Lecture 4
Solving Problems by Searching:
Informed Search

Thanapon Noraset
Faculty of ICT, Mahidol University

Adapted from AIMA by Stuart Russell and Peter Norvig, and UC Berkeley CS188 by Dan Klein and Pieter Abbeel (ai.berkeley.edu)

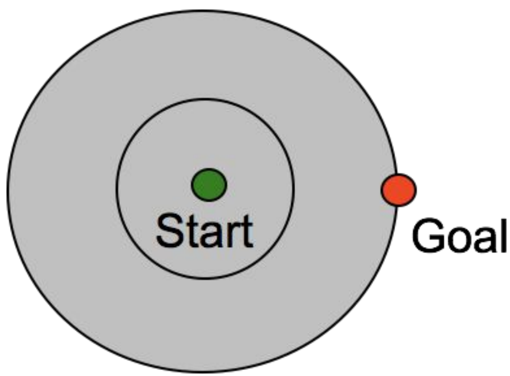
Agenda

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

Uniform Cost Search

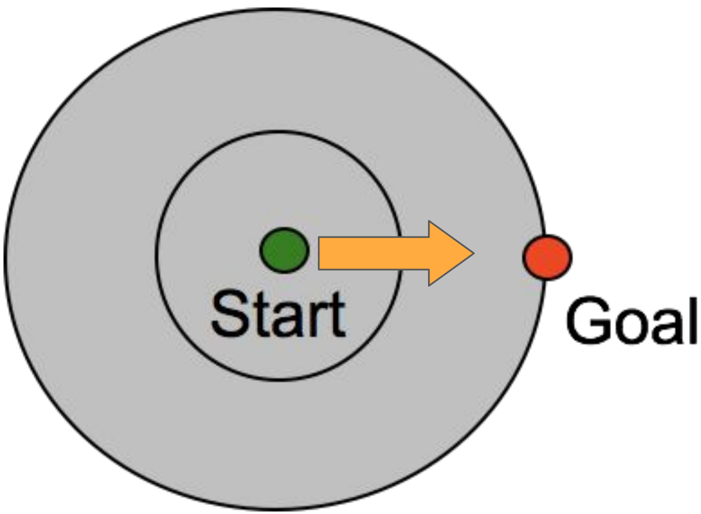
- Work with non-uniform costs
- Complete and Optimal !

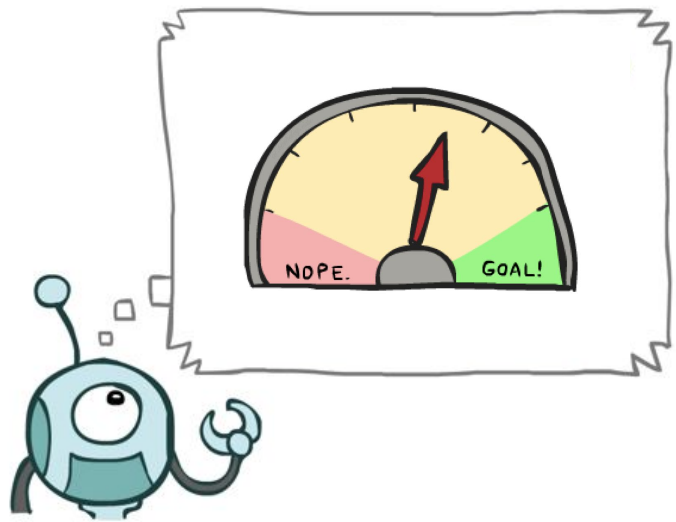
- But, explores every “direction”



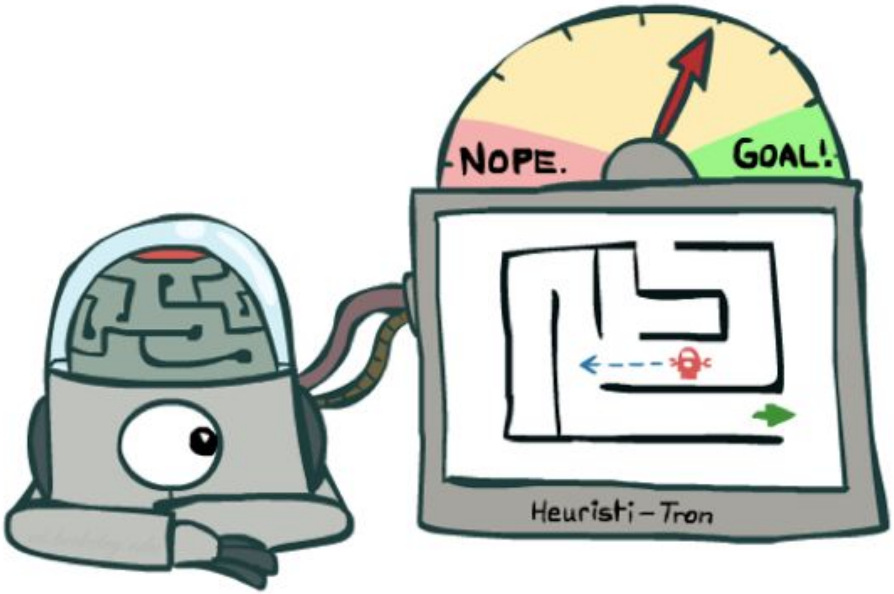
Location of the Goal

- Let's not exploring every "direction"
- We could do better if we know a good direction to the goal





Informed Search Strategies



Search Heuristics

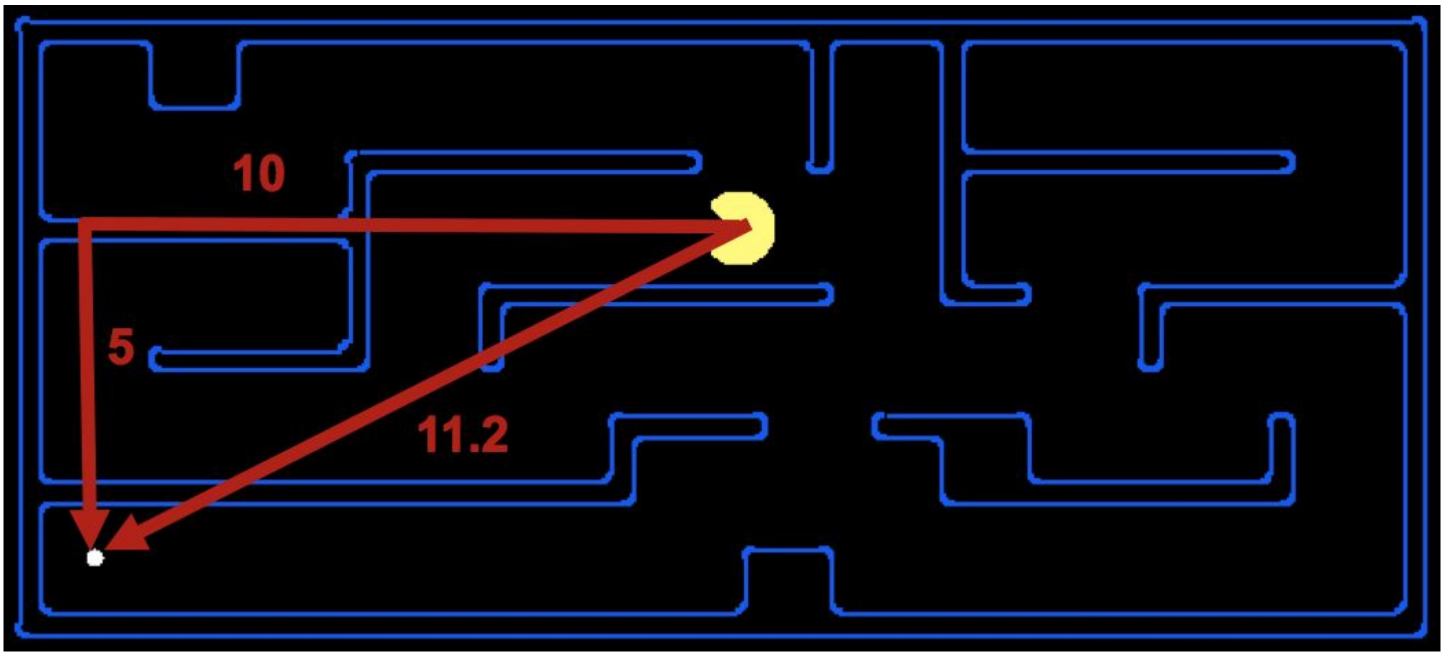
A search heuristics:

- A function that estimates how close a state is to a goal.
- Usually involves domain knowledge for a particular search problem.
- *Preferably easy to compute*.

Search Heuristics: Example

For a path finding problem, heuristics could be:

- Euclidean Distance
- Manhattan Distance



Path Costs vs Heuristic Functions

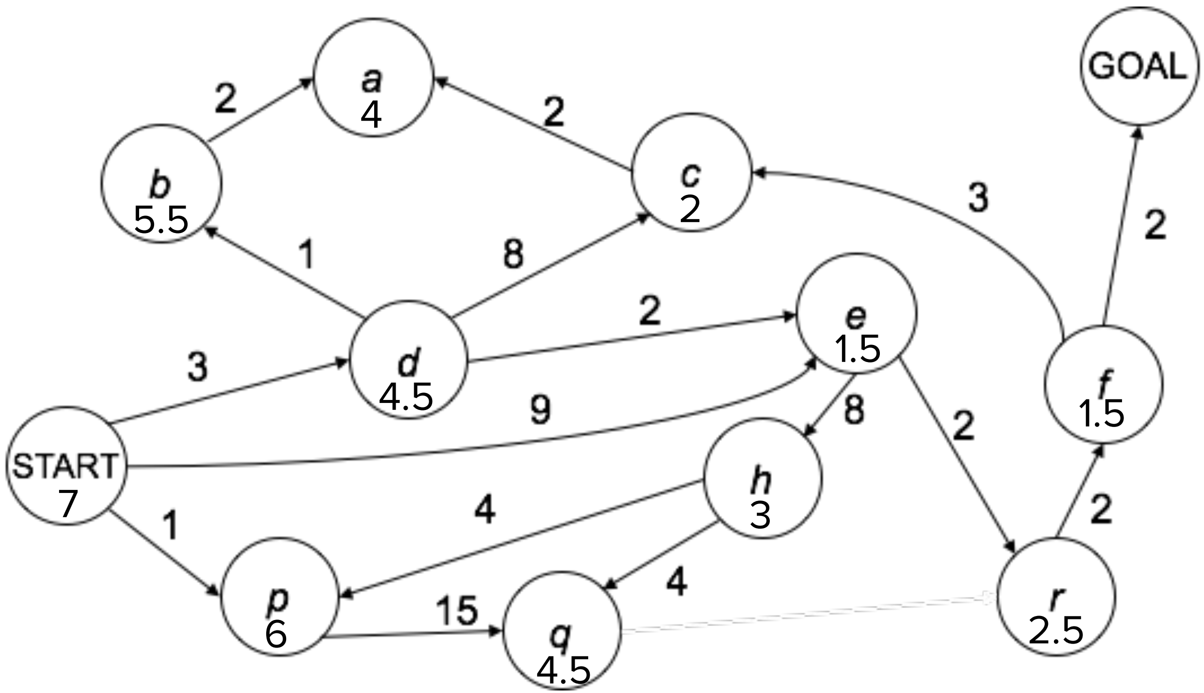
Path Costs

- Sum of costs of each action so far
- Depends on the whole path to the node
- Called backward cost
- *g(n)*

Heuristic Functions

- Proximity to the goal from the node
- Depends on the node alone
- Called forward cost
- *h(n)*

Example: straight-line distance



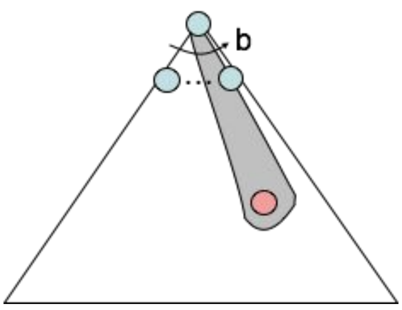
Greedy Best-first Search

Greedy Best-First Search

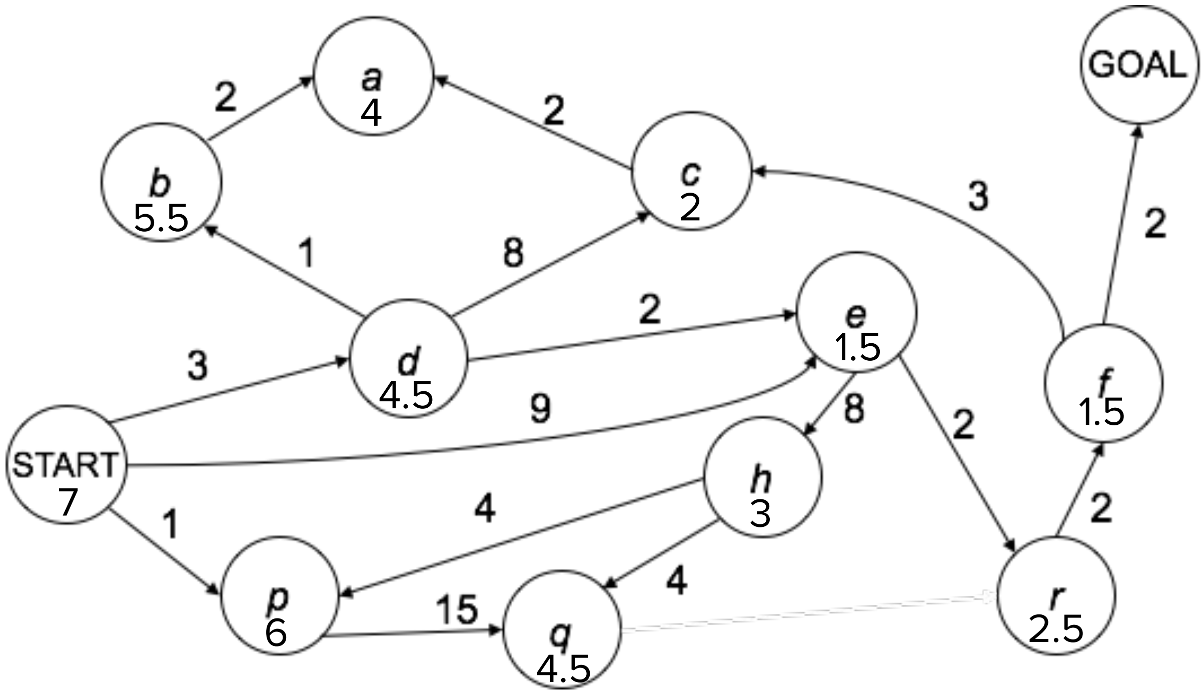
Strategy: Expand a node that heuristic says closest to the goal state

Implementation: Priority Queue
(sorted by h(n))

- Many names: greedy search, best-first search, pure heuristic search



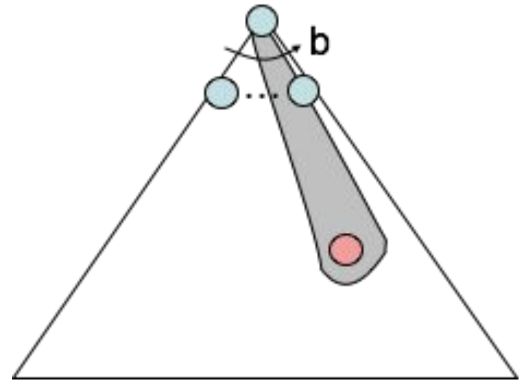
Greedy Best-First Search



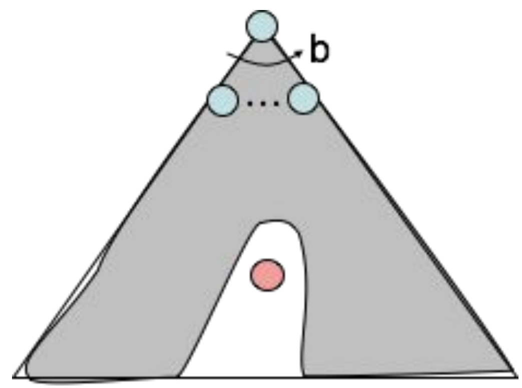
Path cost = ?

Greedy Best-First Search

- Complete?
  - No, if m is infinite
- Optimal?
  - No, it goes straight to the goal
  - But sometimes, not best way
- Time complexity:
  - $O(b^m)$ in the worst case
- Space complexity:
  - $O(b^m)$ in the worst case



Good heuristic



Bad heuristic

A* Search

A* Search

- Combining UCS and Greedy search

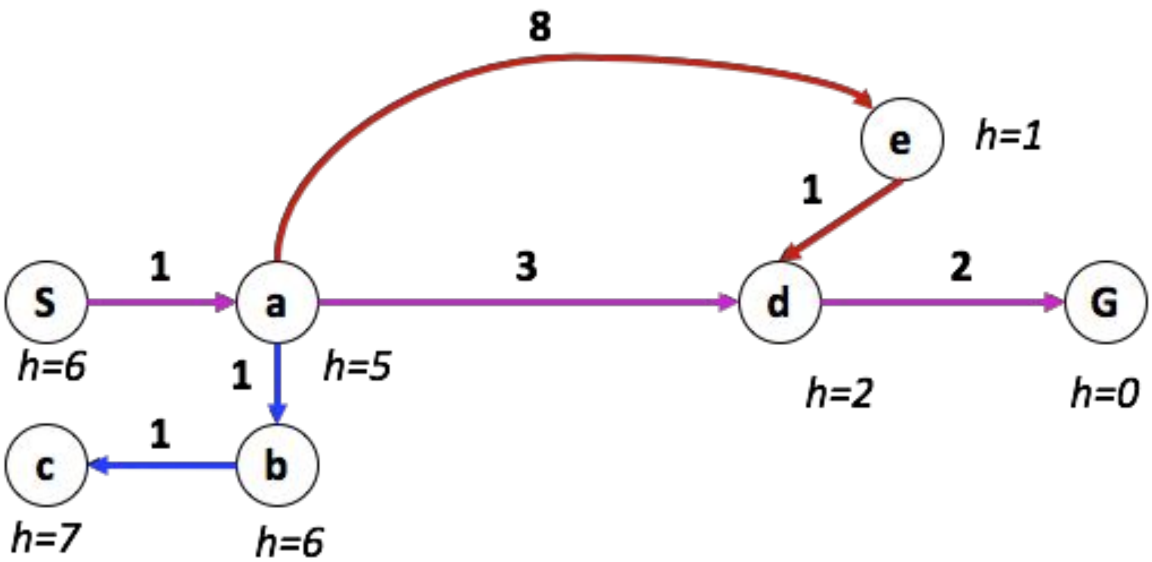
$$f(n) = g(n) + h(n)$$

Strategy: Expand a node that has the lowest *f(n)*

Implementation: Priority Queue

(Sorted by *f(n)*)

A* Search

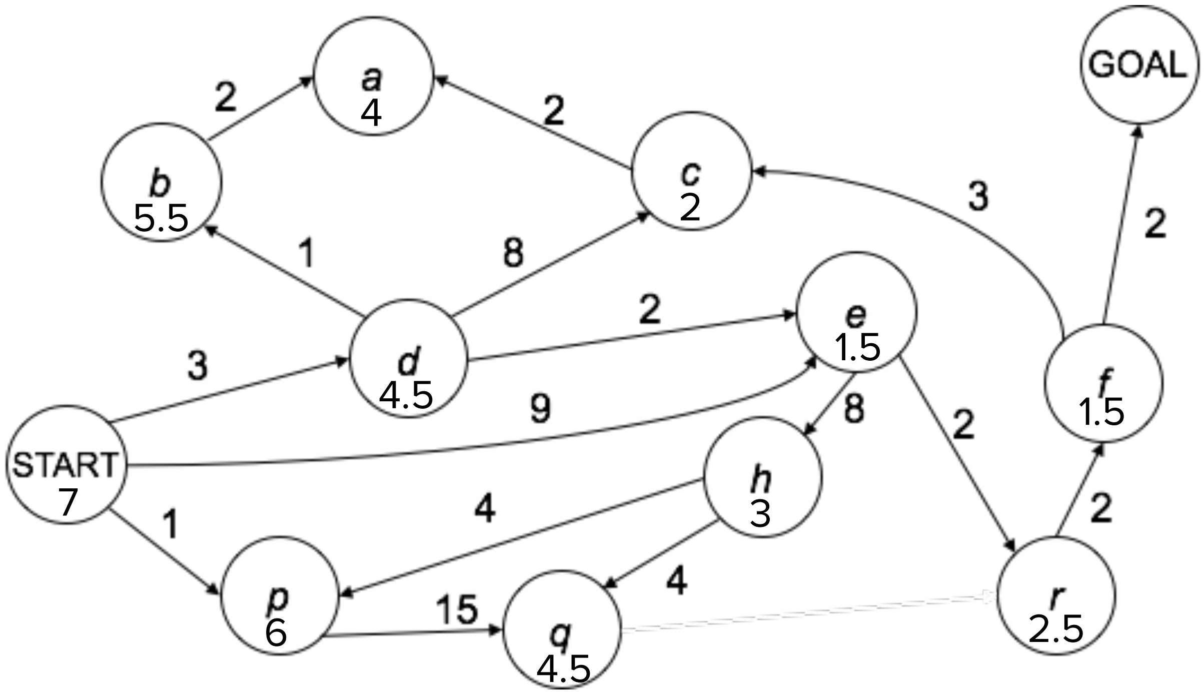


Uniform Cost Search: g(n)

Greedy Search: h(n)

A* Search: f(n) = g(n) + h(n)

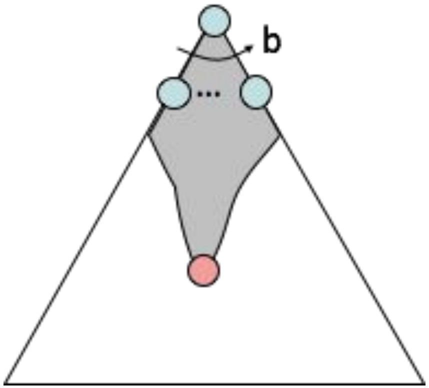
A* Search



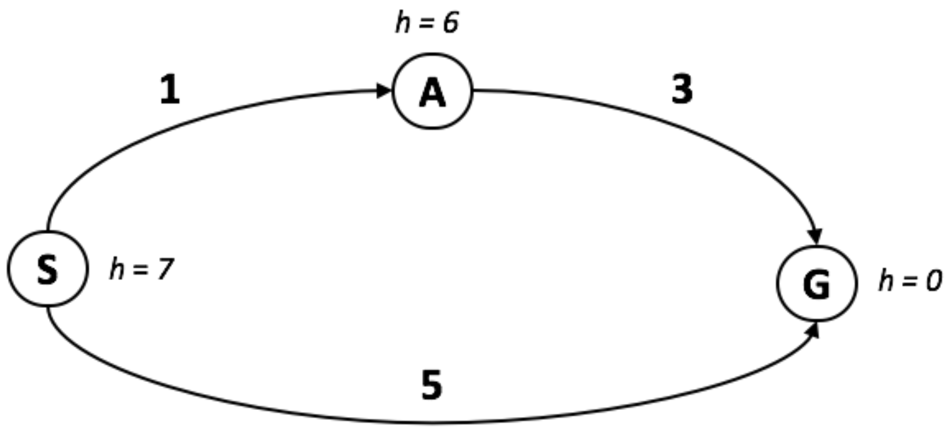
Path cost = ?

A* Search

- Complete?
  - Yes
- Optimal?
  - Yes, if the heuristic is *admissible*
  - More on this later
- Time complexity:
  - $O(b^s)$ in the worst case
- Space complexity:
  - $O(b^s)$ in the worst case



Optimality of A*



actual goal cost (5) < heuristic good goal cost (6)

Heuristics should be *less than* the actual costs!

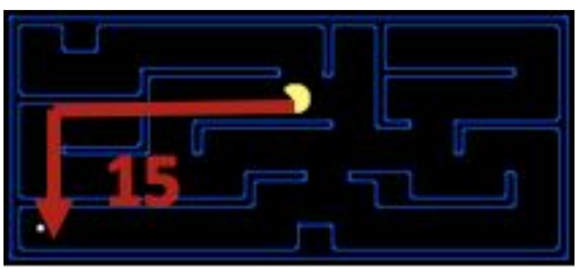
Optimality of A*: Admissibility

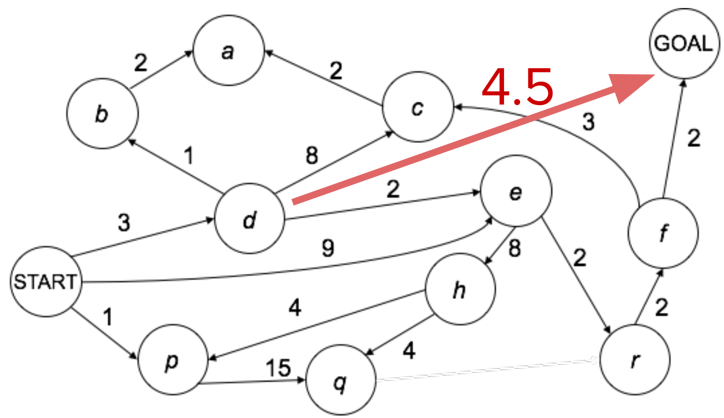
- A heuristic *h* is admissible if:

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

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

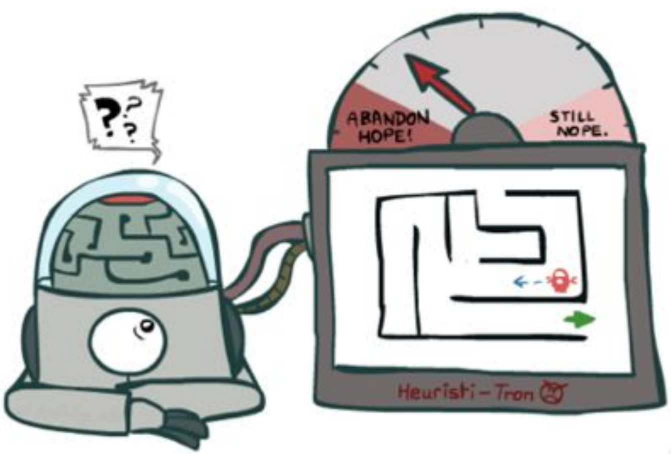
- Examples:



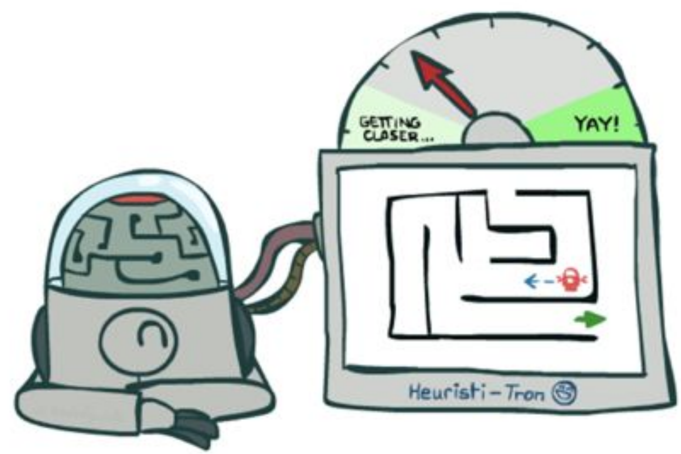


Optimality of A*: Admissibility

Inadmissible (pessimistic) heuristics break A* optimality by trapping good paths in the frontier



Admissible (optimistic) heuristics slow down bad paths, but never outweigh the true costs



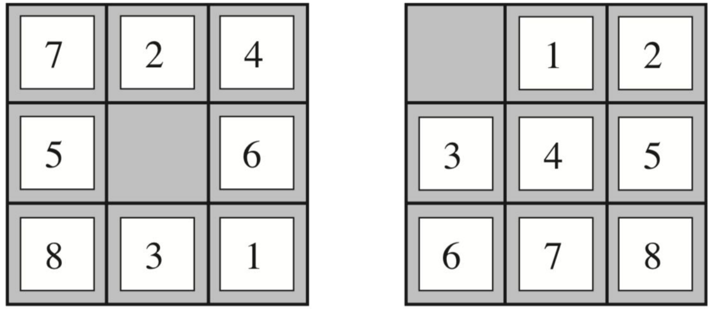
Optimality of A*: Admissibility

Inadmissible (pessimistic) heuristics break A* optimality by trapping good paths in the frontier

Admissible (optimistic) heuristics slow down bad paths, but never outweigh the true costs

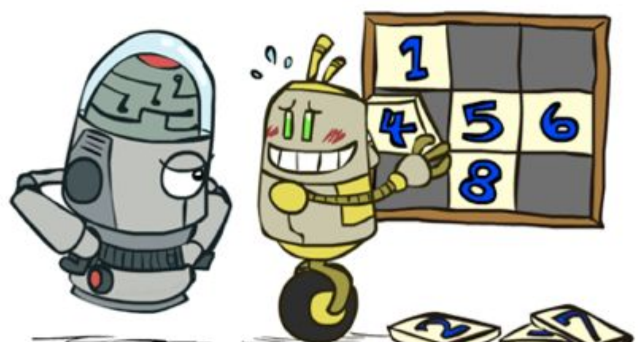
Often, admissible heuristics are solutions to *relaxed problems*, where new actions are available

Example: 8 Puzzle

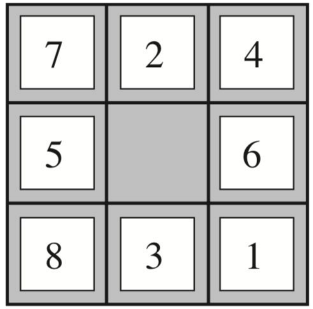


Start State Goal State

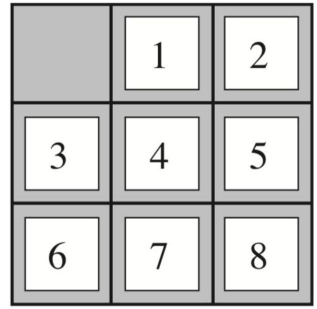
- Heuristic: Number of tiles misplaced (h1)
- h(start state) = ?
- Why is it admissible?



Example: 8 Puzzle

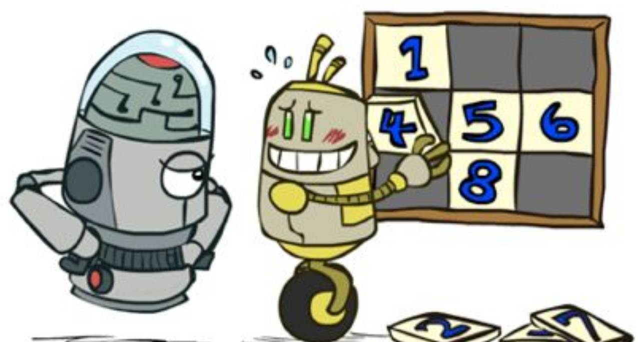


Start State

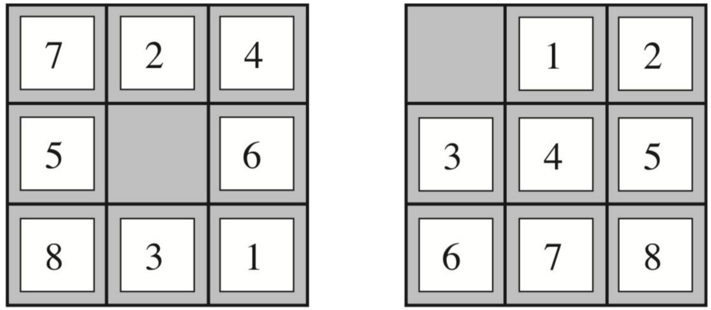


Goal State

- Heuristic: Number of tiles misplaced (h1)
- h(start state) = 8
- Why is it admissible?
  - Any tile that is out-of-place must be moved at least once



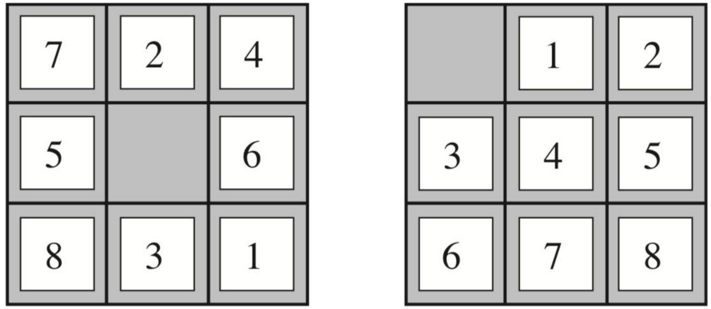
Example: 8 Puzzle



Start State   Goal State

- Heuristic: distance to their goal (h2)
  (i.e. as if we can move thru tiles)
- h(start state) = ?
- Why is it admissible?

Example: 8 Puzzle



Start State Goal State

- Heuristic: distance to their goal (h2)
  (i.e. as if we can move thru tiles)
- h(start state) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18
- Why is it admissible?
  - We can only move one tile one step closer to the goal

Comparison h1 vs h2

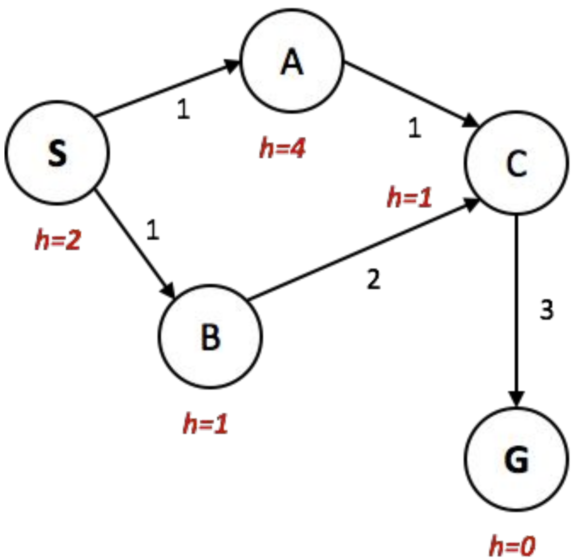
	Search Cost (nodes generated)		
d	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	3644035	227	73
14	–	539	113
16	–	1301	211
18	–	3056	363
20	–	7276	676
22	–	18094	1219
24	–	39135	1641

- As heuristics get closer to the true cost, we will expand fewer nodes
- But we usually do more work per node to compute the heuristic

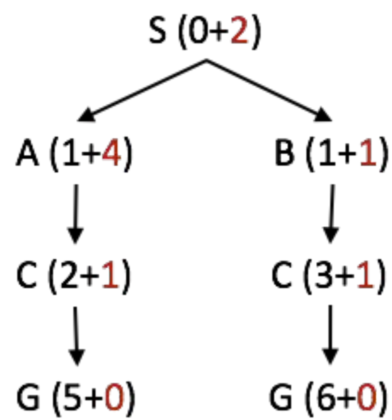
A* Graph Search

- Heuristics of A* Graph Search require a stricter property than *admissible*
- The heuristics have to be <u>*consistent*</u>

State space graph



Search tree



A* Graph Search

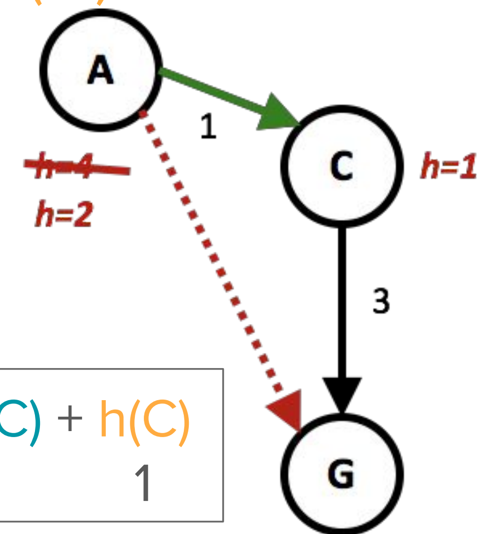
Admissibility (optimistic):

$$h(n) \leq \text{actual cost from n to Goal}$$

Consistency (monotonicity):

$$h(n) \leq \text{cost from n to n'} + h(n')$$

where *n'* is a child of *n*



$$\underset{2}{h(A)} \leq \underset{1}{\text{cost(A to C)}} + \underset{1}{h(C)}$$

Summary

- Heuristic functions, *h(n)*, inform an agent on the proximity of the goal
  - We often use a relaxed problems to design heuristics
- Greedy Best-First Search uses only h(n)
- A* Search uses $f(n) = g(n) + h(n)$
  - Optimal if h(n) is *admissible* (for tree search)
  - *consistent* (for graph search)

Quiz!