CSC 665: Artificial Intelligence

Informed Search

Pooyan Fazli

San Francisco State University

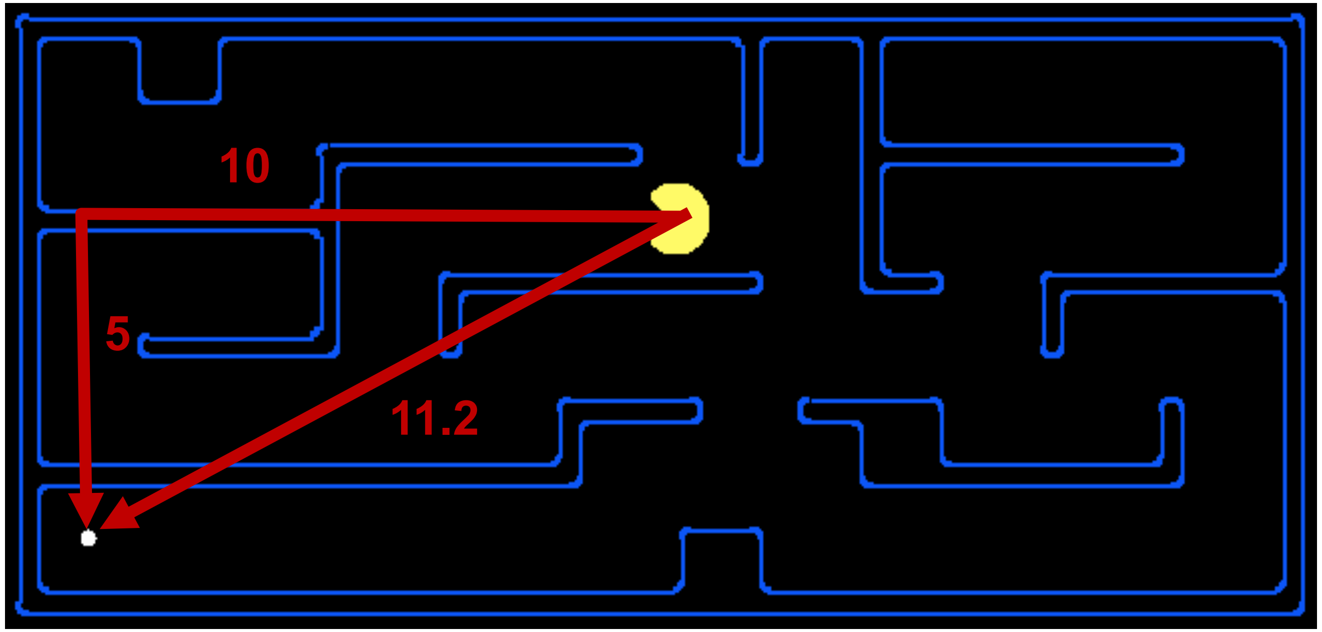
Today

- Informed Search
  - Heuristics
  - Best-First Search
  - A* Search

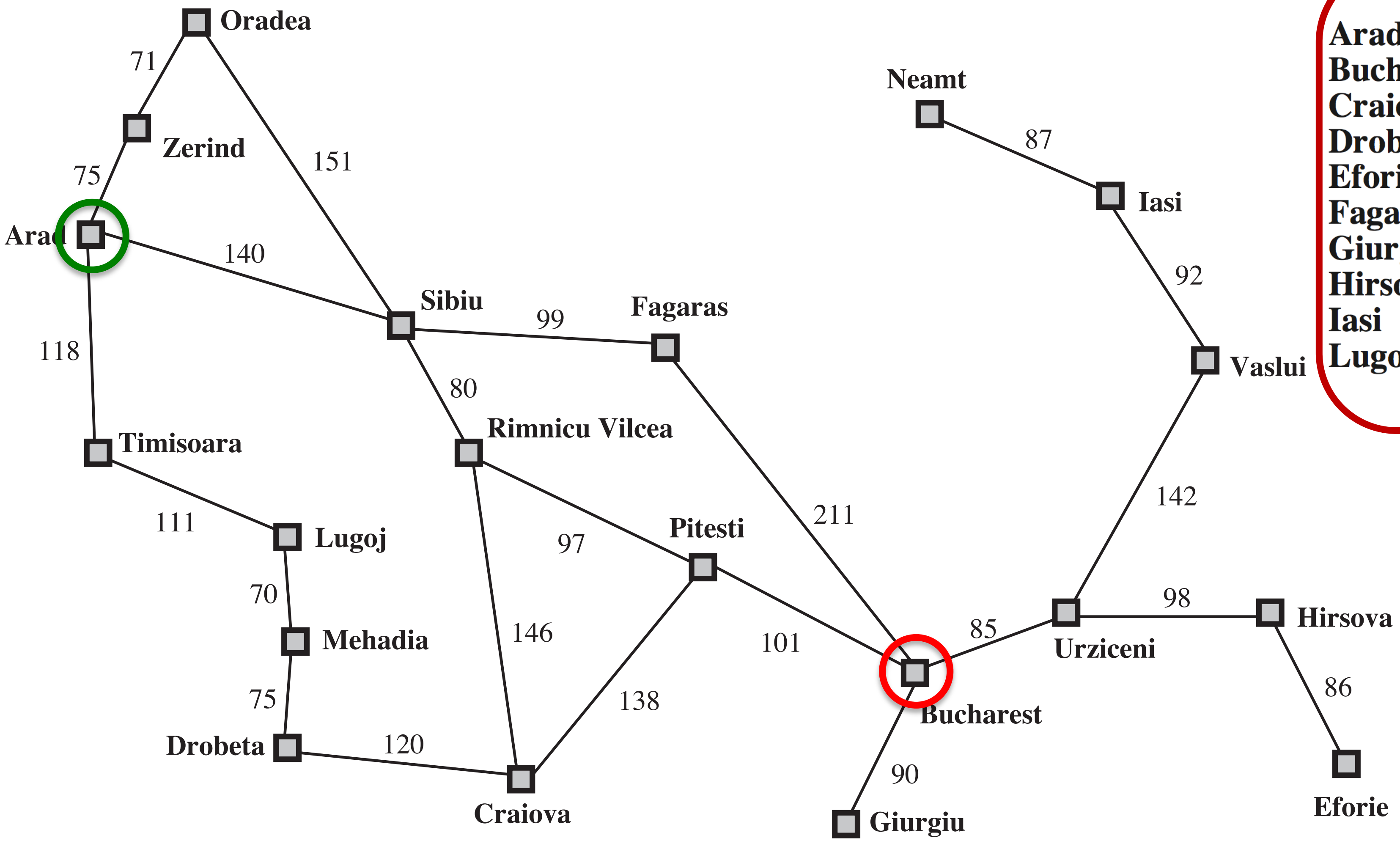
Heuristics

Search Heuristics

- A heuristic is:
  - A function that *estimates* how close a state is to a goal
  - Designed for a particular search problem
  - Examples: Manhattan distance, Euclidean distance for pathing



Example: Euclidean distance to Bucharest

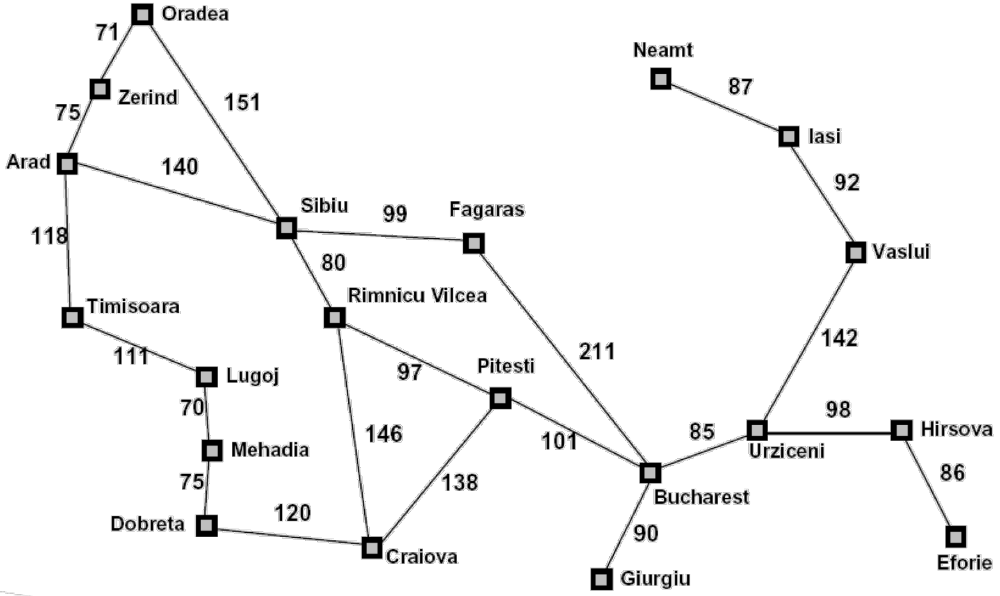


Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

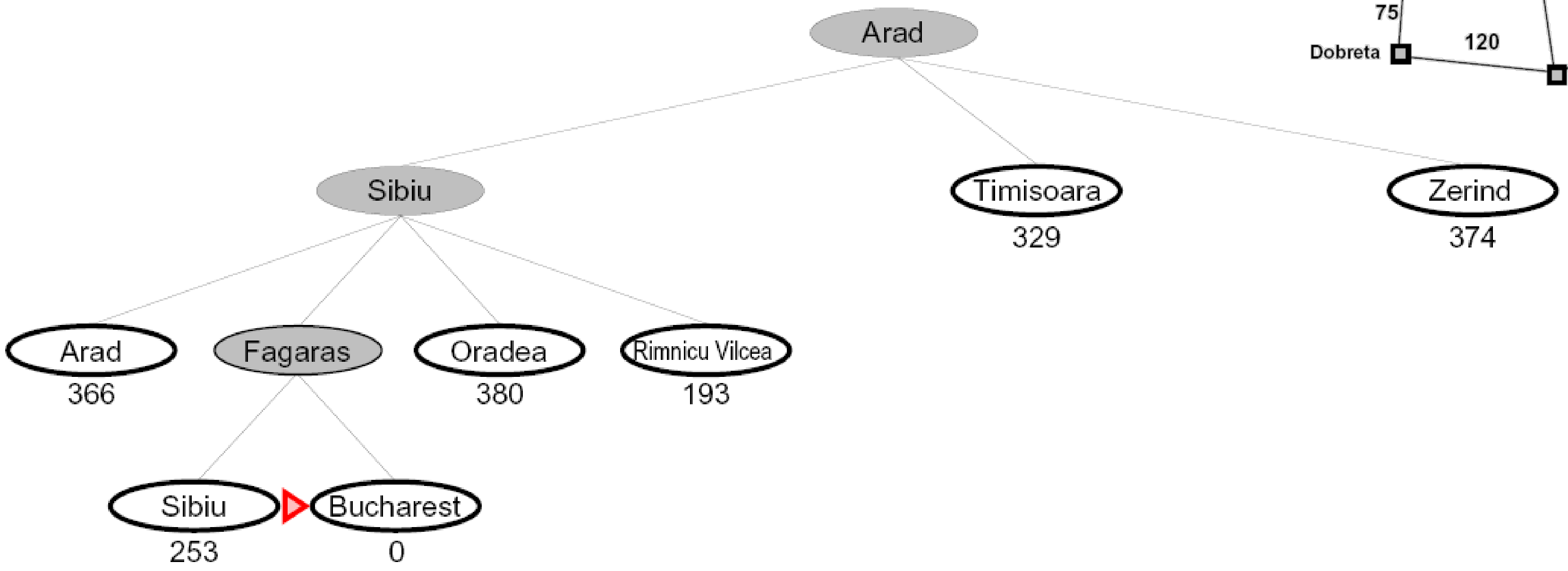
h(x)

Best-First Search

Best-First Search

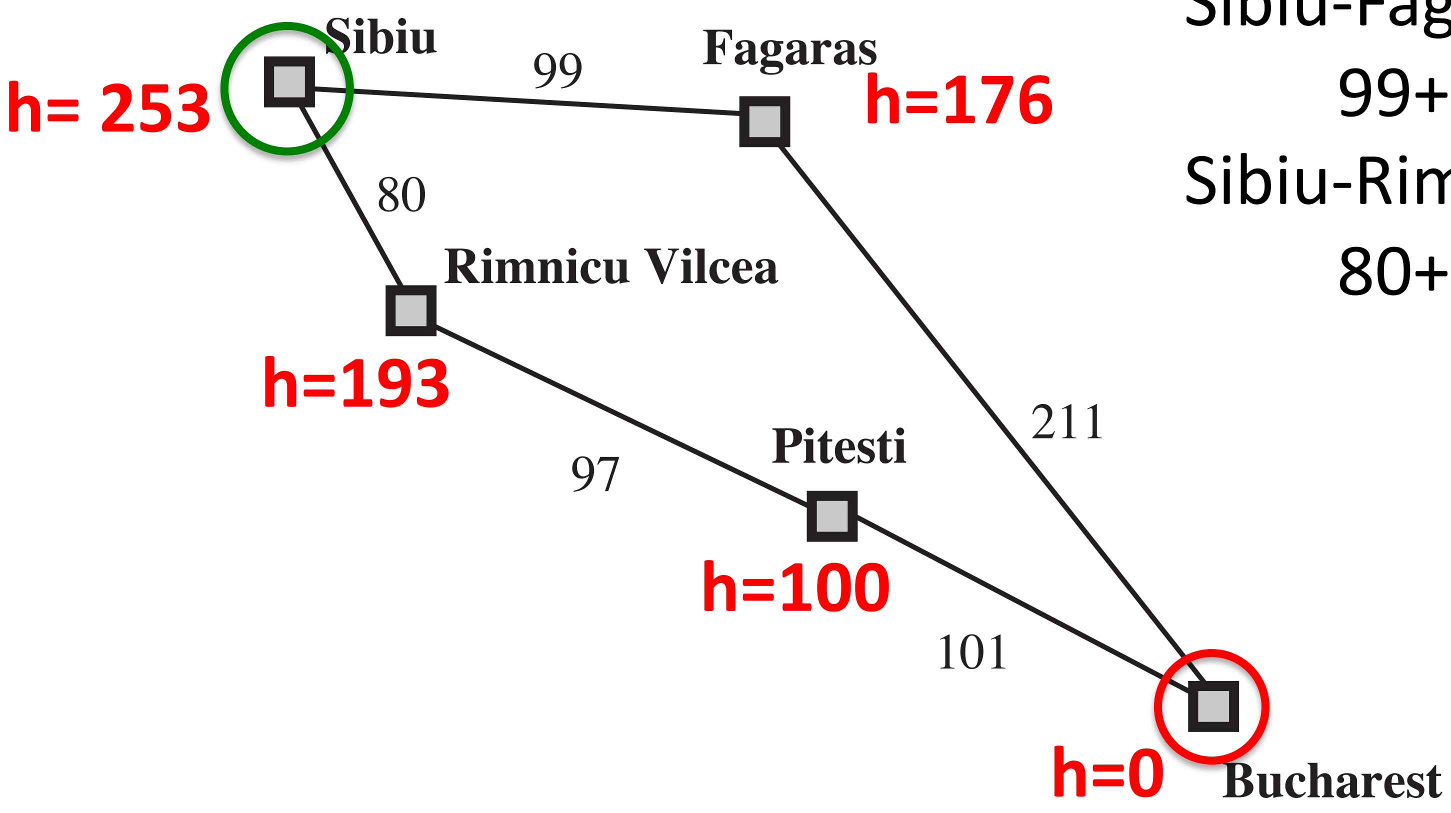


- Expand the node that seems closest…



- What can go wrong?

Best-First Search



Sibiu-Fagaras-Bucharest =

99+211 = **310**

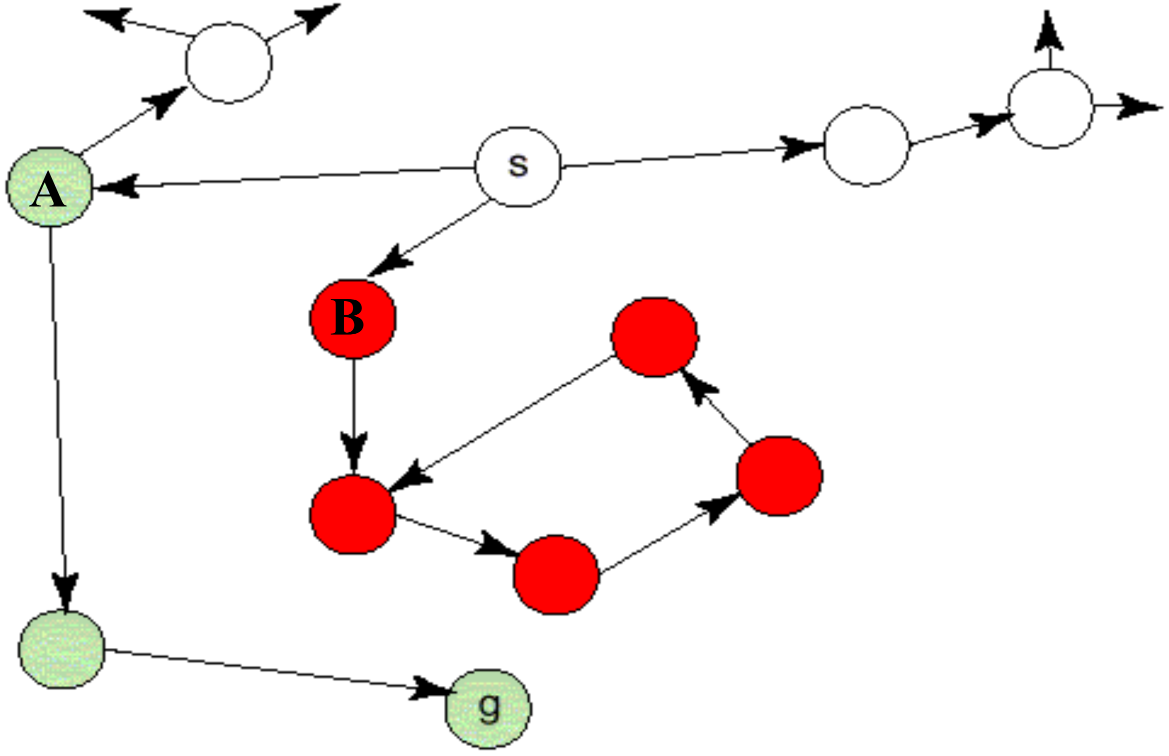
Sibiu-Rimnicu Vilcea-Pitesti-Bucharest =

80+97+101 = **278**

Best-First Search

- Idea: always choose the node on the frontier with the smallest h value.
- Best-First Search treats the frontier as a priority queue ordered by h.

Best-First Search



- A low heuristic value can mean that a cycle gets followed forever -> not complete

Best-First Search Properties

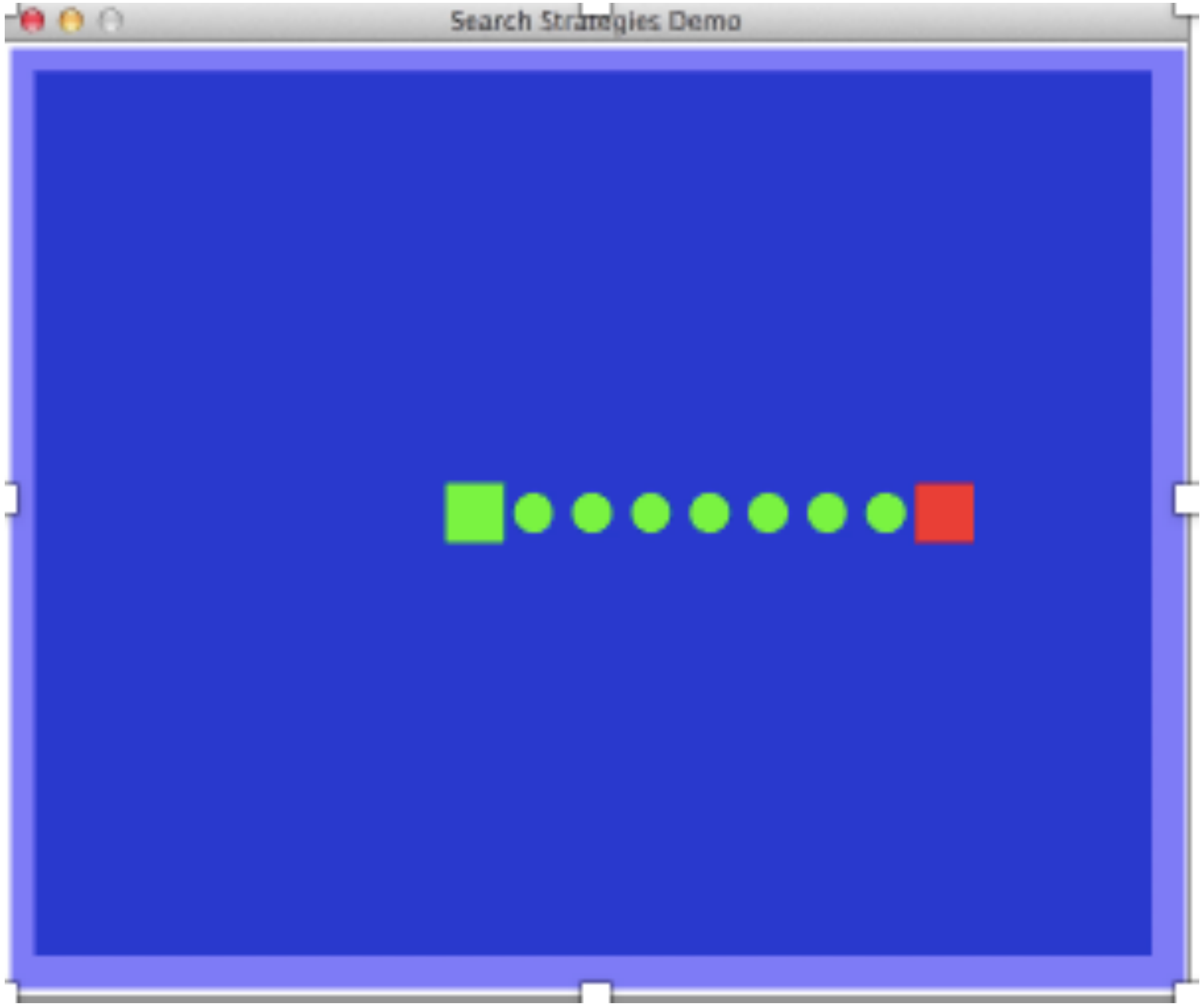
- Complete? No, see the example in the last slide
- Optimal? No, see the Romanian map example
- Time Complexity $O(b^m)$ ✓ $O(m^b)$ $O(bm)$ $O(b+m)$

  - Worst case: has to explore all nodes

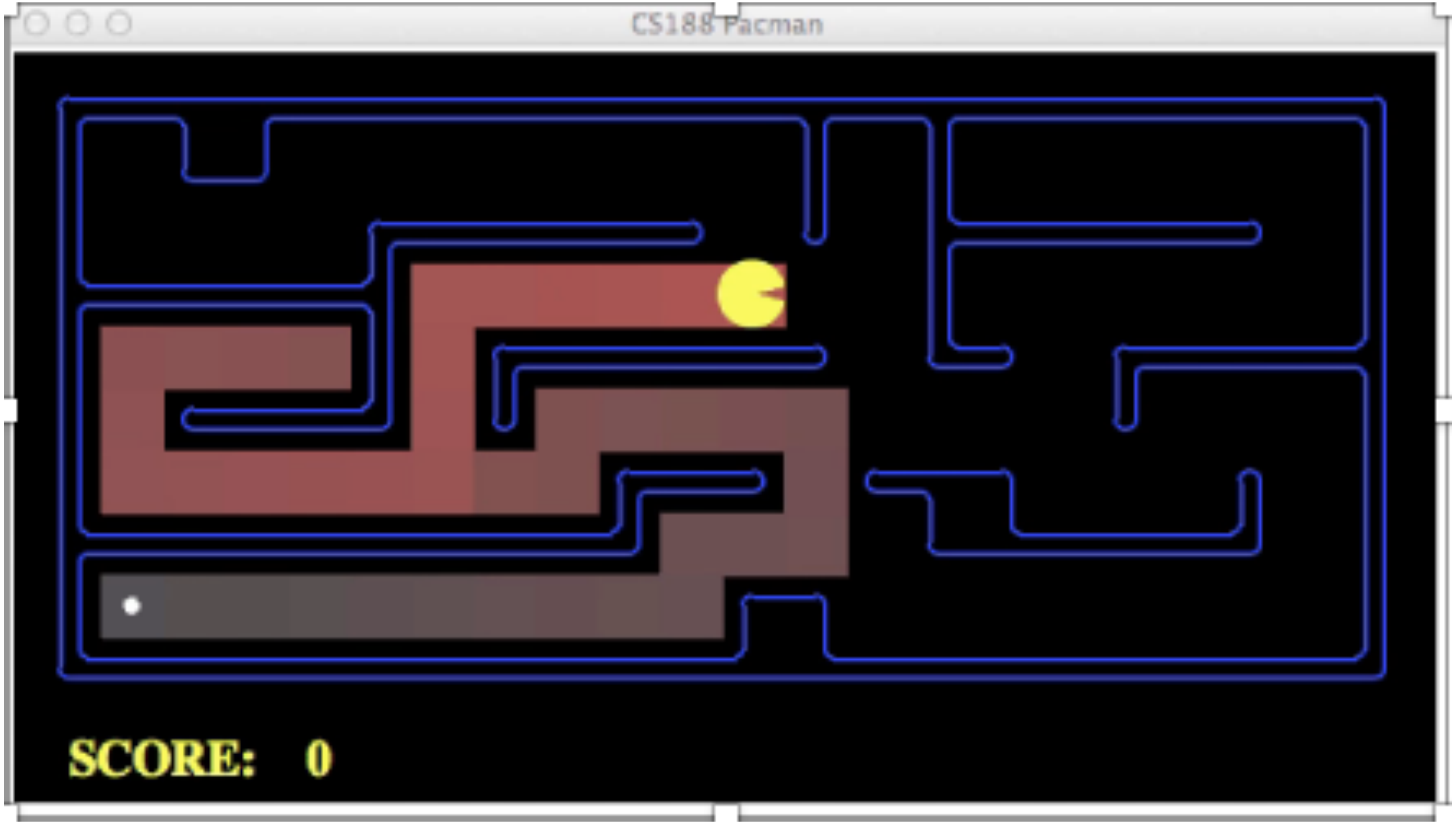
- Space Complexity $O(b^m)$ ✓ $O(m^b)$ $O(bm)$ $O(b+m)$

  - Heuristic could be chosen to emulate BFS:
    E.g. $h(n) = 0$

Video of Demo Contours BestFS (Empty)



Video of Demo Contours BestFS (Pacman Small Maze)



A* Search

A* Search

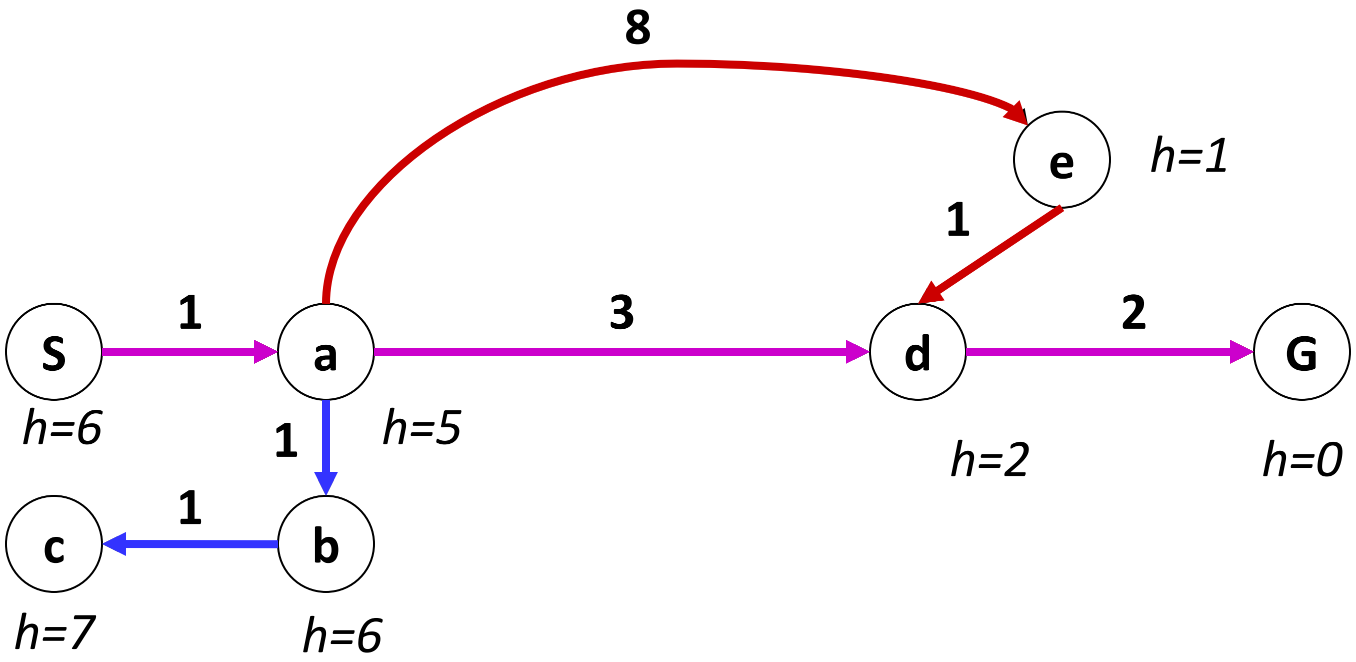
Uniform Cost Search

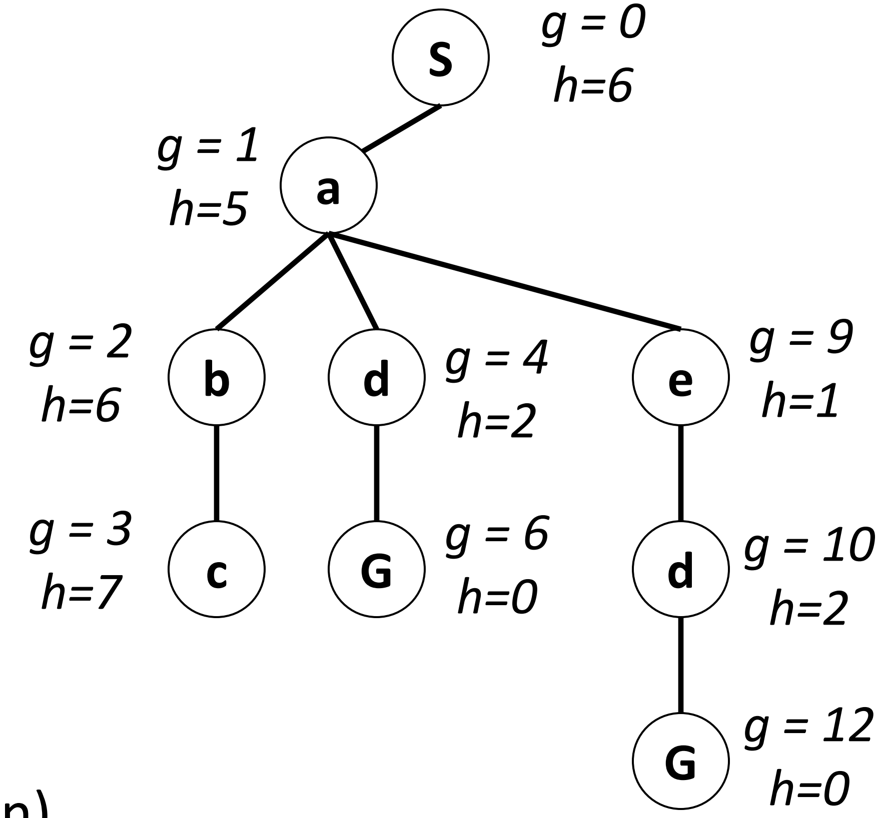
Best-First Search

A*

Combining UCS and BestFS

- Uniform-Cost-Search orders by path cost, or *backward cost* g(n)
- Best-First orders by goal proximity, or *forward cost* h(n)



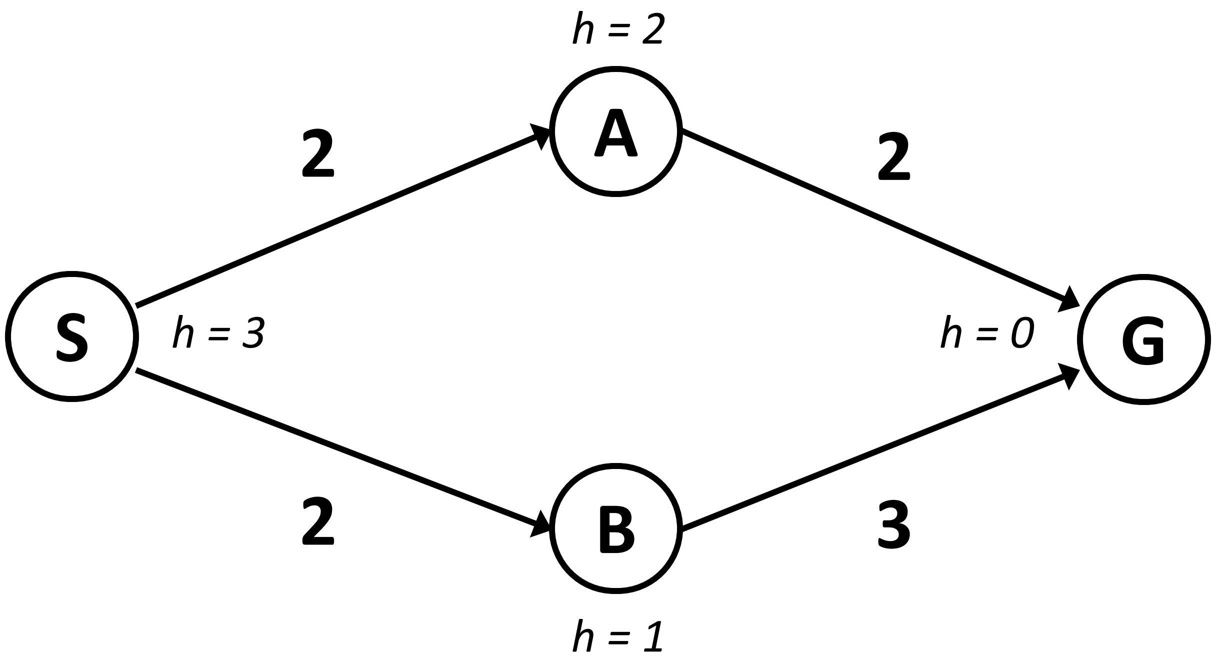


- A* Search orders by the sum: f(n) = g(n) + h(n)

Example: Teg Grenager

When should A* terminate?

- Should we stop when we enqueue (frontier) a goal?



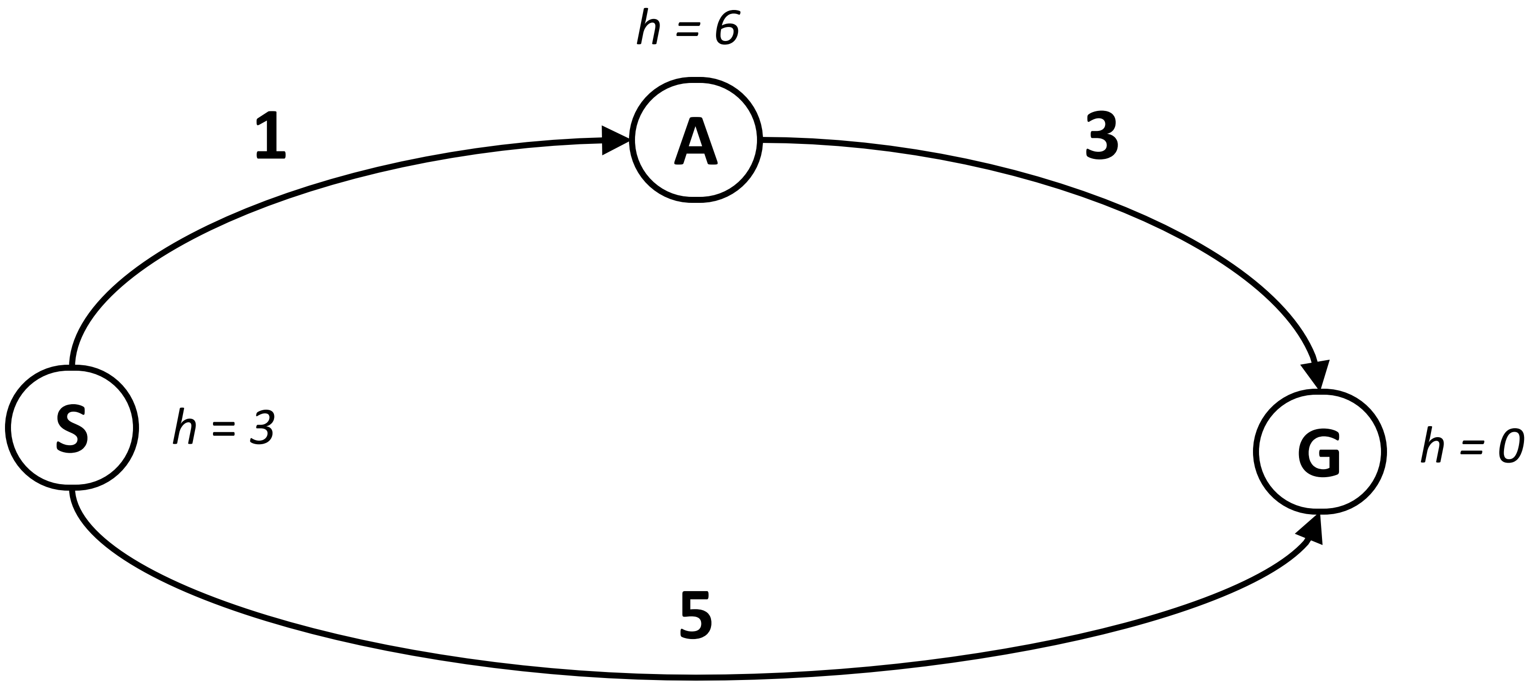
- No: only stop when we dequeue (frontier) a goal

Time / Space Complexity of A^*

- Time complexity is $O(b^m)$ the heuristic could be completely uninformative and the edge costs could all be the same, meaning that A^* does the same thing as BFS.

- Space complexity is $O(b^m)$ like BFS, A^* maintains a frontier which grows with the size of the tree.

Is A* Optimal?



- What went wrong?
  - Actual goal cost < estimated goal cost
- We need estimates to be less than equal to the actual costs!

Admissible Heuristics

Admissible Heuristics

- A heuristic *h* is *admissible* if:

$$0 \leq h(n) \leq h^*(n)$$ *For all nodes n*

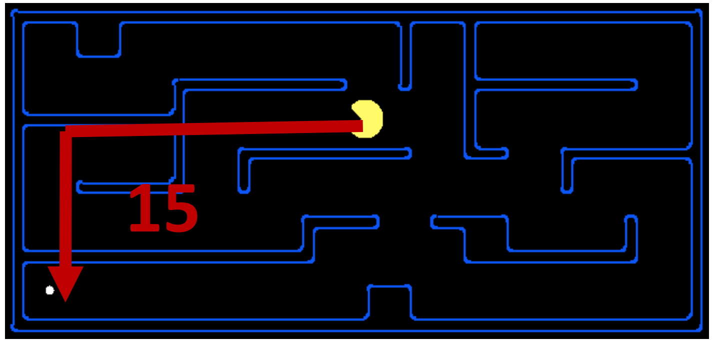
, where $h^*(n)$ is the true cost to a nearest goal,

- For all nodes, it is an *underestimate* of the cost to any goal.

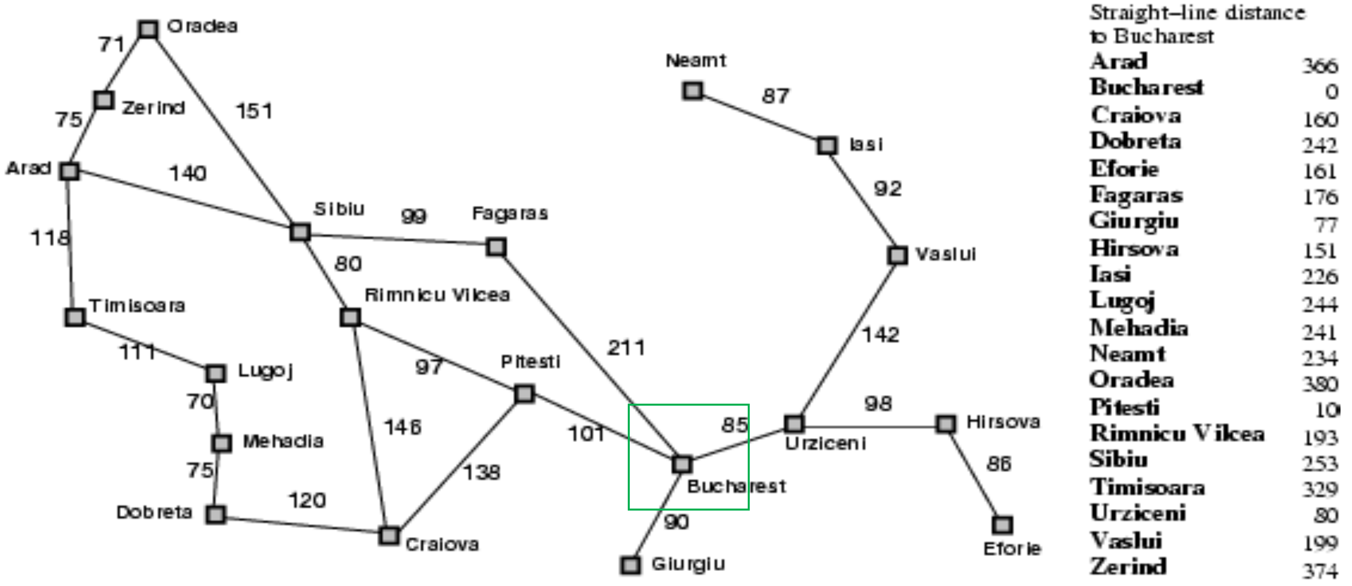
- Coming up with admissible heuristics is most of what's involved in using A* in practice.

Admissible Heuristics

- Example 1: is the Manhattan distance admissible?
  - Yes!

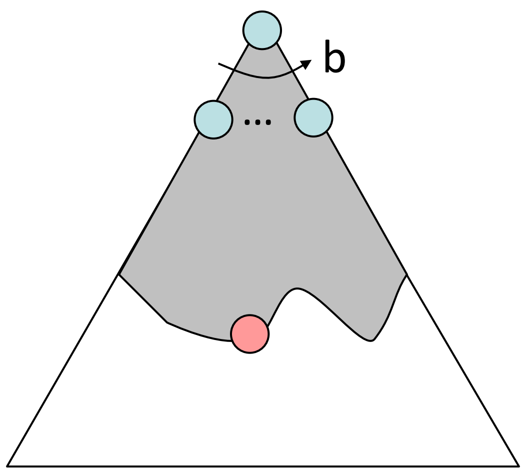


- Example 2: is the straight-line distance admissible?
  - Yes! The shortest distance between two points is a line.

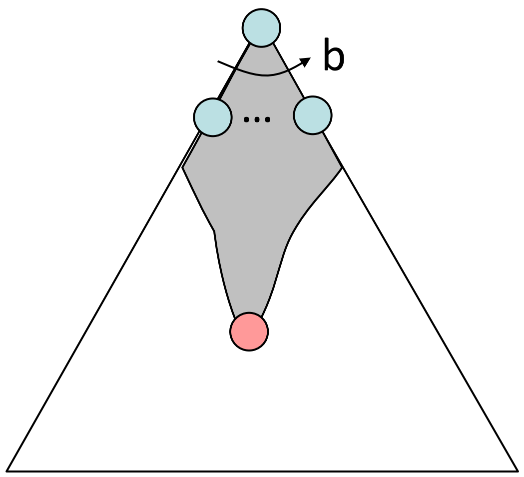


UCS vs A*

Uniform-Cost-Search

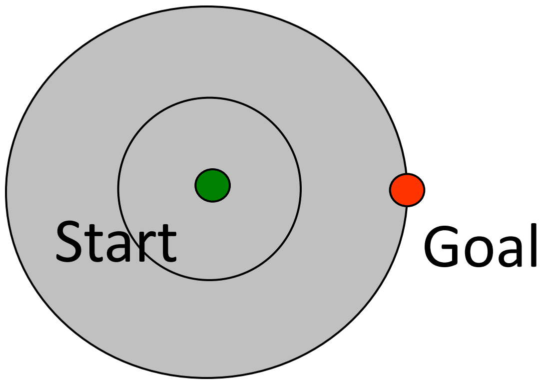


A*

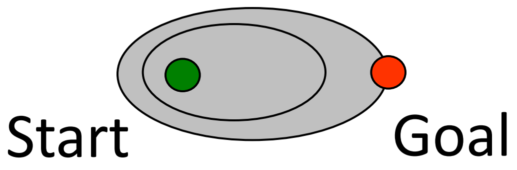


UCS vs A* Contours

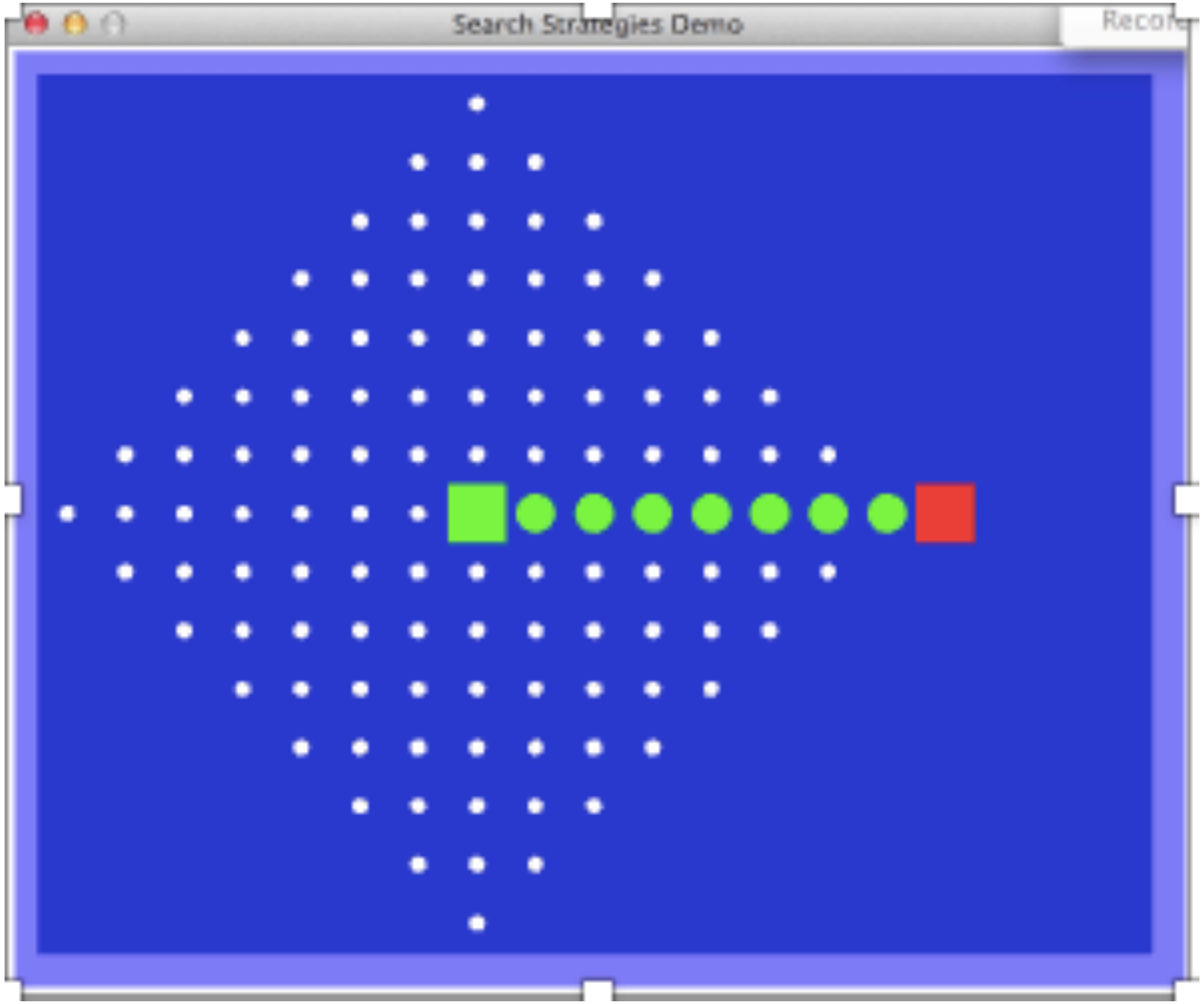
- Uniform-Cost Search expands equally in all "directions"



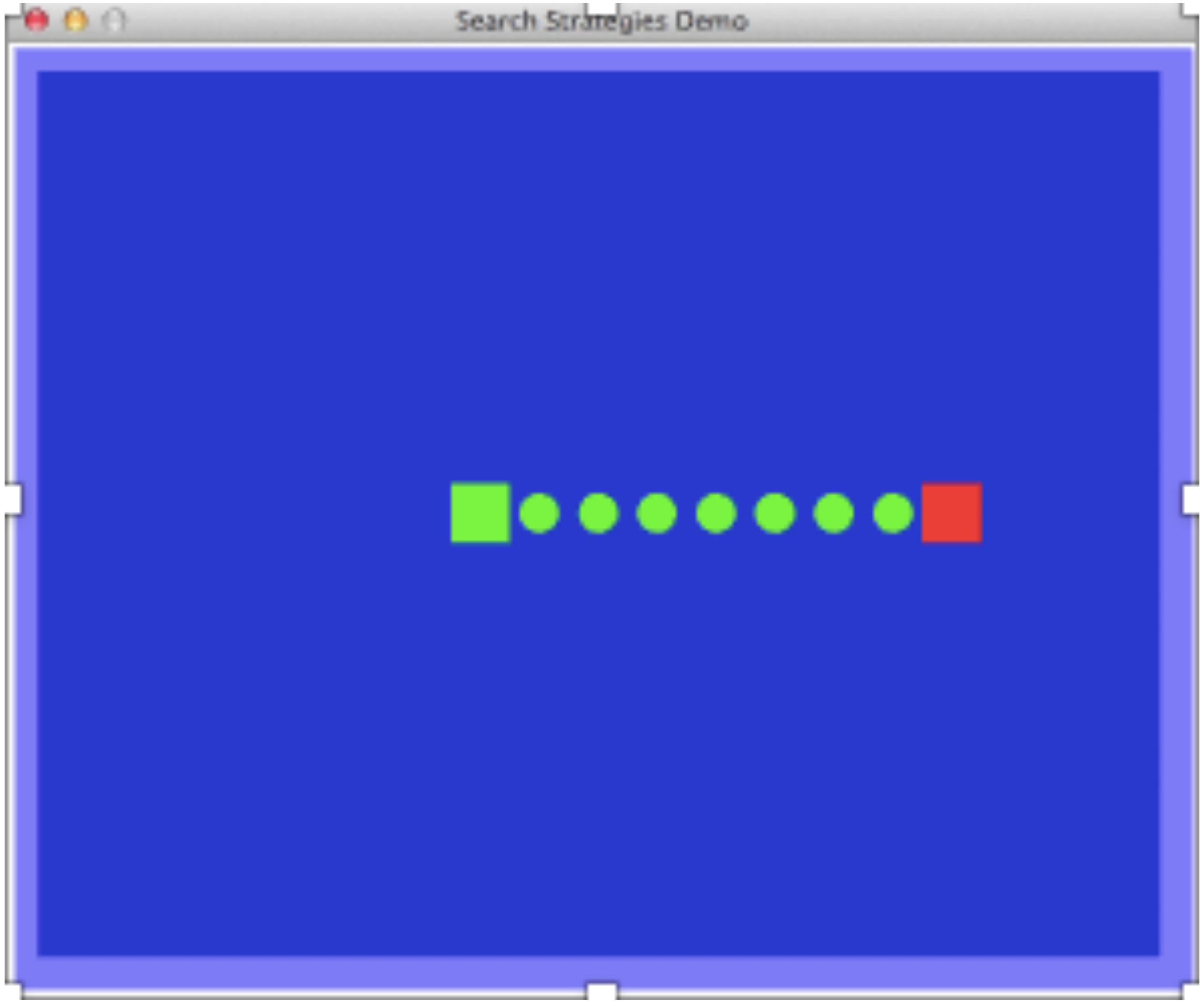
- A* expands mainly toward the goal, but does hedge its bets to ensure optimality



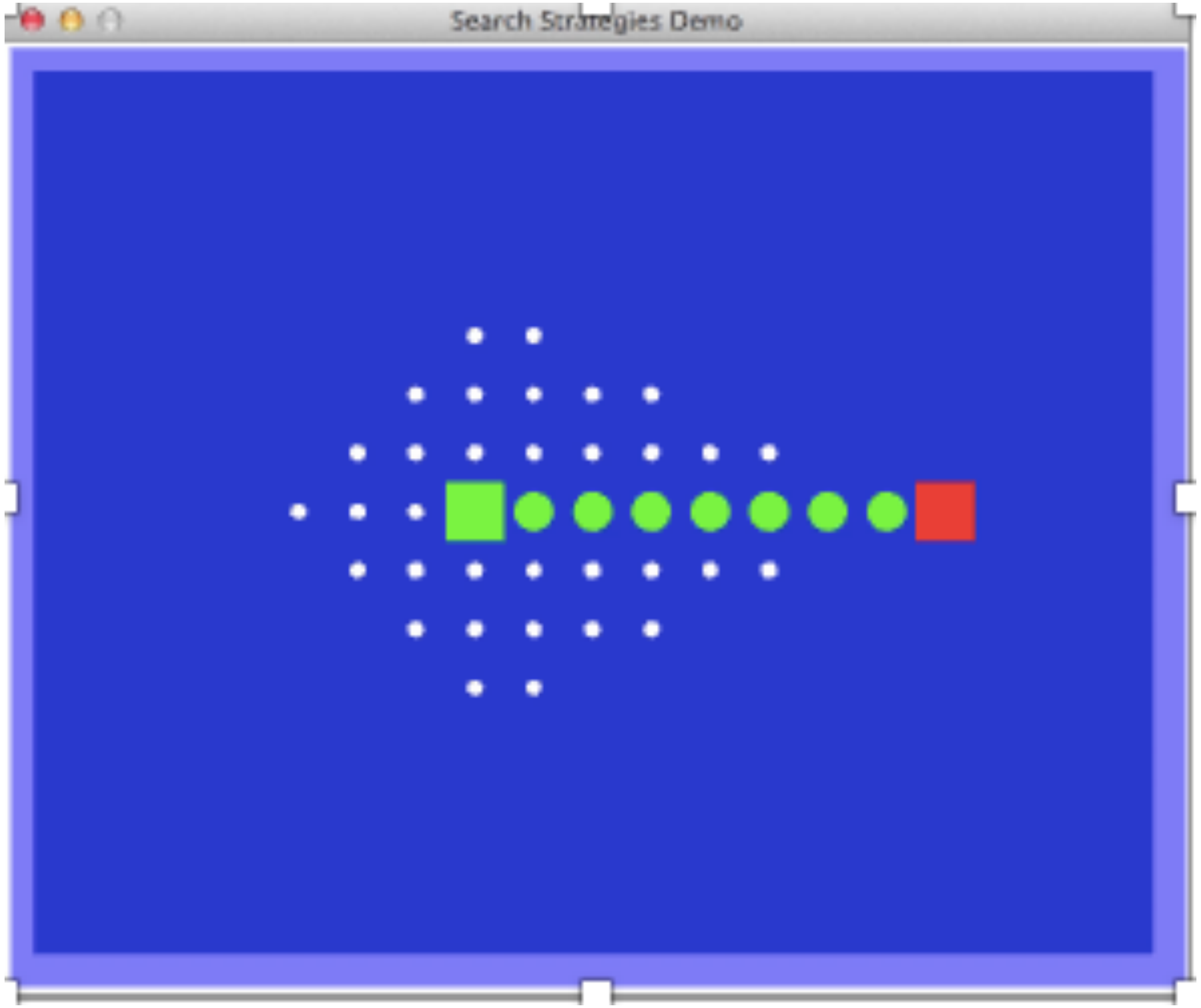
Video of Demo Contours (Empty) -- UCS



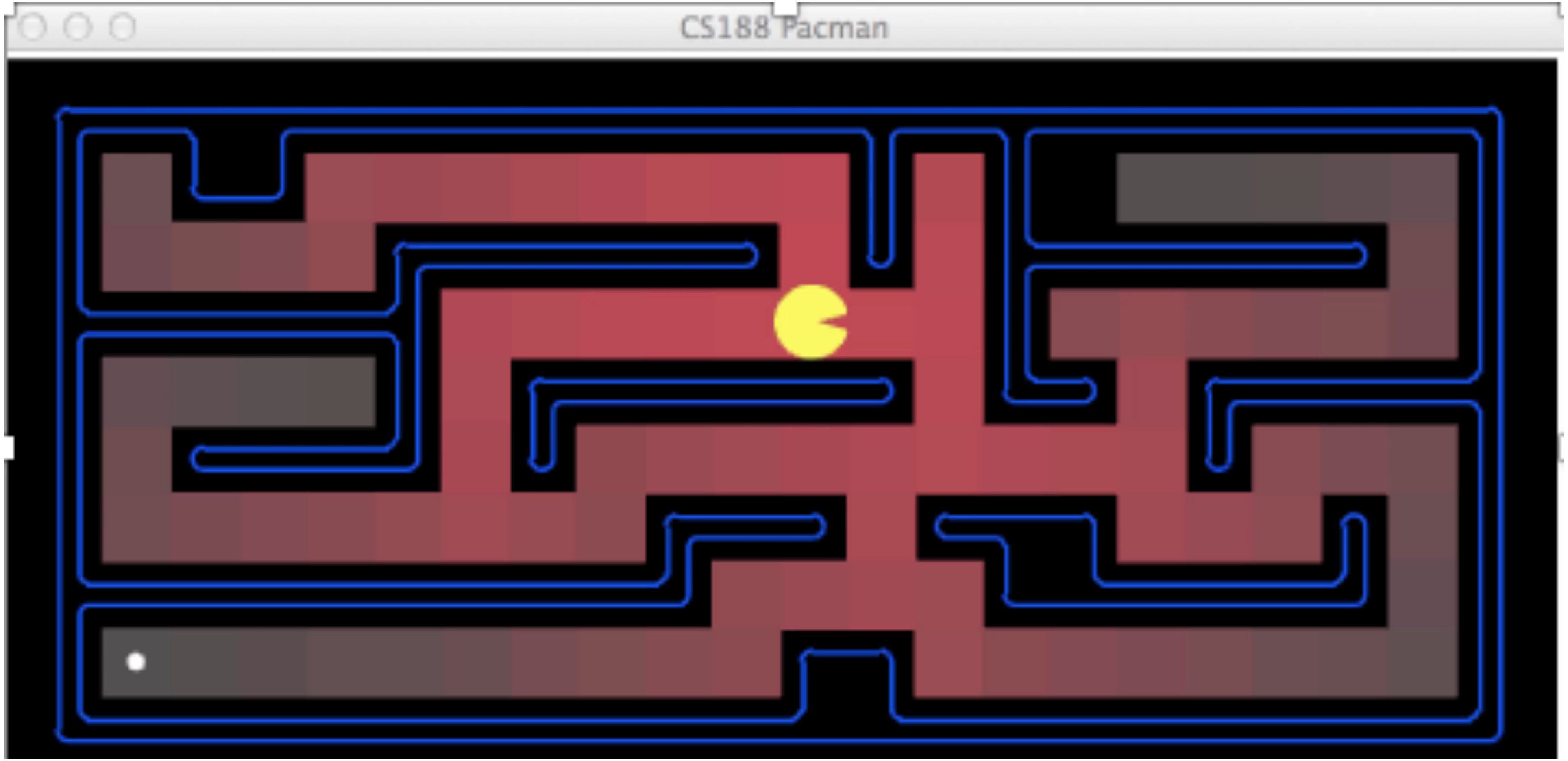
Video of Demo Contours (Empty) -- BestFS



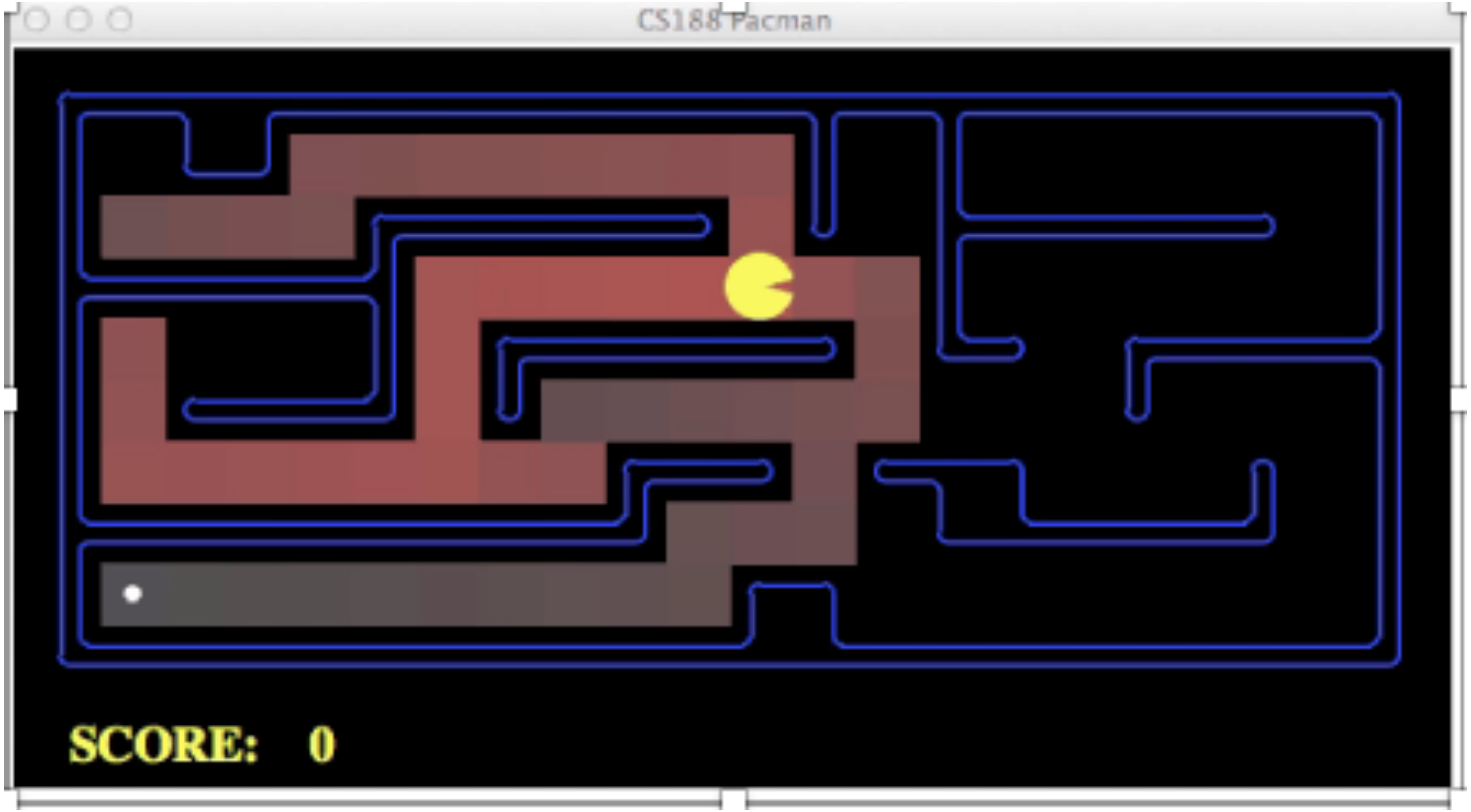
Video of Demo Contours (Empty) – A*



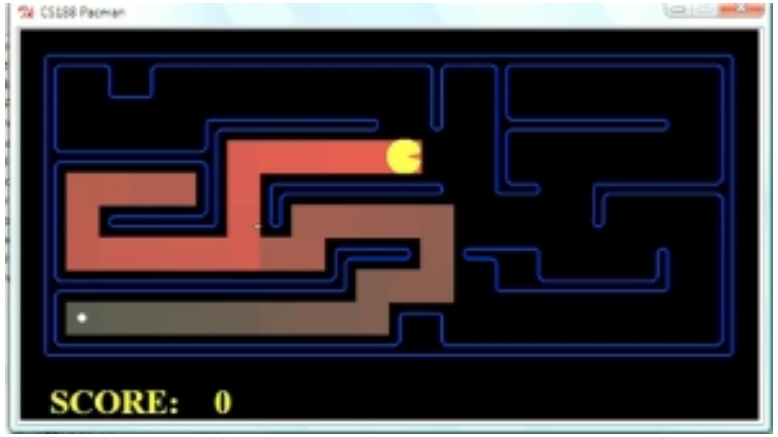
Video of Demo Contours Pacman Small Maze -UCS

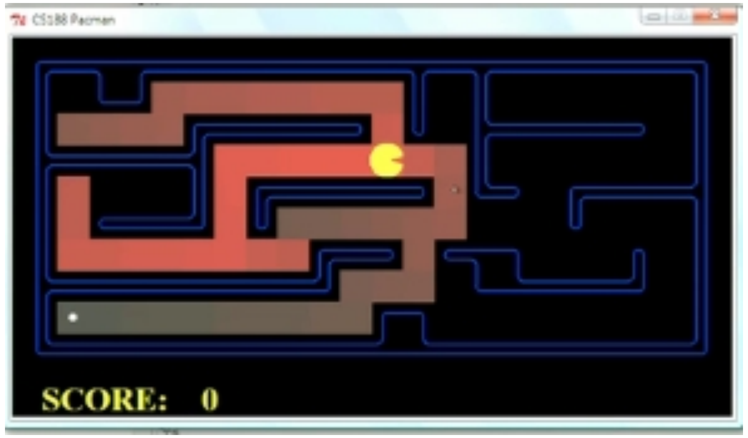


Video of Demo Contours Pacman Small Maze – A*



Comparison





Best-First Search

Uniform-Cost Search

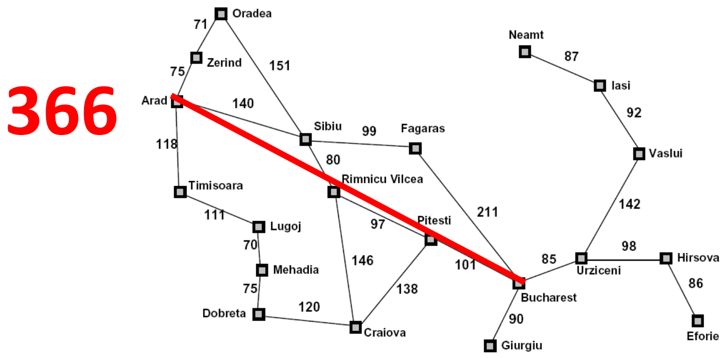
A*

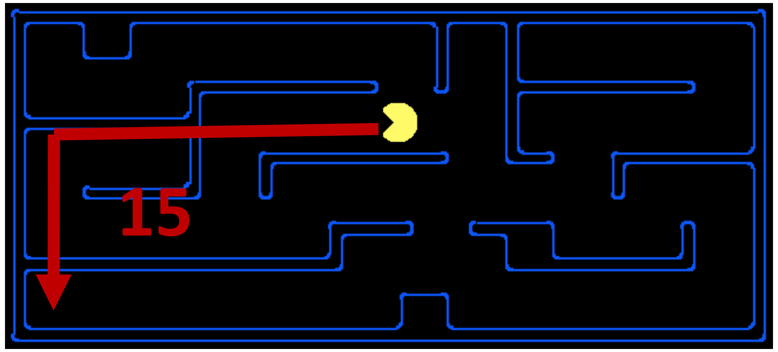
Creating Heuristics

Creating Admissible Heuristics

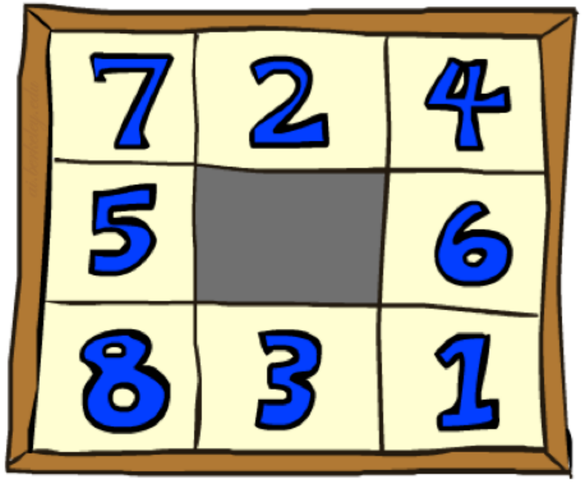
- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics

- Often, admissible heuristics are solutions to *relaxed problems*, where new actions are available

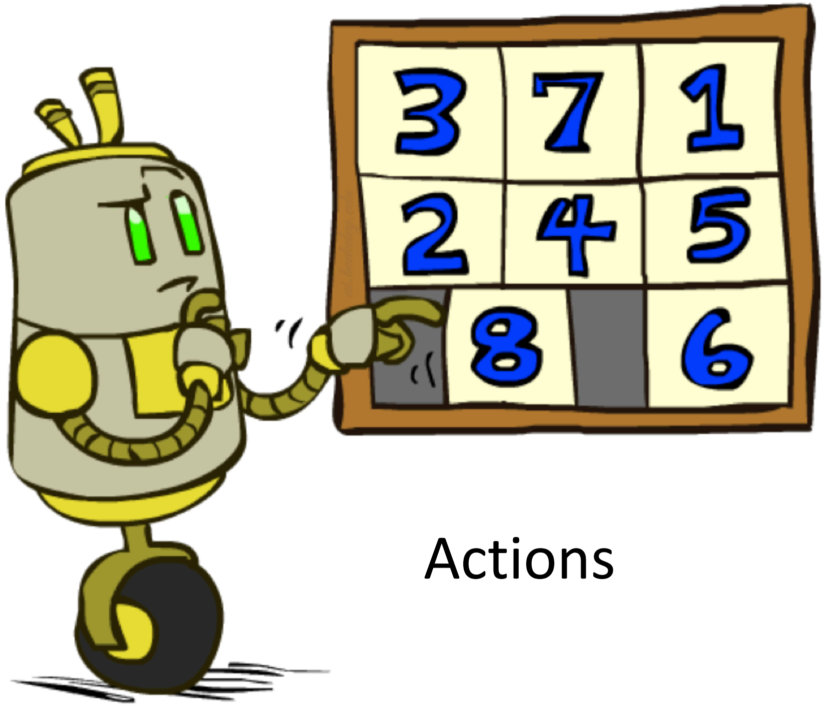




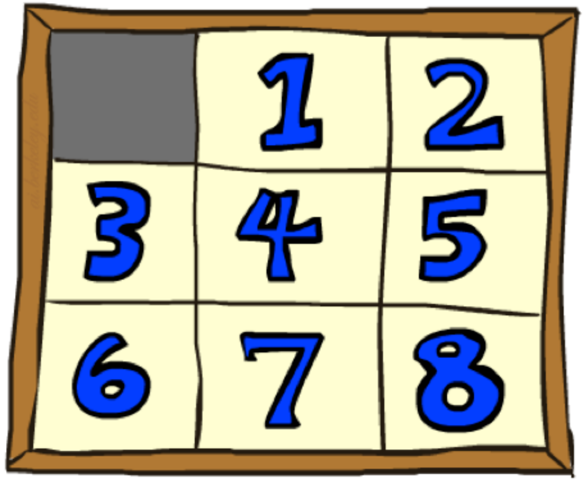
Example: 8 Puzzle



Start State



Actions

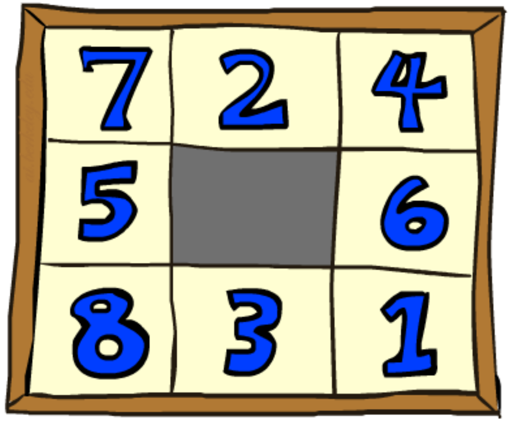


Goal State

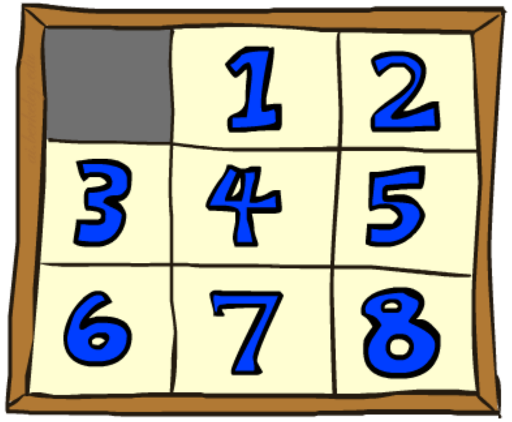
- What are the states?
- How many states?
- What are the actions?
- How many successors from the start state?
- What should the costs be?

8 Puzzle I

- Heuristic: Number of tiles misplaced
- Why is it admissible?
- h(start) = 8
- This is a *relaxed-problem* heuristic



Start State



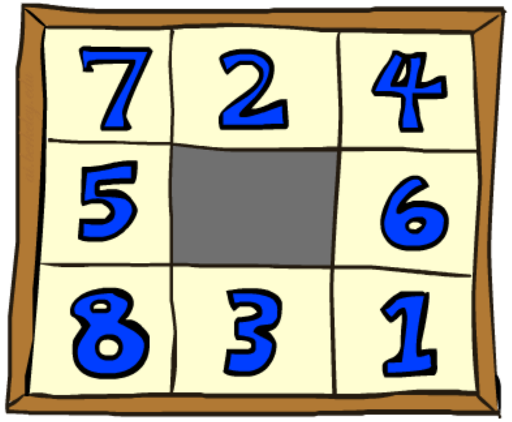
Goal State

	Average nodes expanded when the optimal path has…		
	…4 steps	…8 steps	…12 steps
UCS	112	6,300	3.6×10^6
A*, TILES	13	39	227

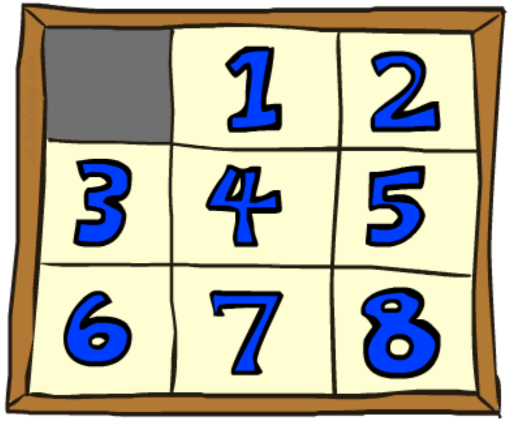
Statistics from Andrew Moore

8 Puzzle II

- What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?
- Total *Manhattan* distance
- Why is it admissible?
- h(start) = 3 + 1 + 2 + … = 18



Start State



Goal State

	Average nodes expanded when the optimal path has…		
	…4 steps	…8 steps	…12 steps
A*, TILES	13	39	227
A*, MANHATTAN	12	25	73

8 Puzzle III

- How about using the *actual cost* as a heuristic?
  - Would it be admissible?
  - Would we save on nodes expanded?
  - What’s wrong with it?

- With A*: a trade-off between quality of estimate and work per node
  - As heuristics get closer to the true cost, you will expand fewer nodes but usually do more work per node to compute the heuristic itself

Semi-Lattice of Heuristics

Trivial Heuristics, Dominance

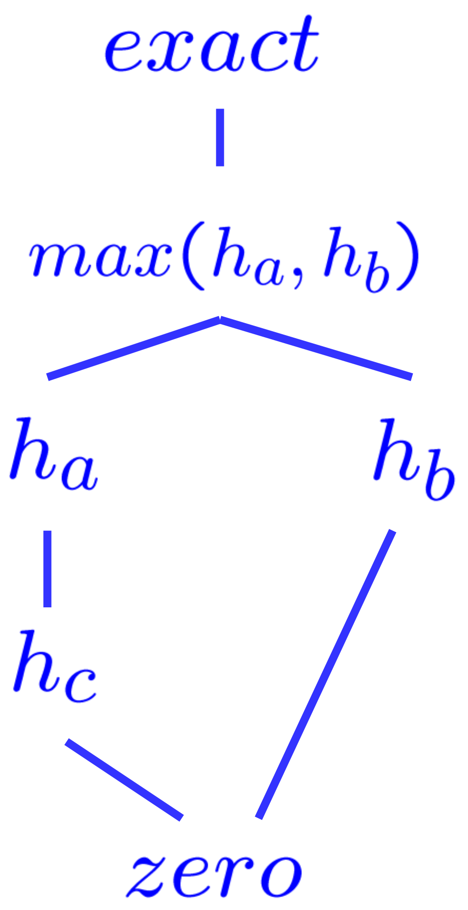
- Dominance: $h_a \geq h_c$ if

$$\forall n : h_a(n) \geq h_c(n)$$

- Heuristics form a semi-lattice:
  - Max of admissible heuristics is admissible

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

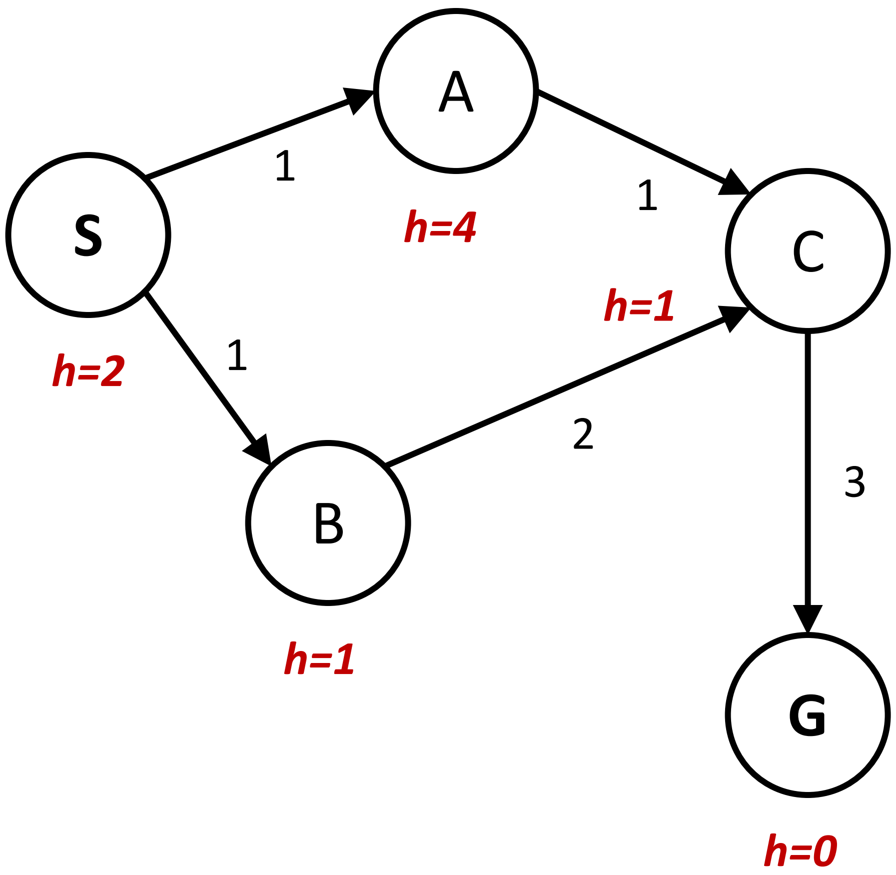
- Trivial heuristics
  - Bottom of lattice is the zero heuristic
  - Top of lattice is the exact heuristic



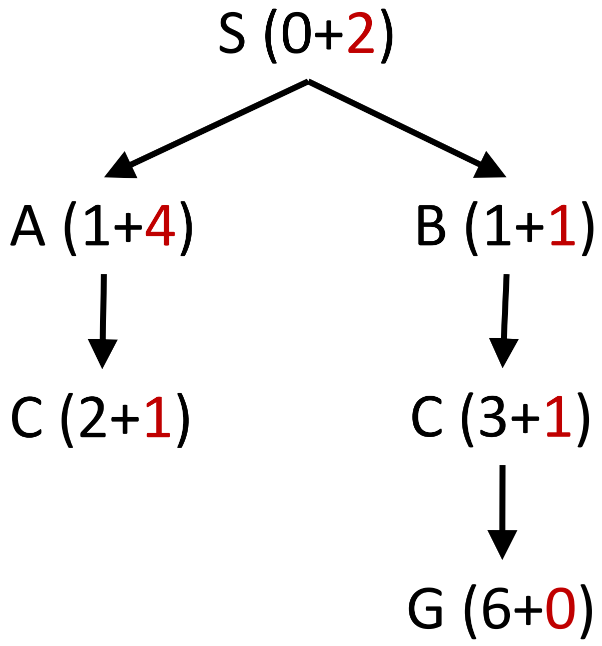
Consistency of Heuristics

A* Graph Search Gone Wrong?

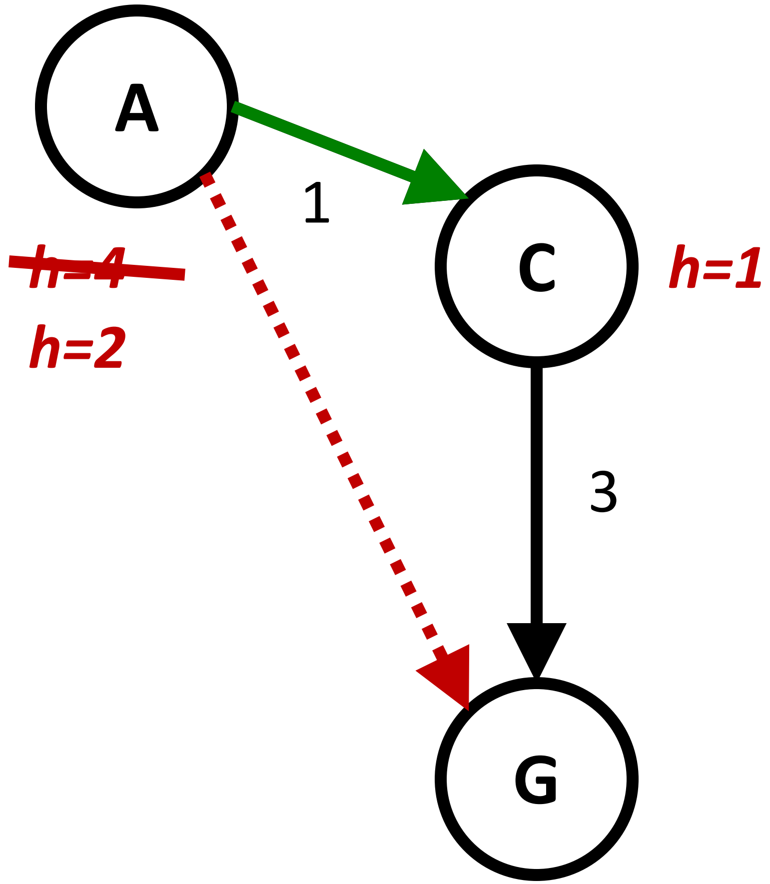
State space graph



Search tree



Consistency of Heuristics



- Main idea: estimated heuristic costs ≤ actual costs
  - Admissibility: heuristic cost ≤ actual cost to goal

    h(A) ≤ actual cost from A to G

  - Consistency: heuristic "arc" cost ≤ actual cost for each arc

    h(A) – h(C) ≤ cost(A to C)

- Consequences of consistency:
  - The f value along a path never decreases

    h(A) ≤ cost(A to C) + h(C)

  - A* graph search is optimal

A* Optimality

- Trees:
  - A* is optimal if heuristic is admissible

- Graphs:
  - A* optimal if heuristic is consistent

- Consistency implies admissibility

- In general, most natural admissible heuristics tend to be consistent, especially if from relaxed problems

A*: Summary

- A* uses both backward costs and (estimates of) forward costs
- A* is optimal with admissible / consistent heuristics
- Heuristic design is key: often use relaxed problems

Search Methods Summary

	Complete	Optimal	Time	Space
DFS	N (Y if no cycles)	N	$O(b^m)$	$O(mb)$
BFS	Y	Y	$O(b^m)$	$O(b^m)$
UCS	Y Arc Costs > 0	Y Arc Costs > 0	$O(b^m)$	$O(b^m)$
Best First	N	N	$O(b^m)$	$O(b^m)$
A*	Y Arc Costs > 0 *h* admissible	Y Arc Costs > 0 *h* admissible	$O(b^m)$	$O(b^m)$

Reading

- Chapters 3.5-6 in the AIMA textbook