TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education
HCMUTE
Faculty of Information Technology

Informed Search Algorithms

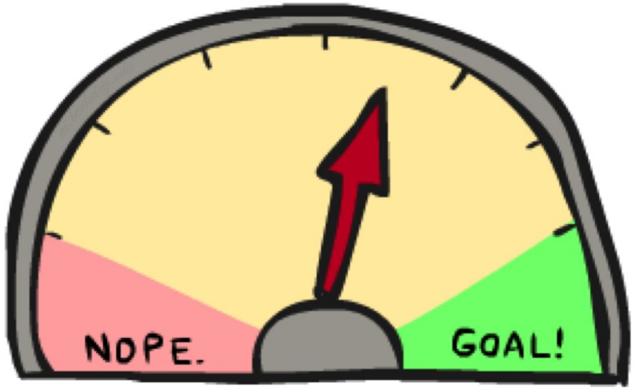
(ARTIFICIAL INTELLIGENCE)

Instructor: Assoc. Prof. PhD. Hoàng Văn Dũng
Email: dunghv@hcmute.edu.vn

(slides adapted and revised from Dan Klein, Pieter Abbeel, Anca Dragan, et al)

Content

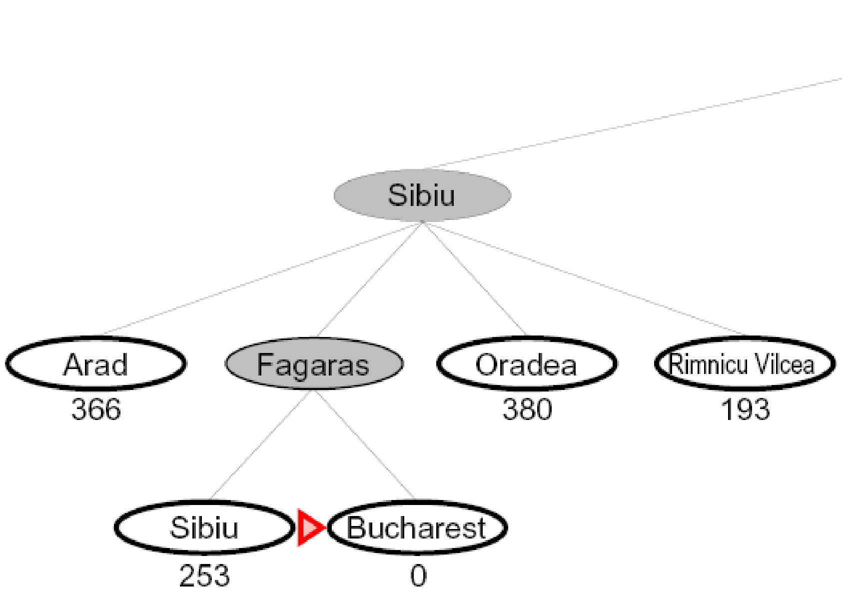
- Informed Search
  - –Best first search
  - –Heuristics
  - –A* Search

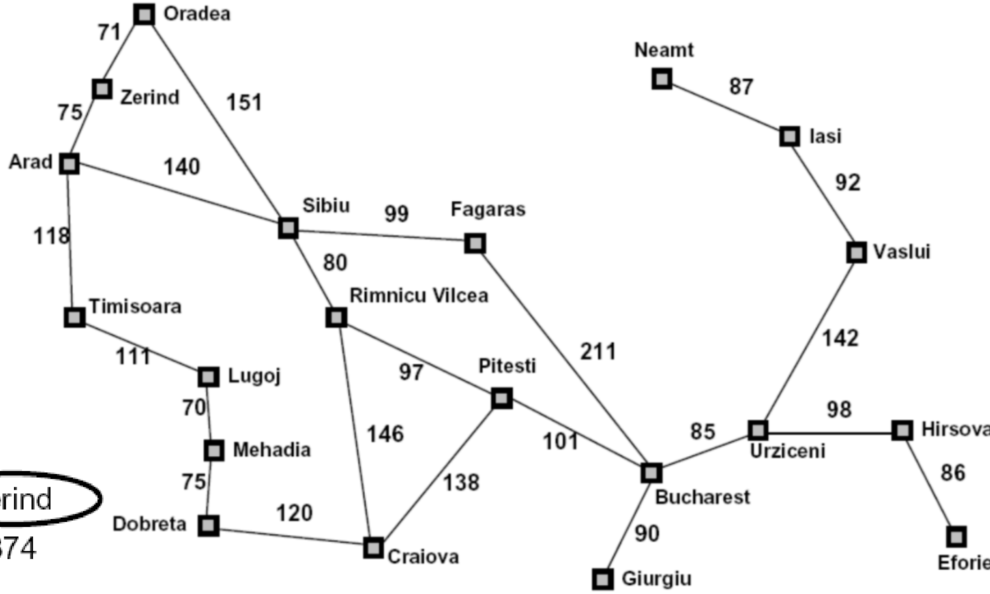


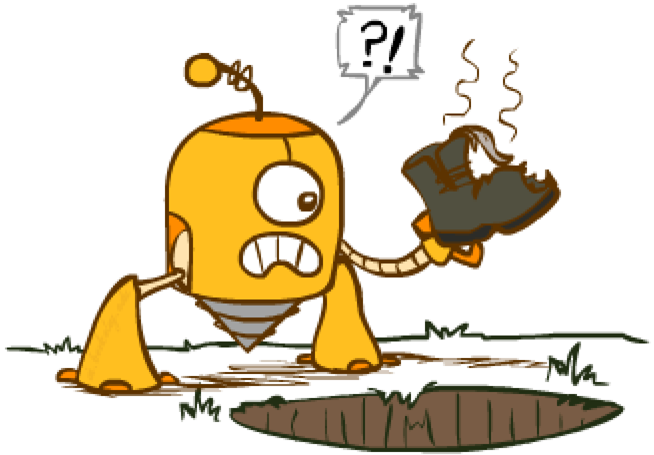
Greedy Search

Greedy Search

- Expand the node that seems closest…



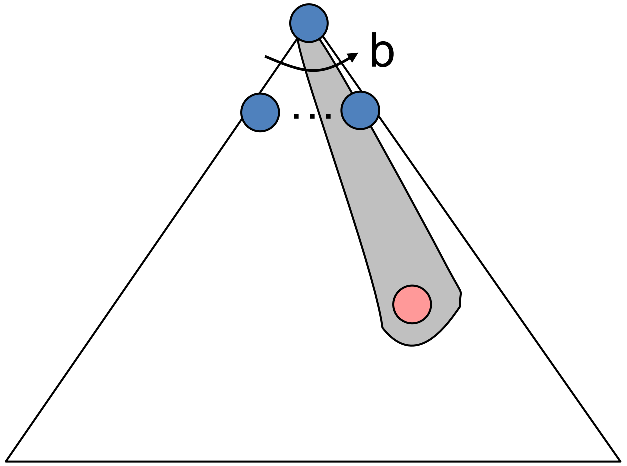




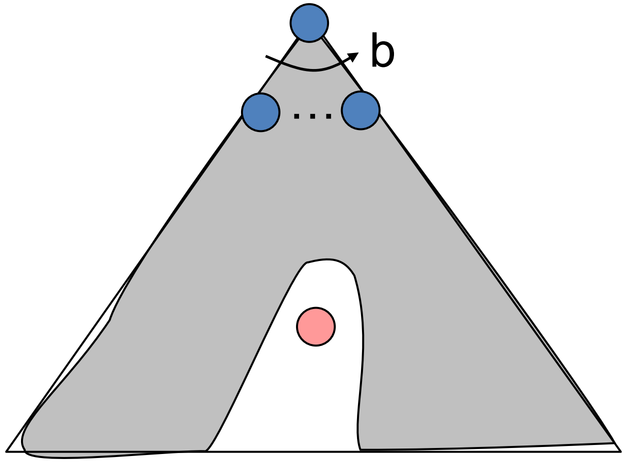
- What can go wrong?

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



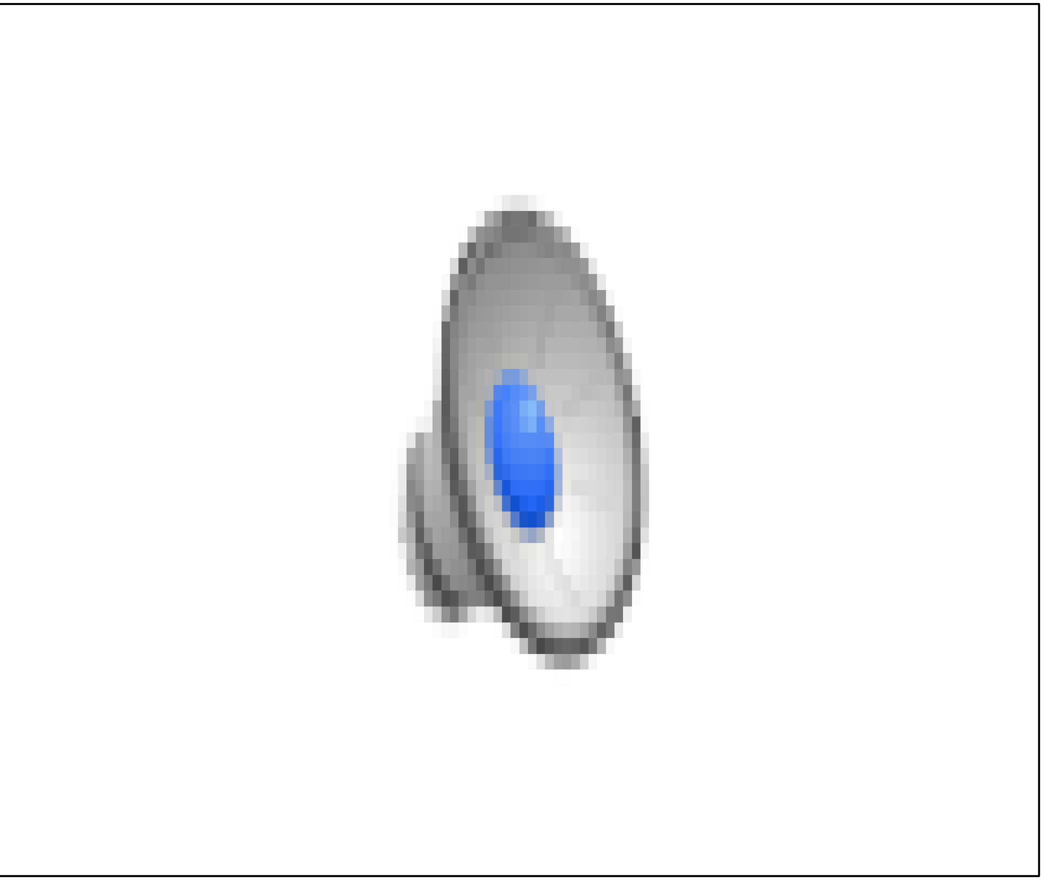
- A common case:
  - Best-first takes you straight to the (wrong) goal

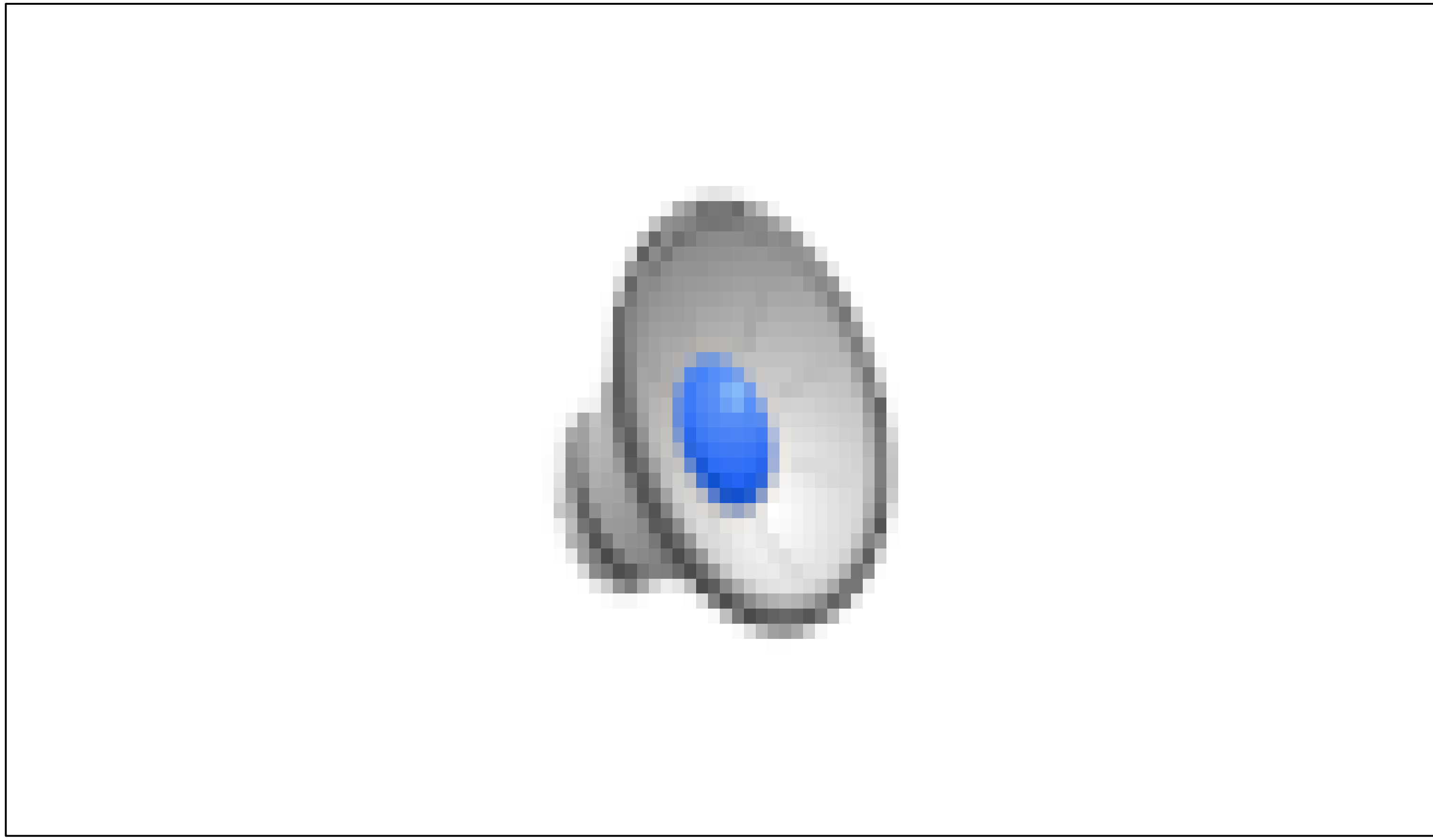


- Worst-case: like a badly-guided DFS

Greedy Search

- Video of Demo Greedy





A* Search

A* Search

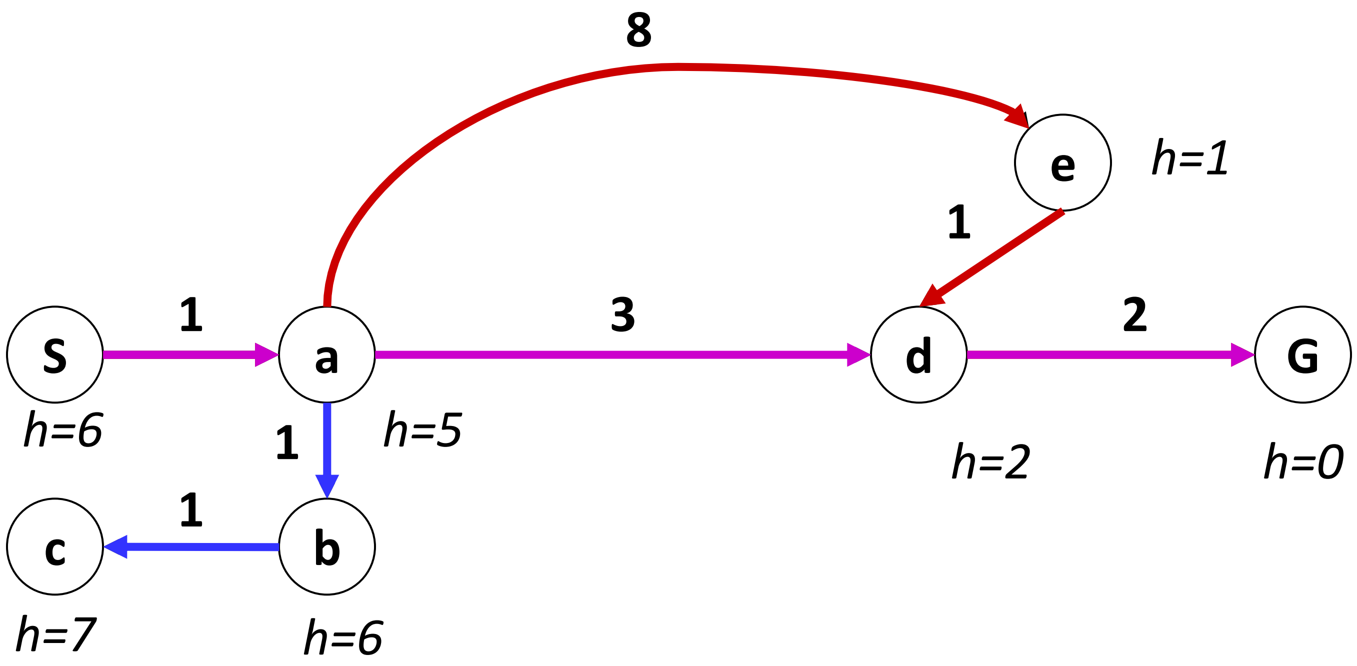
UCS

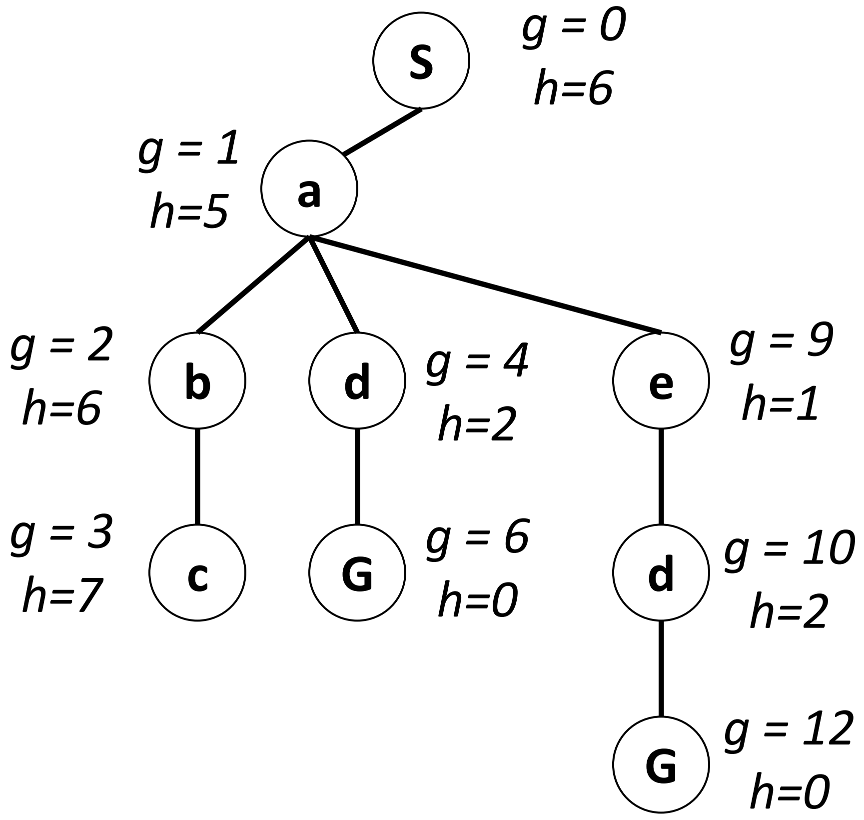
Greedy

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)





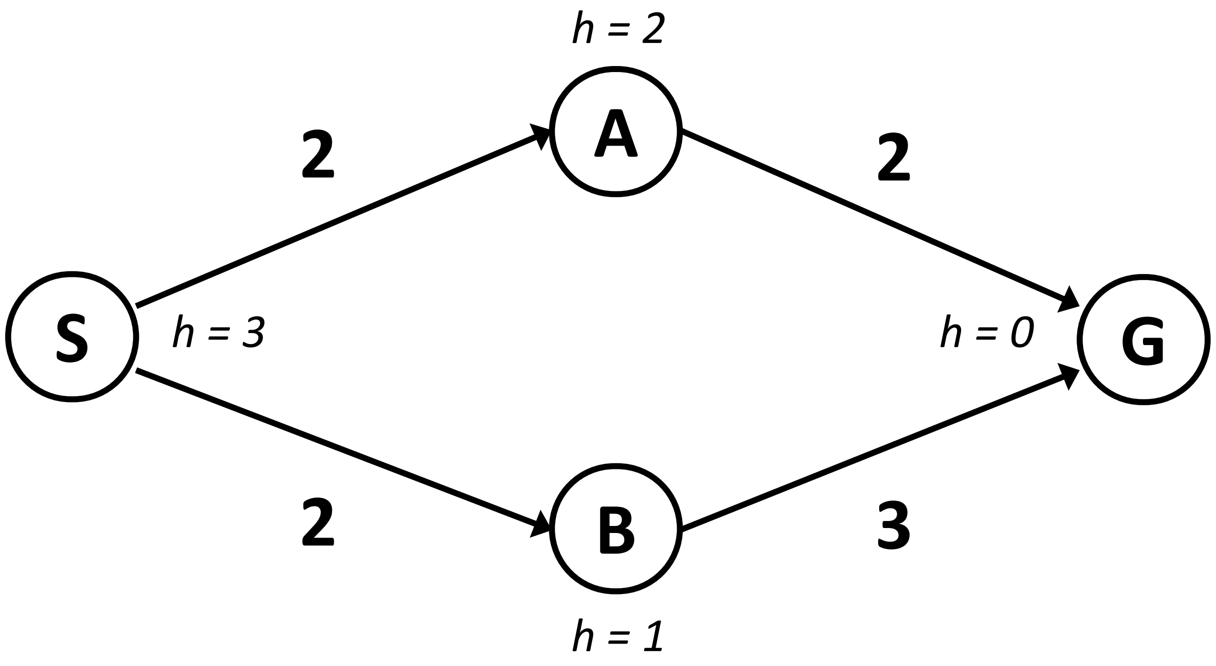
- A* Search orders by the sum: f(n) = g(n) + h(n)

Example: Teg Grenager

A* Search

When should A* terminate?

- Should we stop when we enqueue a goal?

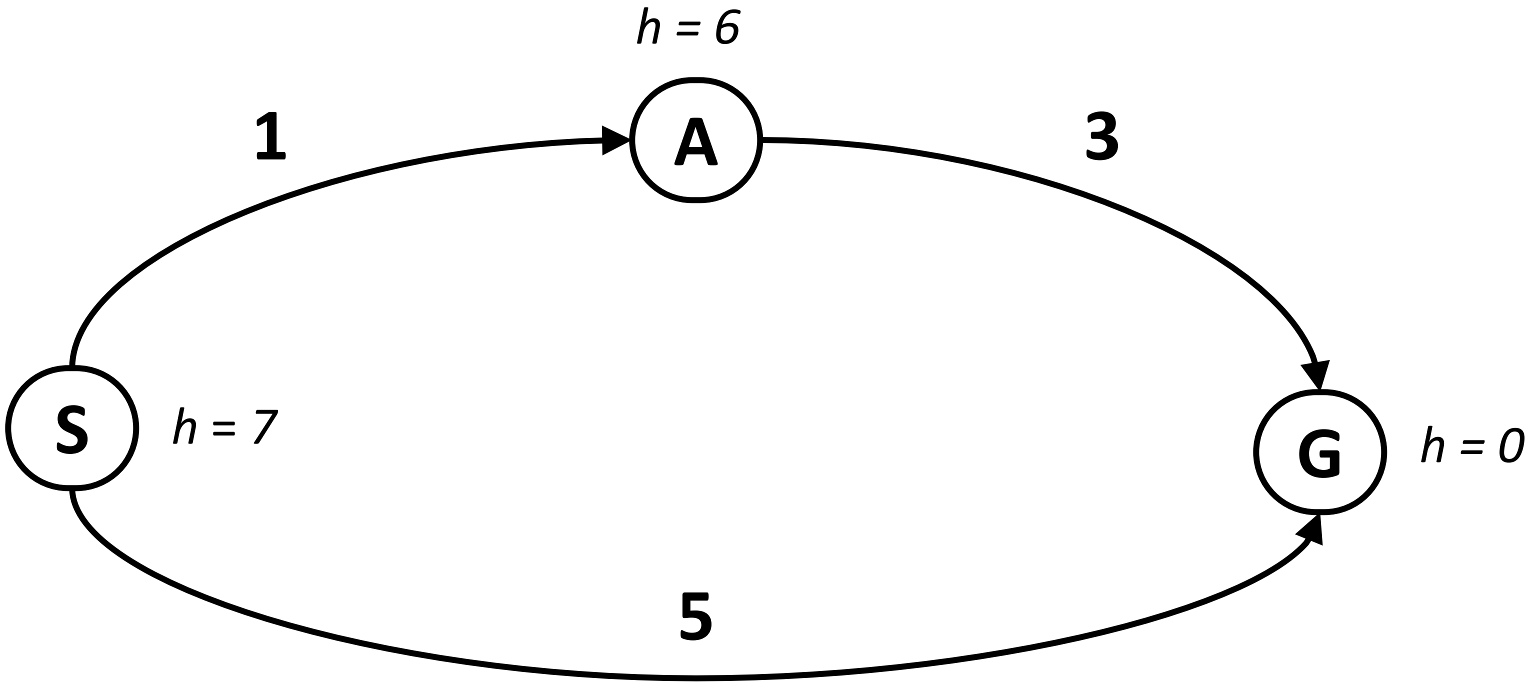


- No: only stop when we dequeue a goal

h = 1

A* Search

Is A* Optimal?



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

A* Search

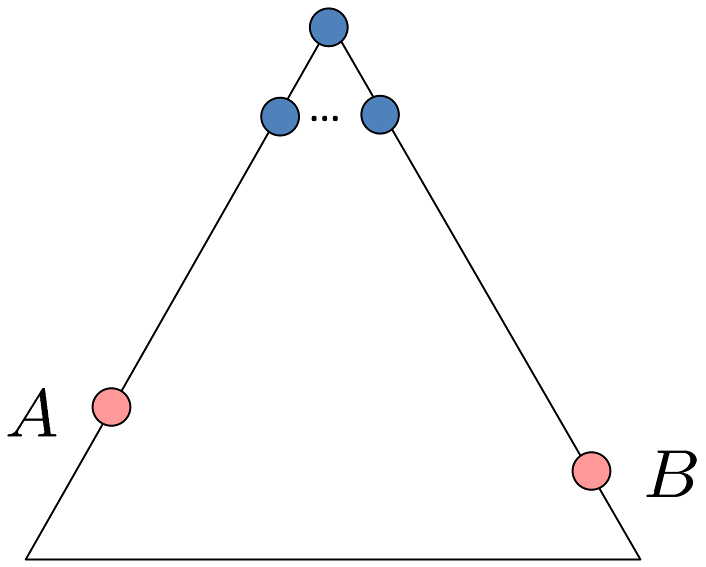
Optimality of A* Tree Search

Assume:

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

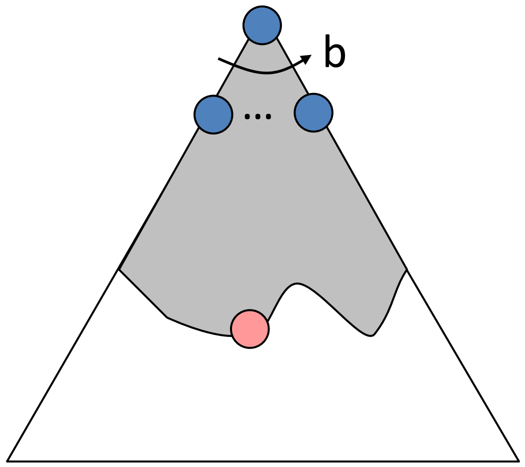
Claim:

- A will exit the fringe before B

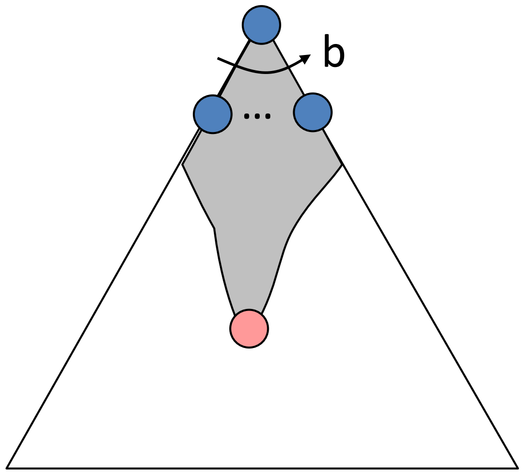


Properties of A*

Uniform-Cost

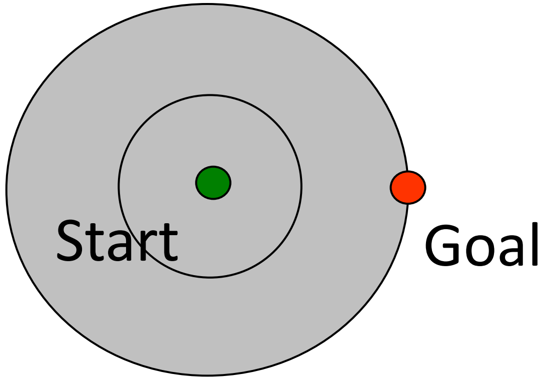


A*

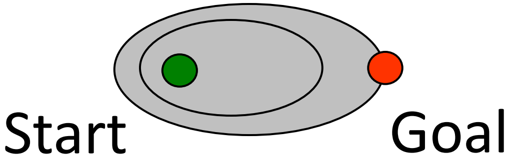


UCS vs A* Contours

- Uniform-cost expands equally in all "directions"

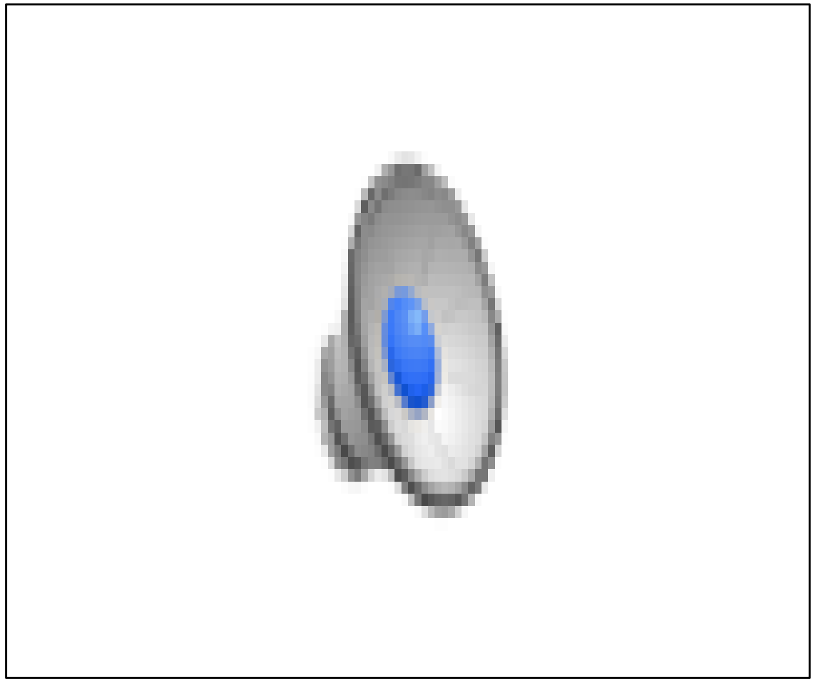


- A* expands mainly toward the goal, but does hedge its bets to ensure optimality

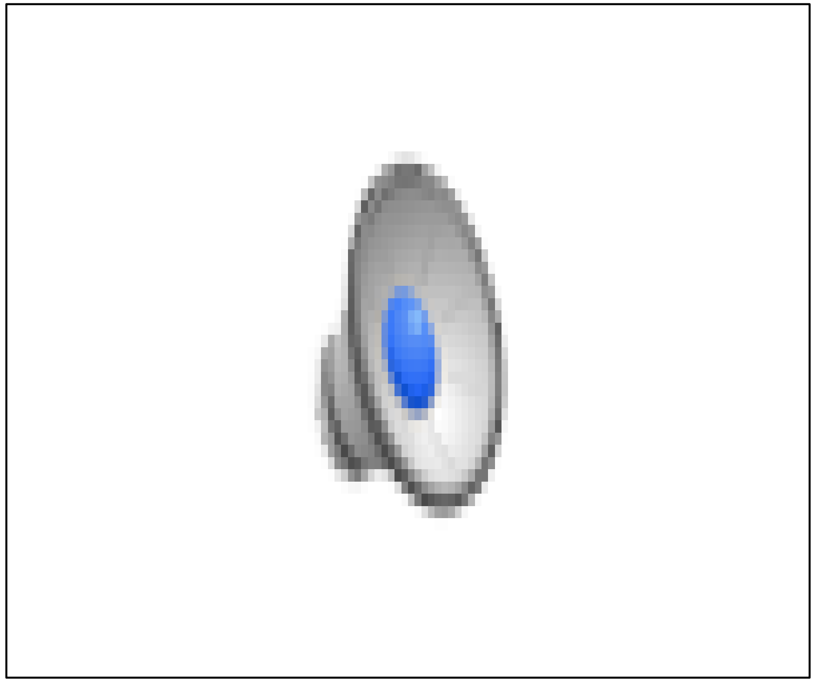


Comparison

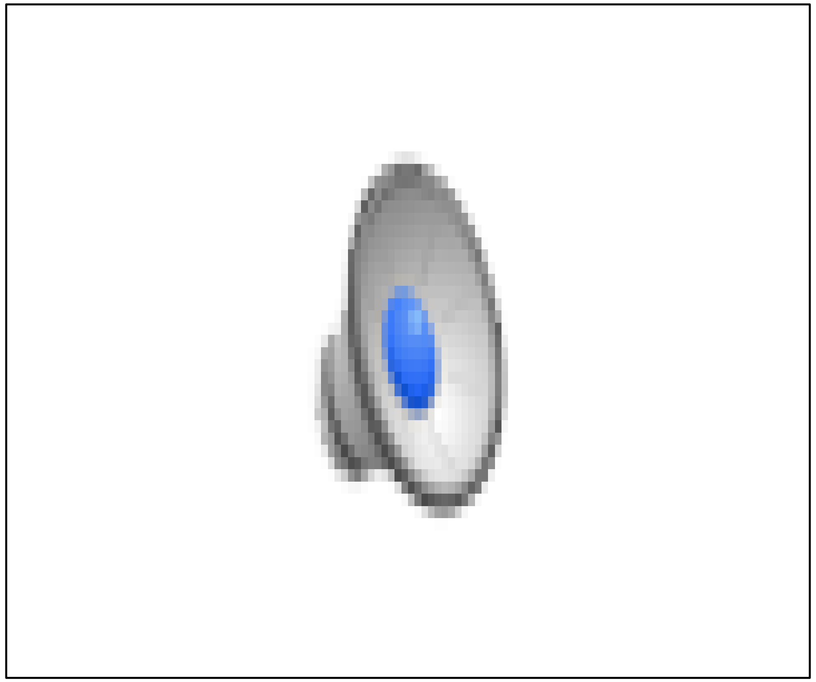
- Video of Demo Contours (Empty)



UCS



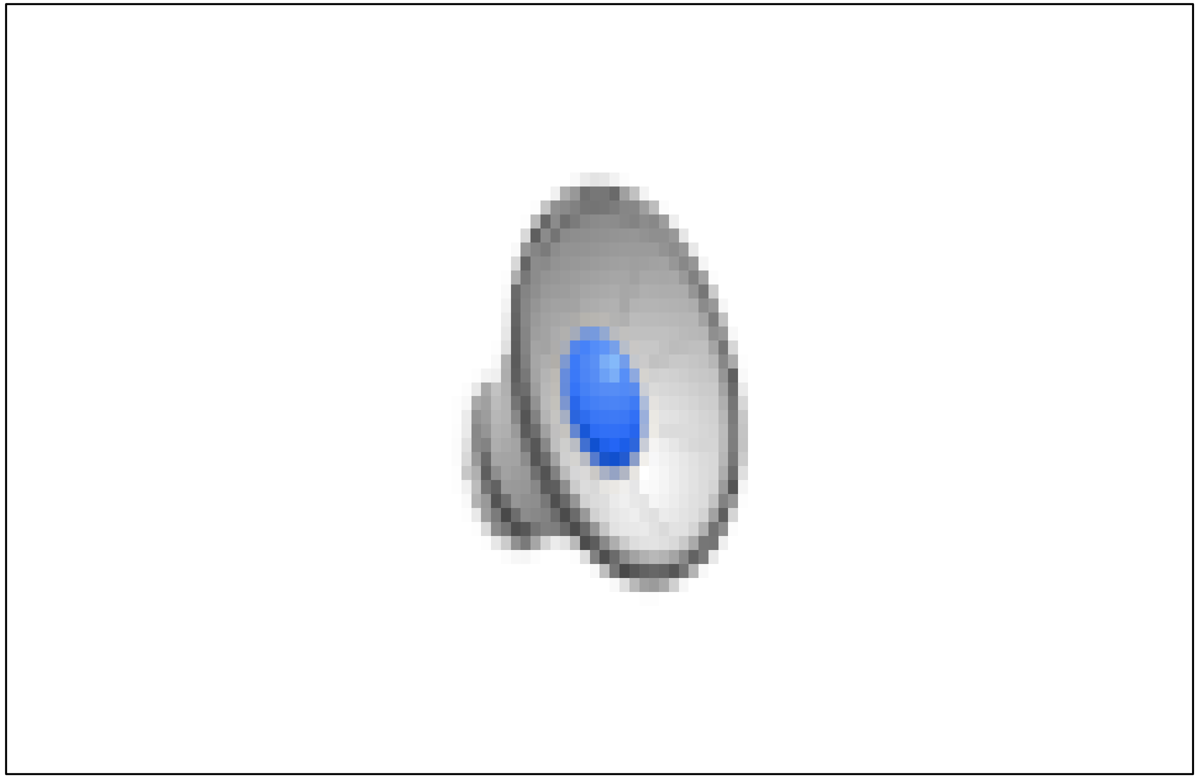
Greedy



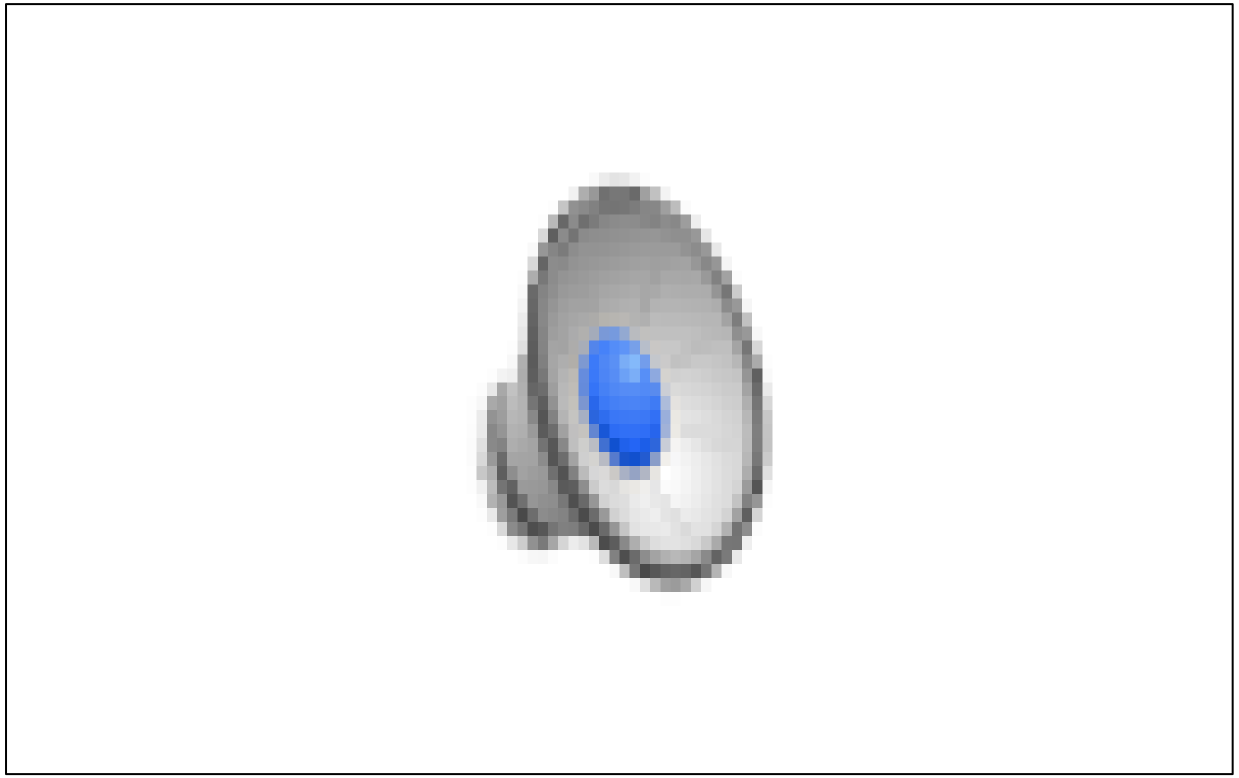
A*

Comparison

- Video of Demo Contours (Pacman Small Maze)



Greedy

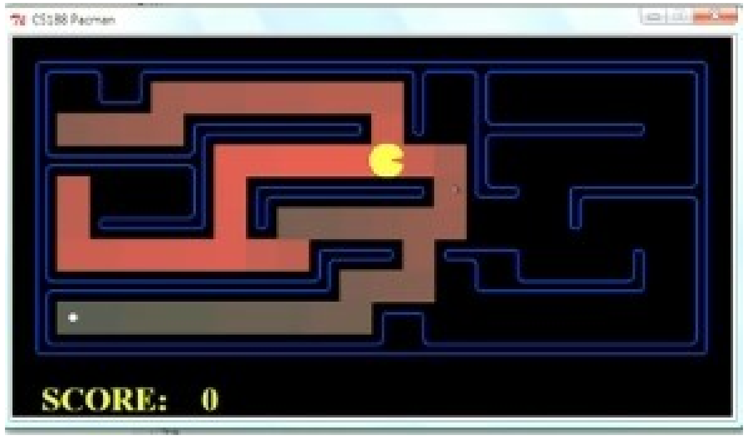


A*

Comparison

Greedy

Uniform Cost

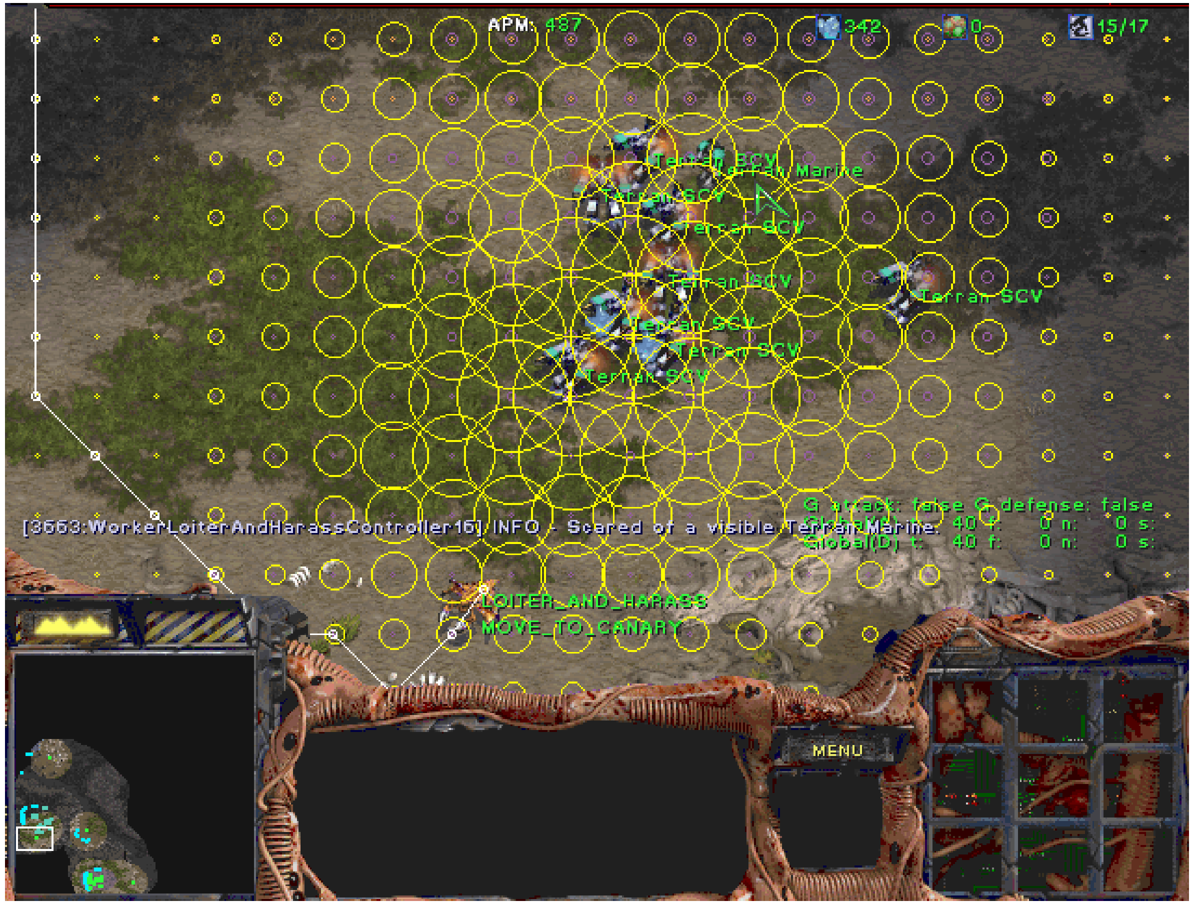


A*

A* search

A* Applications

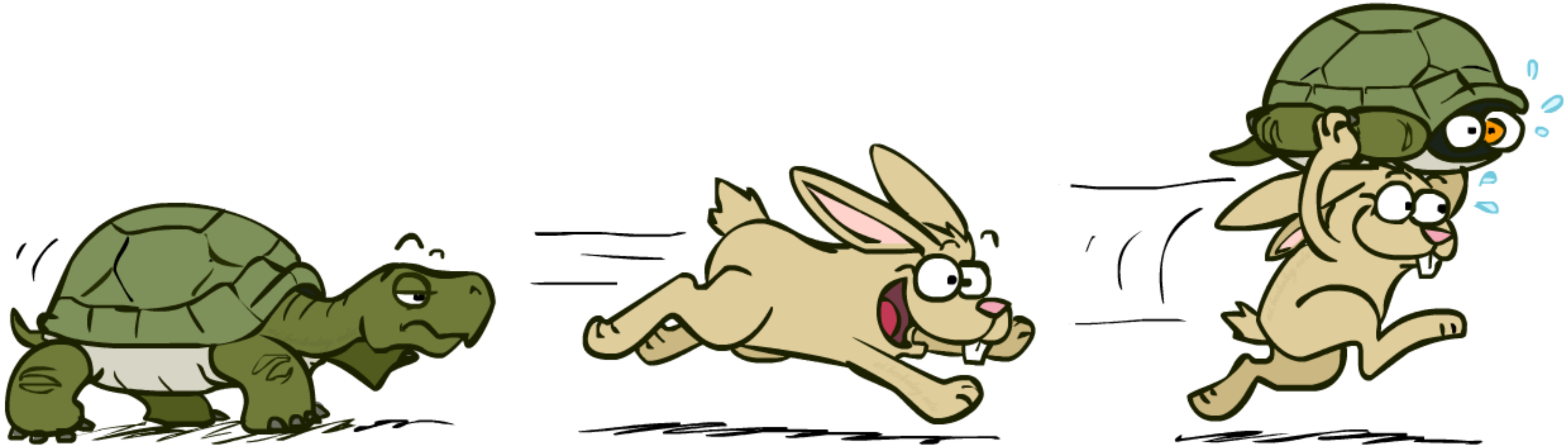
- Video games
- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition
- …



A* search

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



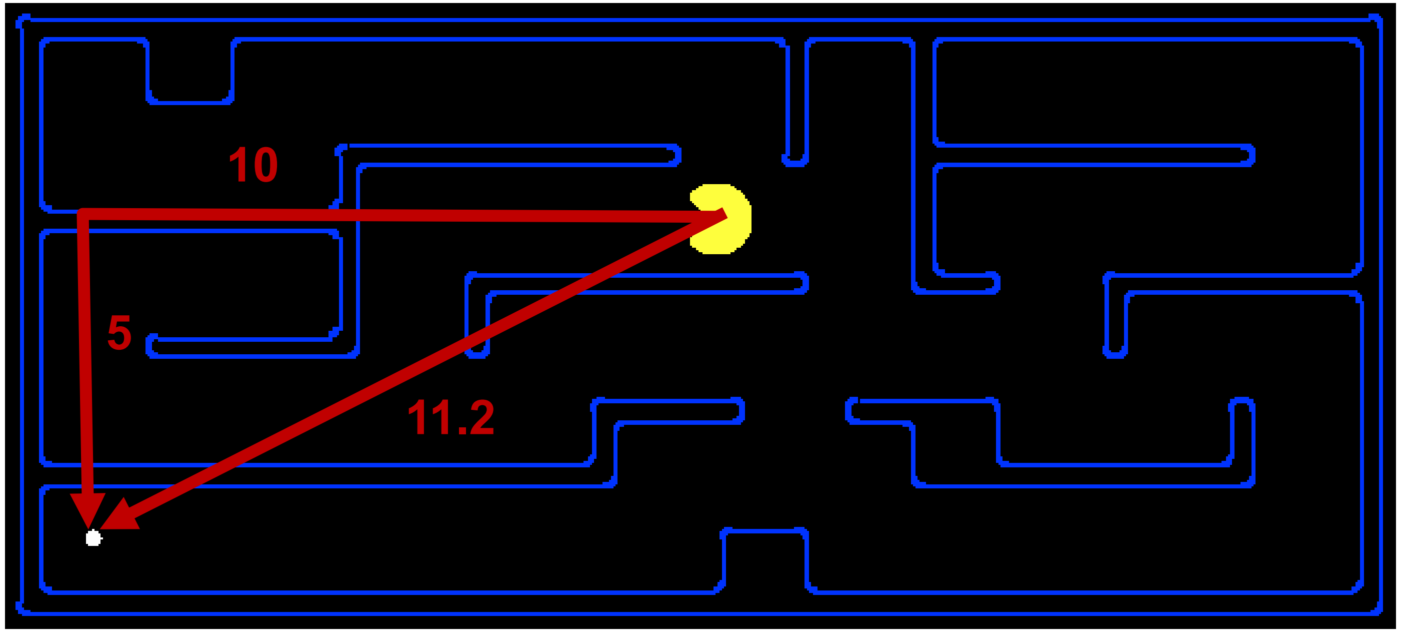
Search Heuristics

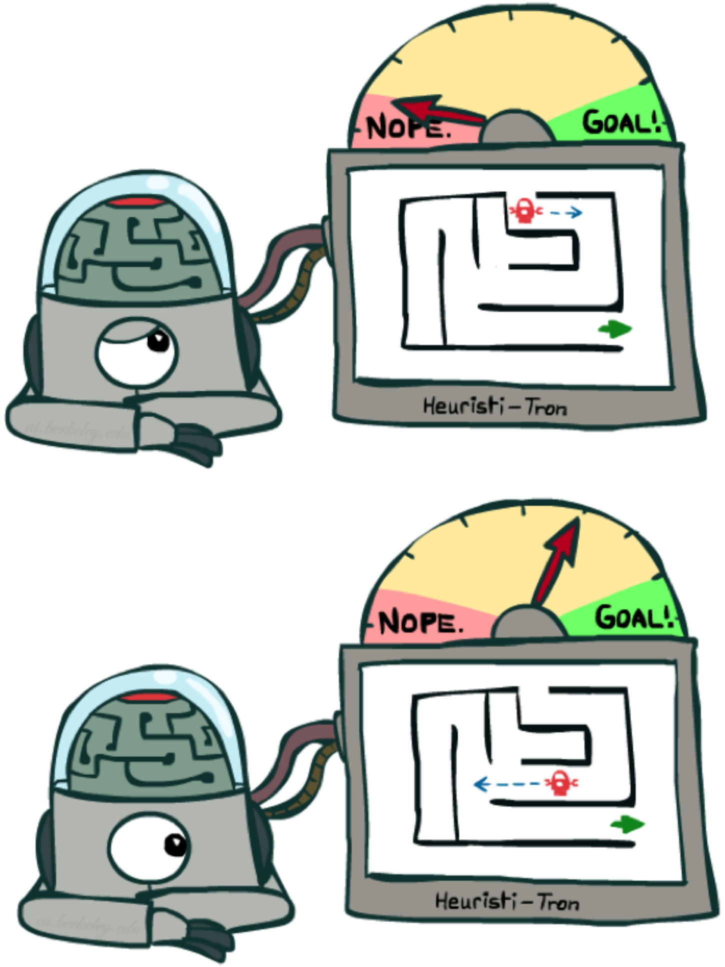
- Heuristic is:
  - A function that *estimates* how close a state is to a goal

Function h(s) with s is a state

  - Designed for a particular search problem

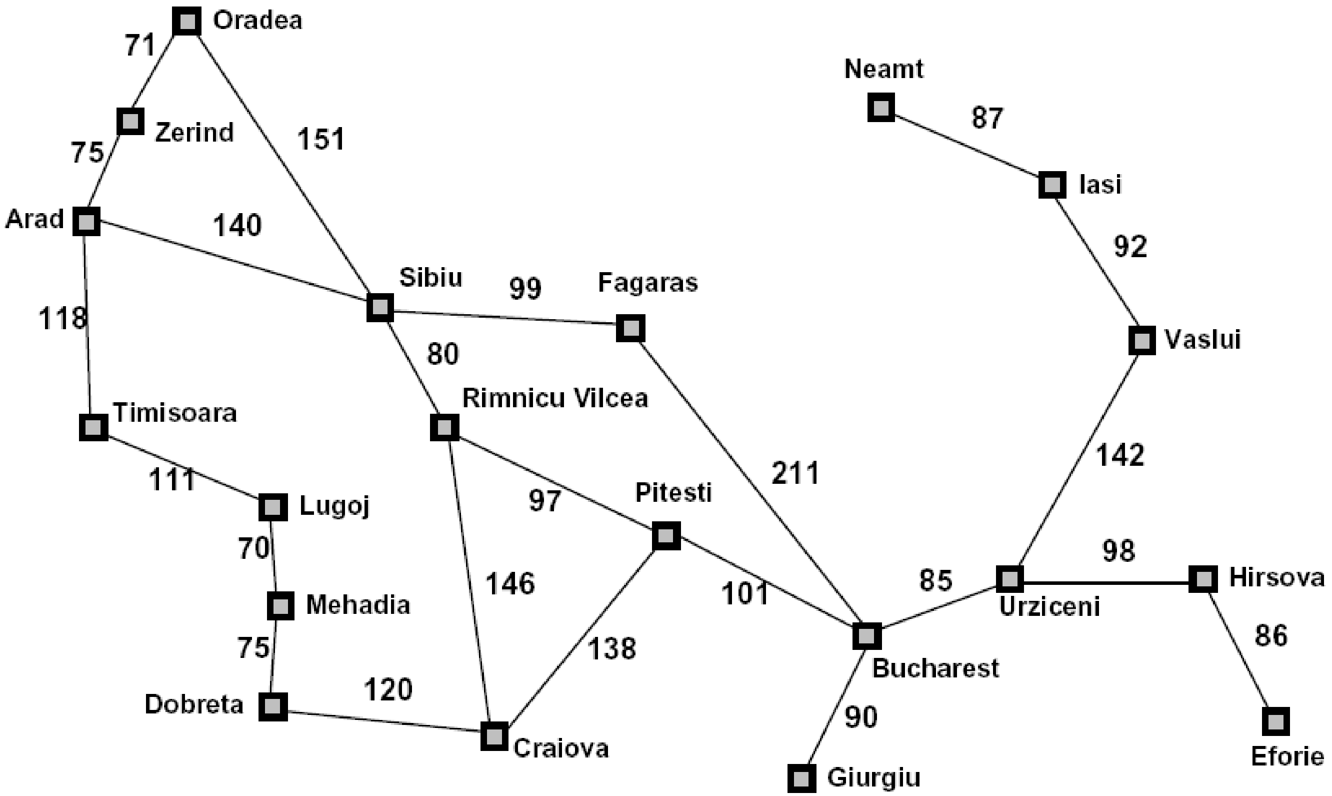
Examples: Manhattan distance, Euclidean distance for pathing





Search Heuristics

- Example: Heuristic Function



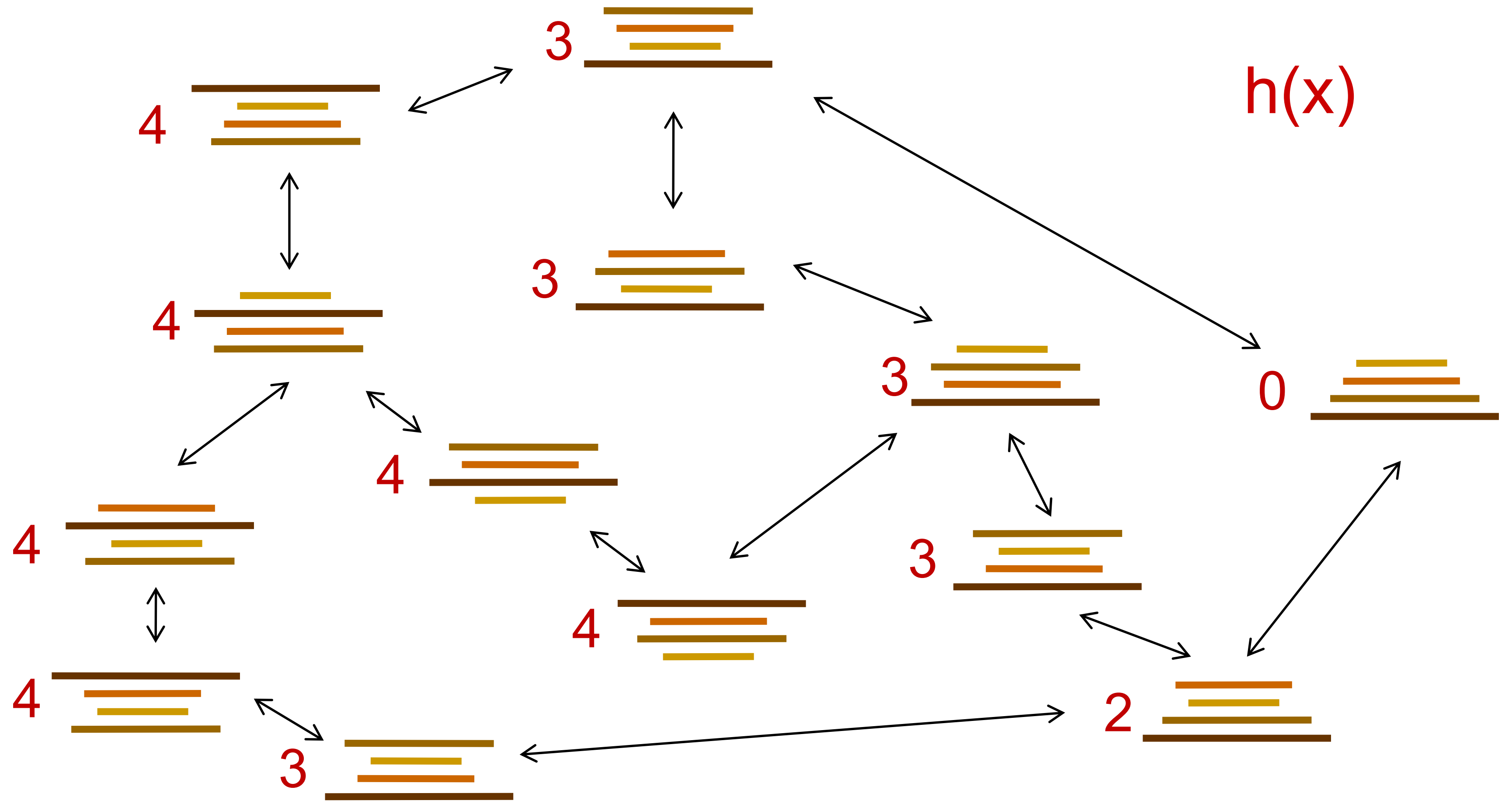
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

Search Heuristics

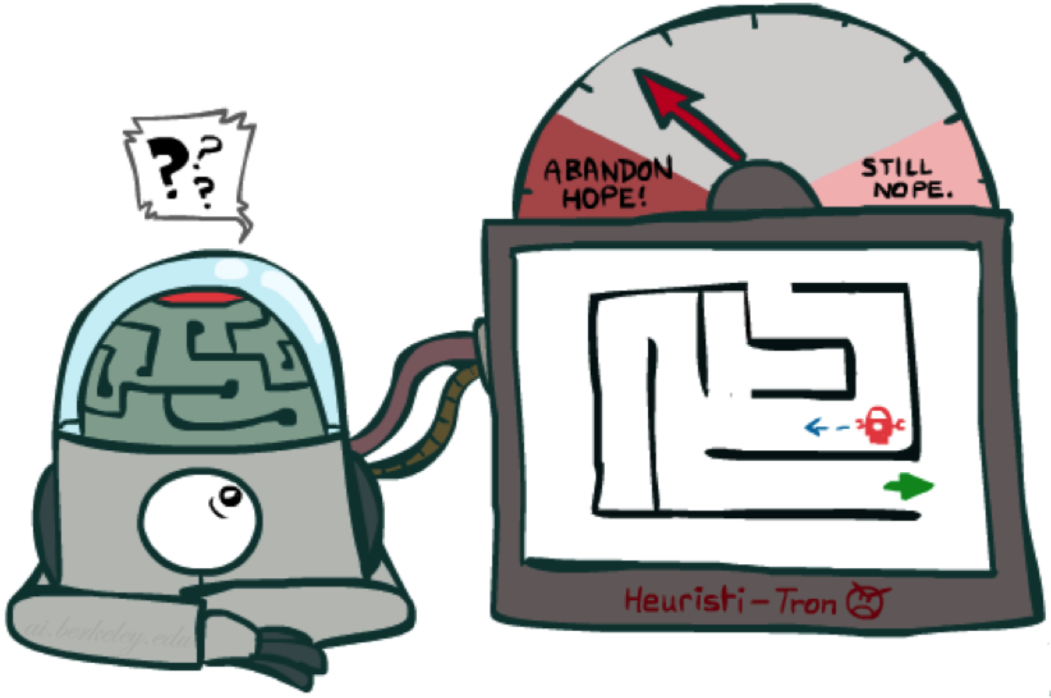
- Example: Heuristic Function

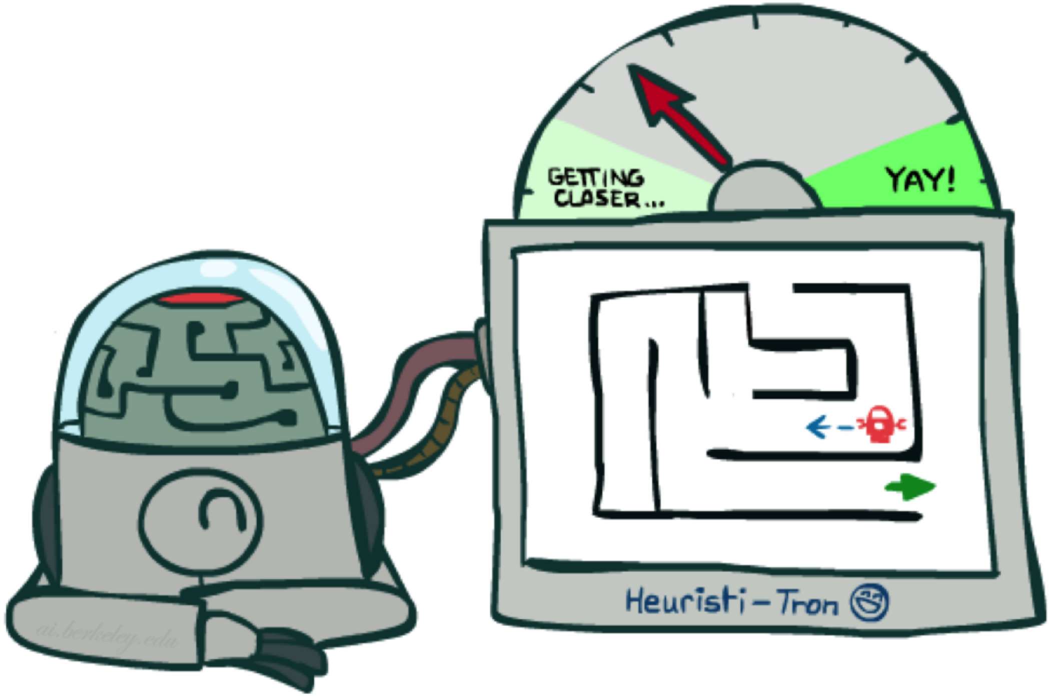
Heuristic: the number of the largest pancake that is still out of place



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

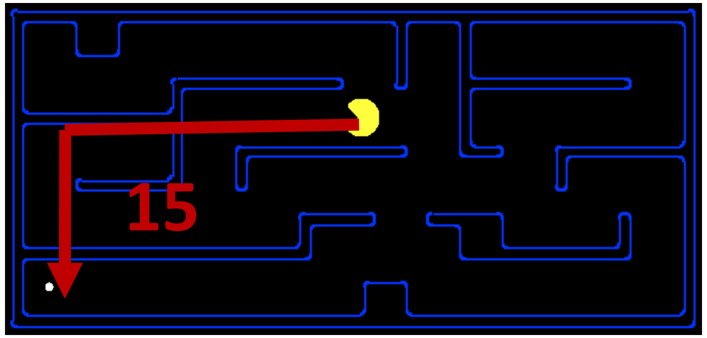
Admissible Heuristics

- 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:



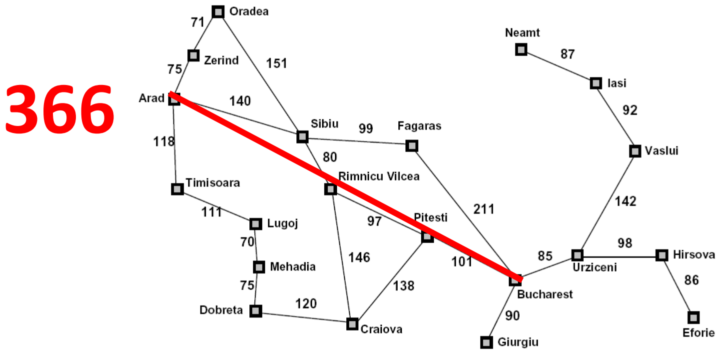
4

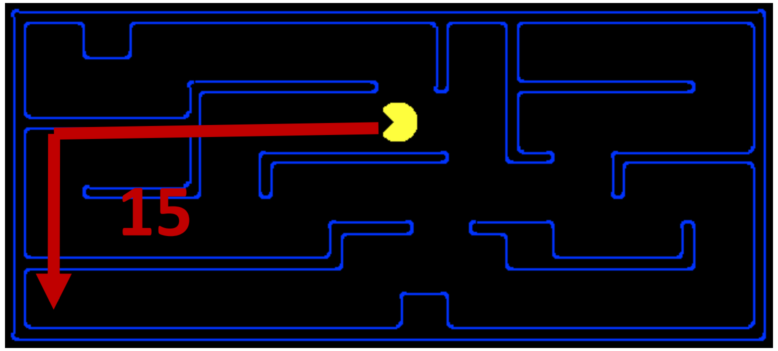
- Coming up with admissible heuristics is most of what's involved in using A* in practice.

Creating Admissible Heuristics

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

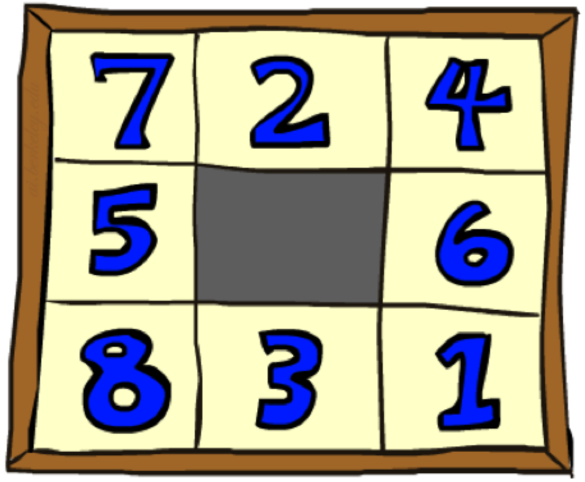
- Often, admissible heuristics are solutions to *relaxed problems*, where new actions are available



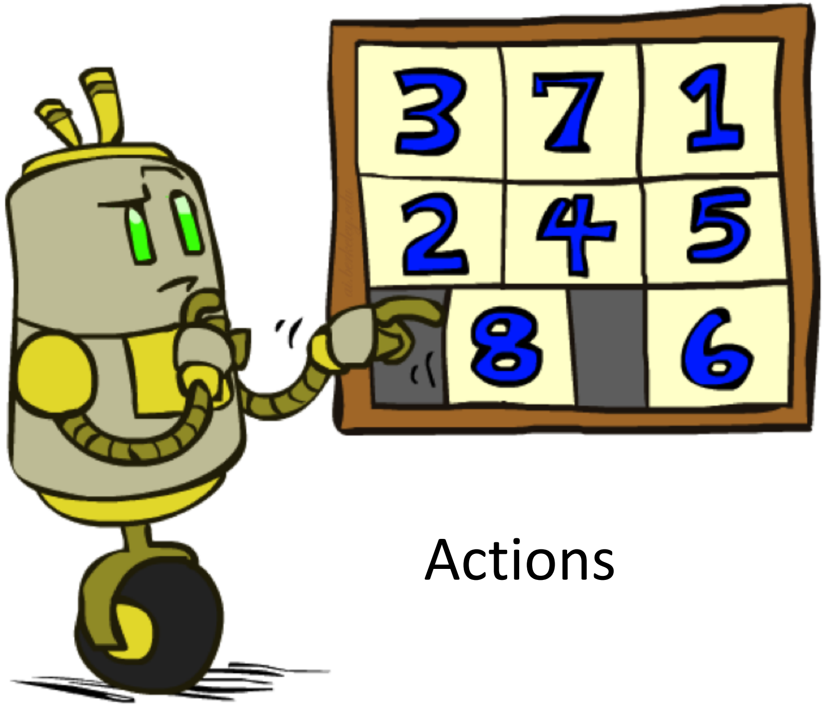


- Inadmissible heuristics are often useful too

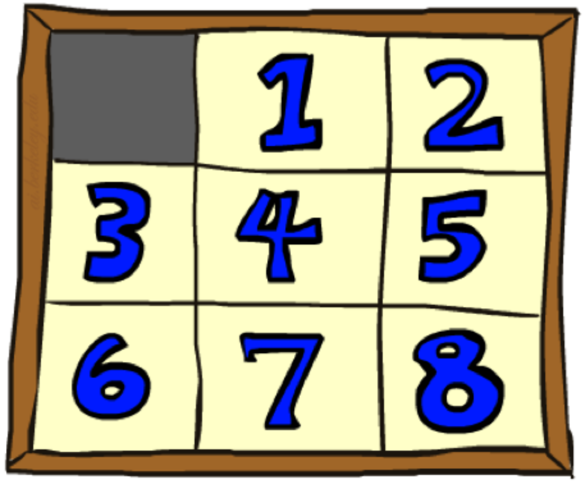
Example: 8 Puzzle



Start State



Actions



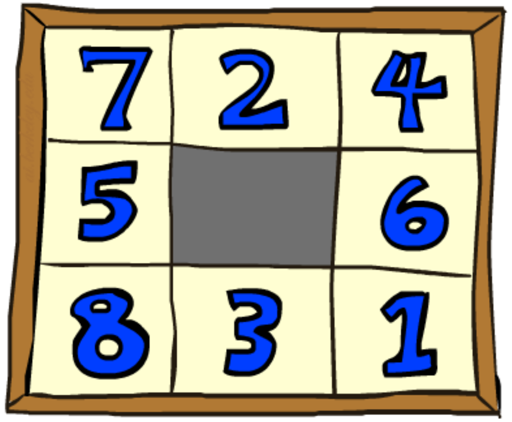
Goal State

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

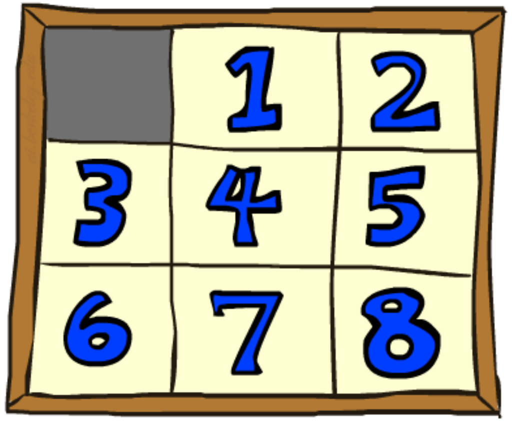
8 Puzzle heuristics

8 Puzzle I

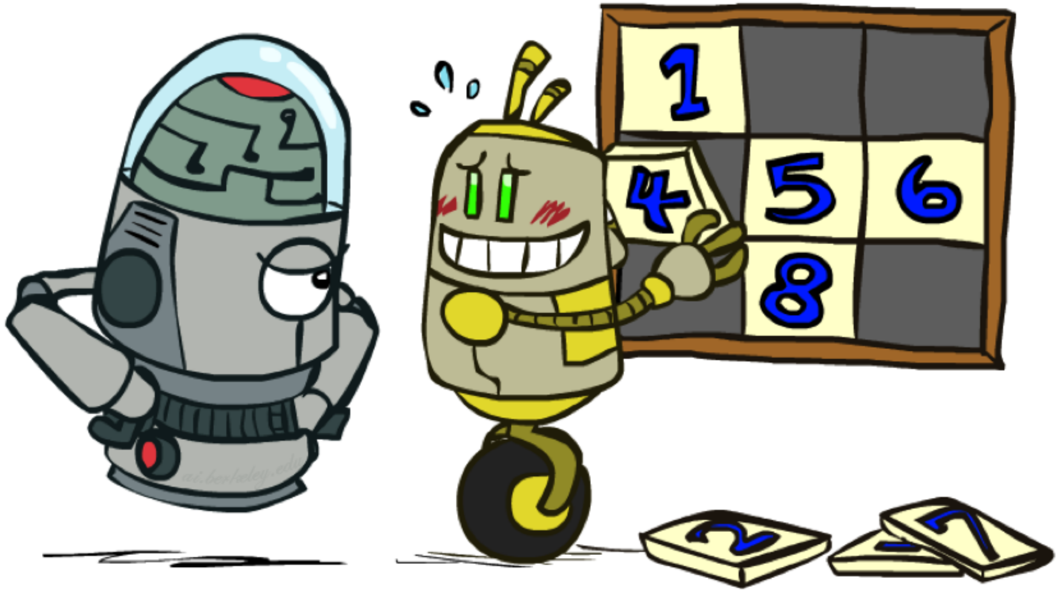
- Heuristic: Number of tiles misplaced
- Why is it admissible?
- $h_1(\text{start}) = 8$
- This is a *relaxed-problem* heuristic



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
TILES	13	39	227

Statistics from Andrew Moore

8 Puzzle heuristics

4	6	5
7	2	
1	3	8

admissible?

3	1	8
7	4	2
6	5	

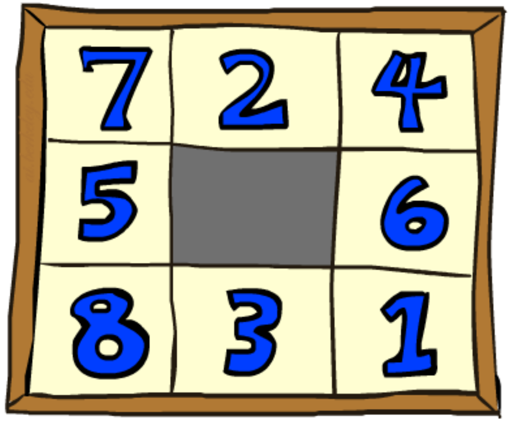
tiles misplaced

Tile	n	Goal
1	1	0
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0
7	0	0
8	1	0
h1(n)	**7**	
h1(goal)	**0**	

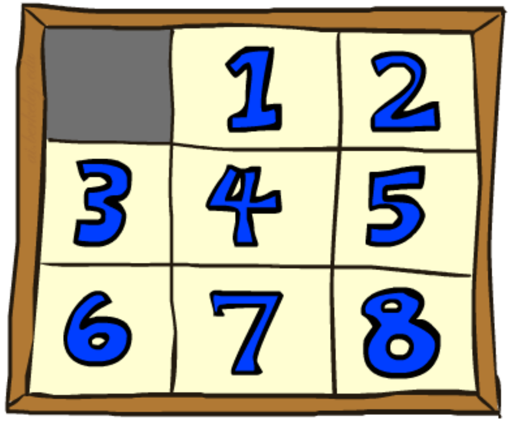
8 Puzzle heuristics

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
TILES	13	39	227
MANHATTAN	12	25	73

8 Puzzle heuristics

4	6	5
7	2	
1	3	8

admissible?

3	1	8
7	4	2
6	5	

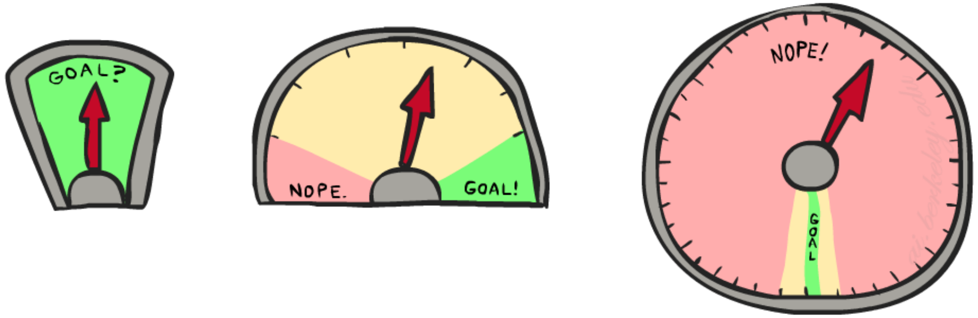
Manhattan **distance**

Tile	n	Goal
1	3	
2	1	
3	3	
4	2	
5	3	
6	3	
7	0	
8	2	
h_2(n)	**17**	
h_2(goal)	**0**	

8 Puzzle heuristics

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

8 Puzzle with A*

- Solving 8 Puzzle with A* using tree search

3	1	8
4	2	
7	6	5

start

3	1	8
7	4	2
6	5	

goal

- Heuristic function h(n)= the number of misplaced tiles.

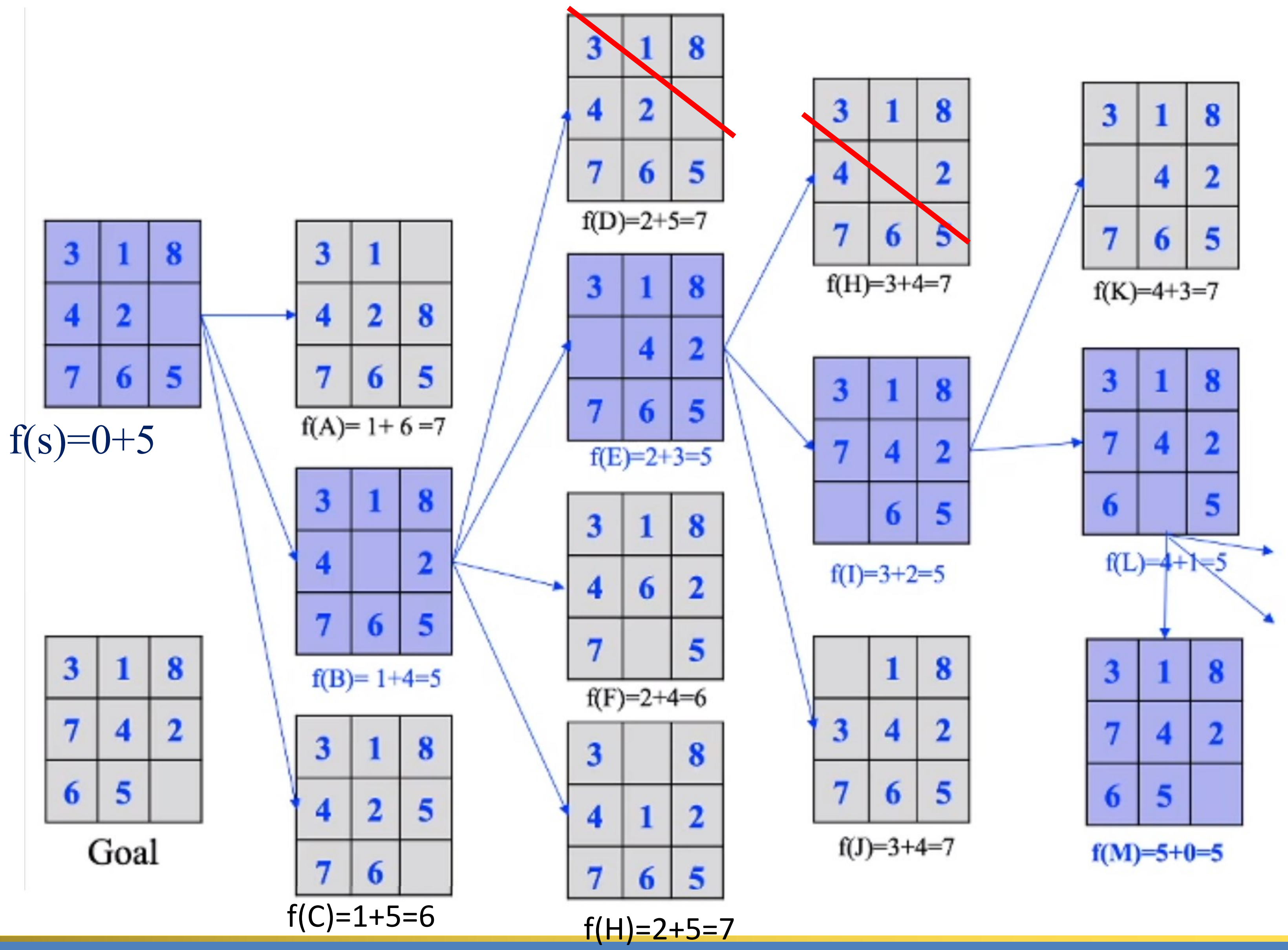
- Steps of A*

Application of A*

8 Puzzle with A*

- Heuristic function h(n)= the number of misplaced tiles.

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



Assignment

- Sử dụng A* để tìm lời giải cho bài toán 8-puzzle với hàm h tính theo số miếng số sai vị trí, và g tính theo số bước dịch chuyển.

3	1	8
7	4	2
6	5	

start

4	6	5
7	2	
1	3	8

goal

Assignment

- Sử dụng A* để tìm lời giải cho bài toán 8-puzzle với hàm h tính theo khoảng cách Manhattan miếng số sai vị trí và g tính theo số bước dịch chuyển.

3	1	8
7	4	2
6	5	

start

4	6	5
7	2	
1	3	8

goal

Thanks for your attention!

Q&A