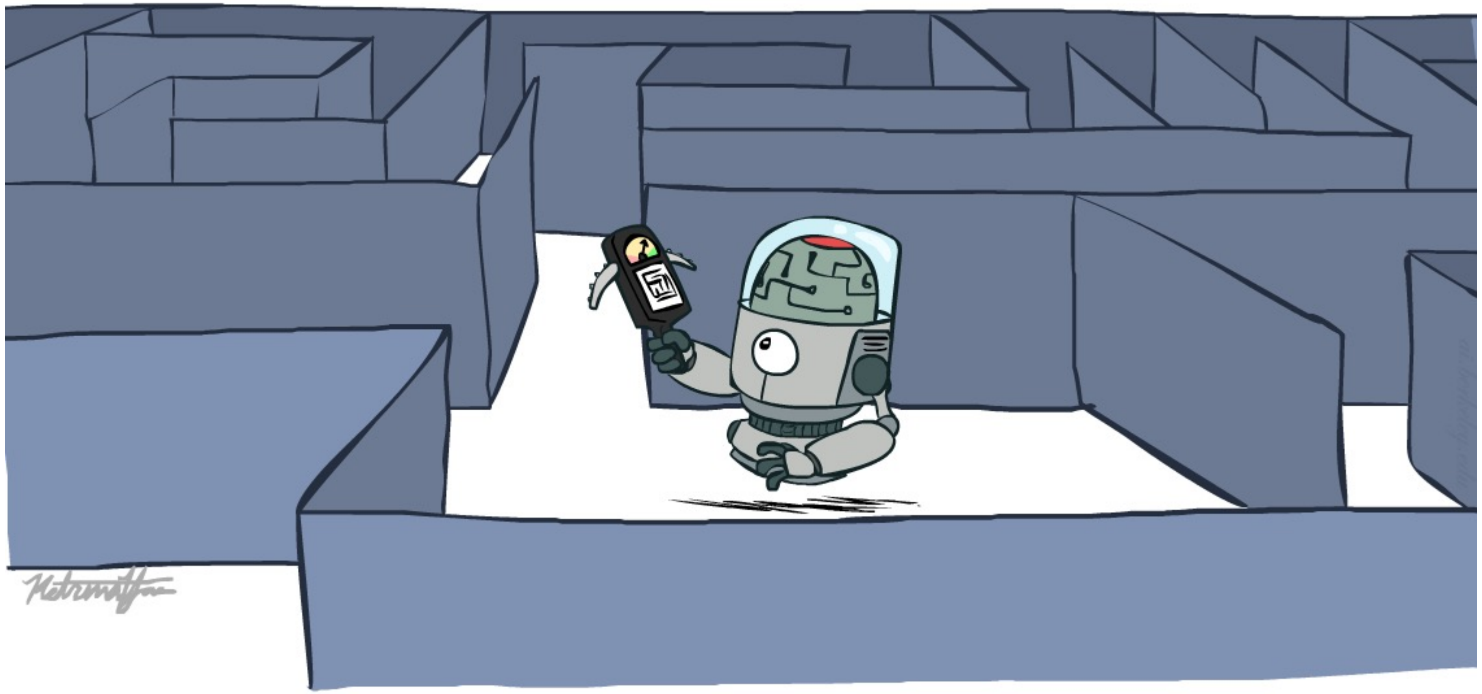
Introduction to Artificial Intelligence

Informed Search



Instructor: Tatjana Petrov

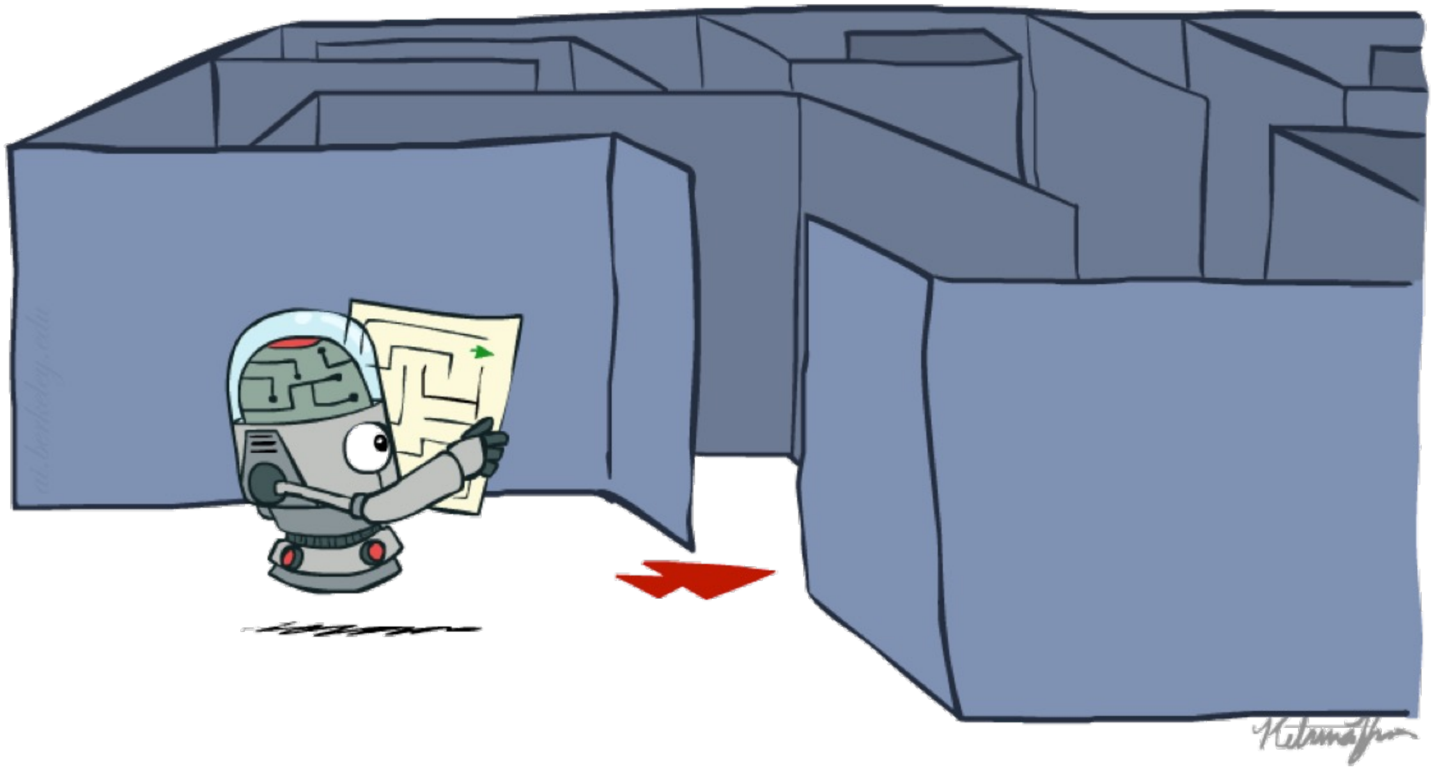
University of Trieste, Italy

[slides adapted from Dan Klein, Pieter Abbeel, Stuart Russell, et al for CS188 Intro to AI at UC Berkeley. All materials available at http://ai.berkeley.edu.]

Today

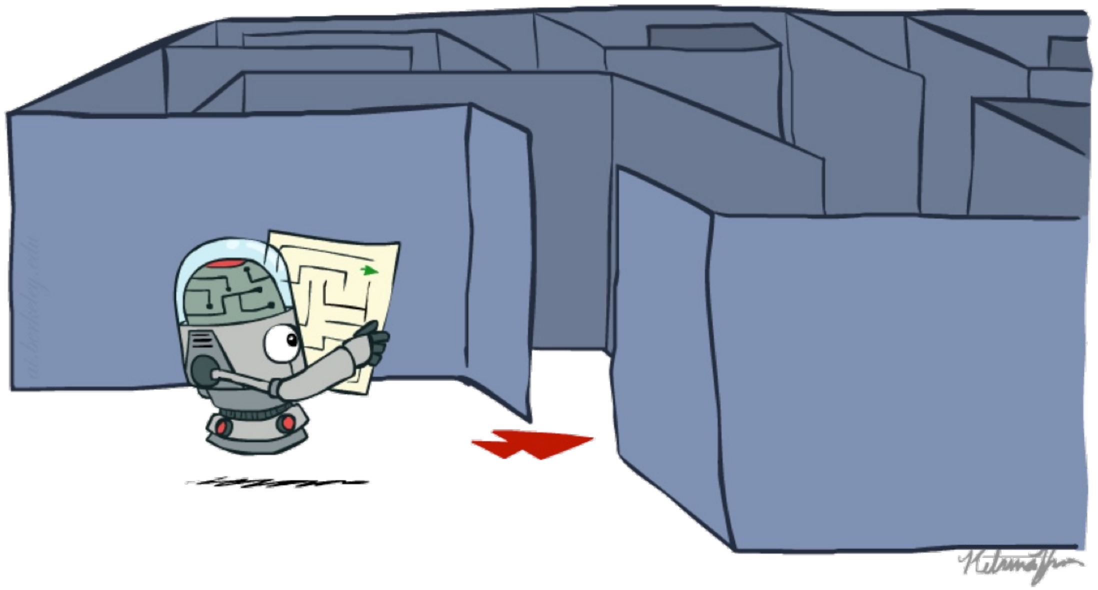
- Informed Search
  - Heuristics
  - Greedy Search
  - A* Search

Recap: Search



Recap: Search

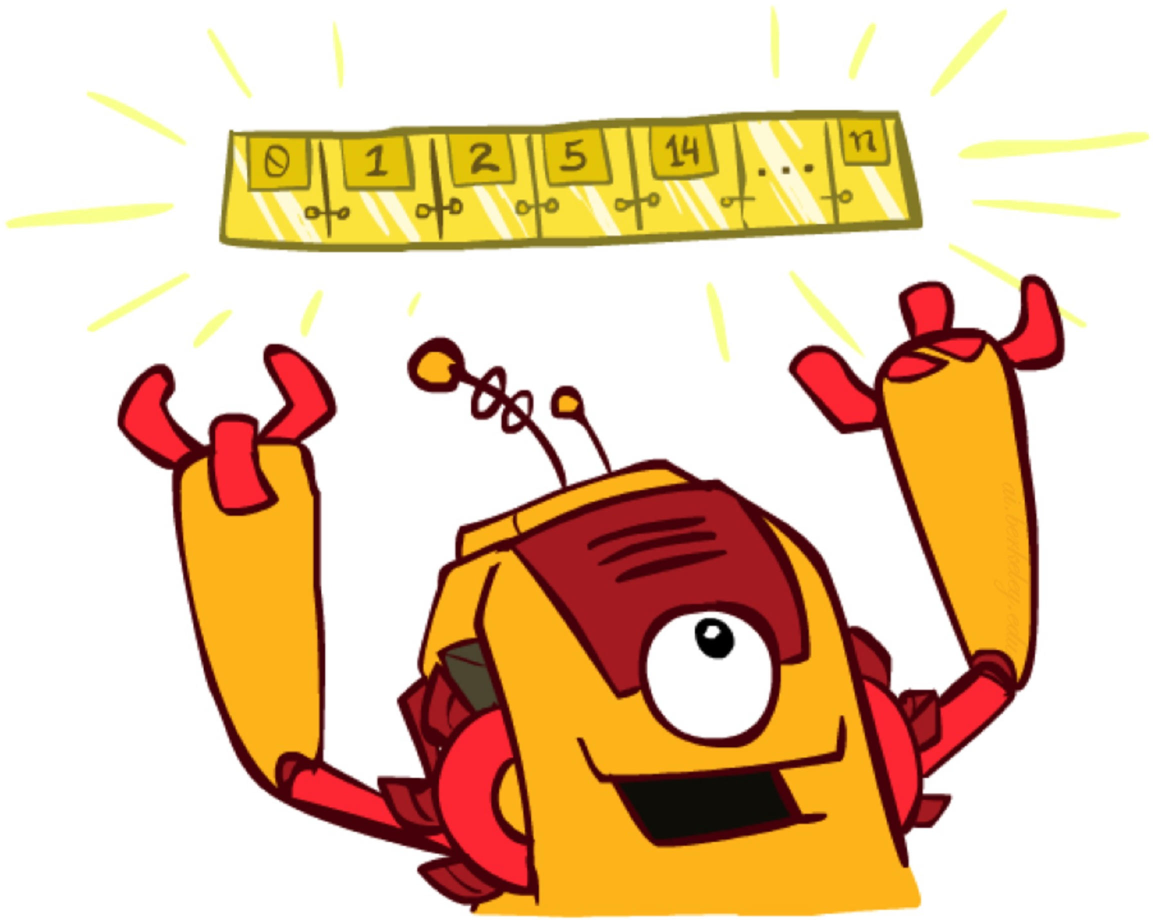
- Search problem:
  - States (configurations of the world)
  - Actions and costs
  - Successor function (world dynamics)
  - Start state and goal test



- Search tree:
  - Nodes: represent plans for reaching states
  - Plans have costs (sum of action costs)
- Search algorithm:
  - Systematically builds a search tree
  - Chooses an ordering of the fringe (unexplored nodes)
  - Optimal: finds least-cost plans

The One Queue

- All these search algorithms are the same except for fringe strategies
  - Conceptually, all fringes are priority queues (i.e. collections of nodes with attached priorities)
  - Practically, for DFS and BFS, you can avoid the log(n) overhead from an actual priority queue, by using stacks and queues
  - Can even code one implementation that takes a variable queuing object



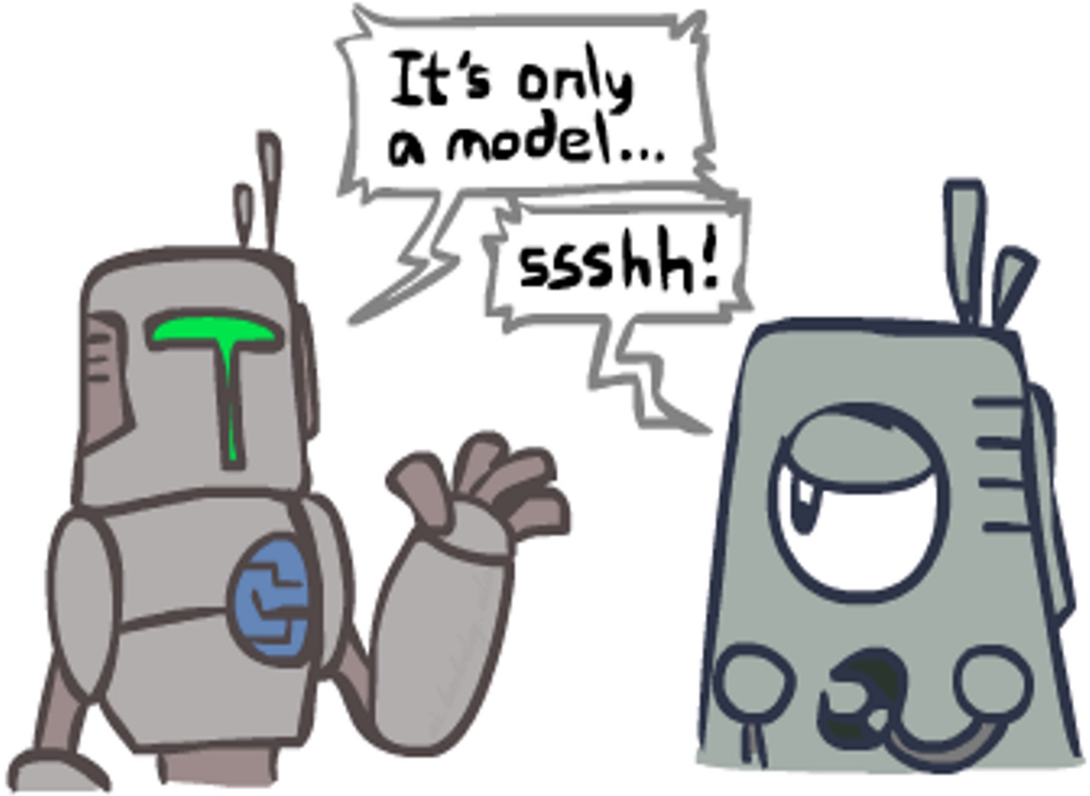
Search and Models

Search operates over models of the world

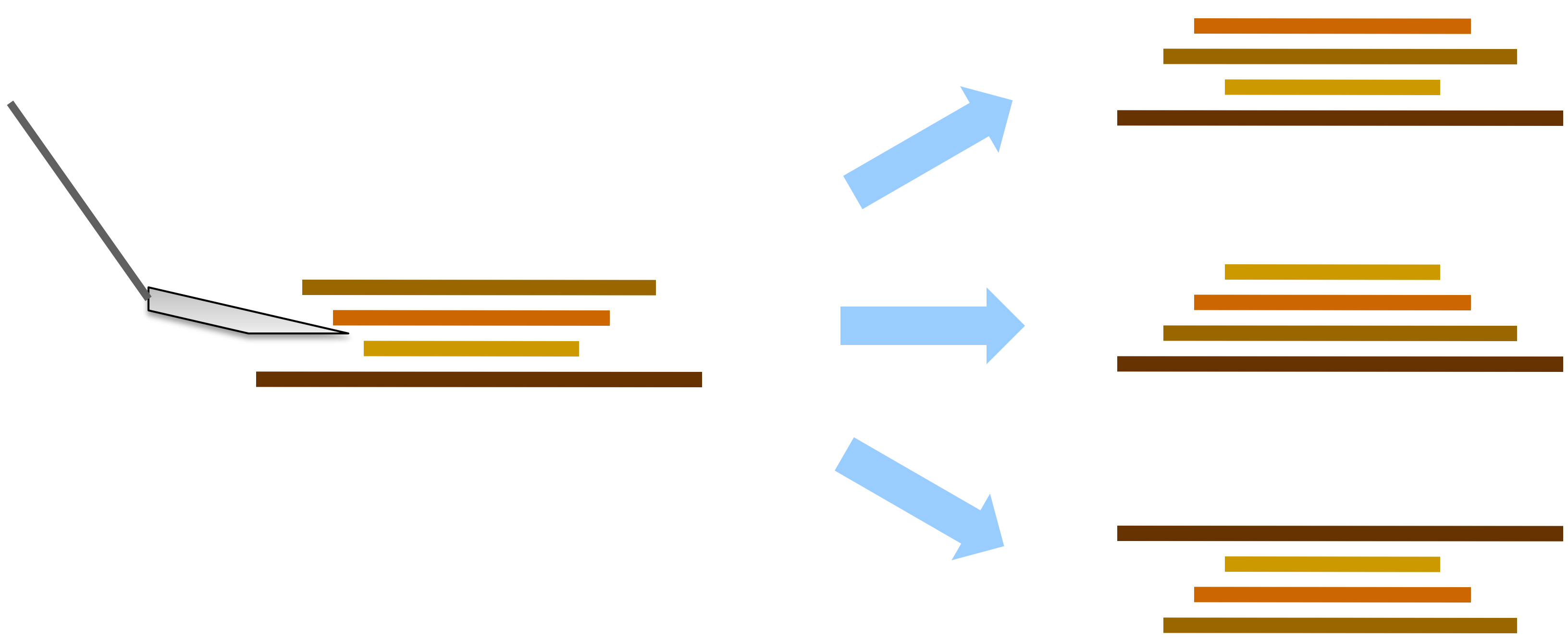
The agent doesn't actually try all the plans out in the real world!

Planning is all "in simulation"

Your search is only as good as your models…



Example: Pancake Problem



Cost: Number of pancakes flipped

Example: Pancake Problem

BOUNDS FOR SORTING BY PREFIX REVERSAL

William H. GATES

Microsoft, Albuquerque, New Mexico

Christos H. PAPADIMITRIOU*†

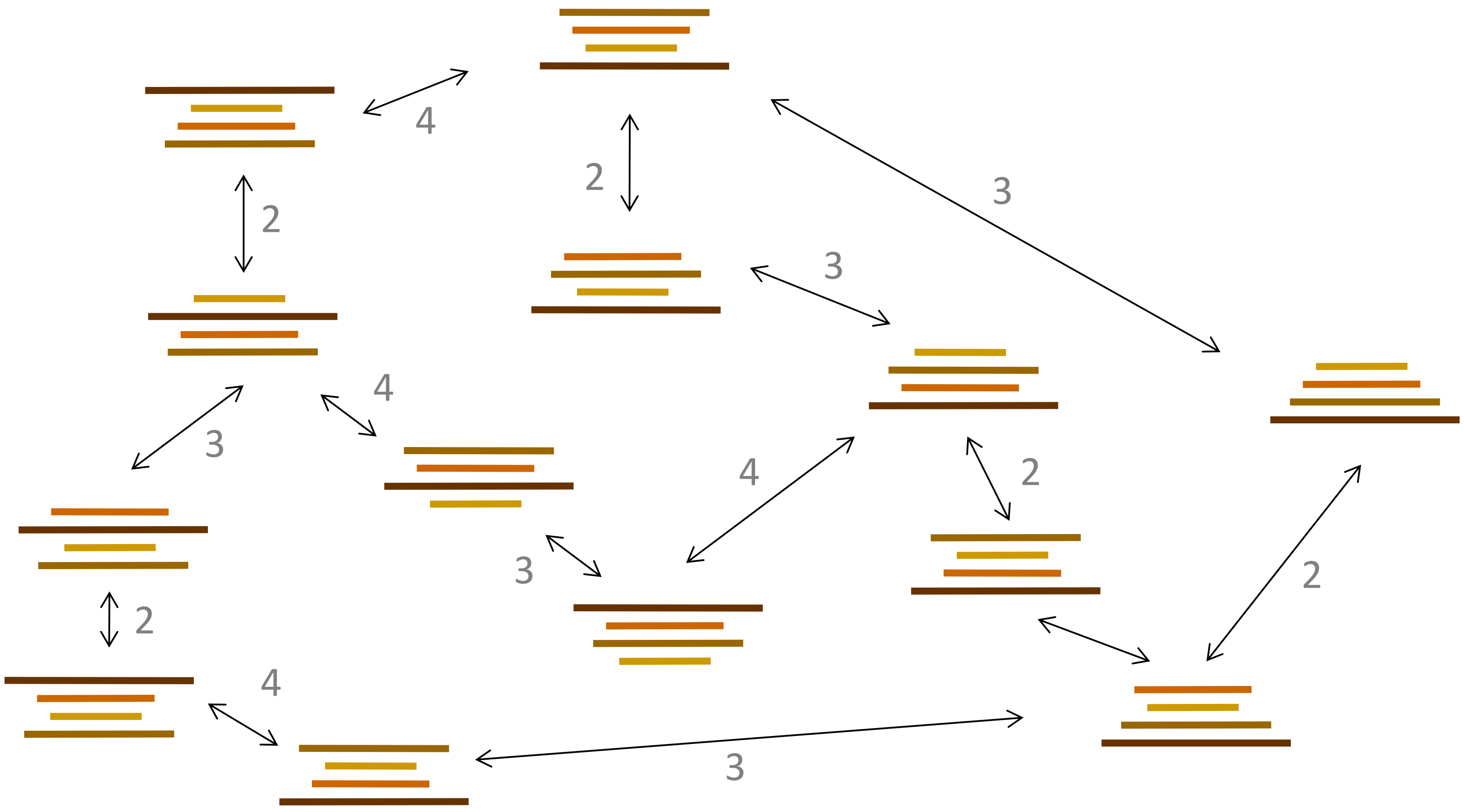
Department of Electrical Engineering, University of California, Berkeley, CA 94720, U.S.A.

Received 18 January 1978
Revised 28 August 1978

For a permutation σ of the integers from 1 to n, let $f(\sigma)$ be the smallest number of prefix reversals that will transform σ to the identity permutation, and let $f(n)$ be the largest such $f(\sigma)$ for all σ in (the symmetric group) S_n. We show that $f(n) \leqslant (5n+5)/3$, and that $f(n) \geqslant 17n/16$ for n a multiple of 16. If, furthermore, each integer is required to participate in an even number of reversed prefixes, the corresponding function $g(n)$ is shown to obey $3n/2 - 1 \leqslant g(n) \leqslant 2n+3$.

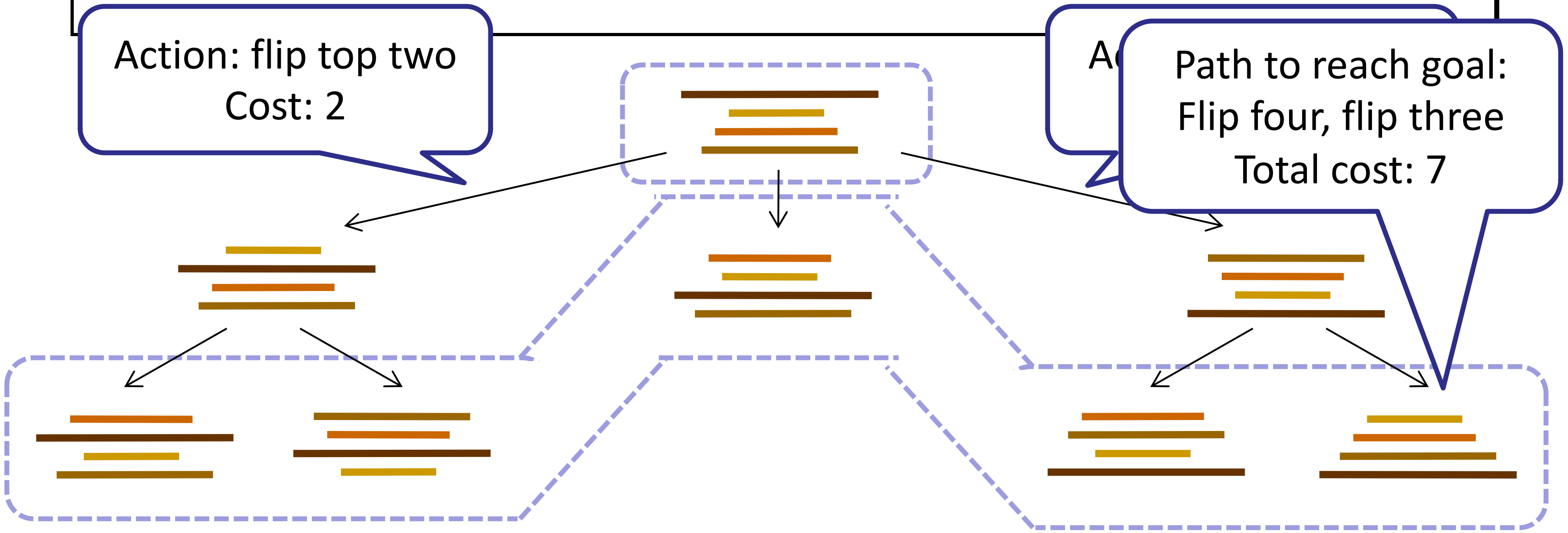
Example: Pancake Problem

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
```



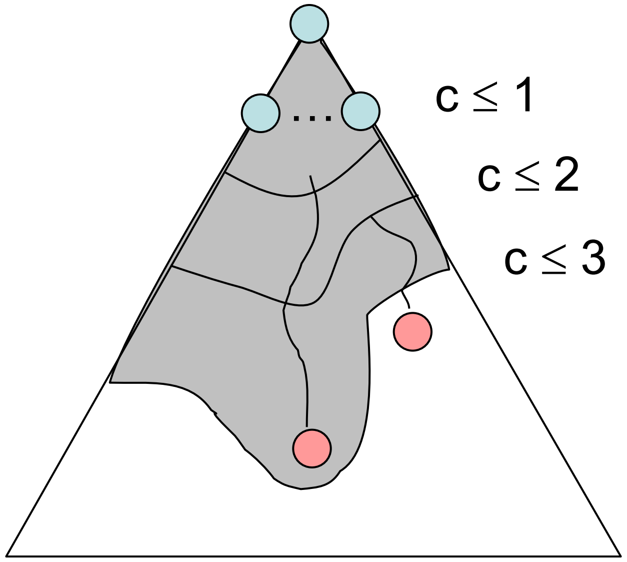
Uninformed Search

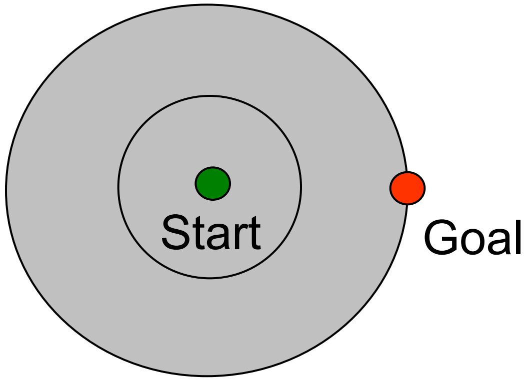
Uniform Cost Search

- Strategy: expand lowest path cost

- The good: UCS is complete and optimal!

- The bad:
  - Explores options in every “direction”
  - No information about goal location

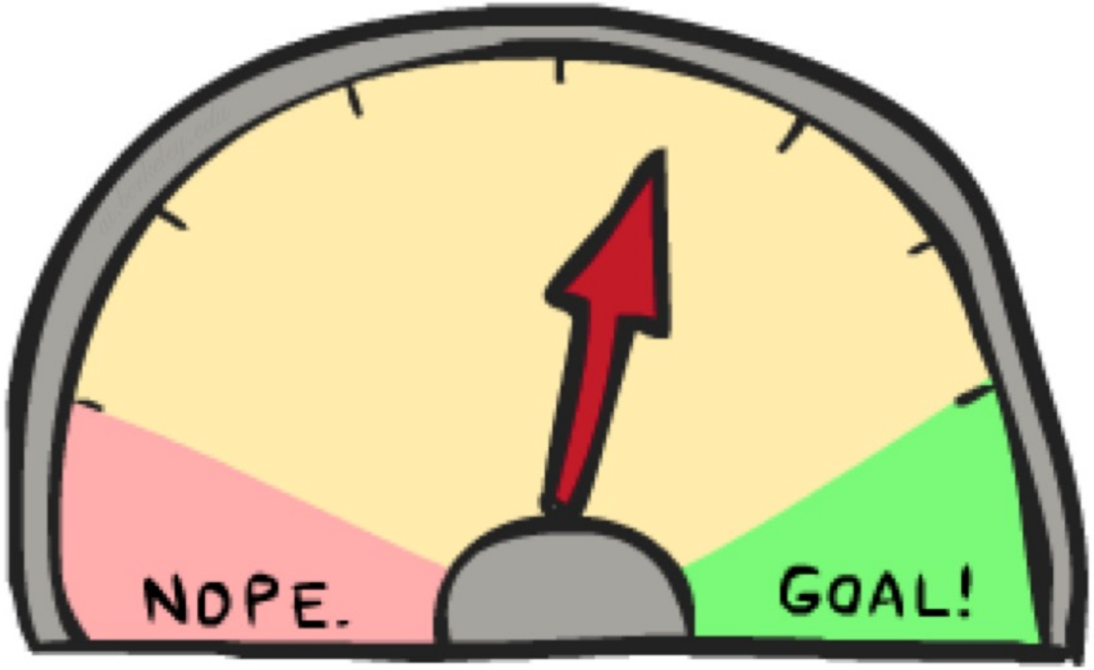




Video of Demo Contours UCS Empty

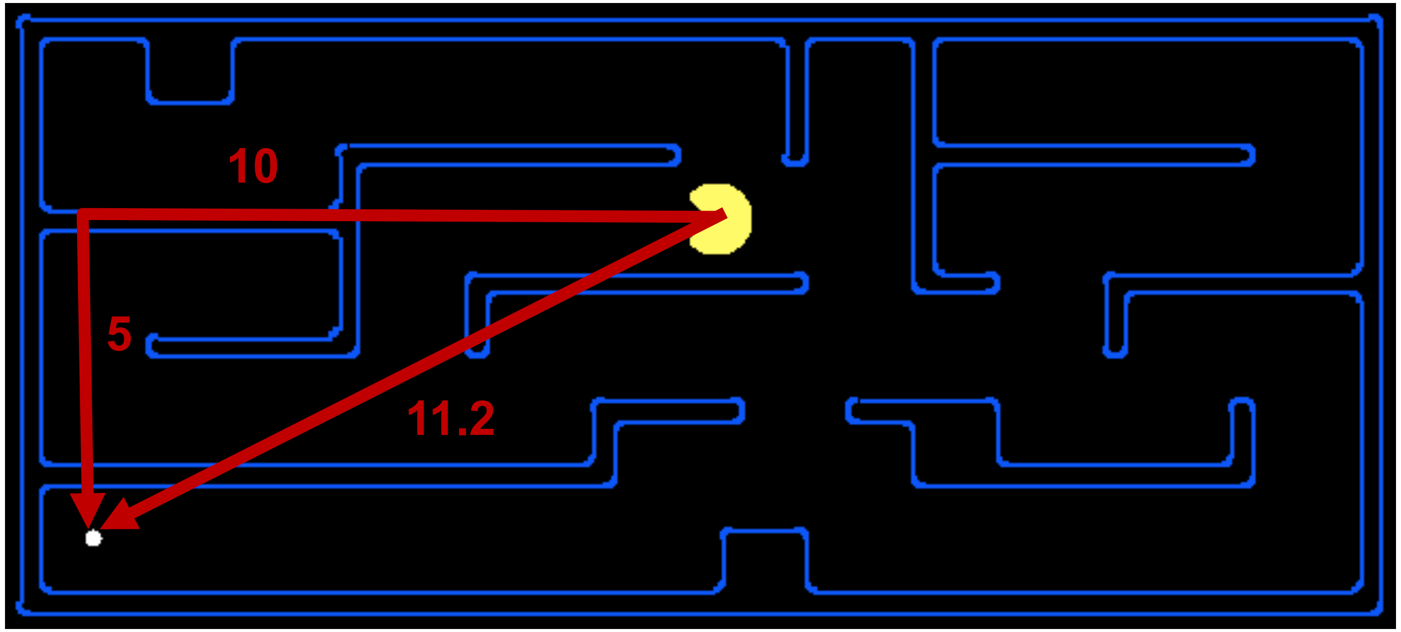
Video of Demo Contours UCS Pacman Small Maze

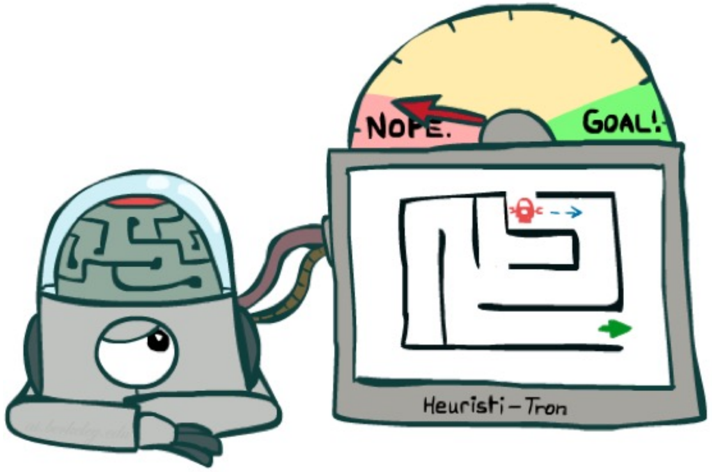
Informed Search

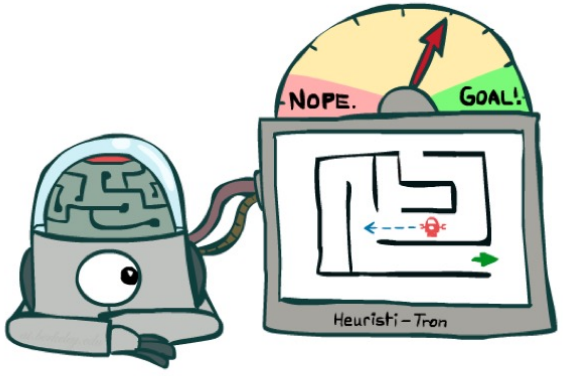


Search Heuristics

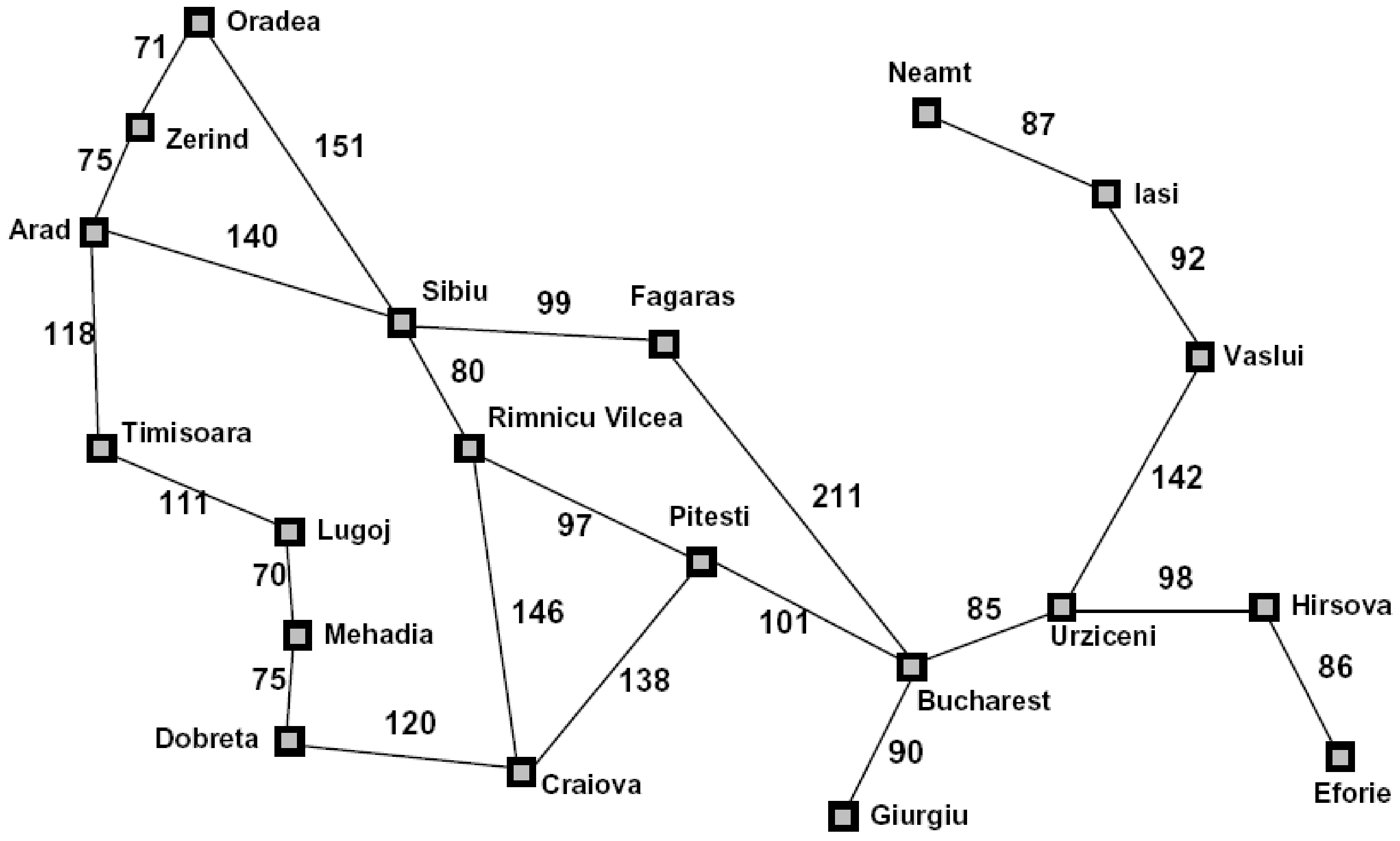
- An heuristic function $h(n)$:
  - *Estimates* how close a state n is to a goal
  - Designed for a particular search problem
  - Examples: Manhattan distance, Euclidean distance for pathing







Example: Heuristic Function

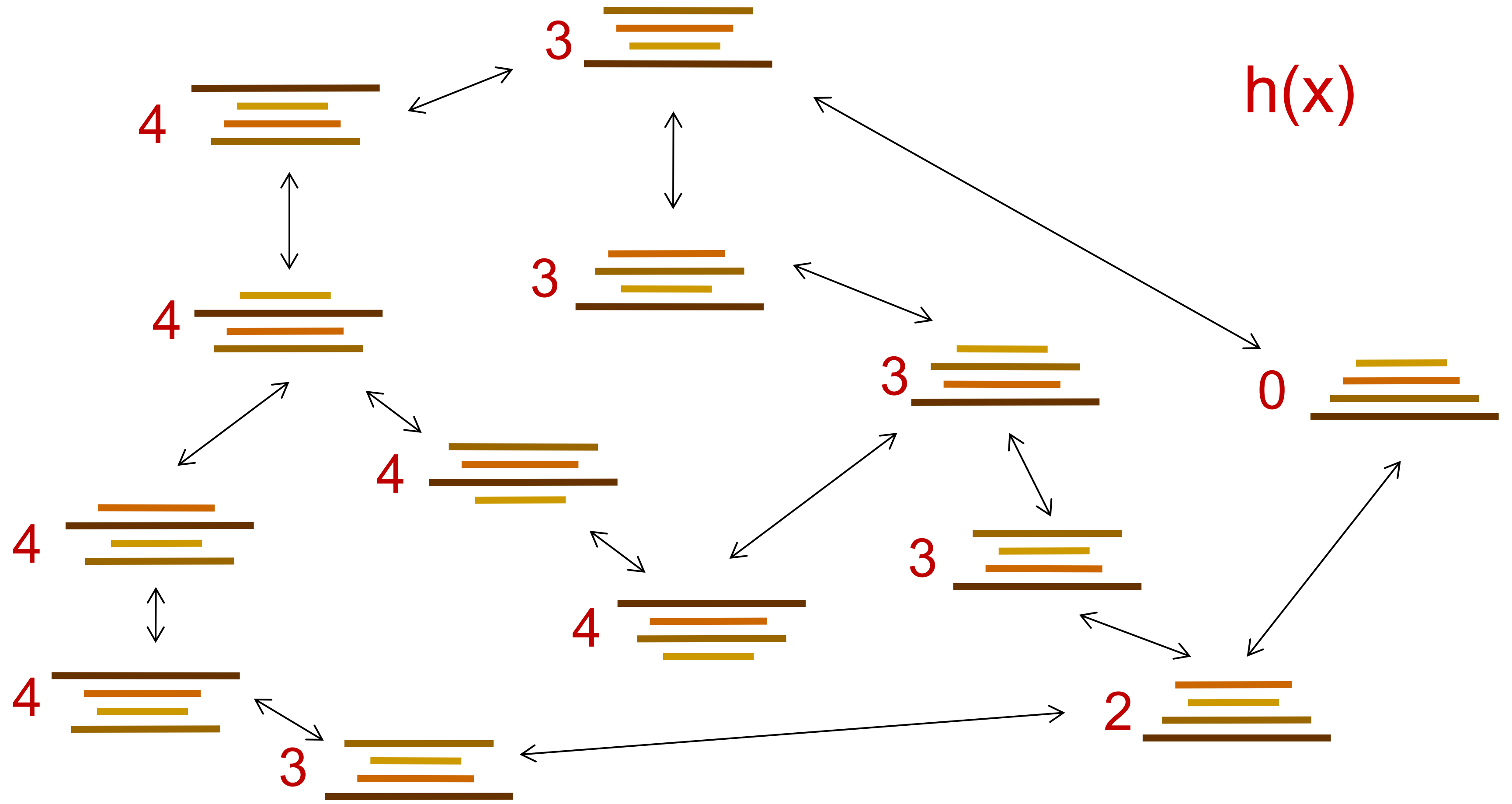


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

Example: Heuristic Function

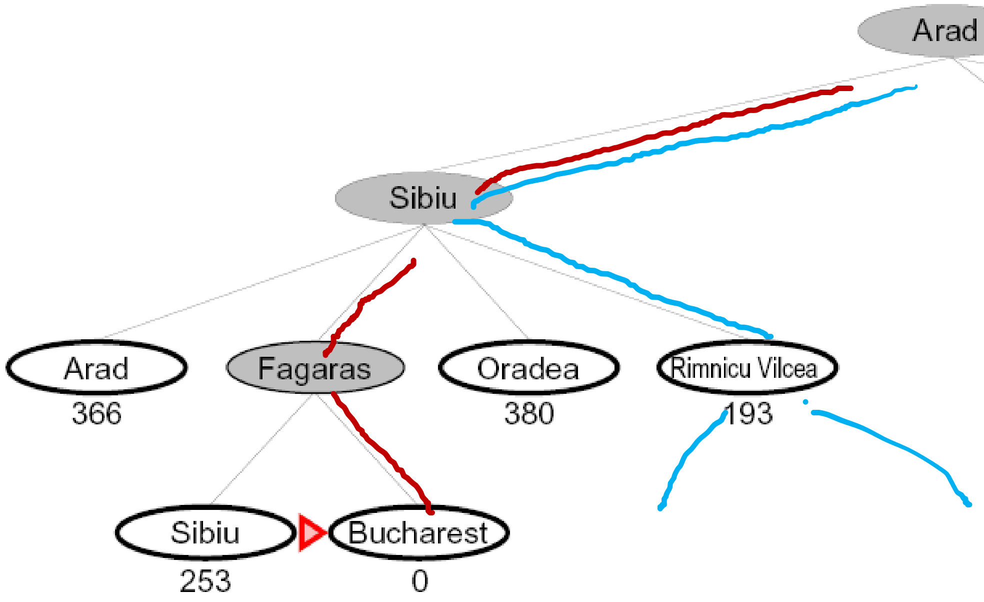
Heuristic: the number of the largest pancake that is still out of place

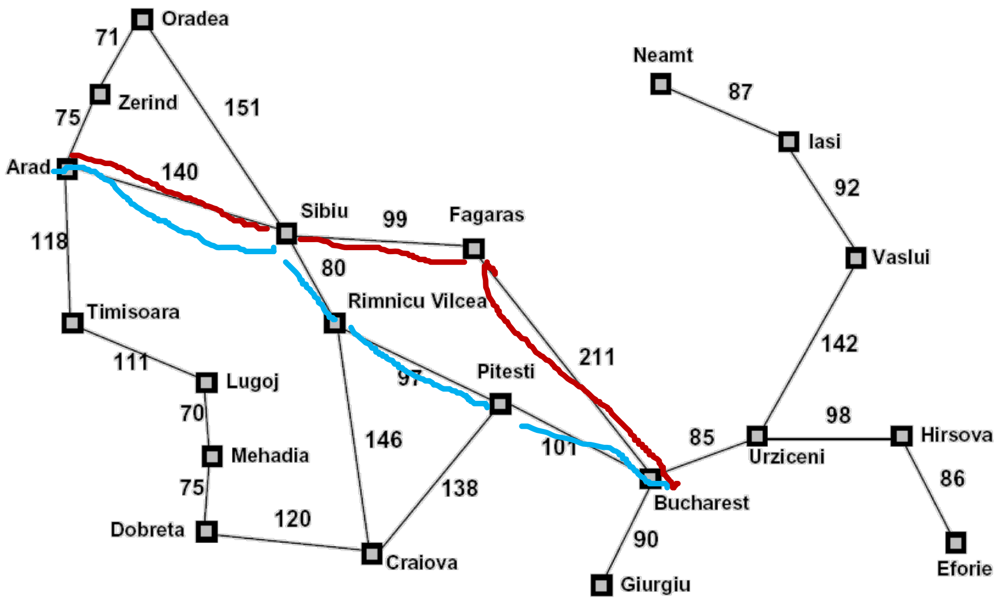


Greedy Search

Greedy Search

- Expand the node that seems closest…
  i.e. evaluation function $f(n) = h(n)$

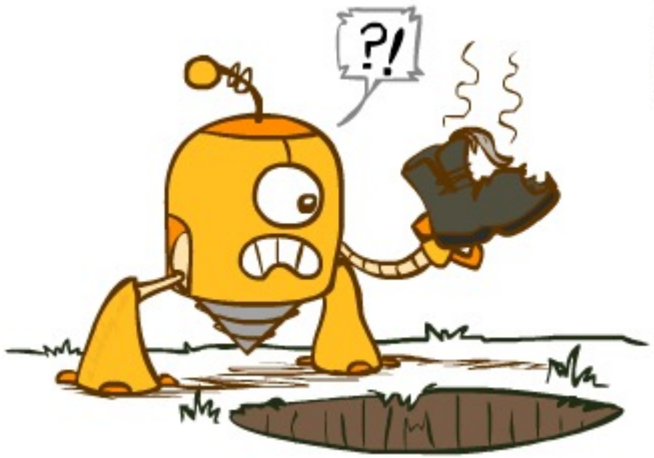




Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

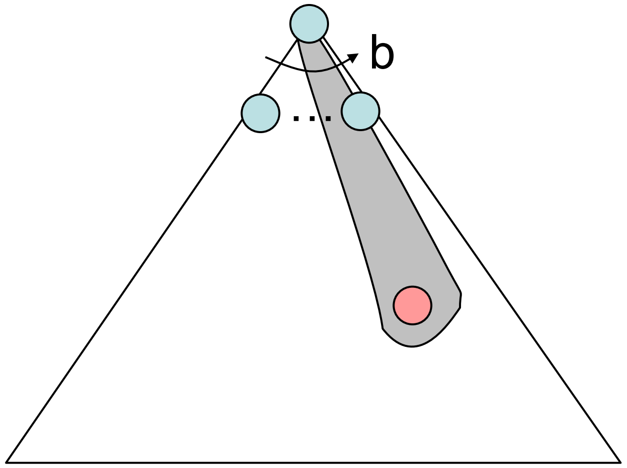
Straight-line distances

- What can go wrong?



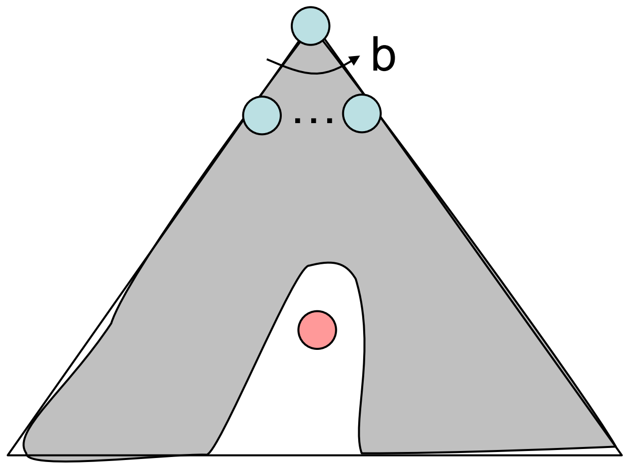
Greedy Search

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

- Worst-case: like a badly-guided DFS

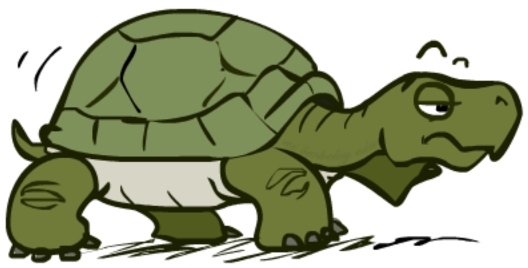


Video of Demo Contours Greedy (Empty)

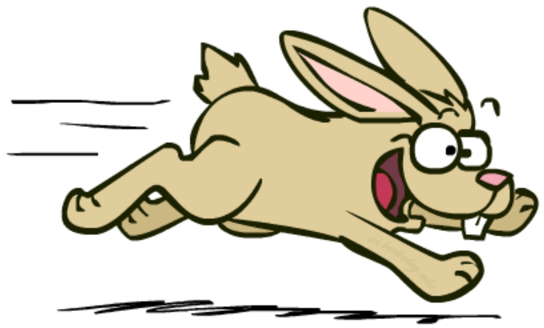
Video of Demo Contours Greedy (Pacman Small Maze)

A* Search

A* Search



UCS

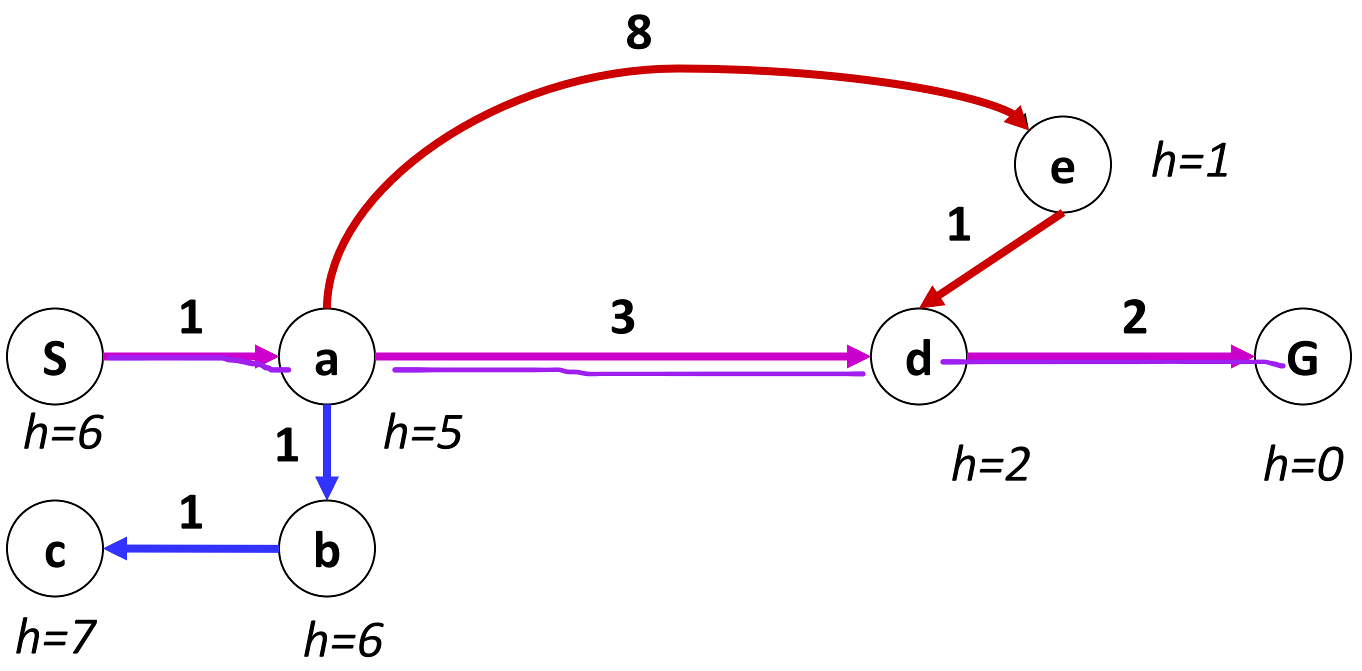


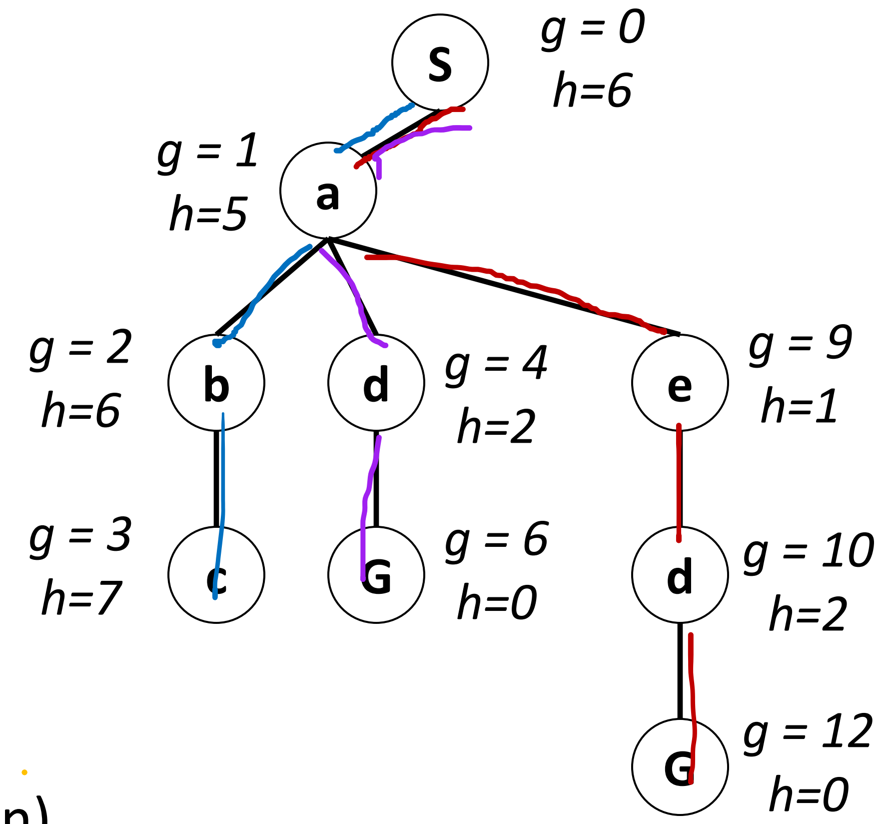
Greedy

A*

Combining UCS and Greedy

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



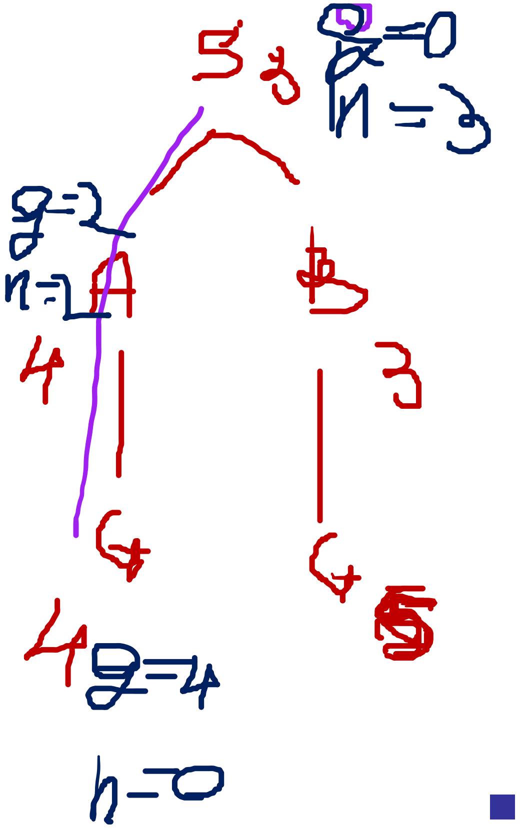


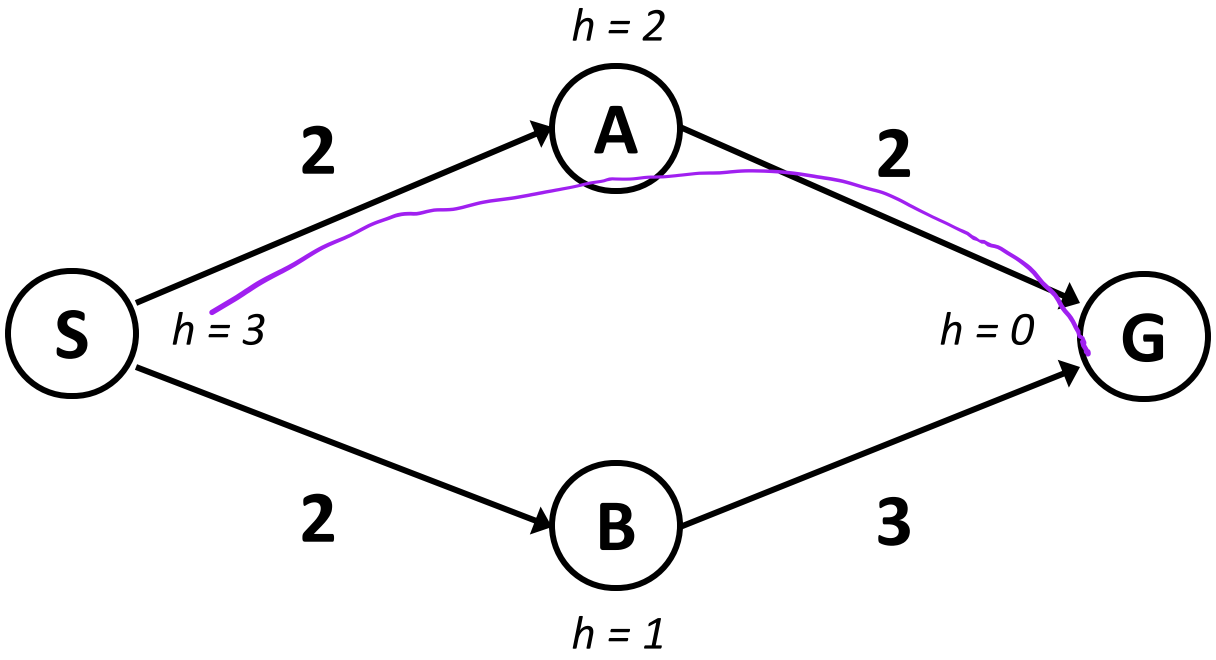
- A* Search orders by the sum: $f(n) = g(n) + h(n)$

Example: Teg Grenager

When should A* terminate?

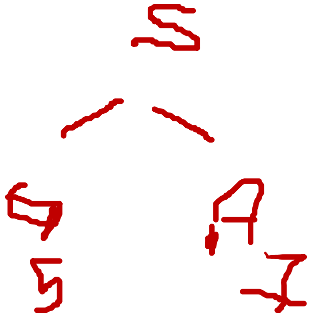
- Should we stop when we enqueue a goal?

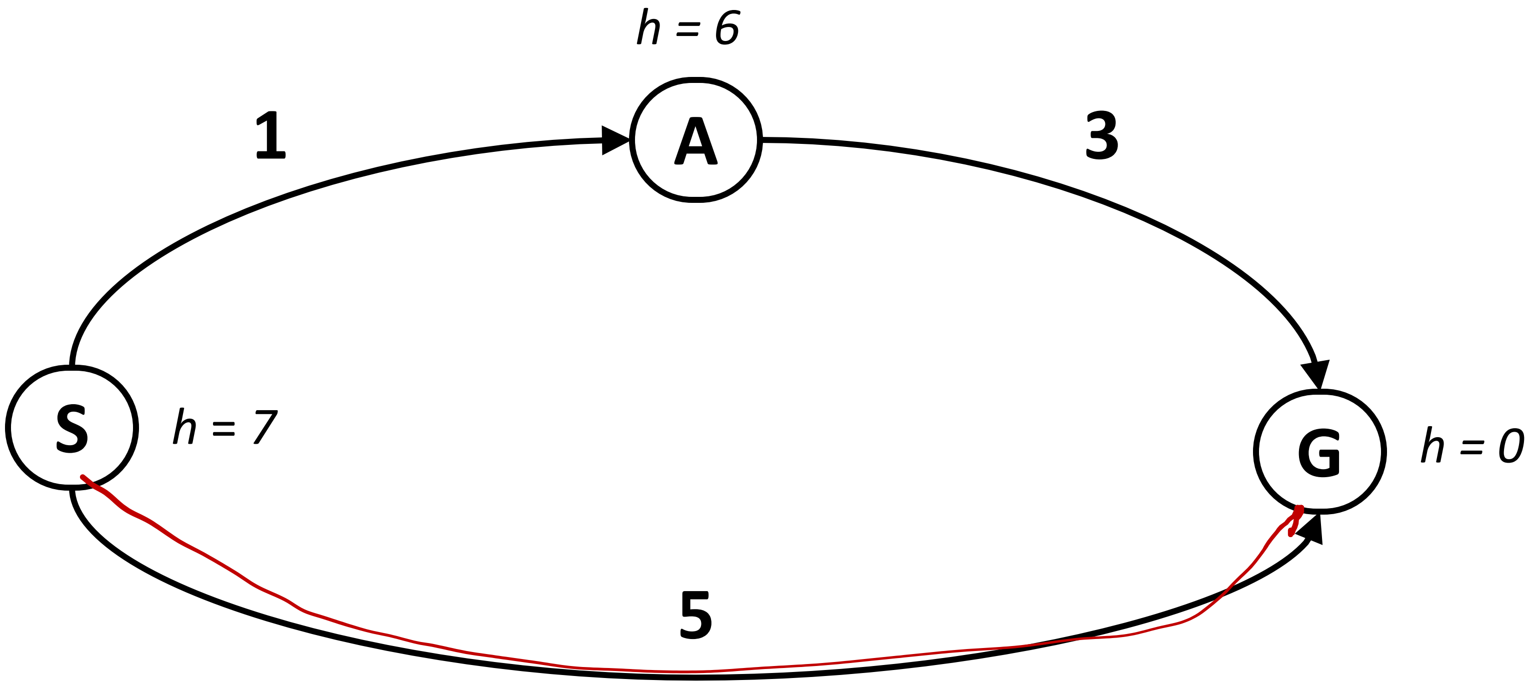




- No: only stop when we dequeue a goal

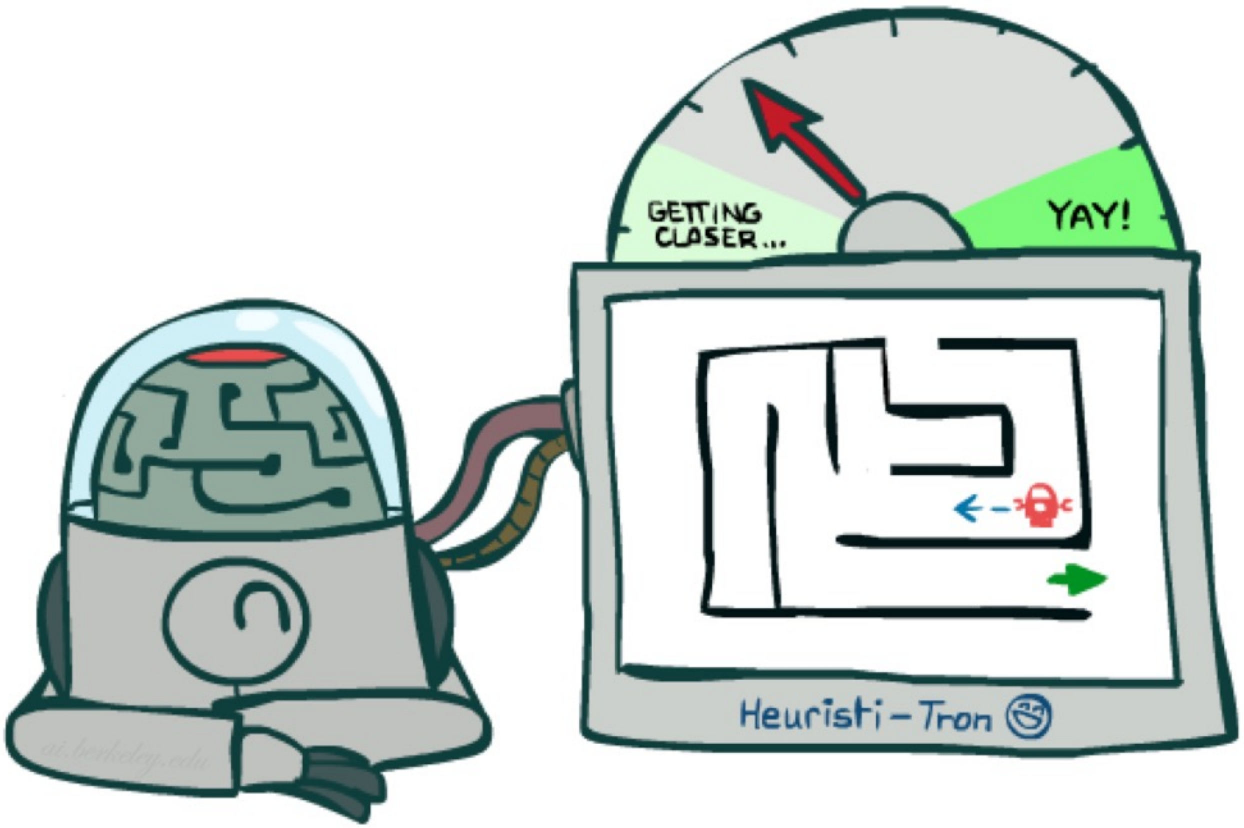
Is A* Optimal?



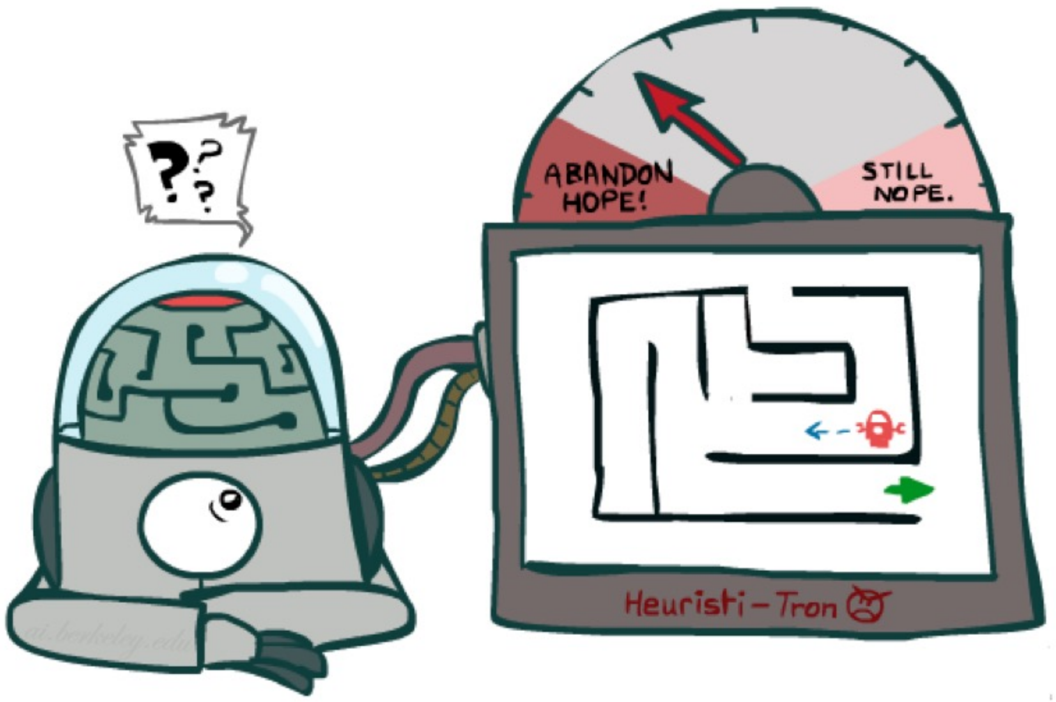


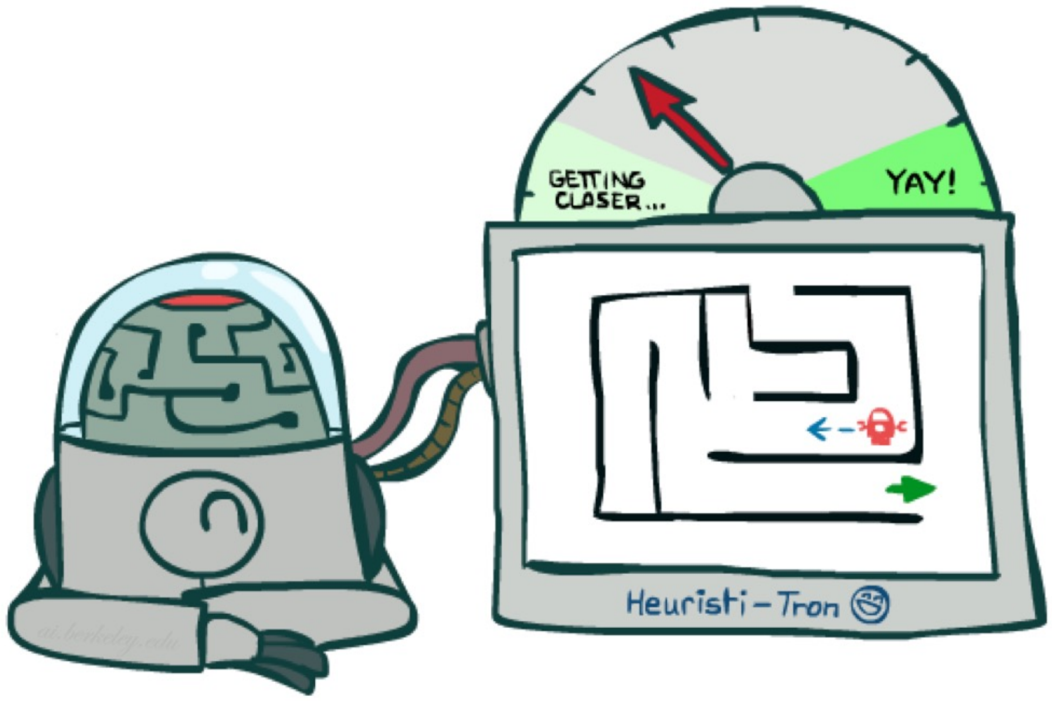
- What went wrong?
- Actual bad goal cost < estimated good goal cost
- We need estimates to be less than actual costs!

Admissible Heuristics



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

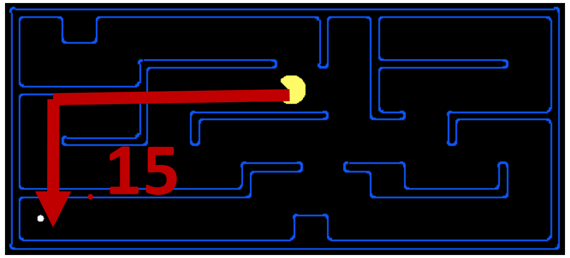
Admissible Heuristics

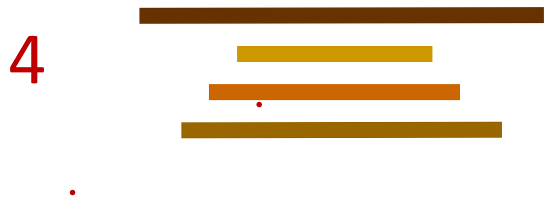
- A heuristic h is *admissible* (optimistic) if:

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

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

- Examples:



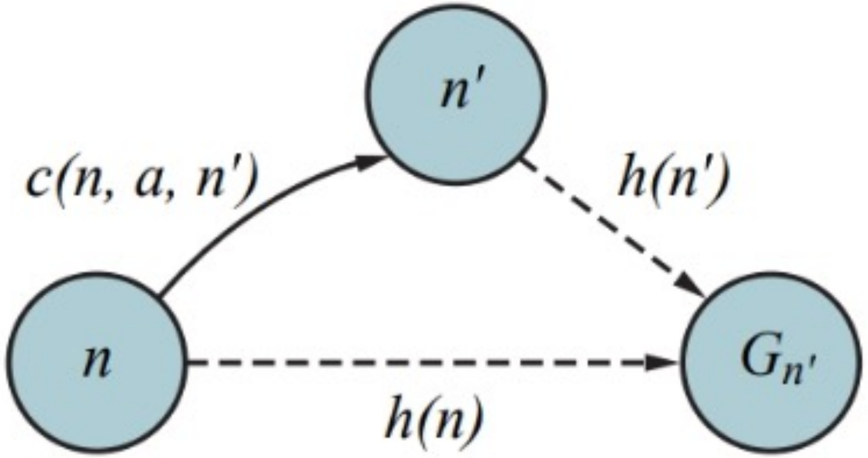


- Coming up with admissible heuristics is most of what's involved in using A* in practice.

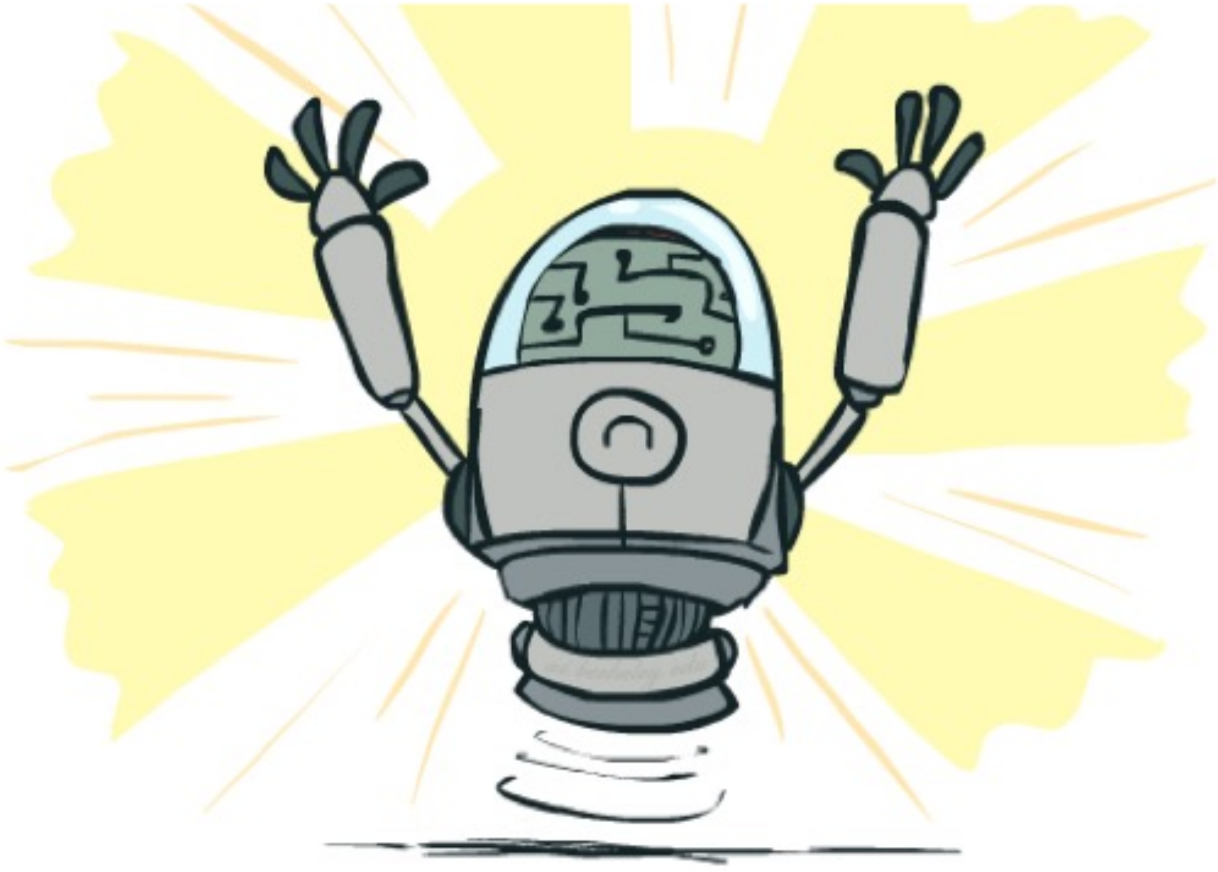
Consistent Heuristics

- A heuristic h is consistent if, for every node n and every successor n' of n generated by an action a, we have:

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



Optimality of A* Tree Search



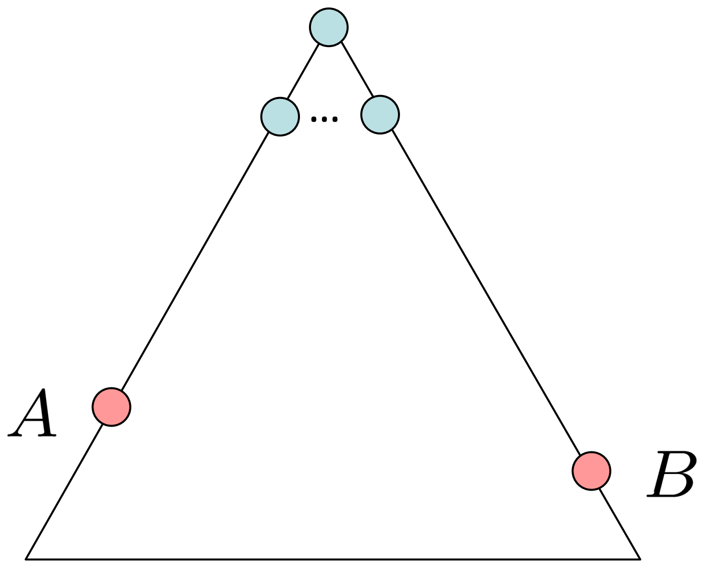
Optimality of A* Tree Search

Assume:

- A is an optimal goal node
- B is a suboptimal goal node
- h is admissible

Claim:

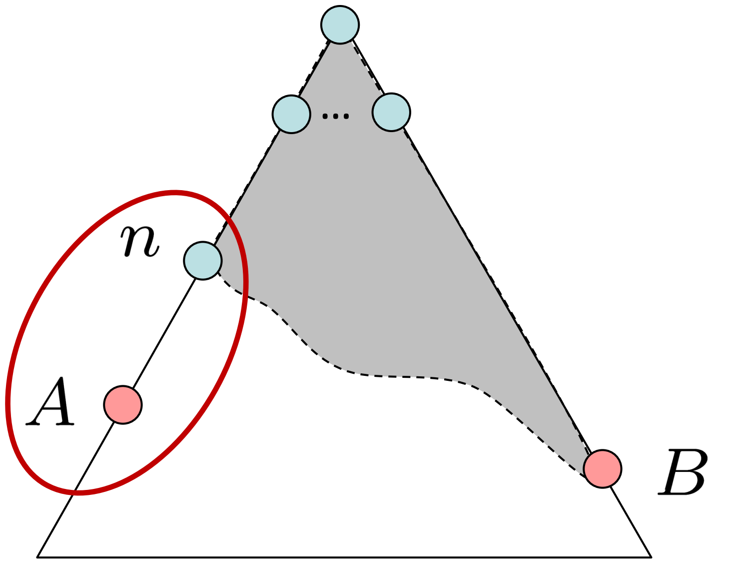
- A will exit the fringe before B



Optimality of A* Tree Search: Blocking

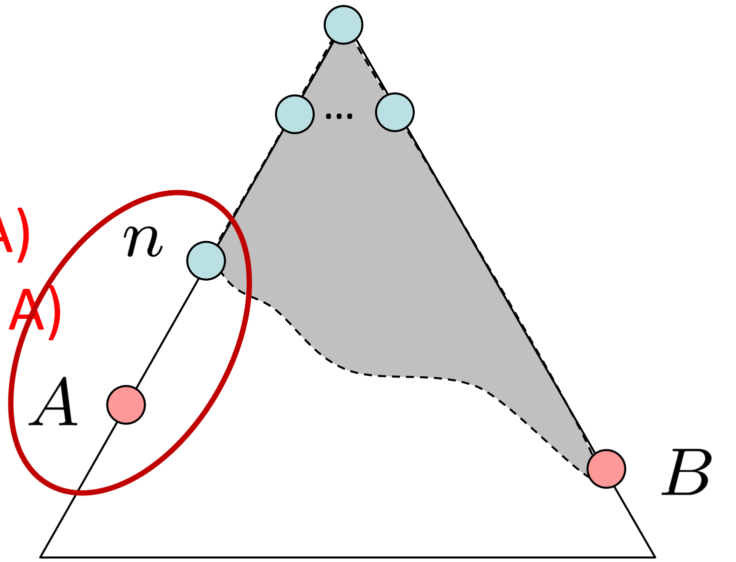
Proof:

- Imagine B is on the fringe
- Some ancestor *n* of A is on the fringe, too (maybe A!)
- Claim: *n* will be expanded before B
  1. $f(n) \leq f(A)$



Optimality of A* Tree Search: Blocking

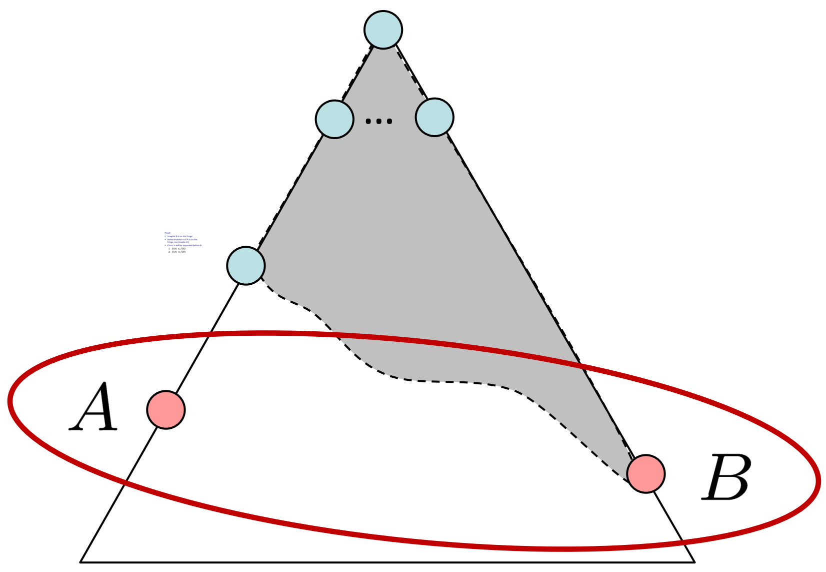
1. $f(n) \leq f(A)$
   - Definition of f-cost says:
     $f(n) = g(n) + h(n)$ =(path cost to n) + (est. cost of n to A)
     $f(A) = g(A) + h(A)$ =(path cost to A) + (est. cost of A to A)
   - The admissible heuristic must underestimate the true cost
     $h(A)$ = (est. cost of A to A) $= 0$
   - So now, we have to compare:
     $f(n) = g(n) + h(n)$
     $f(A) = g(A)$
   - $h(n)$ must be an underestimate of the true cost from n to A
     $g(n) + h(n) \leq g(A)$
     $f(n) \leq f(A)$



Optimality of A* Tree Search: Blocking

Proof:

- Imagine B is on the fringe
- Some ancestor *n* of A is on the fringe, too (maybe A!)
- Claim: *n* will be expanded before B
  1. $f(n) \leq f(A)$
  2. $f(A) \leq f(B)$



Optimality of A* Tree Search: Blocking

2. $f(A) \le f(B)$

- We know that:

  $f(A) = g(A) + h(A)$ = (path cost to A) + (est. cost of A to A)

  $f(B) = g(B) + h(B)$ = (path cost to B) + (est. cost of B to B)

- The heuristic must underestimate the true cost:

  $h(A) = h(B) = 0$

- So now, we have to compare:

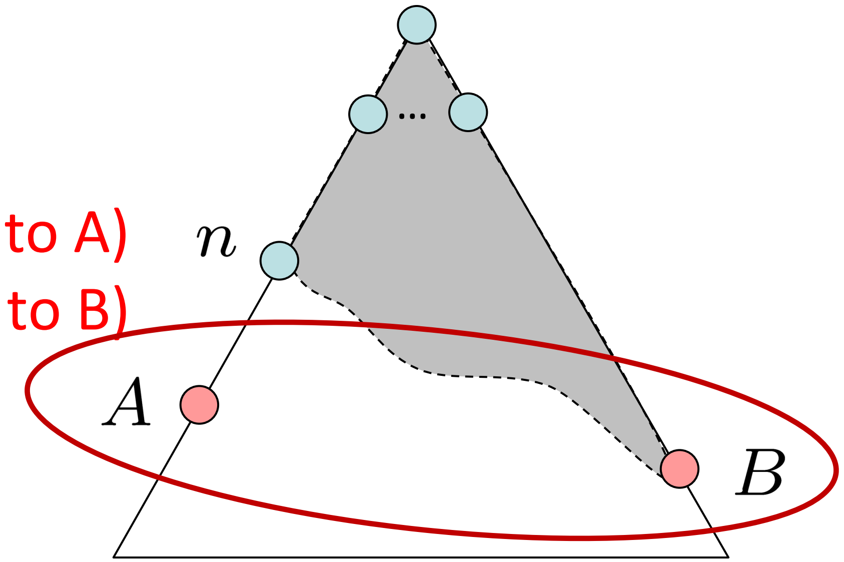
  $f(A) = g(A)$

  $f(B) = g(B)$

- We assumed that B is suboptimal! So

  $g(A) < g(B)$

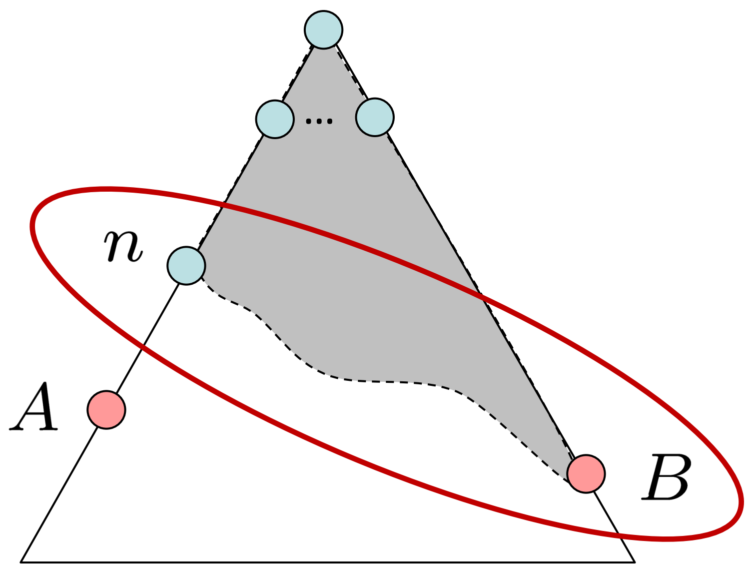
  $f(A) < f(B)$



Optimality of A* Tree Search: Blocking

Proof:

- Imagine B is on the fringe
- Some ancestor *n* of A is on the fringe, too (maybe A!)
- Claim: *n* will be expanded before B
  1. $f(n) \leq f(A)$
  2. $f(A) < f(B)$
  3. n expands before B
- All ancestors of A expand before B
- A expands before B
- A* search is optimal

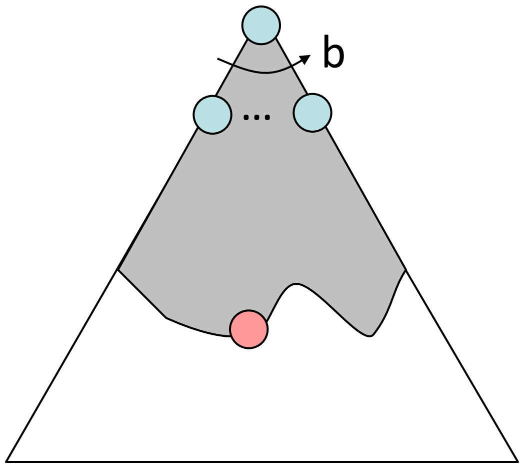


$f(n) \leq f(A) < f(B)$

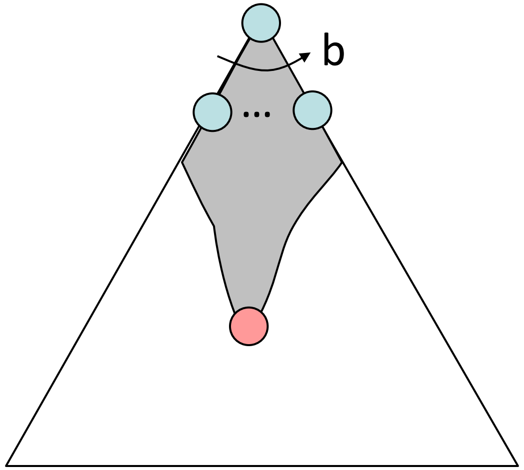
Properties of A*

Properties of A*

Uniform-Cost

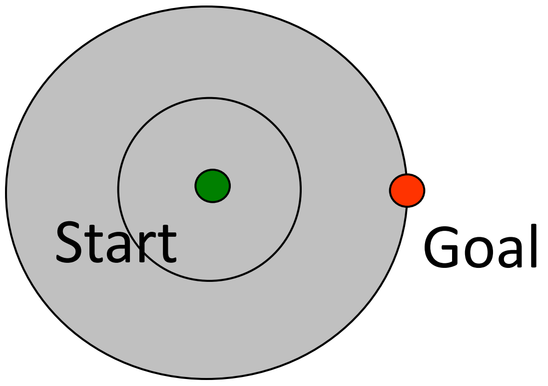


A*

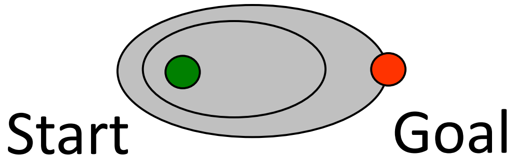


UCS vs A* Contours

- 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

Video of Demo Contours (Empty) -- Greedy

Video of Demo Contours (Empty) – A*

Video of Demo Contours (Pacman Small Maze) – A*

Comparison

Greedy

Uniform Cost

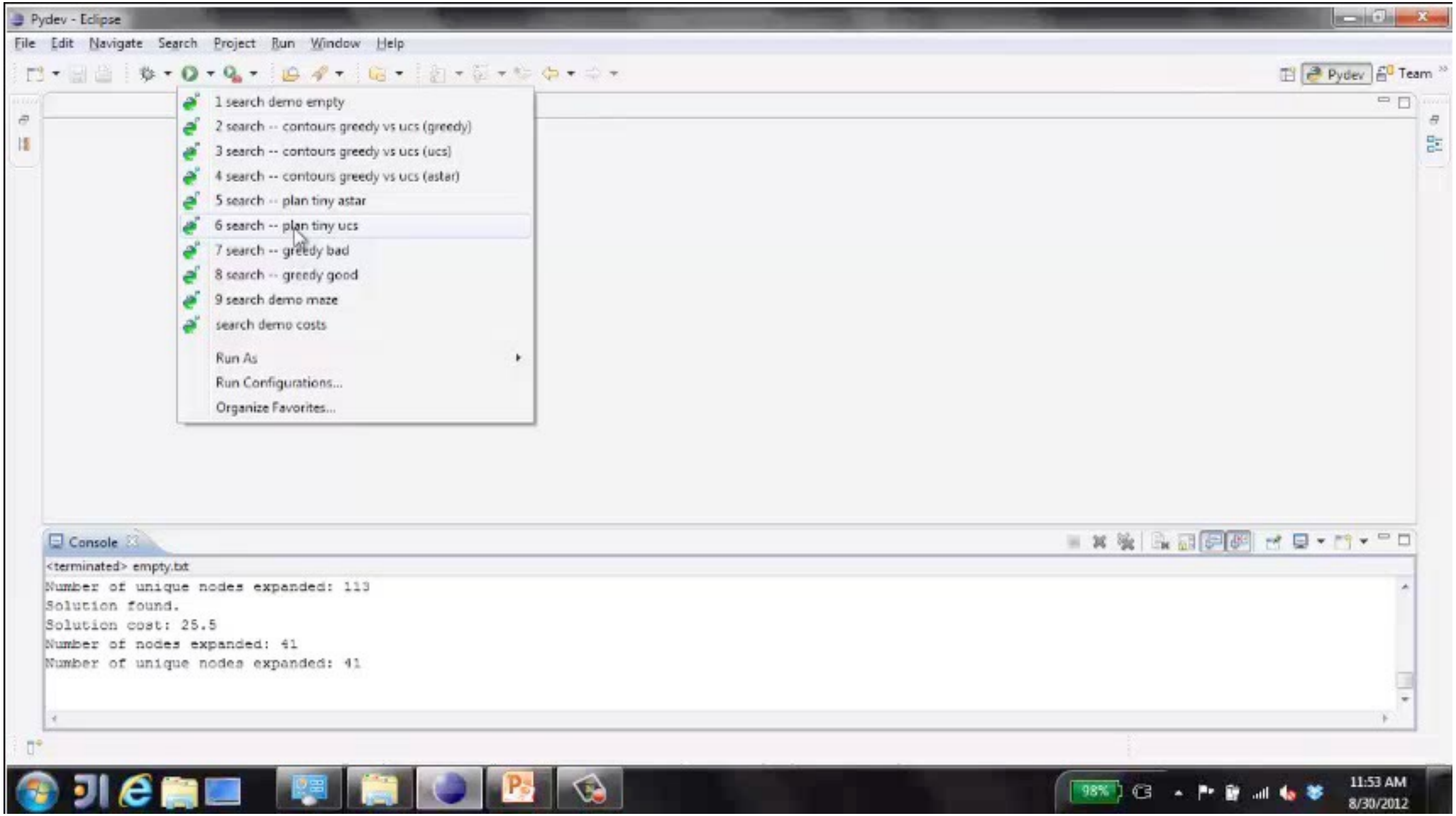
A*

A* Applications

A* Applications

- Video games
- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- …

Video of Demo Pacman (Tiny Maze) – UCS / A*



Video of Demo Empty Water Shallow/Deep – Guess Algorithm