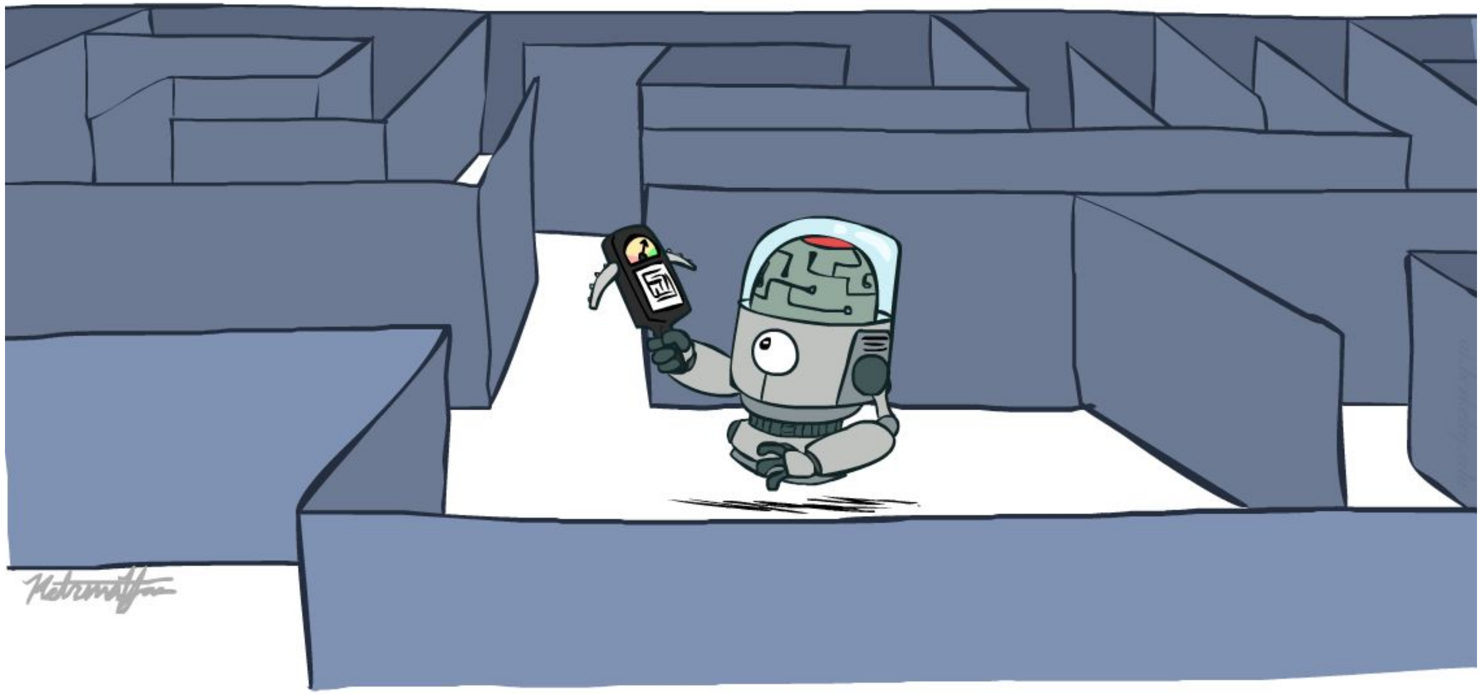
# Informed Search

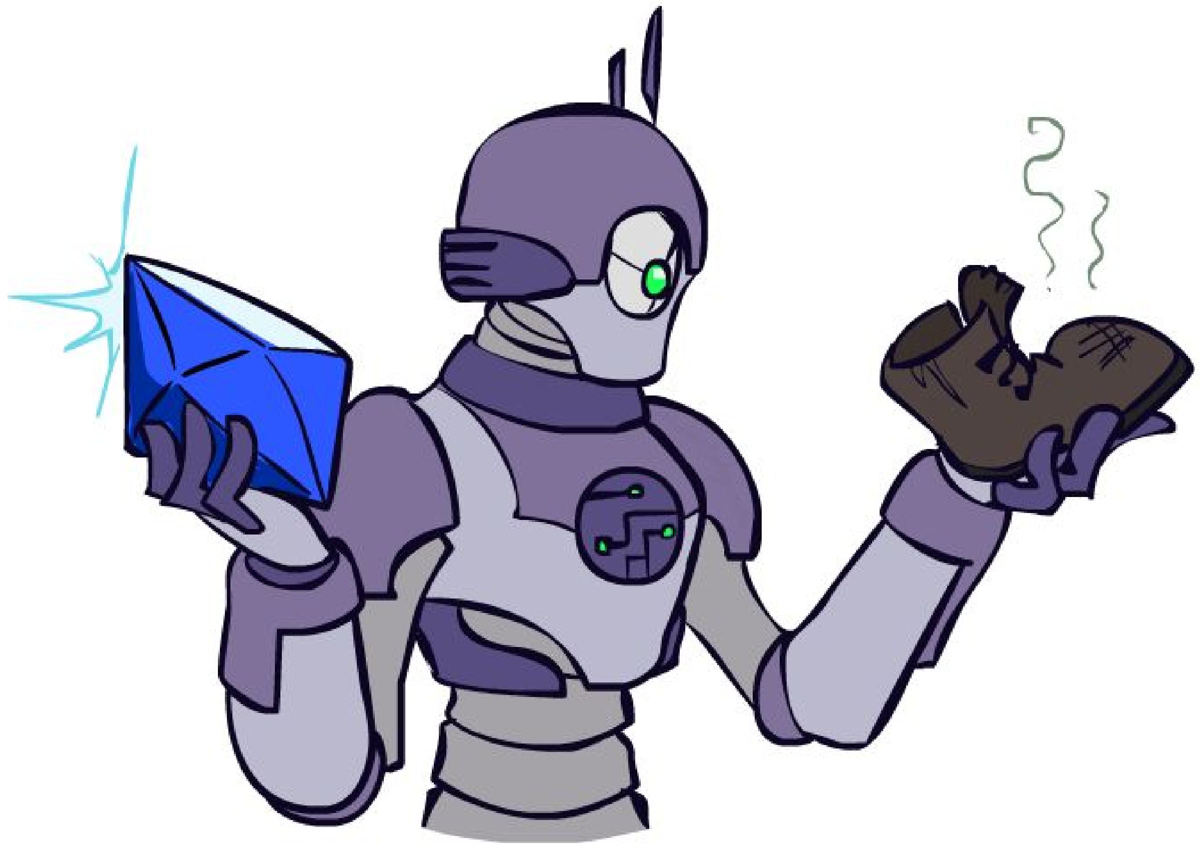


These slides are primarily based on the slides created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley.

The artwork is by Ketrina Yim.

# Today

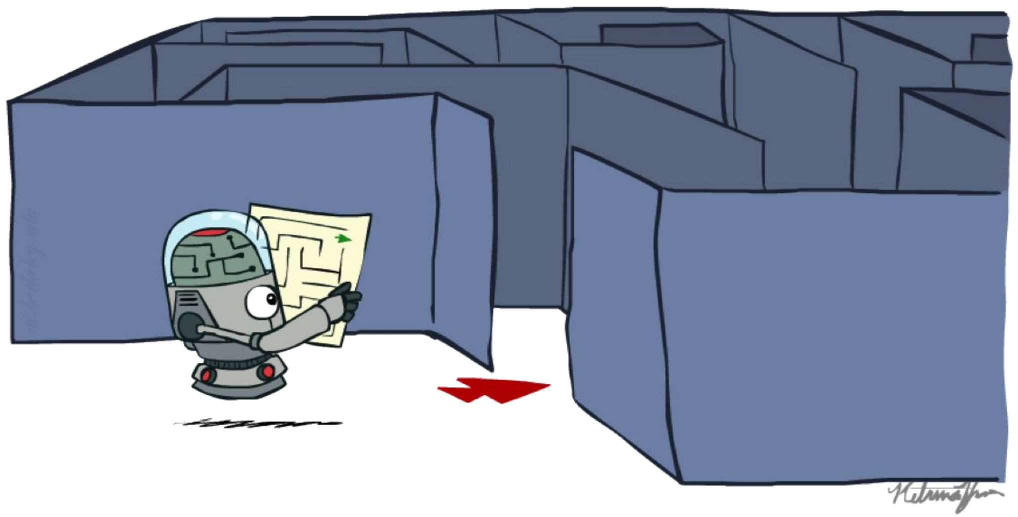
- Informed Search
  - Heuristics
  - Greedy Search
  - A* Search



# Recap: Search

Formulate search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test



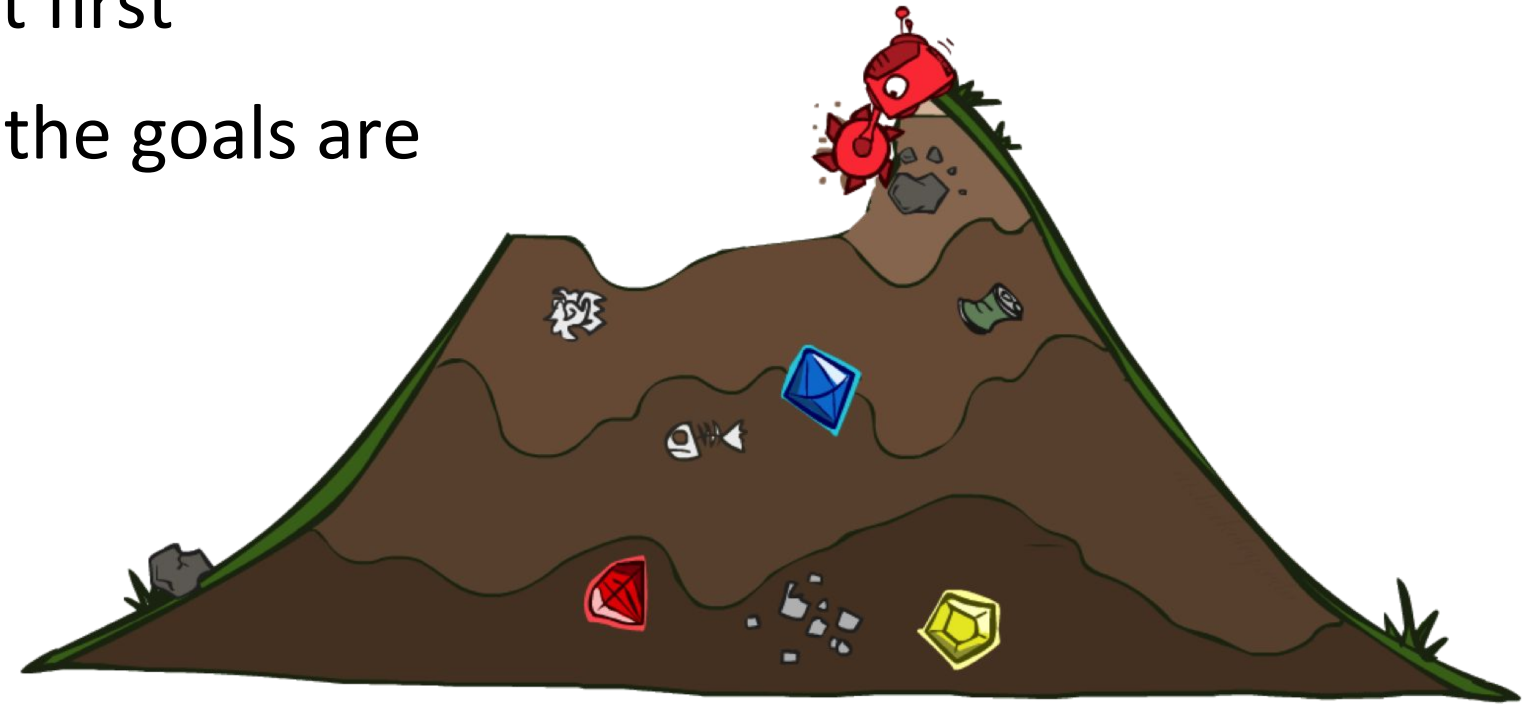
Search for a solution :

- Systematically build a search tree
- Choose an ordering of the fringe (unexplored nodes)
- Optimal: finds least-cost plans

# Uninformed Search

## Uninformed Search

- explores the search tree in a systematic way: top down, left to right, cheapest cost first
- unaware of where the goals are



# Informed Search

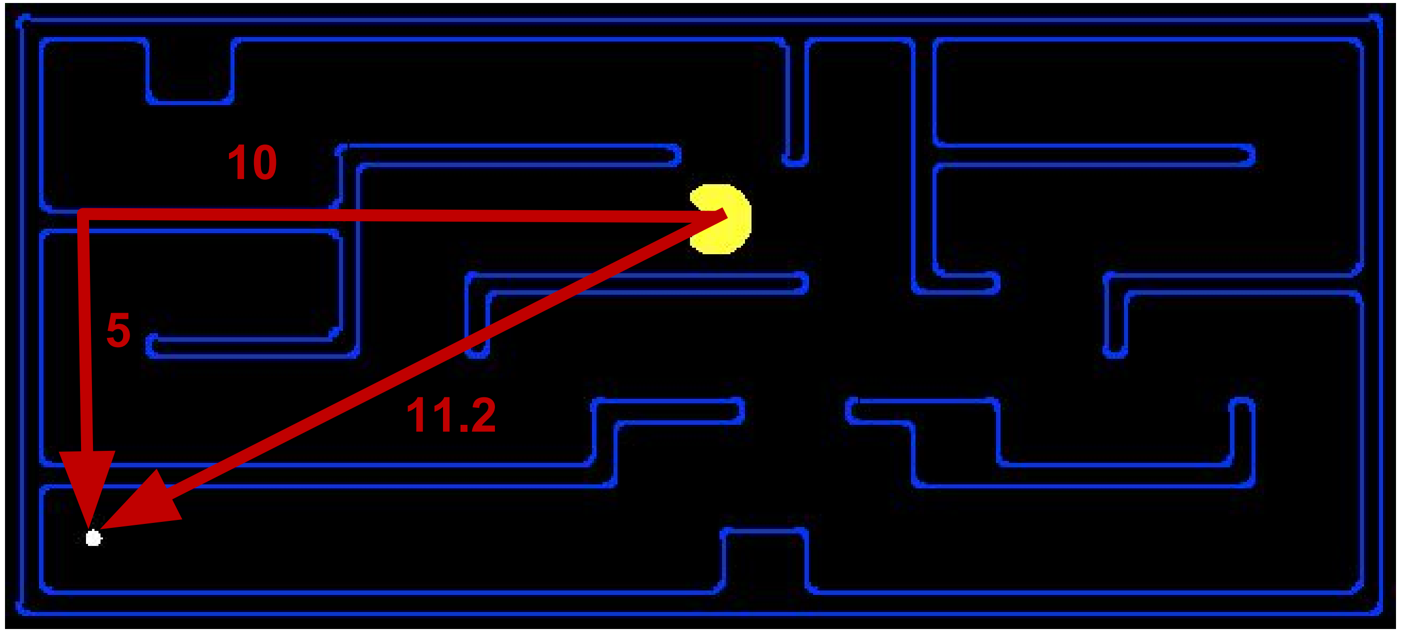
## Key Idea:

- In addition to everything else that our uninformed search was doing, we have something that tells us if we're getting hotter or colder.
- We are able to answer not just whether a state is a goal state or not but how close to the goal is it.

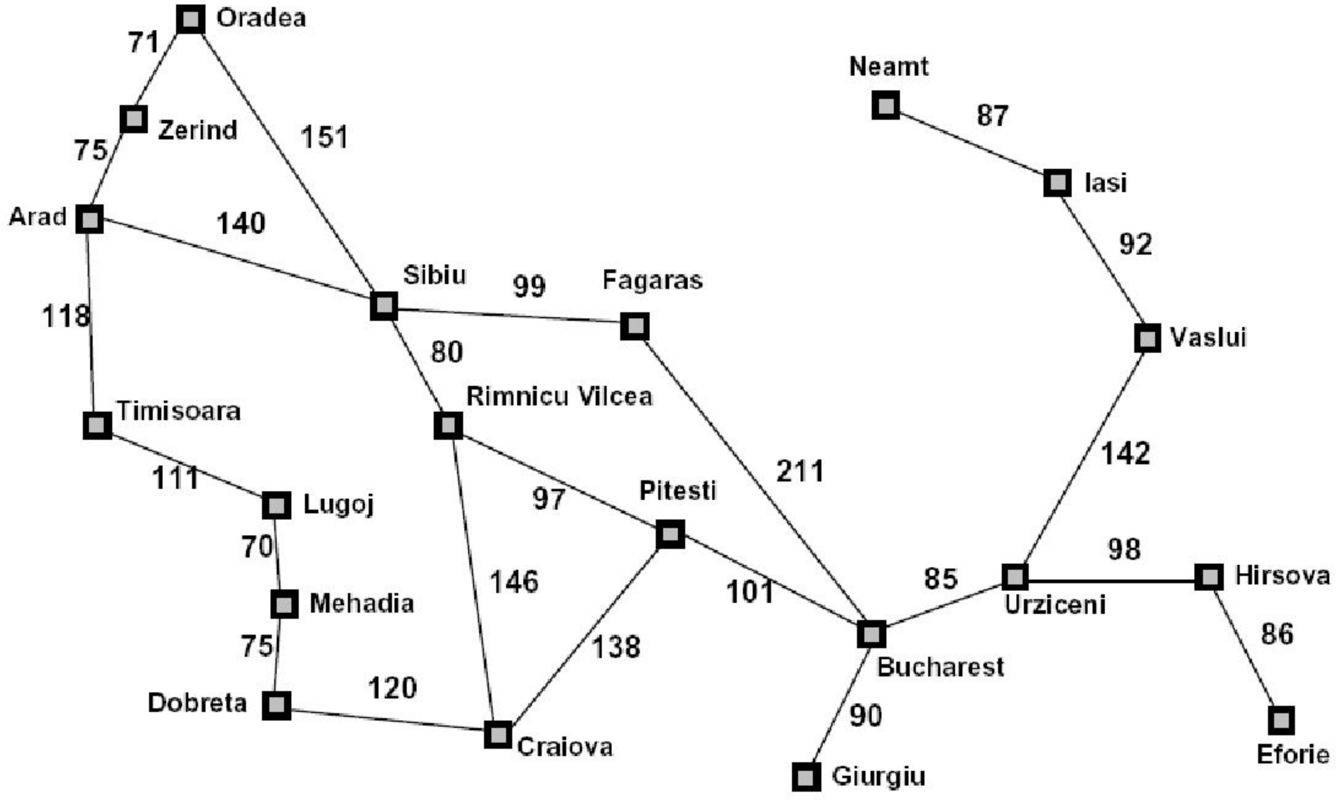
# Search Heuristics

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: Heuristic Function



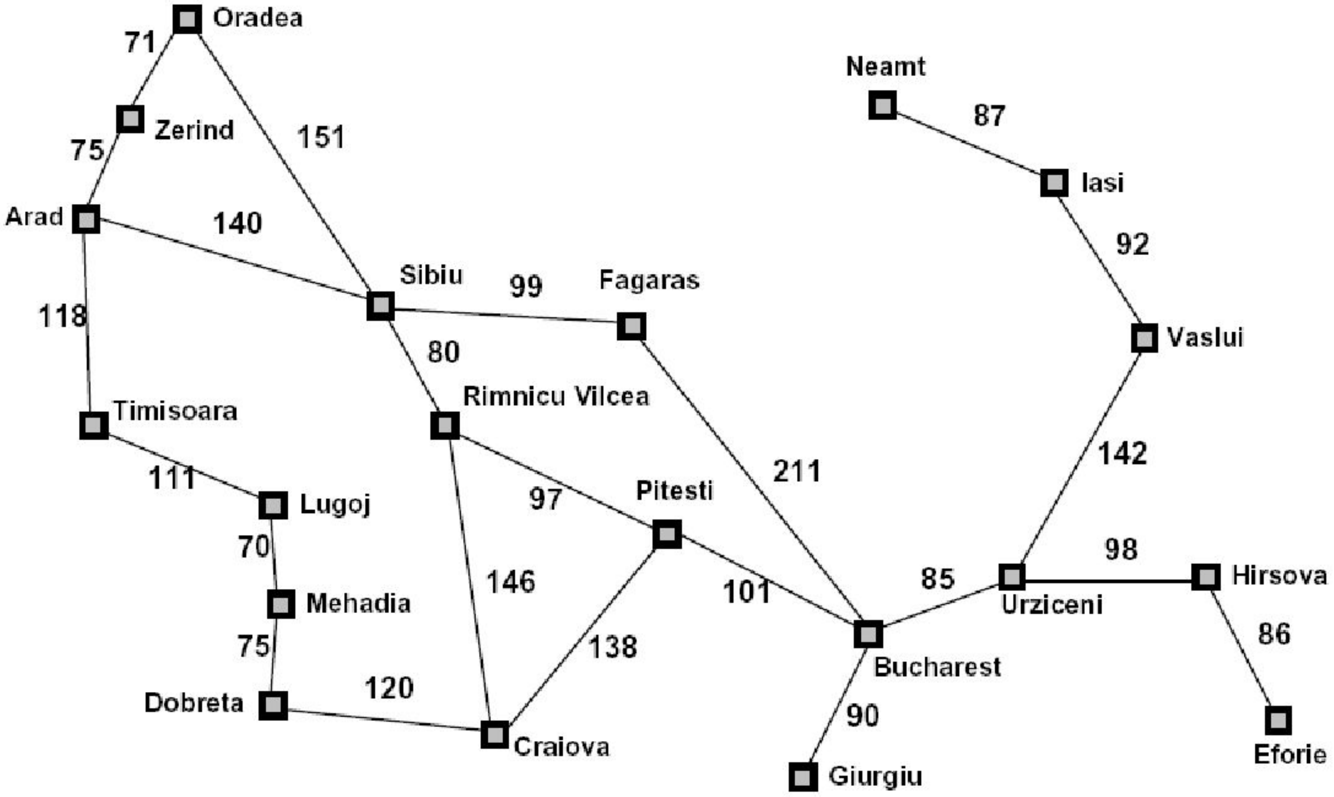

| Straight–line distance to Bucharest | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 178 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 98 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

h(x)

# Greedy Search

- explores the tree in the most promising direction

# Example: Greedy Search

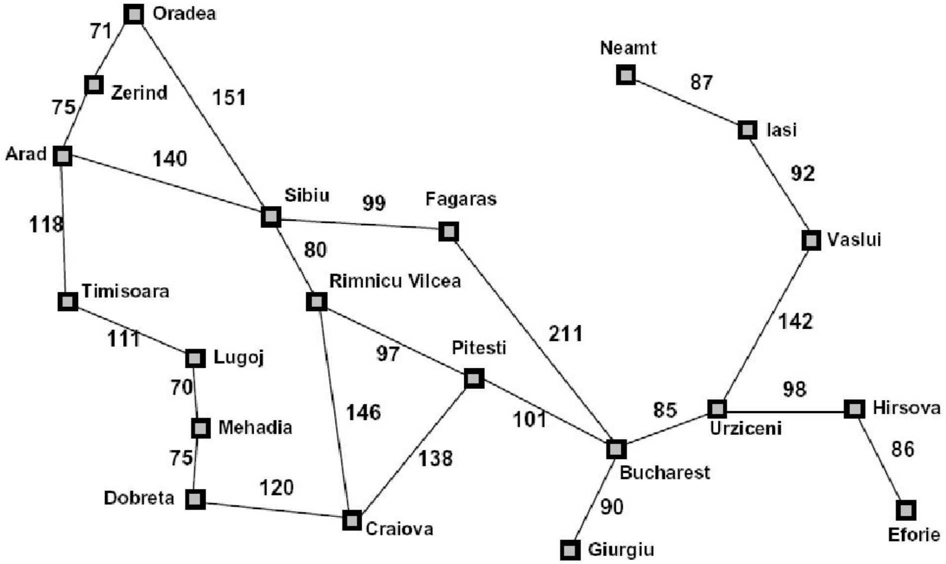



Straight−line distance to Bucharest

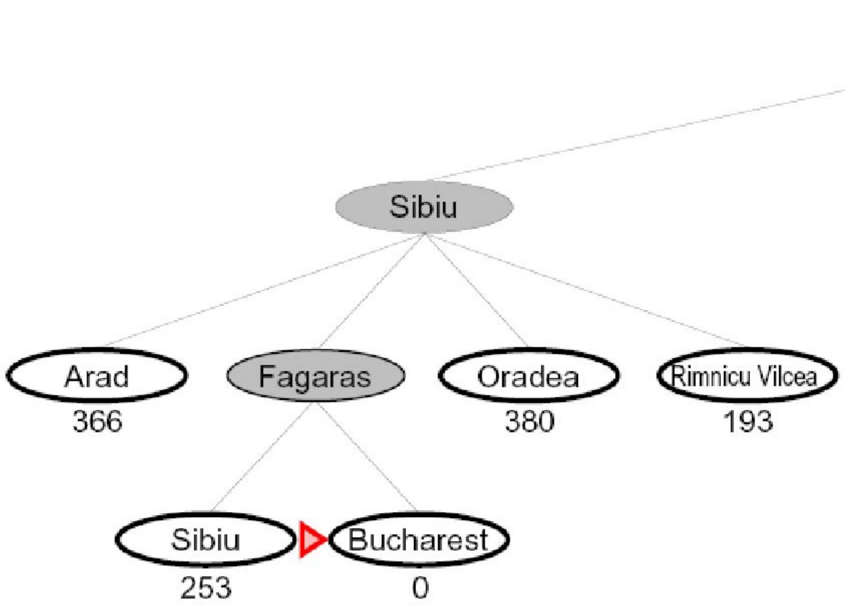
| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 178 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 98 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

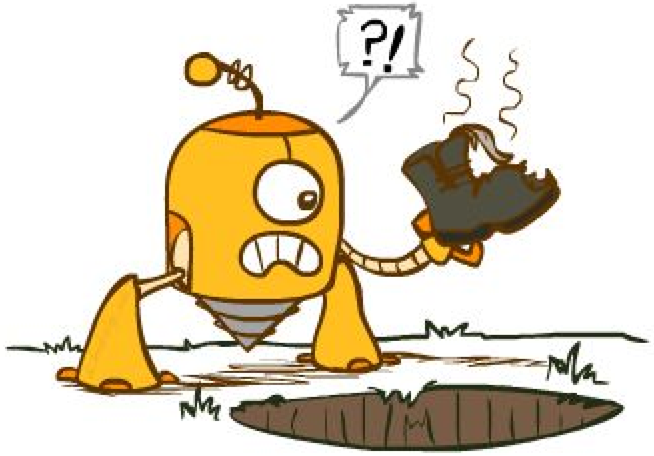
h(x)

# Greedy Search



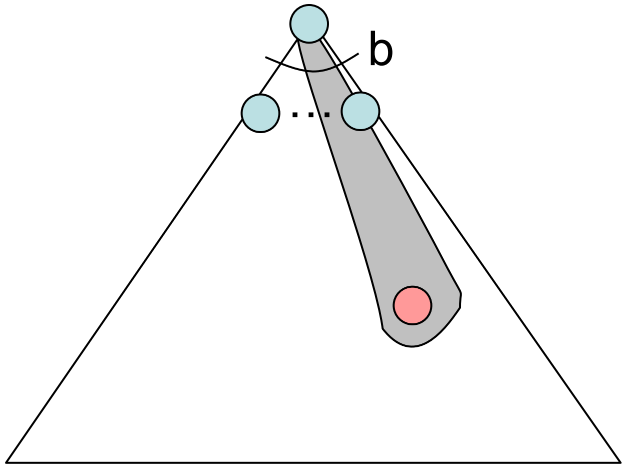

- Expand the node that seems closest…




- What can go wrong?

# Greedy Search

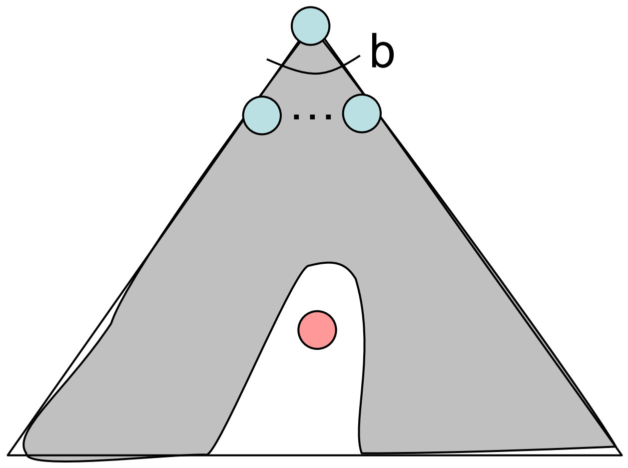



Strategy: expand a node that you think is closest to a goal state

- Heuristic: estimate of distance to nearest goal for each state

A common case:

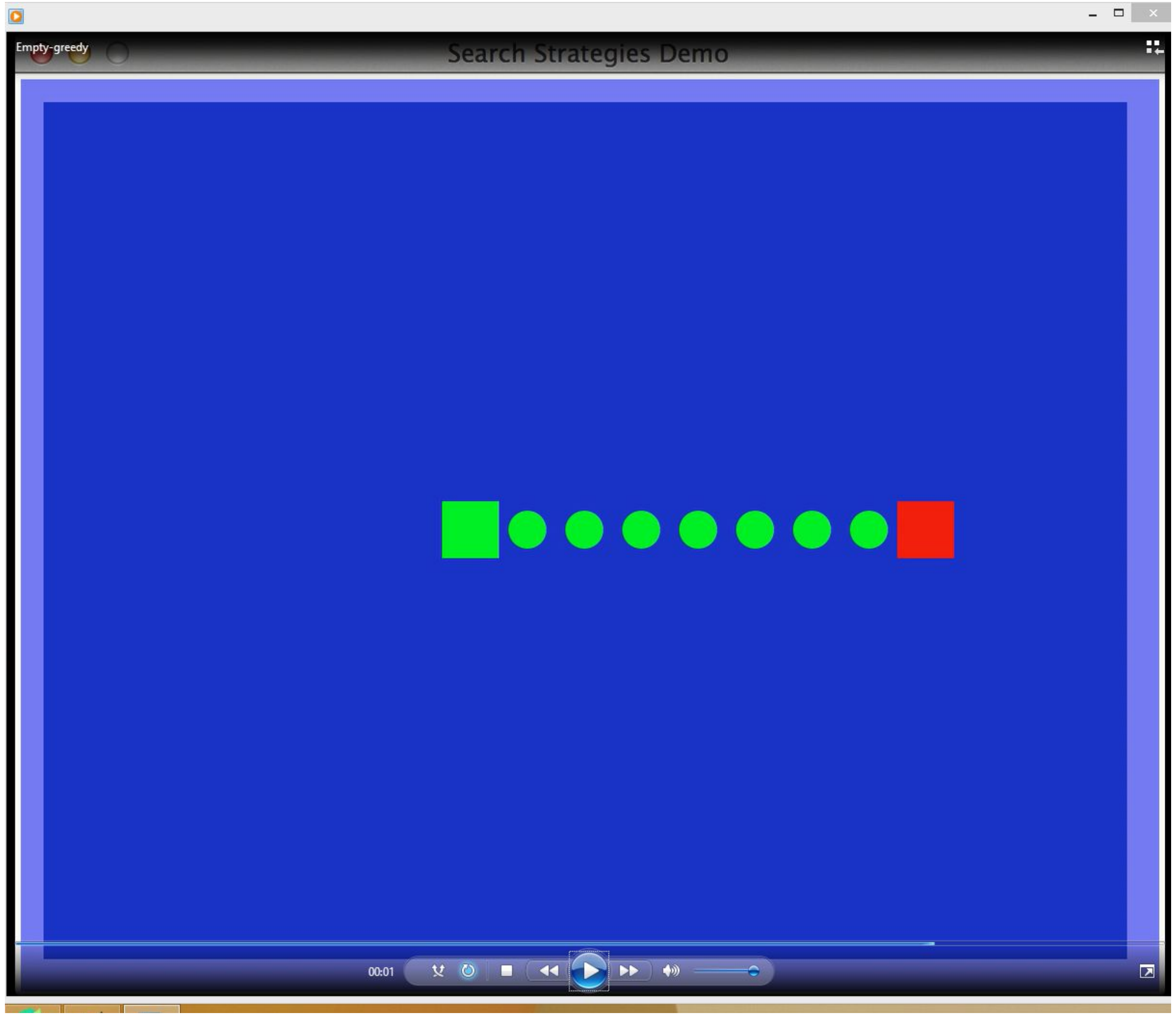
- Greedy search takes you straight to the (wrong) goal



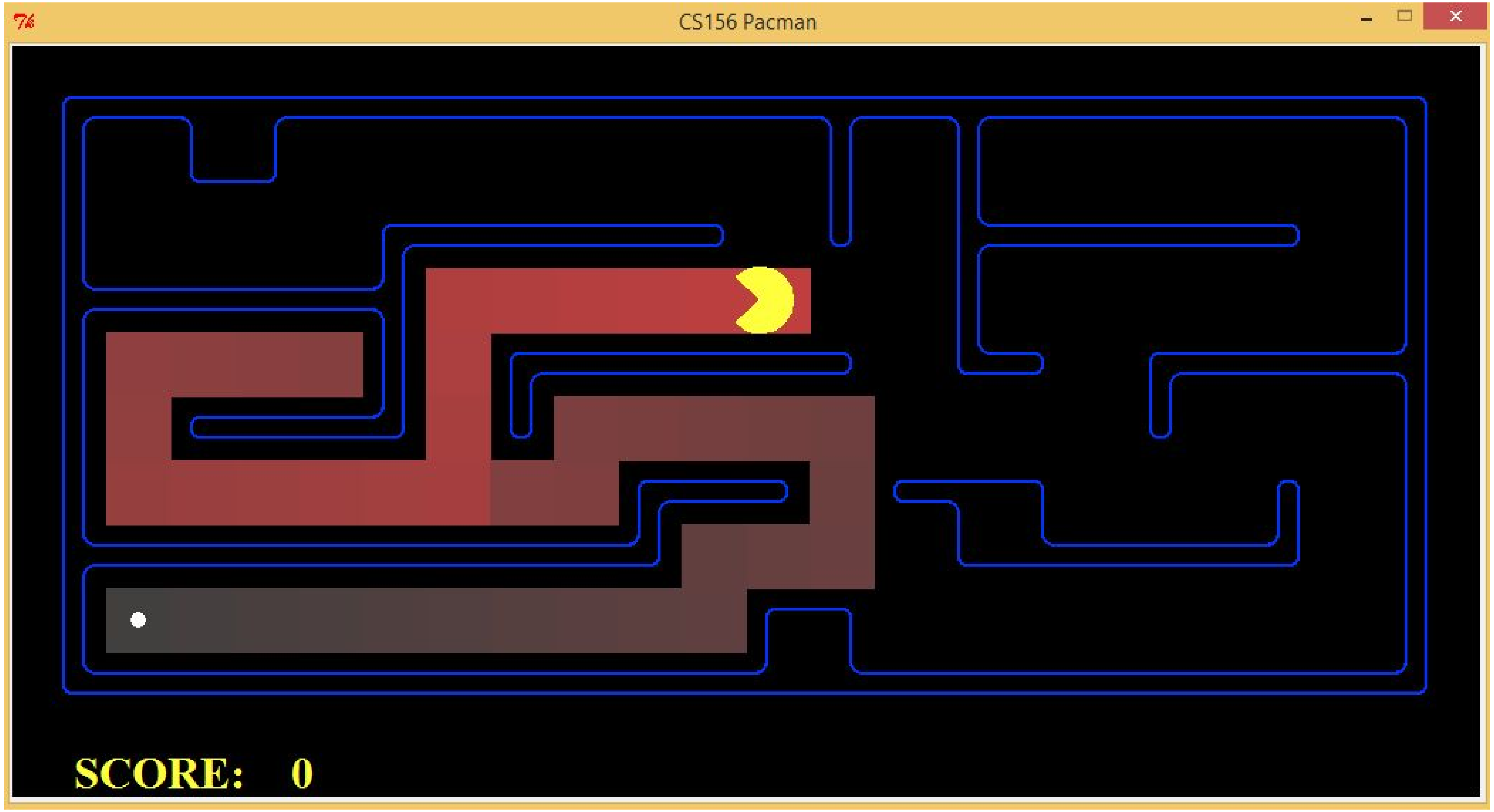

Worst-case: DFS:

- As good as the heuristic
- Bad heuristic: badly guided dfs

# Video of Demo Contours Greedy (Empty)

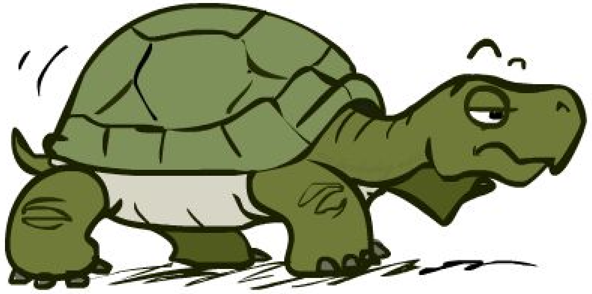


# Demo Greedy (Pacman Small Maze)

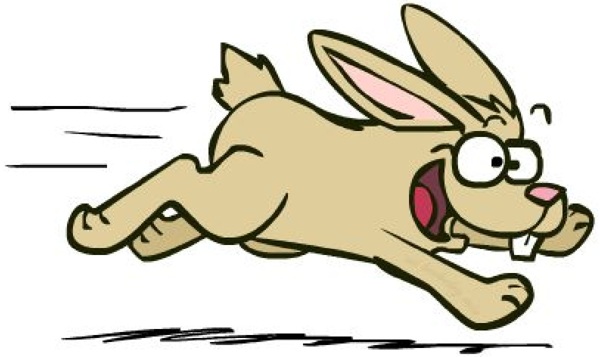


# A* Search

# A* Search



UCS

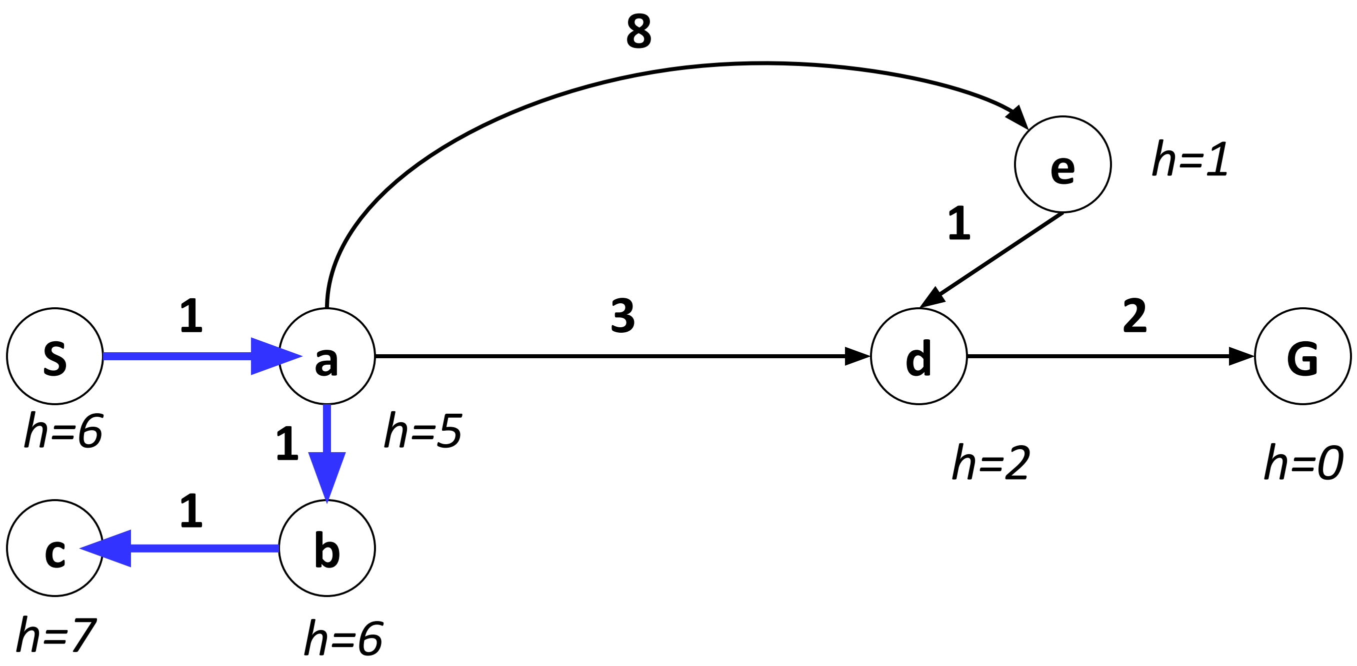


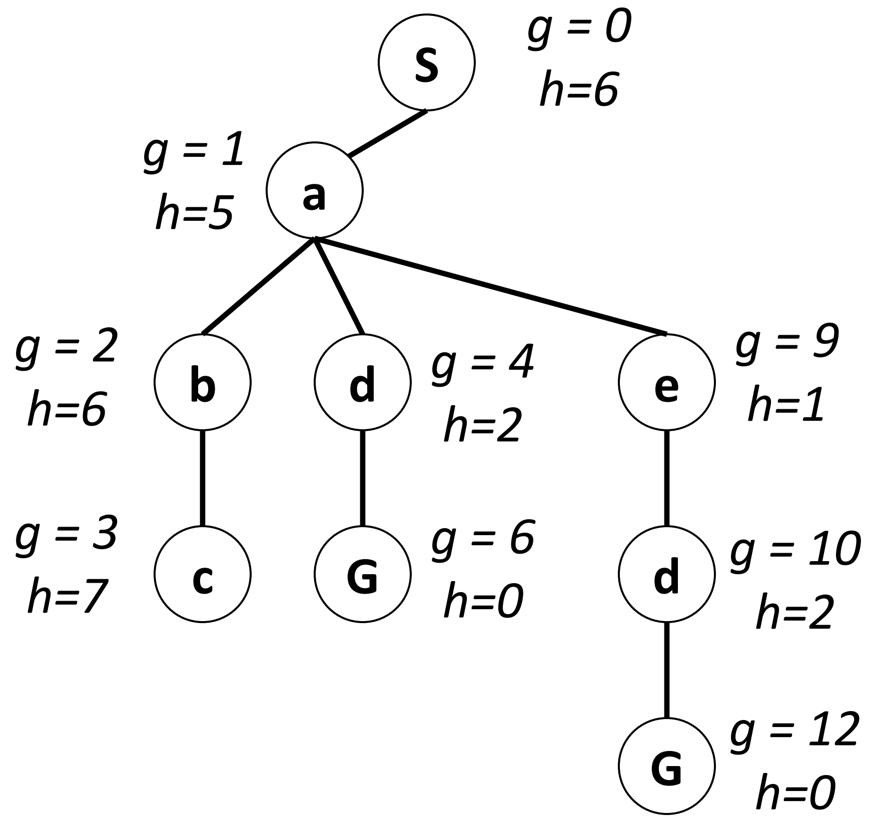
Greedy

A*

# Combining UCS and Greedy

- Uniform-cost orders by path cost, or *backward cost* g(n)

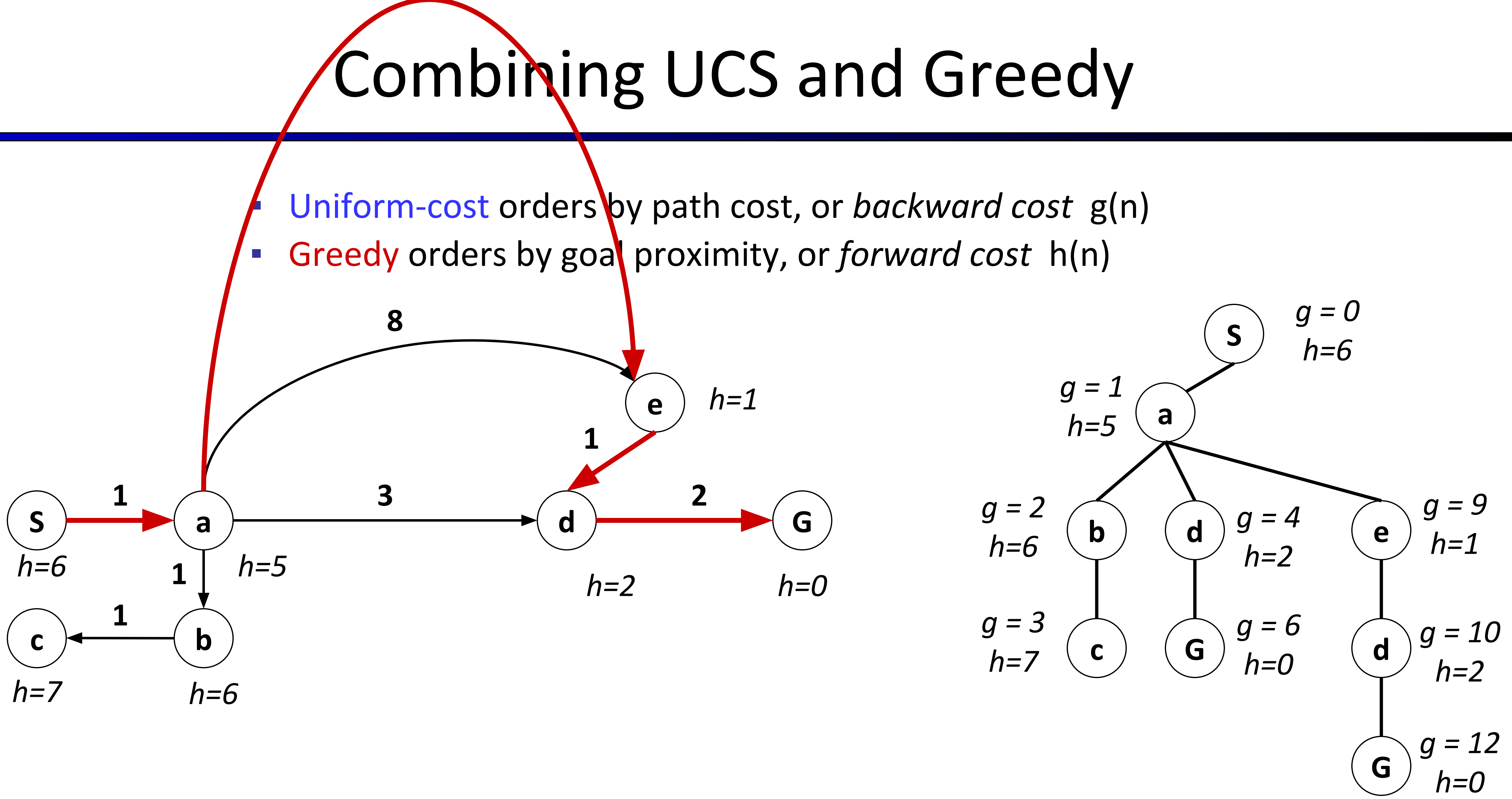



Example: Teg Grenager

# Combining UCS and Greedy

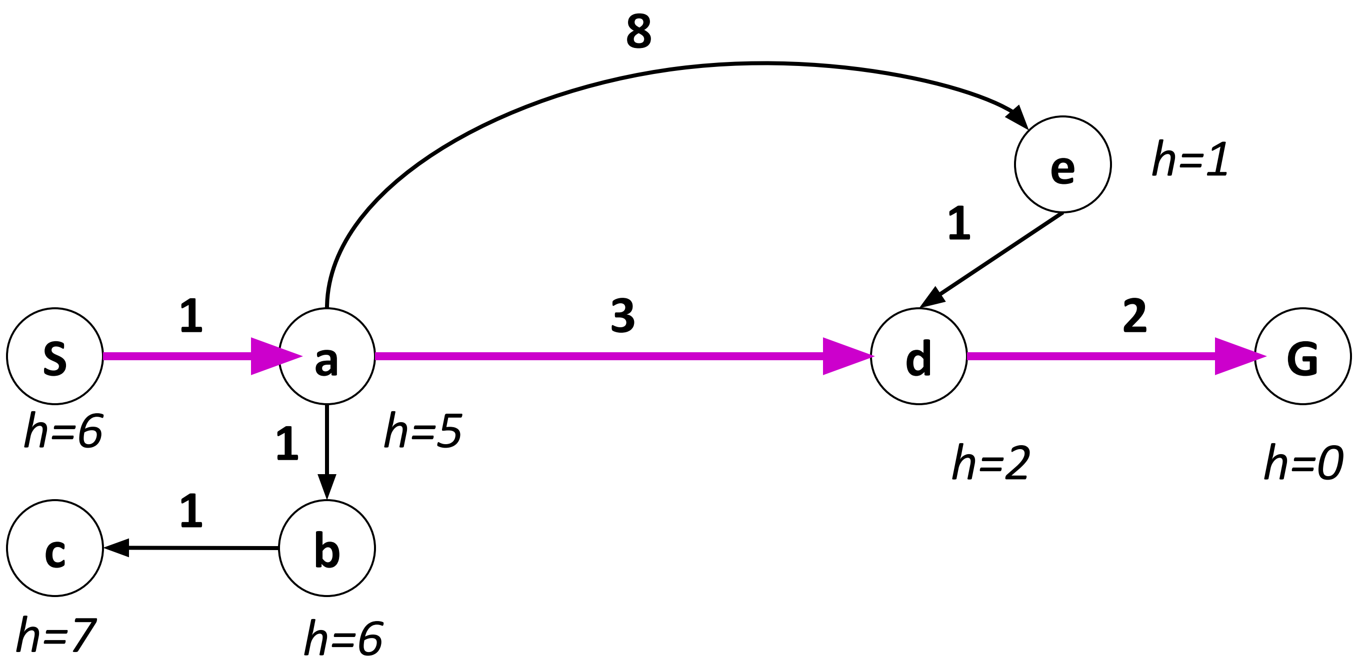
- Uniform-cost orders by path cost, or *backward cost* g(n)
- Greedy orders by goal proximity, or *forward cost* h(n)

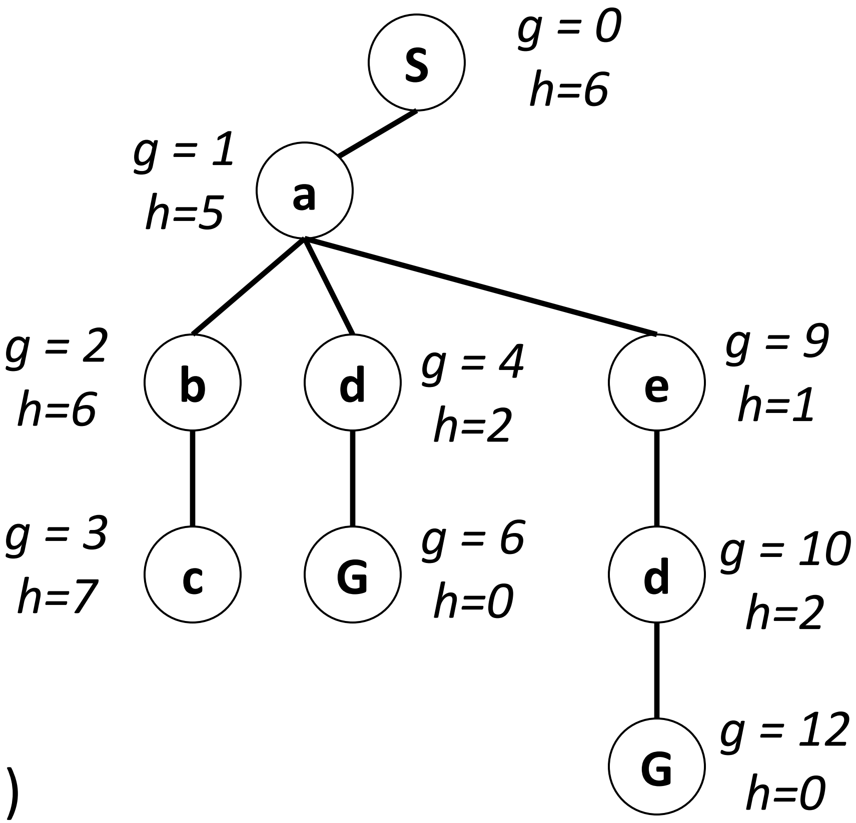



Example: Teg Grenager

# Combining UCS and Greedy

- Uniform-cost orders by path cost, or *backward cost* g(n)
- Greedy orders by goal proximity, or *forward cost* h(n)



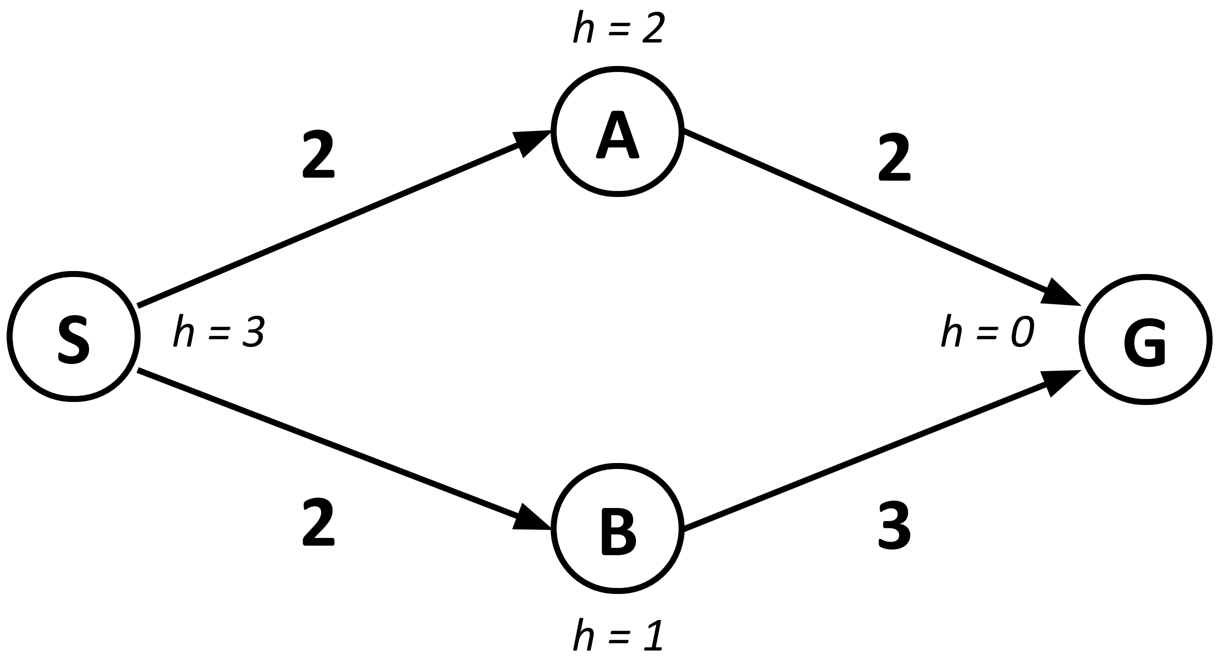




- A* Search orders by the sum: $f(n) = g(n) + h(n)$

Example: Teg Grenager

# When should A* terminate?

Should we stop when we enqueue a goal?

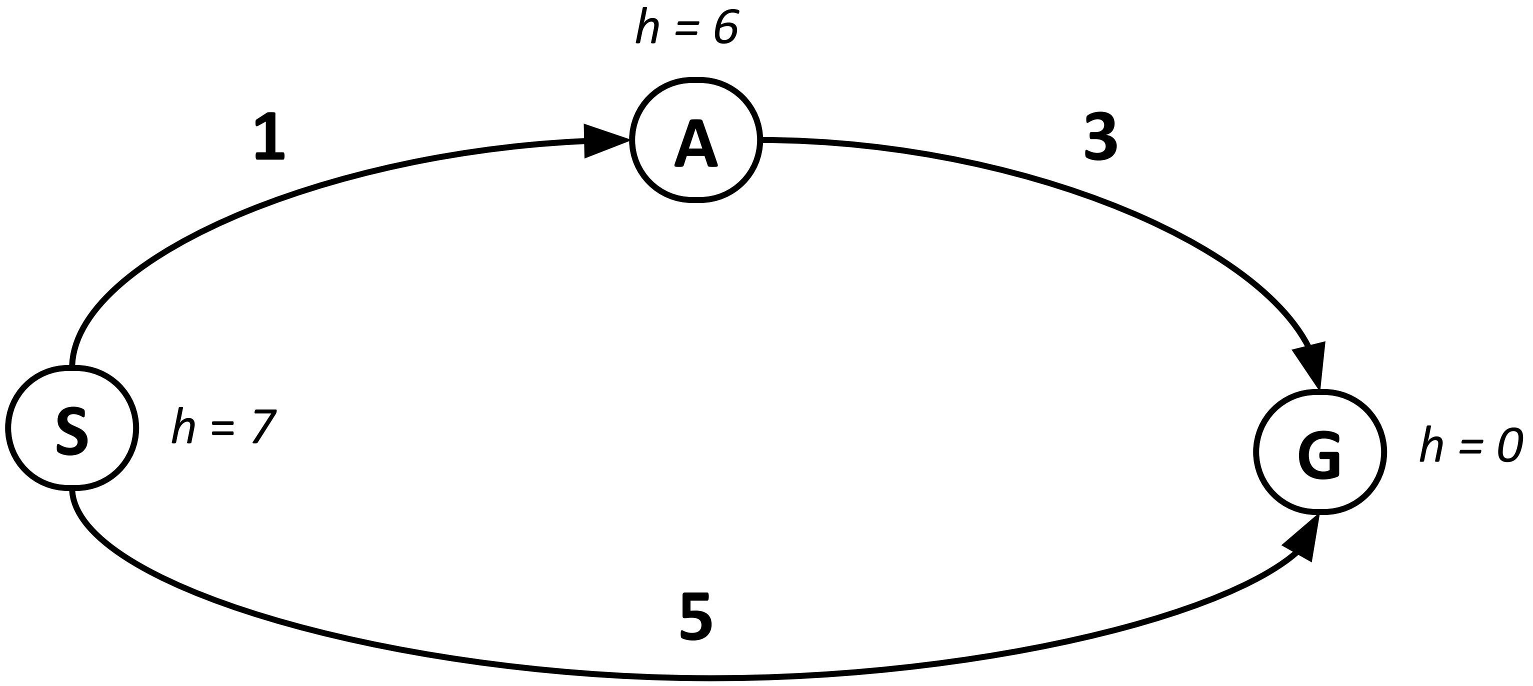



Fringe

| path | g | h | f |
|---|---|---|---|
| ~~S~~ | 0 | 3 | 3 |
| ~~S-A~~ | 2 | 2 | 4 |
| ~~S-B~~ | 2 | 1 | 3 |
| S-B-G | 5 | 0 | 5 |
| S-A-G | 4 | 0 | 4 |

No: only stop when we dequeue a goal

# Is A* Optimal?

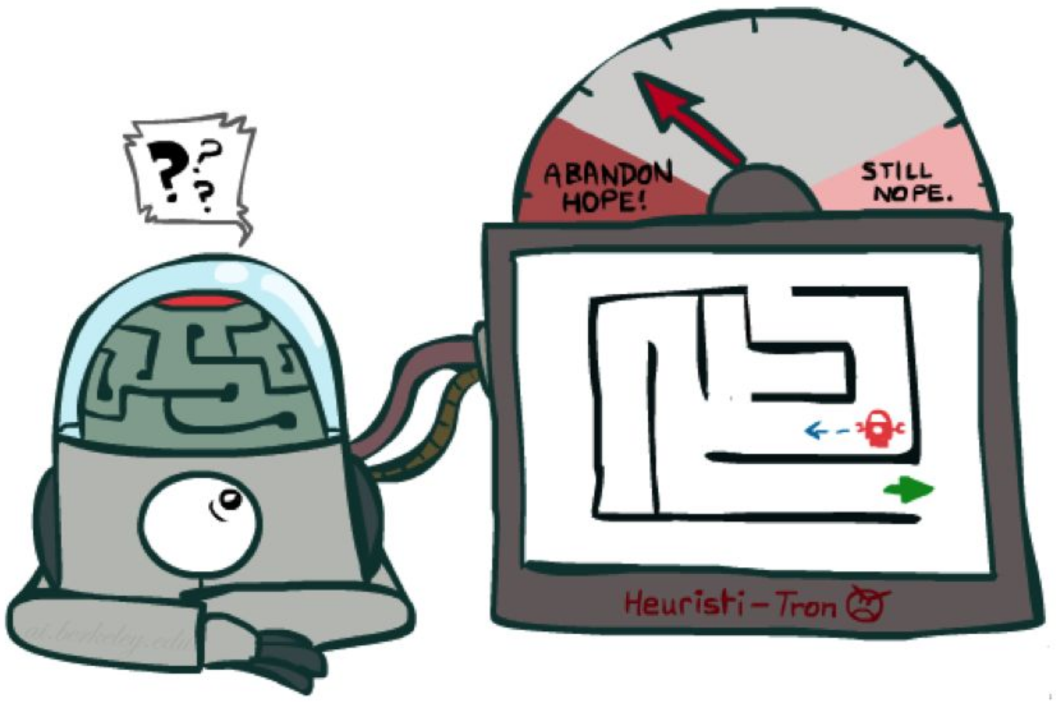



Fringe

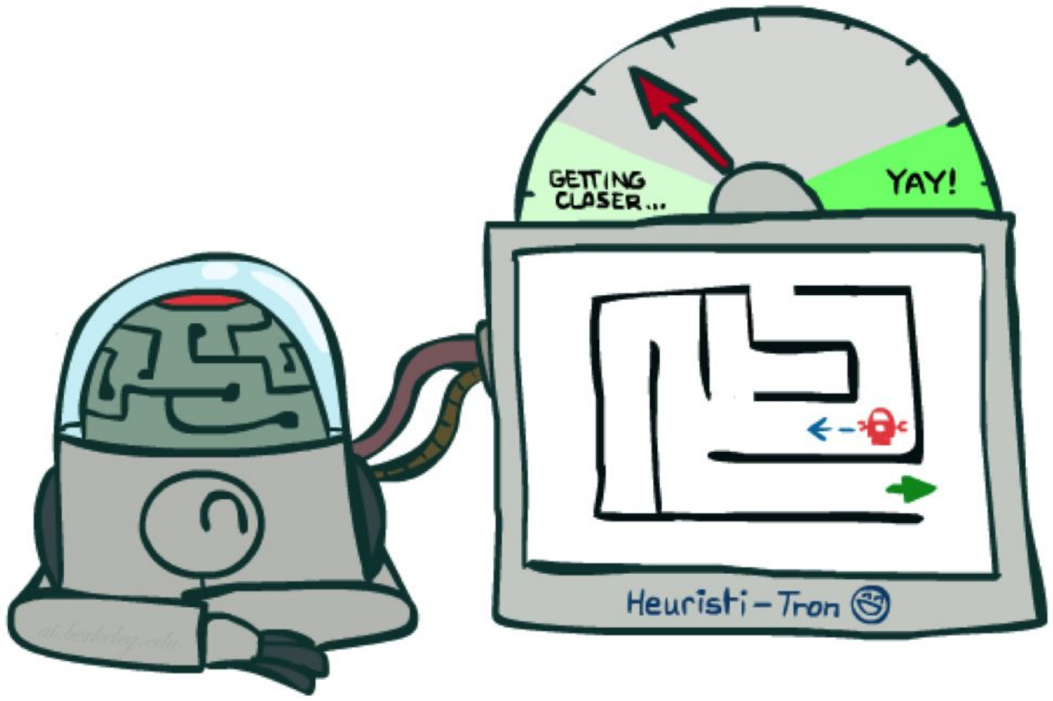
| path | g | h | f |
|---|---|---|---|
| ~~S~~ | 0 | 7 | 7 |
| S-A | 1 | 6 | 7 |
| $\boxed{\text{S-G}}$ | 5 | 0 | 5 |

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

# Idea: Admissibility




Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the fringe




Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs

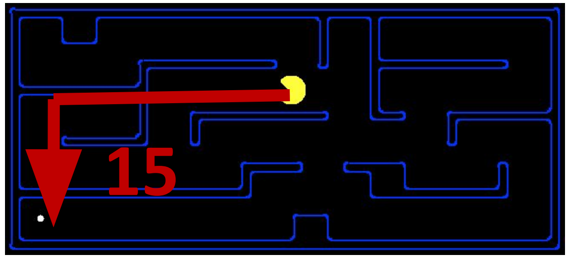
# Admissible Heuristics

- A heuristic $h$ is *admissible* (optimistic) if:

$$0 \le h(n) \le h^*(n)$$

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

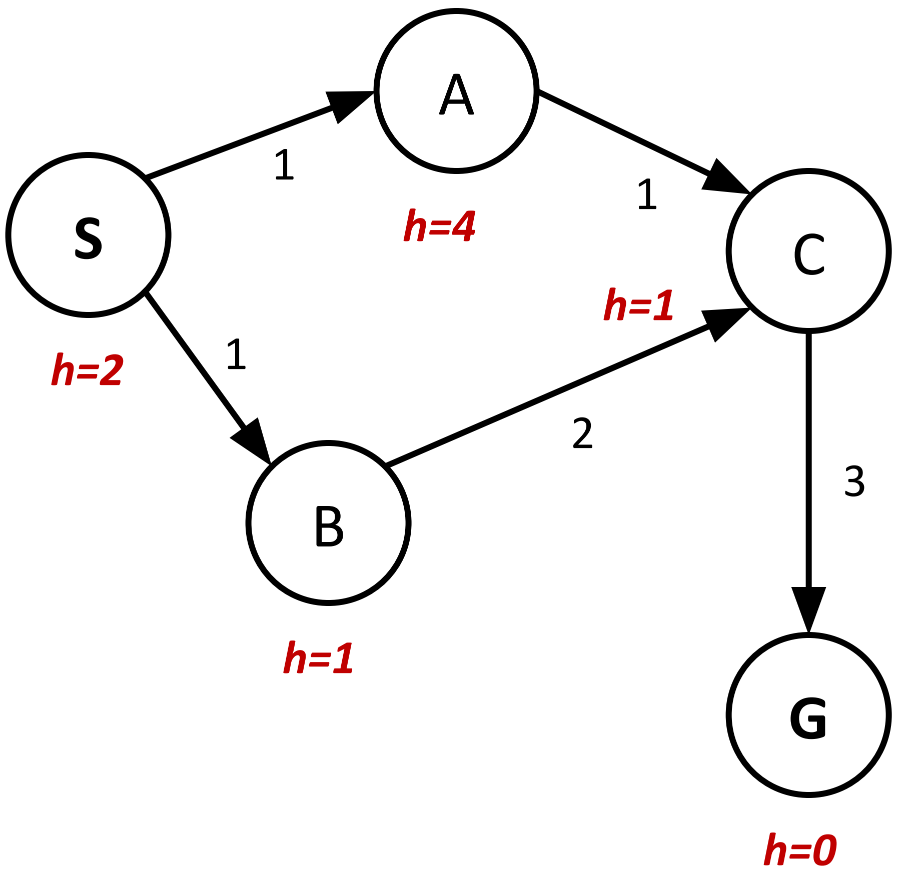
- Example:



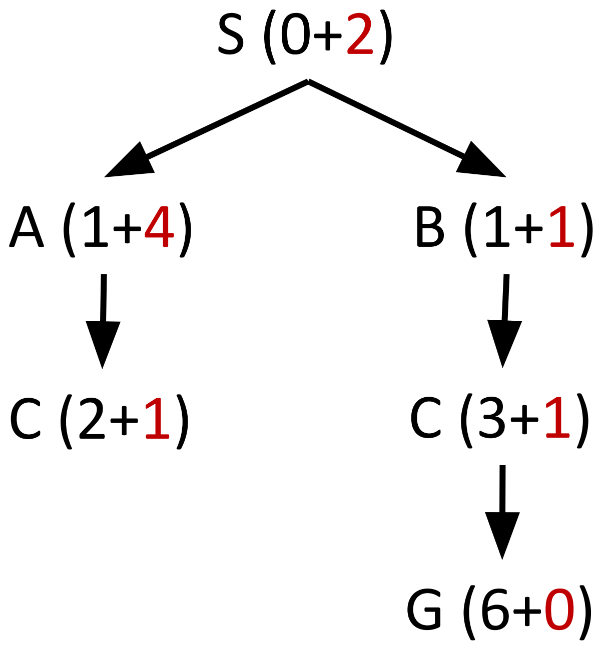

- Coming up with admissible yet useful heuristics is most of what's involved in using A* in practice.

# A* Graph Search Gone Wrong?

State space graph



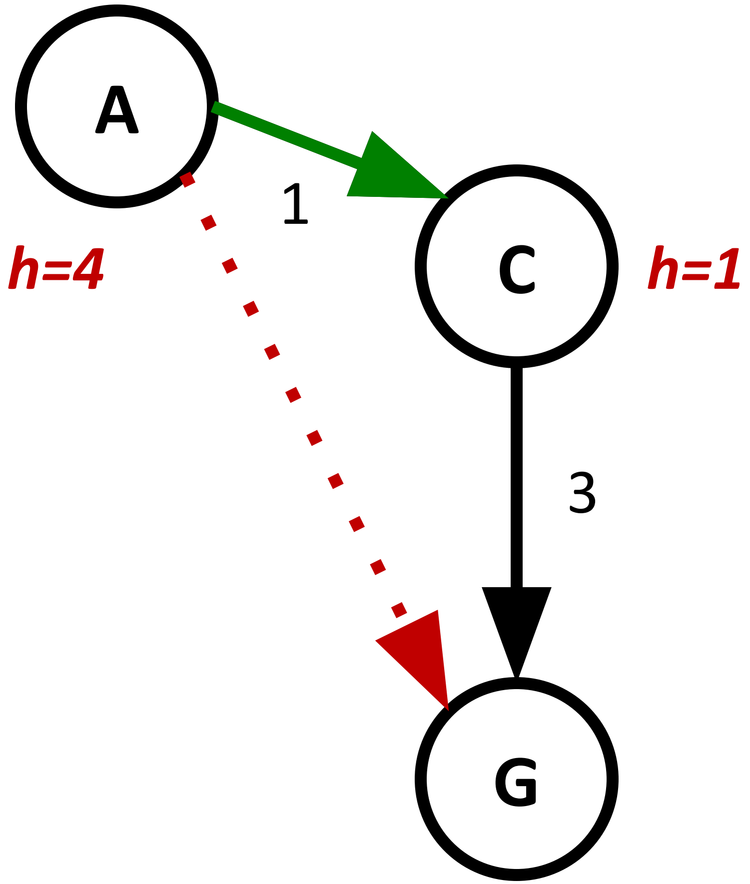

Search tree




Closed set

S B C A

# Consistency of Heuristics



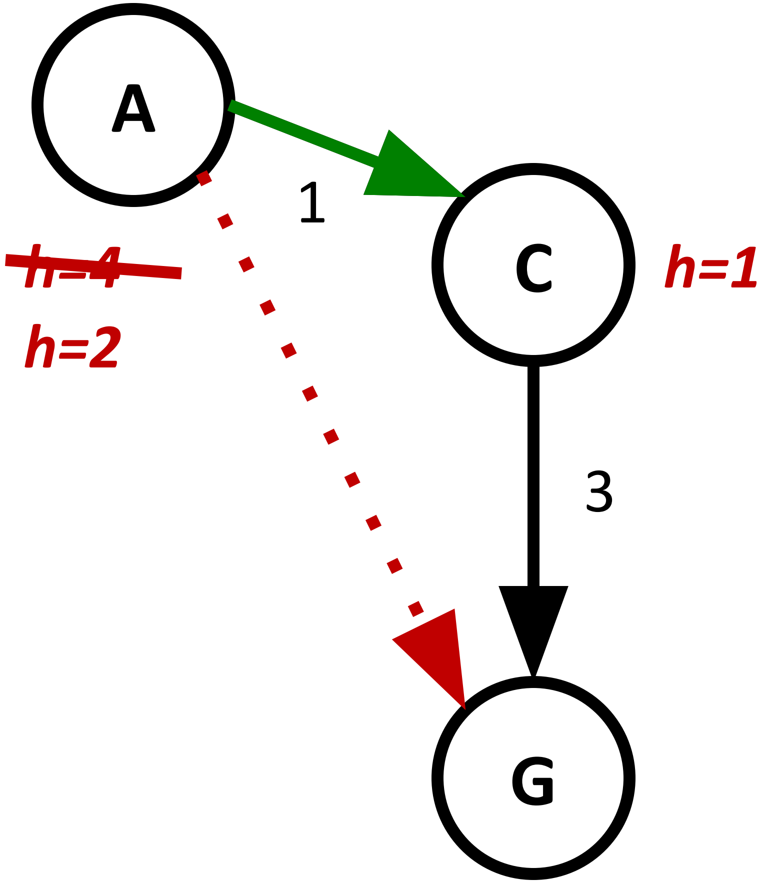

- Main idea: estimated heuristic costs ≤ actual costs
  - Admissibility: heuristic cost ≤ actual cost to goal

    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)

# 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)

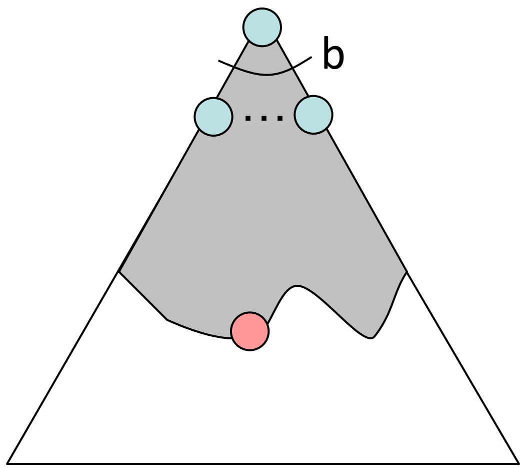
- Consequences of consistency:
  - The f value along a path never decreases

    h(A) ≤ cost(A to C) + h(C)

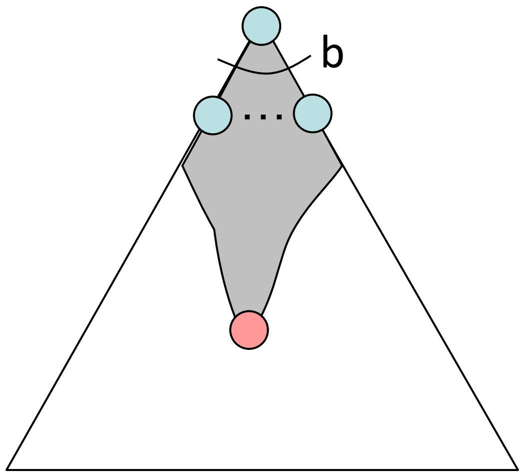
  - A* graph search is optimal

# Properties of A*

Uniform-Cost

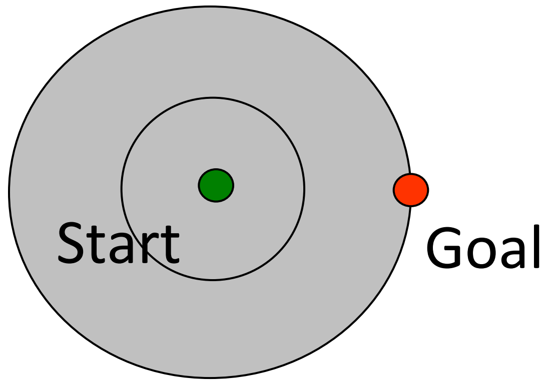



A*

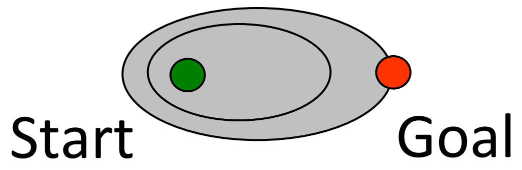


# UCS vs A* Contours

- Uniform-cost expands equally in all "directions"




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



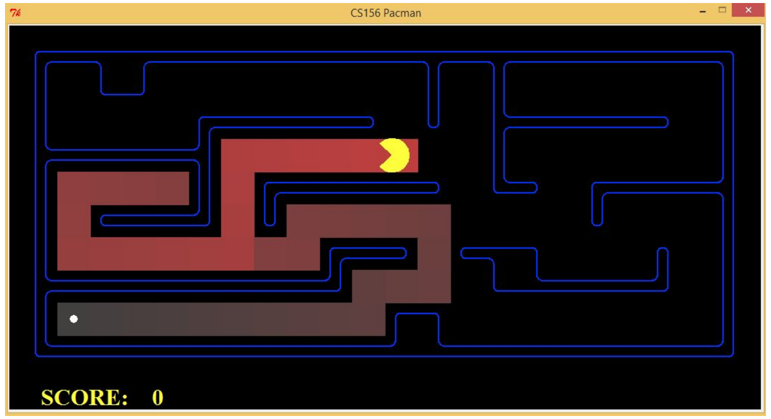
# Video of Demo Contours (Empty)

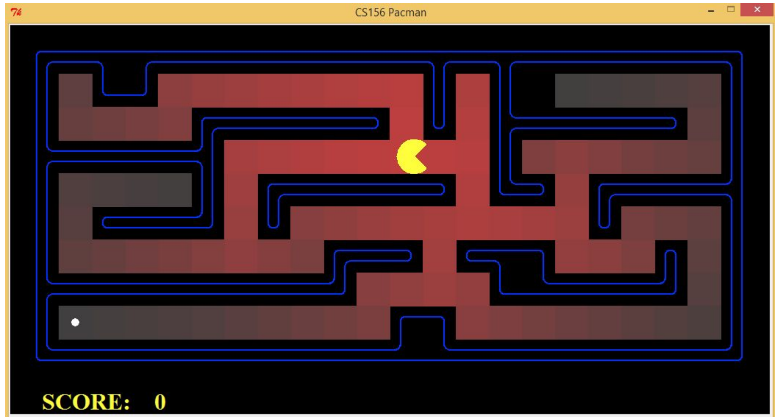
- UCS vs greedy vs A*

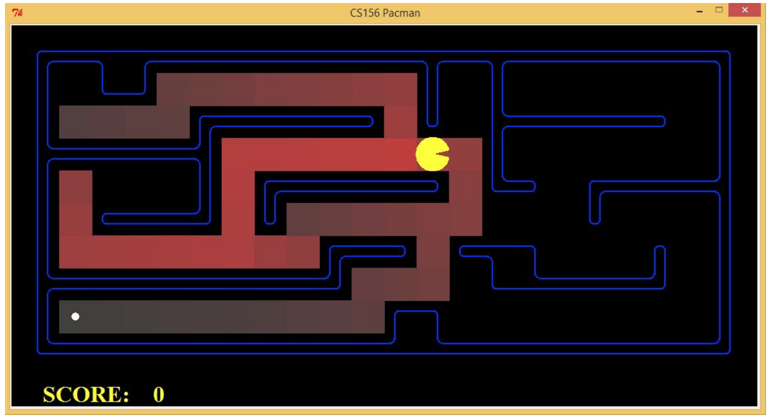
# Demo Pacman Small Maze

- UCS vs greedy vs A*

# Comparison




Greedy

Uniform Cost

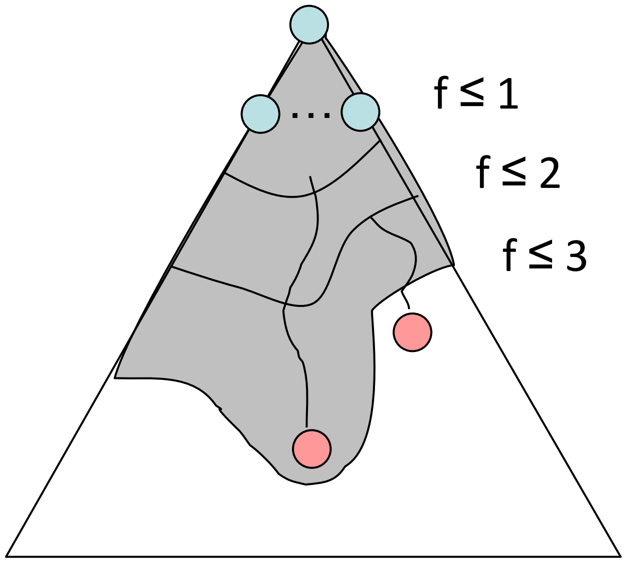
A*

# Optimality of A* Graph Search

Consider what A* does with a consistent heuristic:

1. In tree search, A* expands nodes in increasing total f value (f-contours)
2. For every state s, nodes that reach s optimally are expanded before nodes that reach s suboptimally

⇨ A* graph search is optimal



# Optimality

Tree search:

- A* is optimal if heuristic is admissible
- UCS is a special case (h = 0)

Graph search:

- A* optimal if heuristic is consistent
- UCS optimal (h = 0 is consistent)

Consistency implies admissibility

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