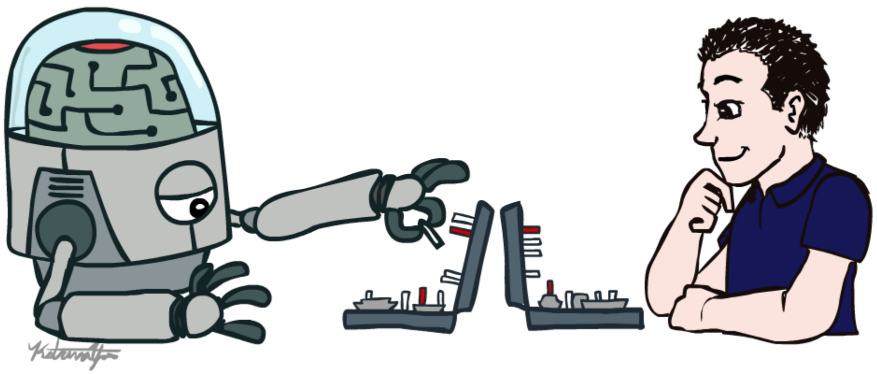
CSE 3521:
Introduction to Artificial Intelligence

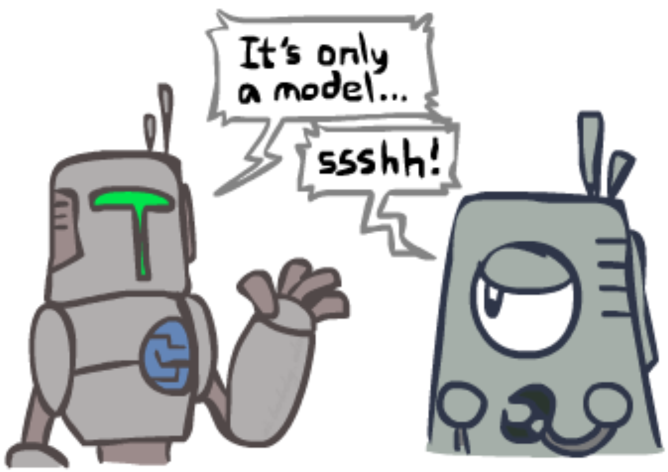


[These slides are partially adapted from the UC Berkeley. CS188 Intro to AI at UC Berkeley]

The Ohio State University

Search and Models

- Search operates over models of the world
  - The agent doesn't actually try all the plans out in the real world!
  - Planning is all "in simulation"
  - Your search is only as good as your models…

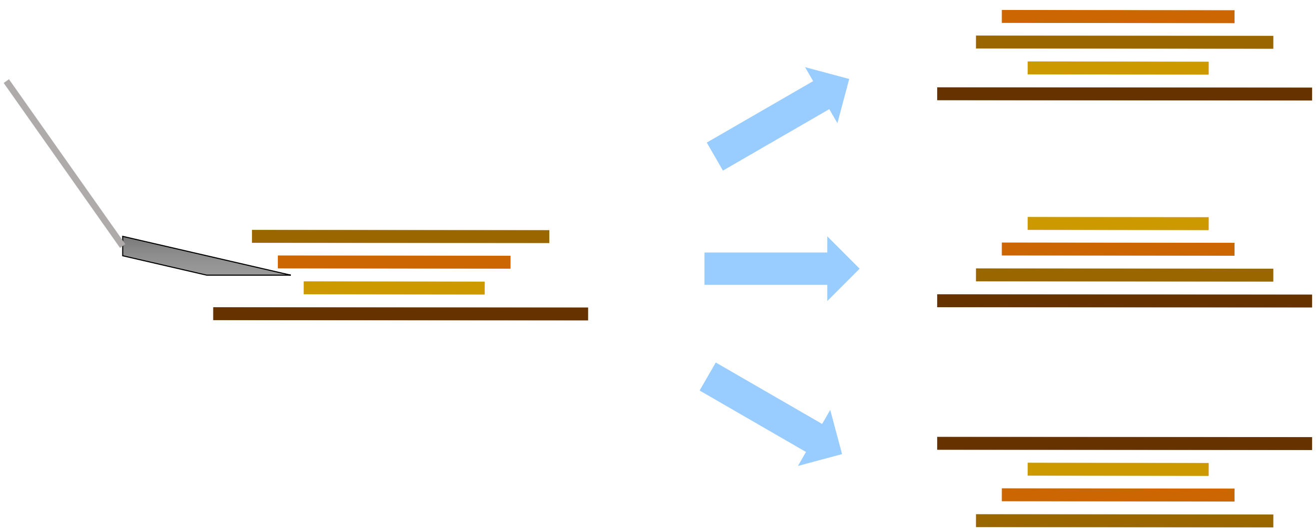


Uninformed vs. Informed Search

- <u>Uninformed</u> search
  - Given no information about problem (other than its definition)
  - Find solutions to problems by systematically generating new states and testing for goal

- <u>Informed</u> search
  - Given some ideas of where to look for solutions
  - Use problem-specific knowledge

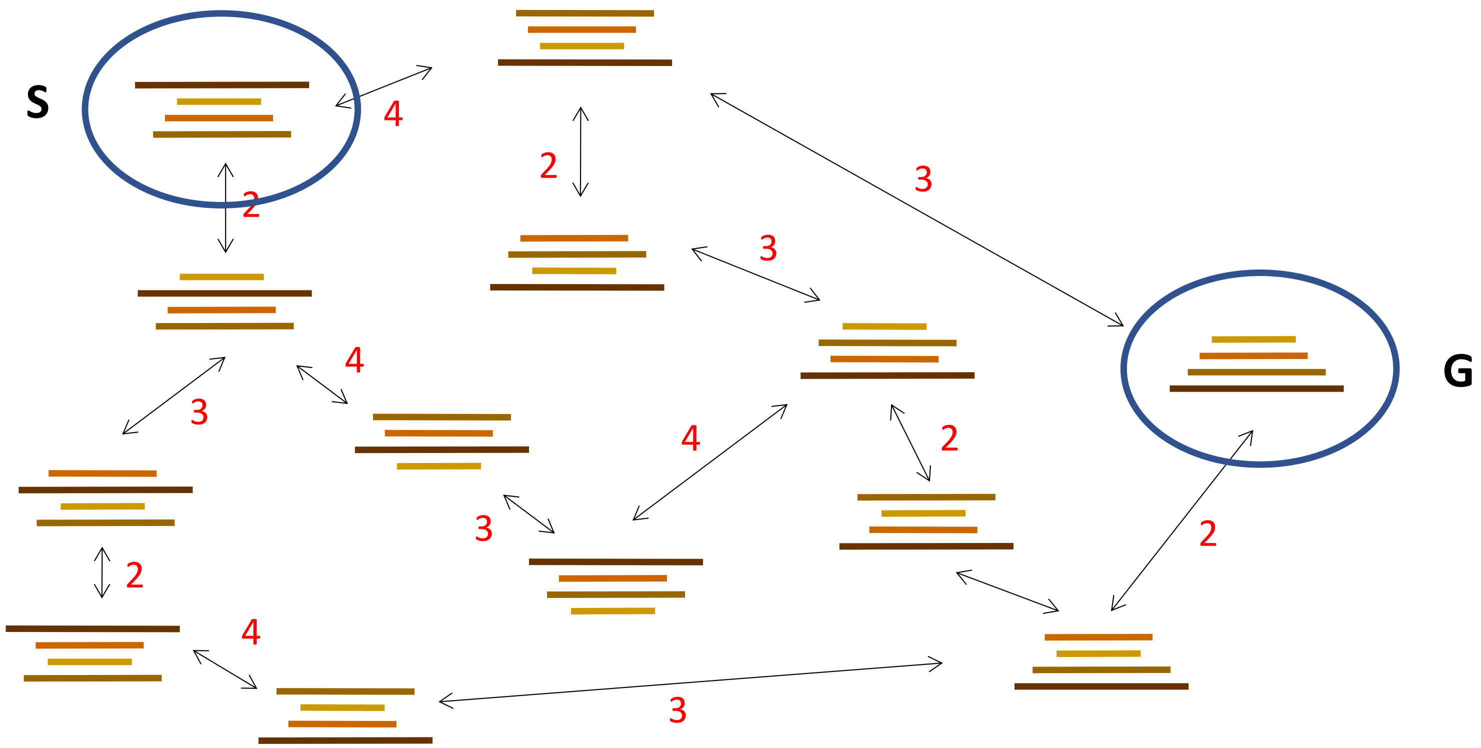
Example: Pancake Problem



Cost: Number of pancakes flipped

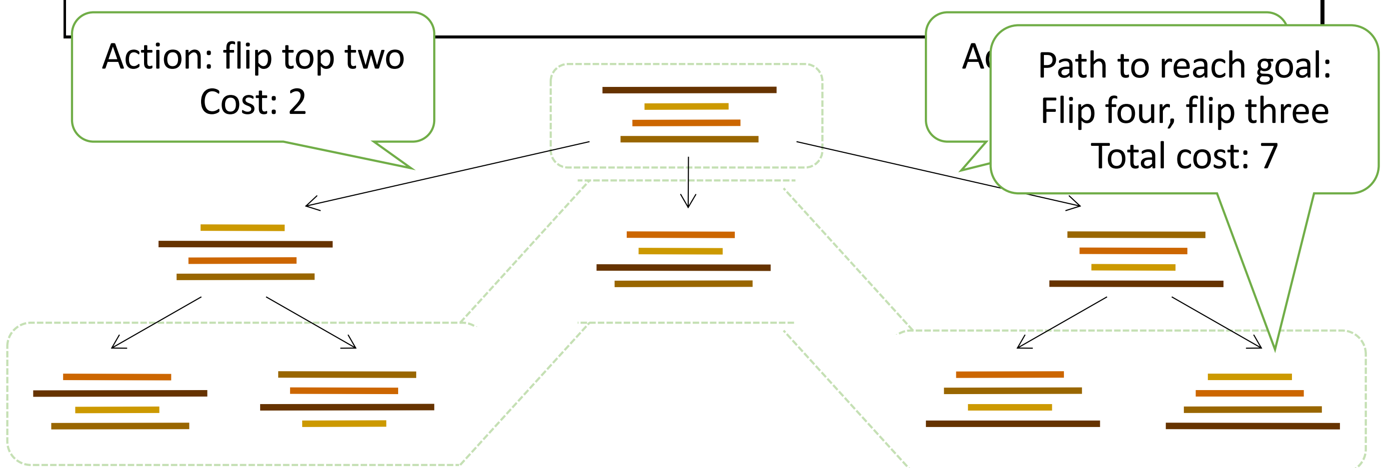
Example: Pancake Problem

State space graph with costs as weights



General Tree Search

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

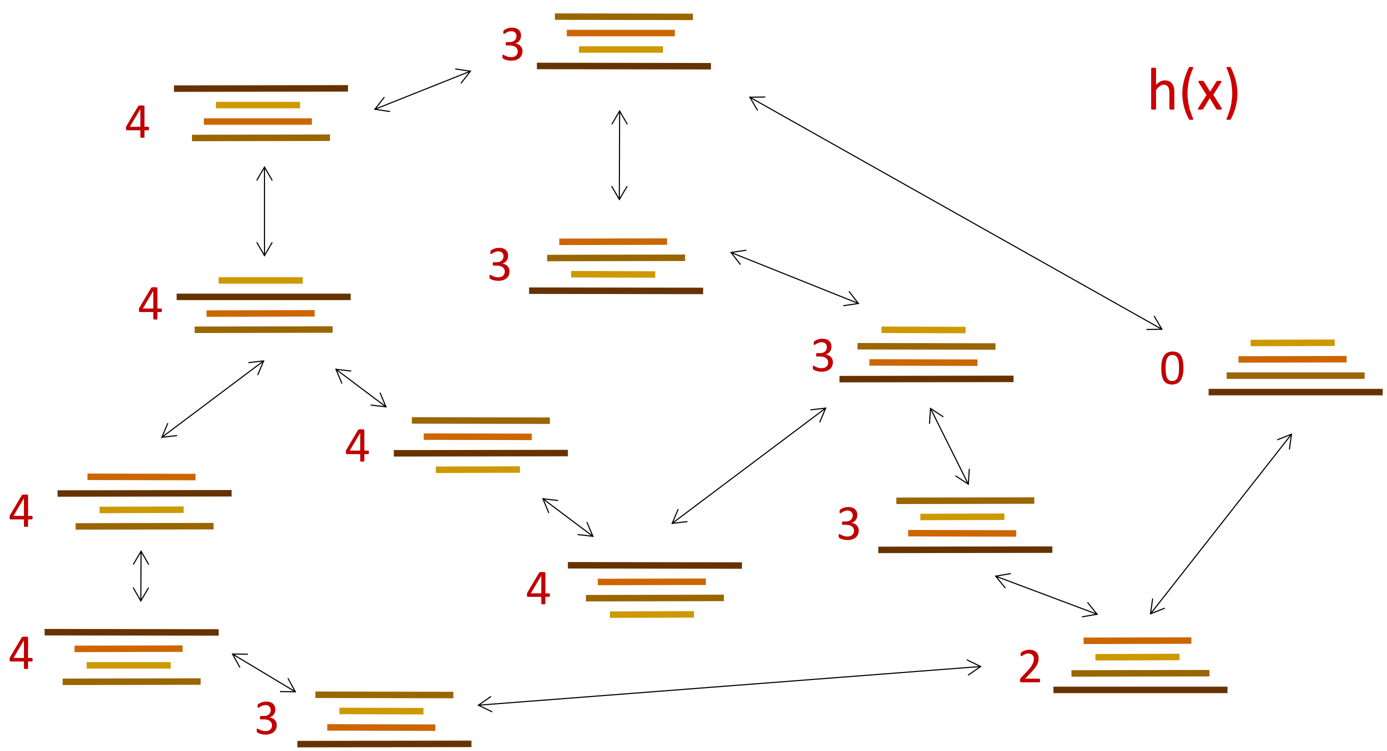


Search Heuristics

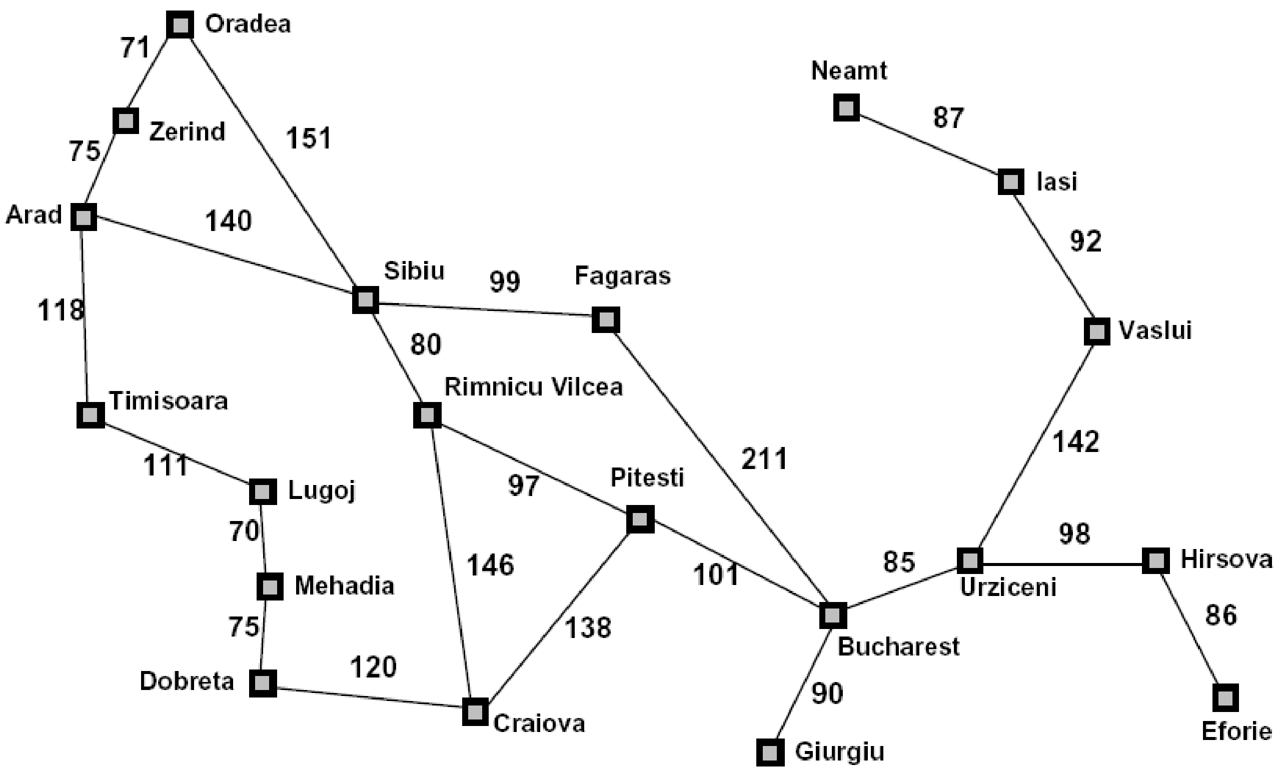
- A heuristic is
  - A function that *estimates* how close a state is to a goal
  - Designed for a particular search problem
  - Examples: Manhattan distance, Euclidean distance for pathing
    - not the exact “path” distance

Example: Heuristic Function

Heuristic: the number of the largest pancake that is still out of place



Example: Heuristic Function

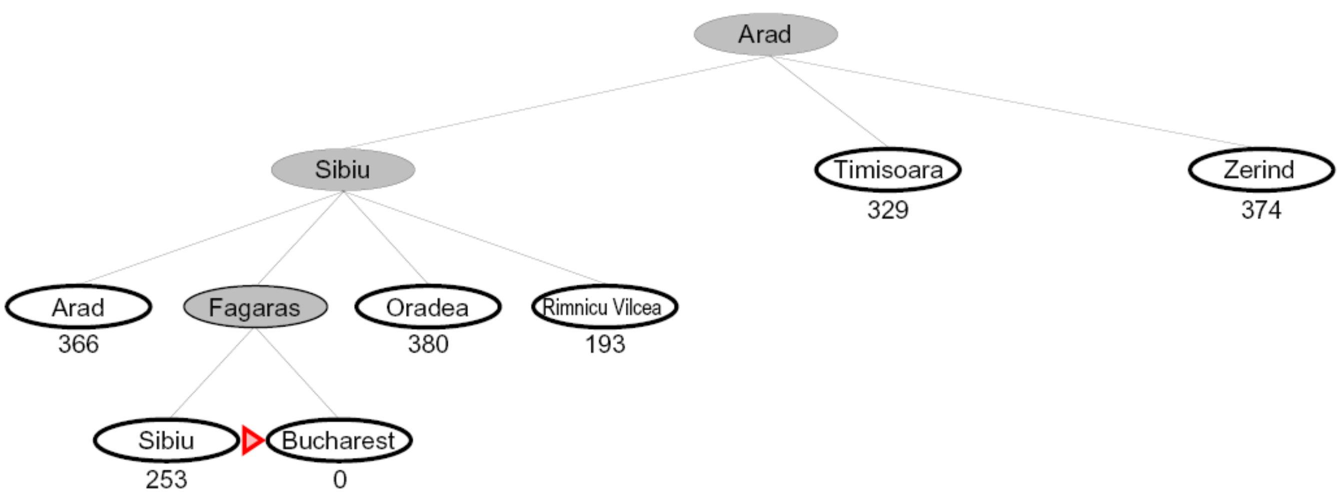


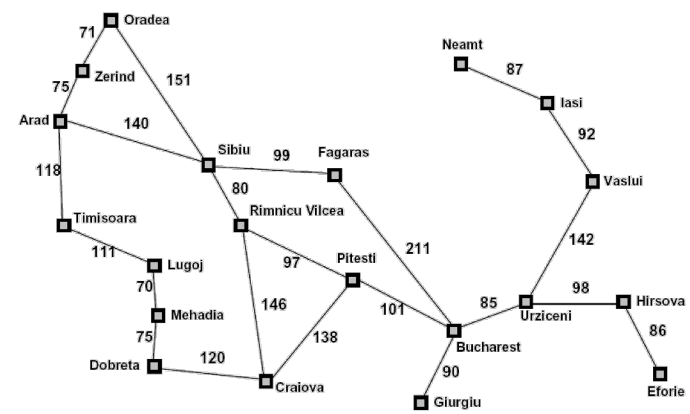
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

Greedy Search

- Expand the node that seems closest…

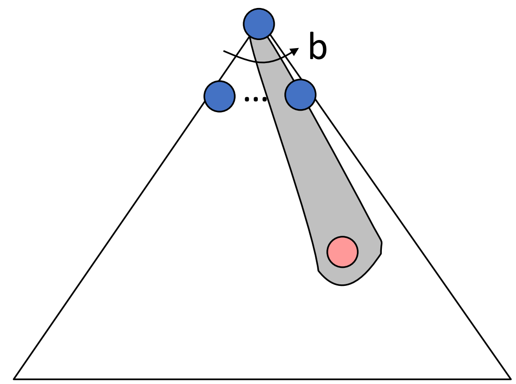




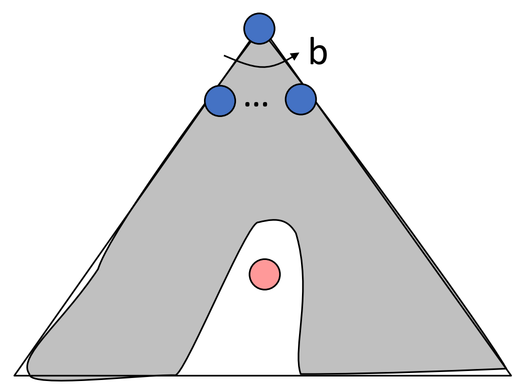
- What can go wrong?
  - Does not guarantee the optimal solution

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:
  - Best-first takes you straight to the (wrong) goal



- Worst-case: like a badly-guided DFS

A* Search

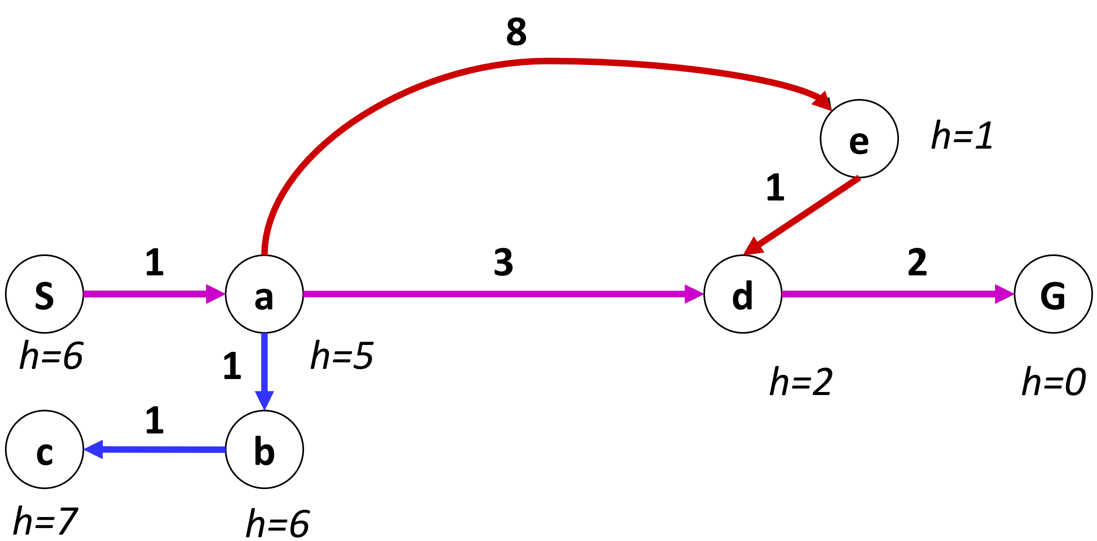
UCS

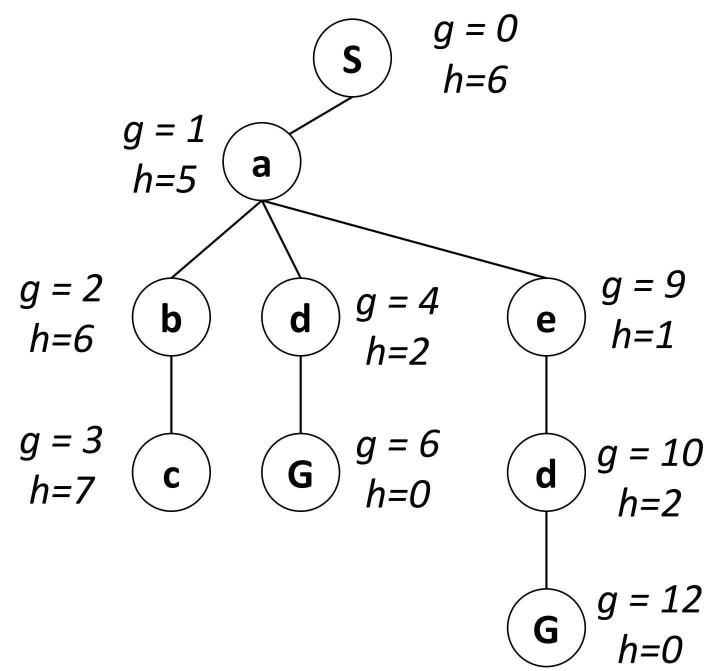
Greedy

A*

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)

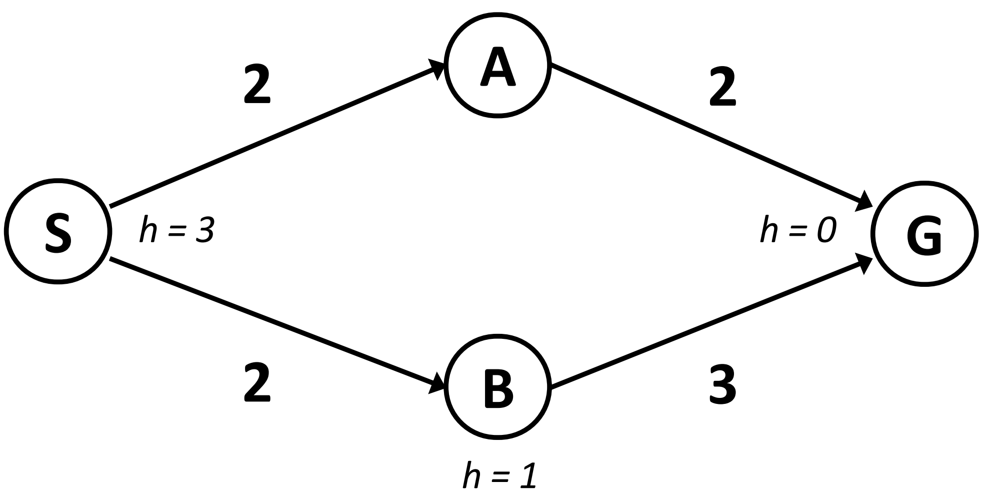




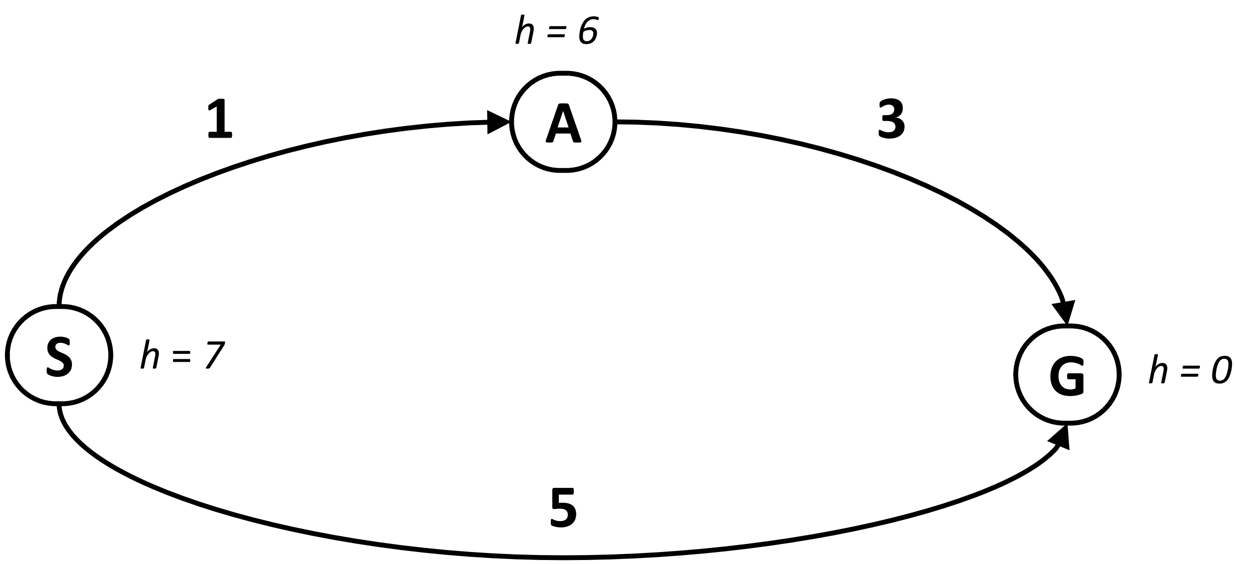
- A* Search orders by the sum: f(n) = g(n) + h(n)

When Should we terminate A*

- Should we stop when we enqueue a goal?
  - No: only stop when we dequeue a goal

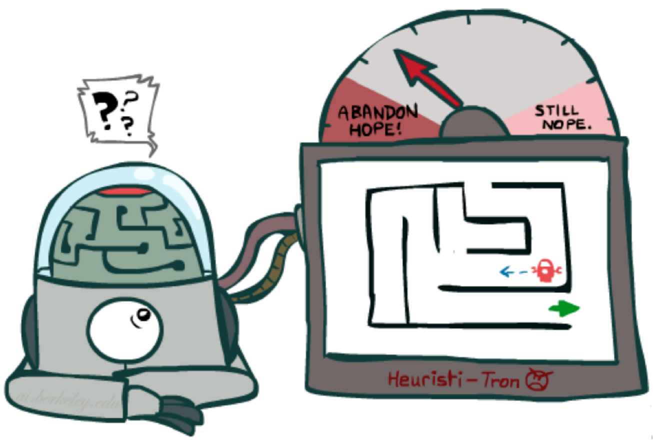


Is A* Optimal?

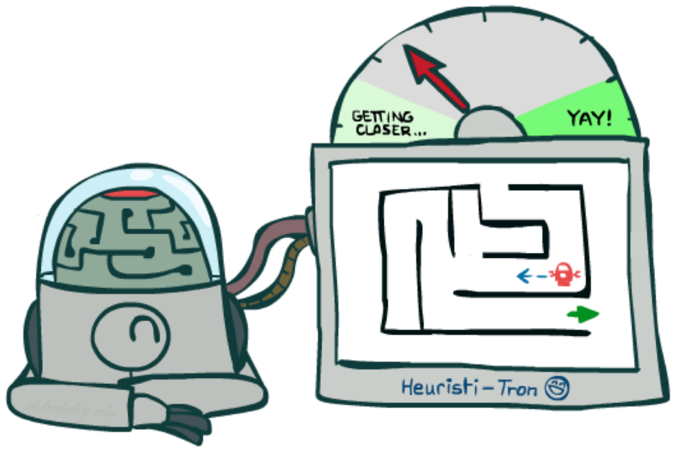


- What went wrong?
- Actual bad goal cost < estimated good goal cost
- We need estimates to be less than actual costs!

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:

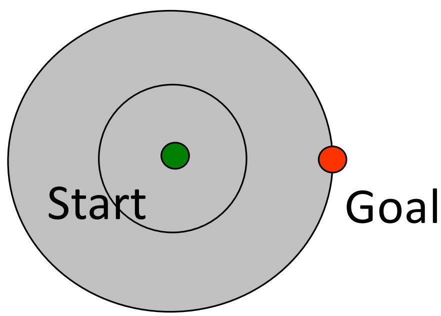
$$0 \leq h(n) \leq h^*(n)$$

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

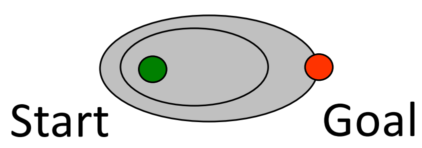
- Coming up with admissible heuristics is most of what's involved in using A* in practice.

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



A* Applications

- Video games
- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition
- …

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