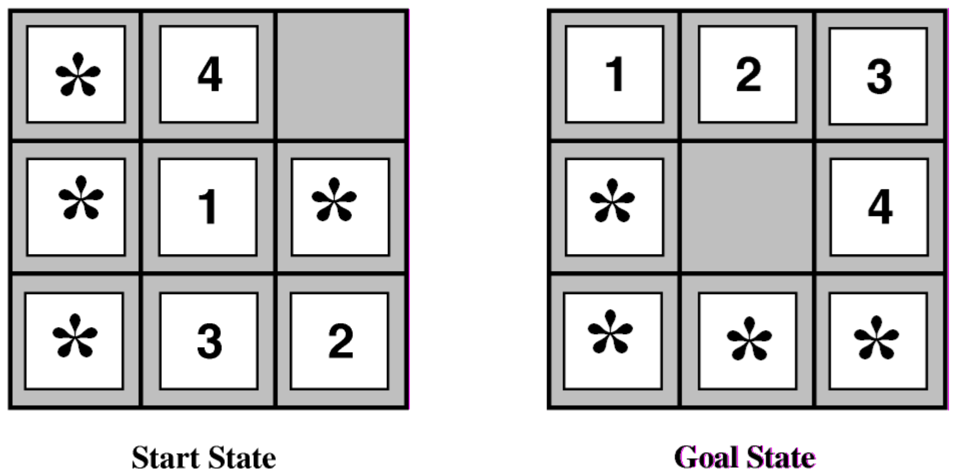
- The cost to solve a sub-problem
  - Store exact solution costs for every possible sub-problem

- Admissible?
  - The cost of the optimal solution to this problem is a lower bound on the cost of the complete problem



Pattern databases heuristics

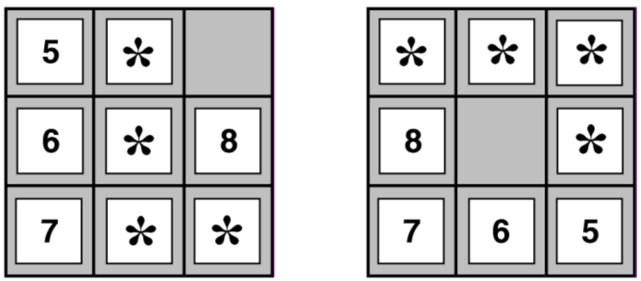
- Storing the exact solution cost for every possible sub-problem instance



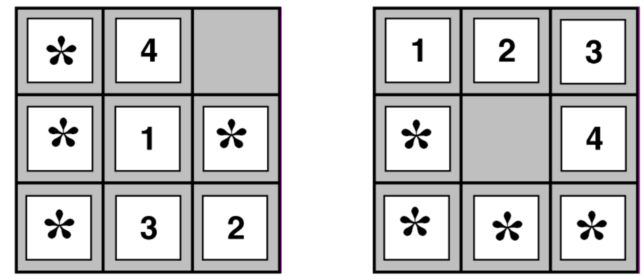
- Combination (taking maximum) of heuristics resulted by different sub-problems
  - 15-Puzzle: 10^3 times reduction in no. of generated nodes vs. h_2

Disjoint pattern databases

- Adding these pattern-database heuristics yields an admissible heuristic?!



Start State Goal State



Start State Goal State

- Dividing up the problem such that each move affects only one sub-problem (disjoint sub-problems) and then adding heuristics
  - 15-puzzle: 10^4 times reduction in no. of generated nodes vs. h_2
  - 24-Puzzle: 10^6 times reduction in no. of generated nodes vs. h_2
  - Can Rubik's cube be divided up to disjoint sub-problems?

Learning heuristics from experience

- Machine Learning Techniques
  - Learn $h(n)$ from samples of optimally solved problems (predicting solution cost for other states)

- Features of state (instead of raw state description)
  - 8-puzzle
    - number of misplaced tiles
    - number of adjacent pairs of tiles that are not adjacent in the goal state
  - Linear Combination of features