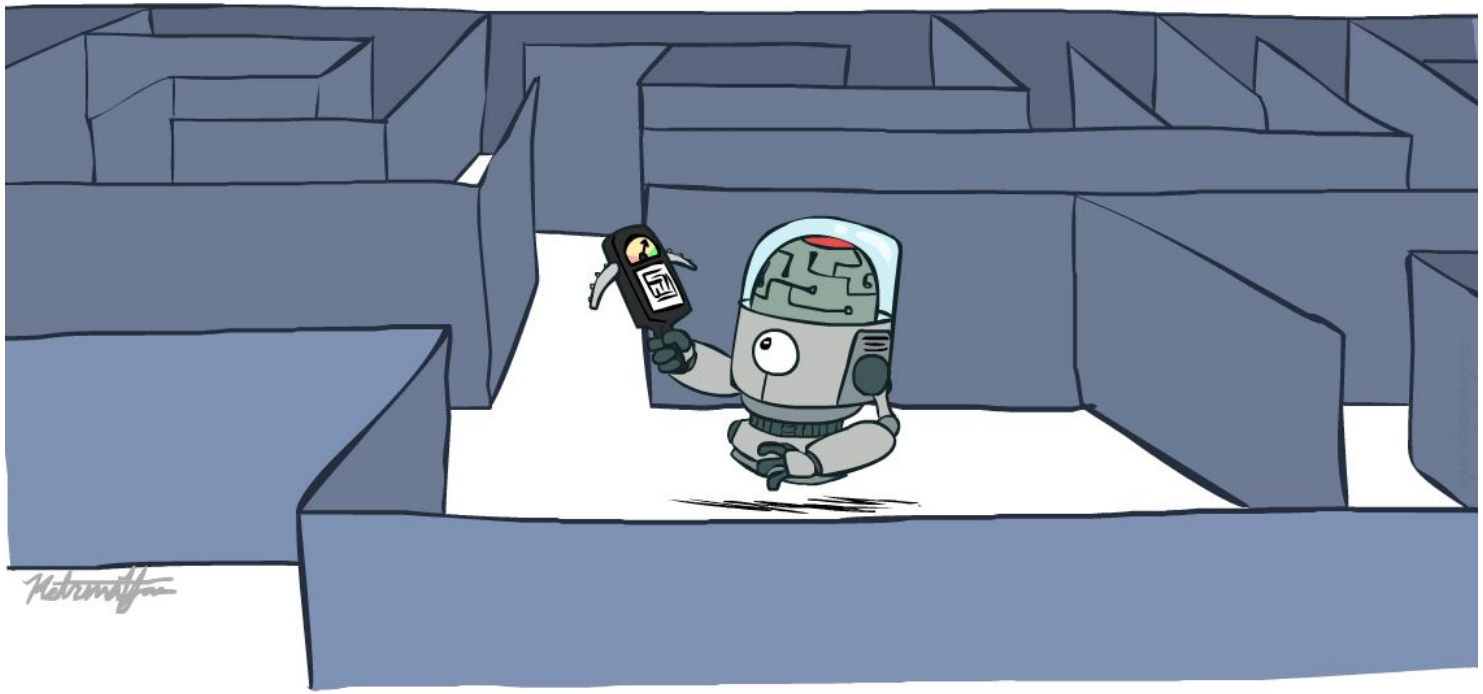
CS 188: Artificial Intelligence

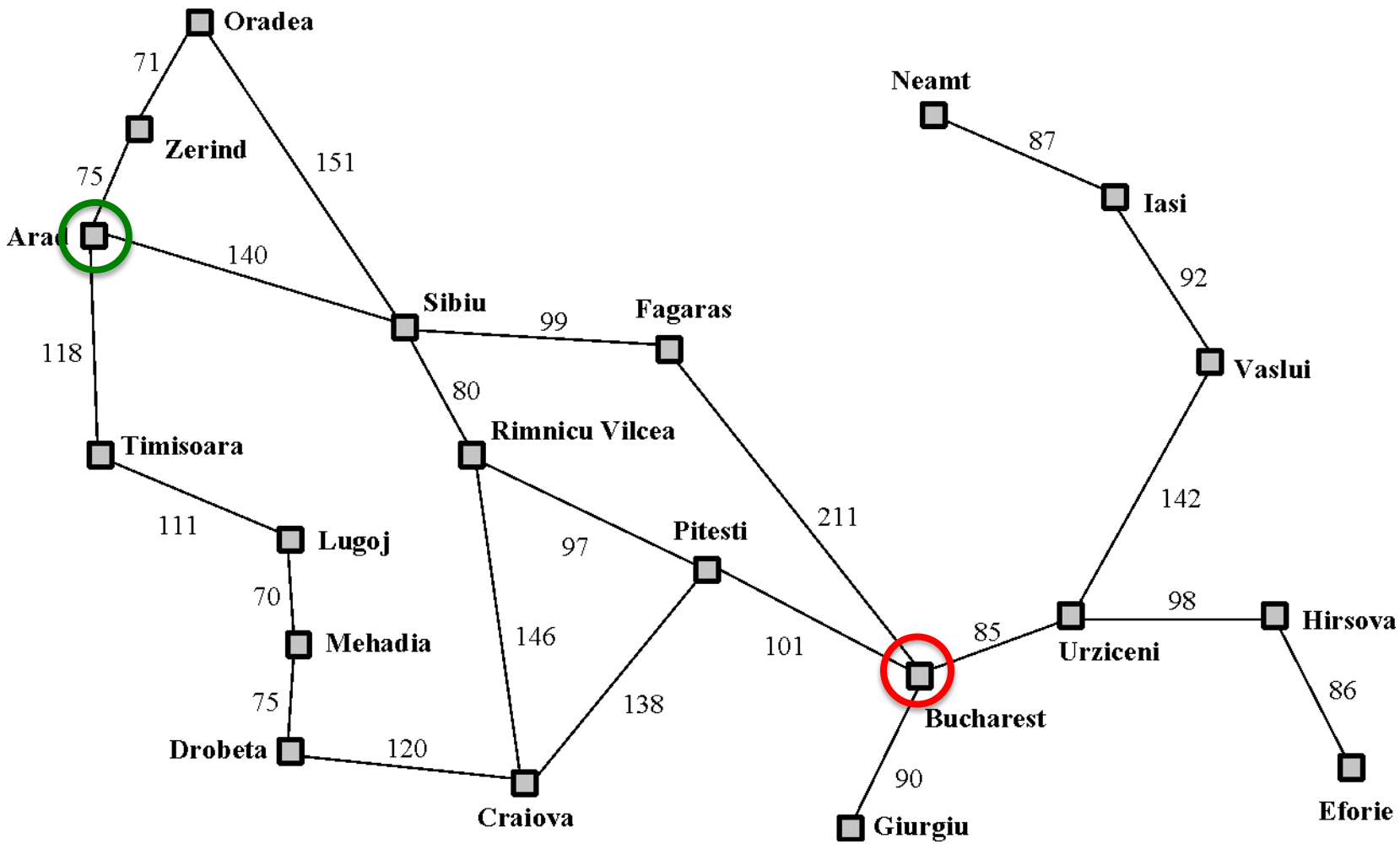
Informed Search



Instructors: Stuart Russell and Dawn Song

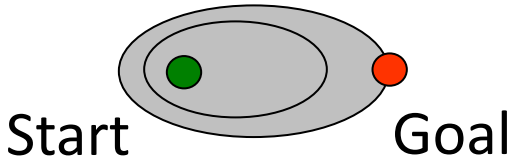
University of California, Berkeley

Example: route-finding in Romania

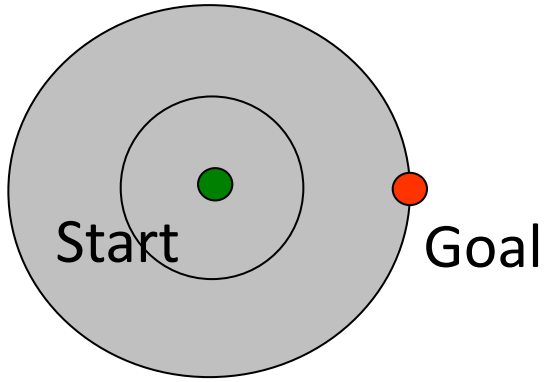


What we would like to have happen

Guide search ***towards the goal*** instead of ***all over the place***



Informed



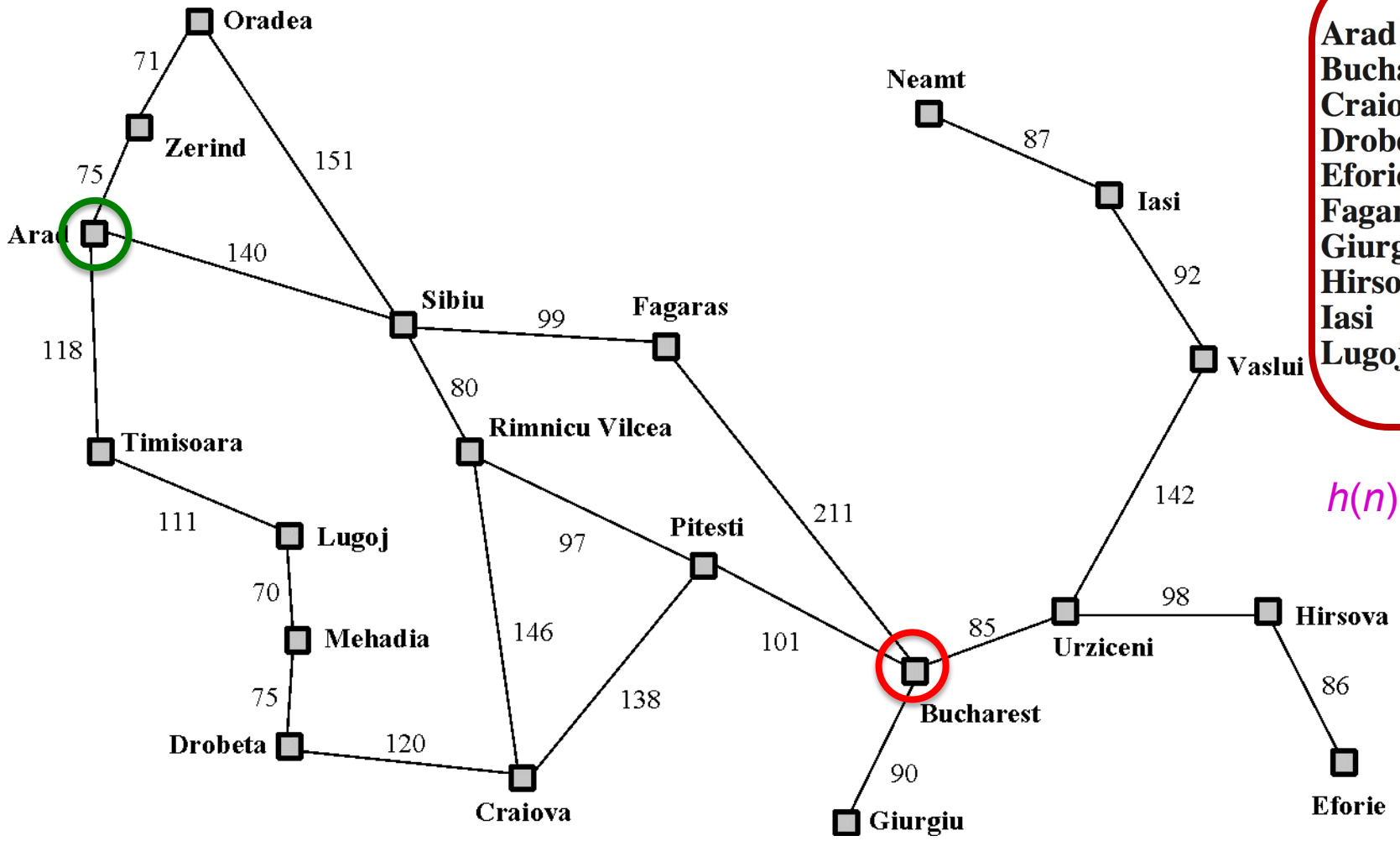
Uninformed

A* Search

A^*: the core idea

- Expand a node n most likely to be on the optimal path
- Expand a node n s.t. the cost of the best solution through n is optimal
- Expand a node n with lowest value of $g(n) + h^*(n)$
  - $g(n)$ is the cost from root to n
  - $h^*(n)$ is the optimal cost from n to the closest goal
- We seldom know $h^*(n)$ but might have a heuristic approximation $h(n)$
- A^* = tree search with priority queue ordered by $g(n) + h(n)$

Example: route-finding in Romania

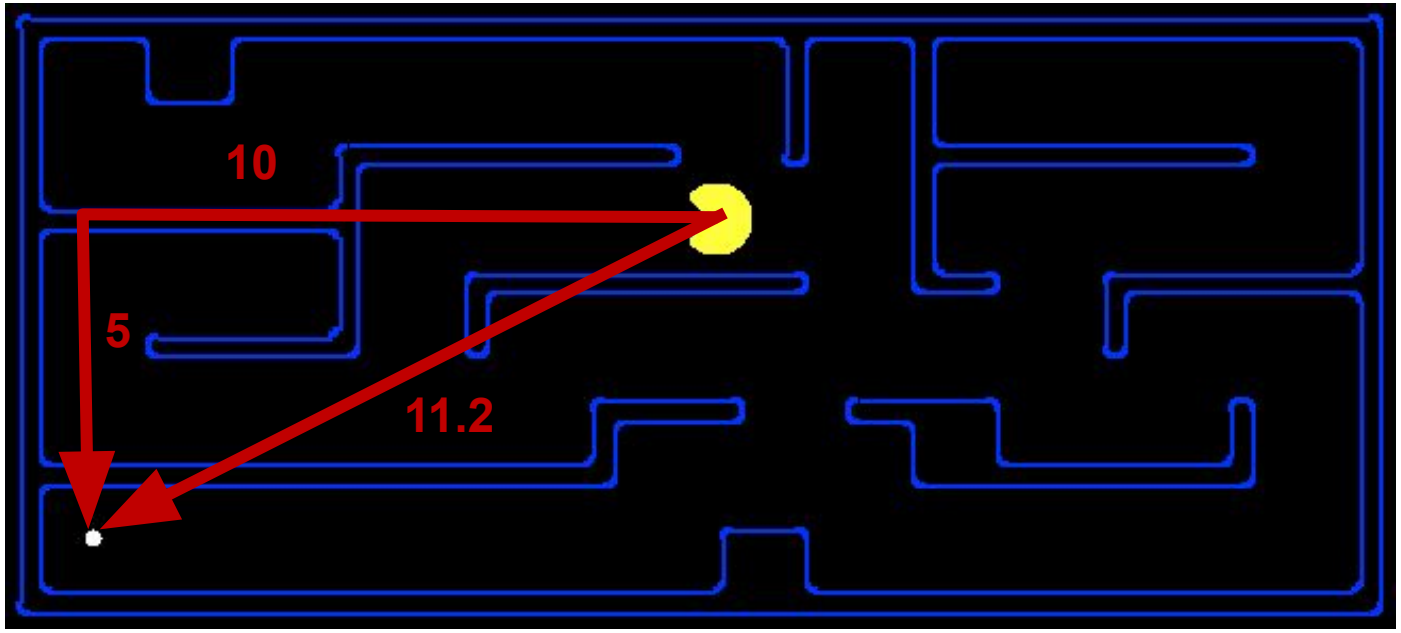


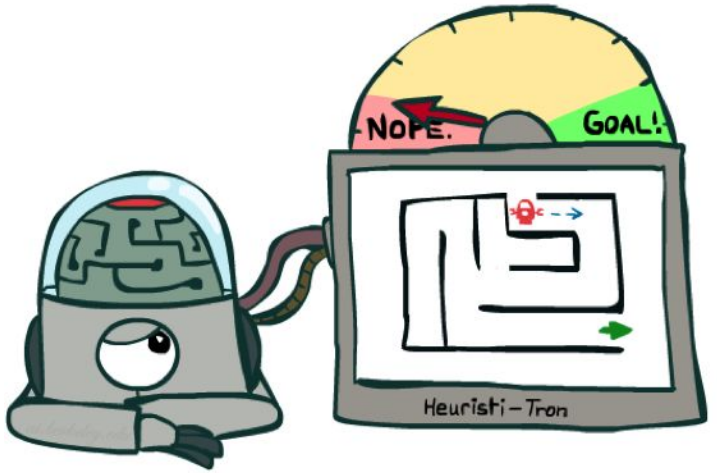
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

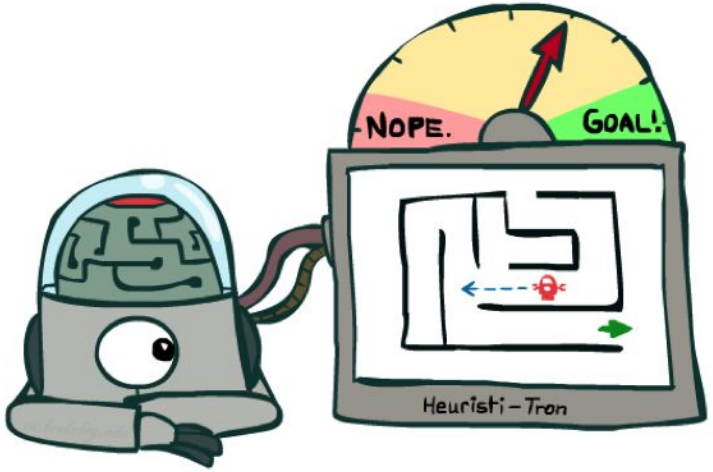
$h(n)$ = straight-line distance to Bucharest

Example: pathing in Pacman

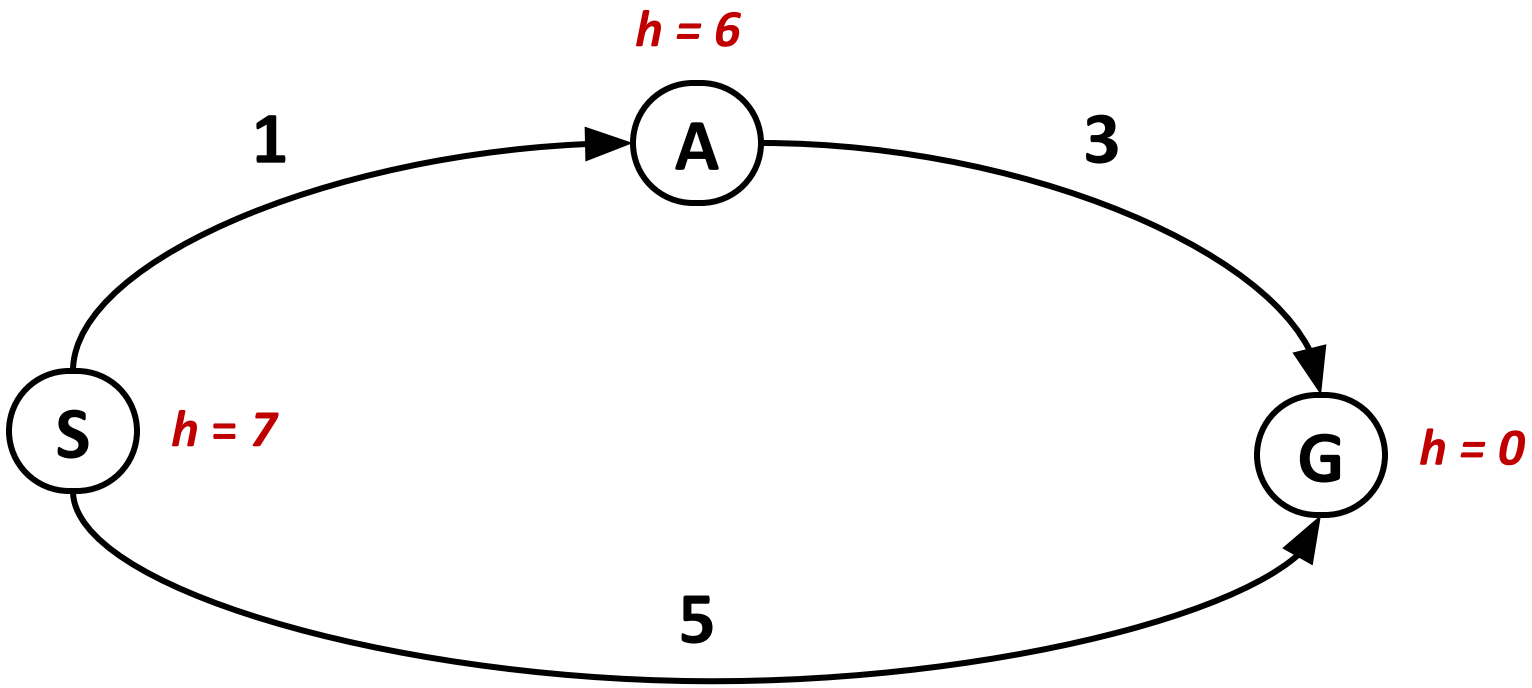
- $h(n)$ = Manhattan distance = $|\Delta x| + |\Delta y|$
- Is Manhattan better than straight-line distance?







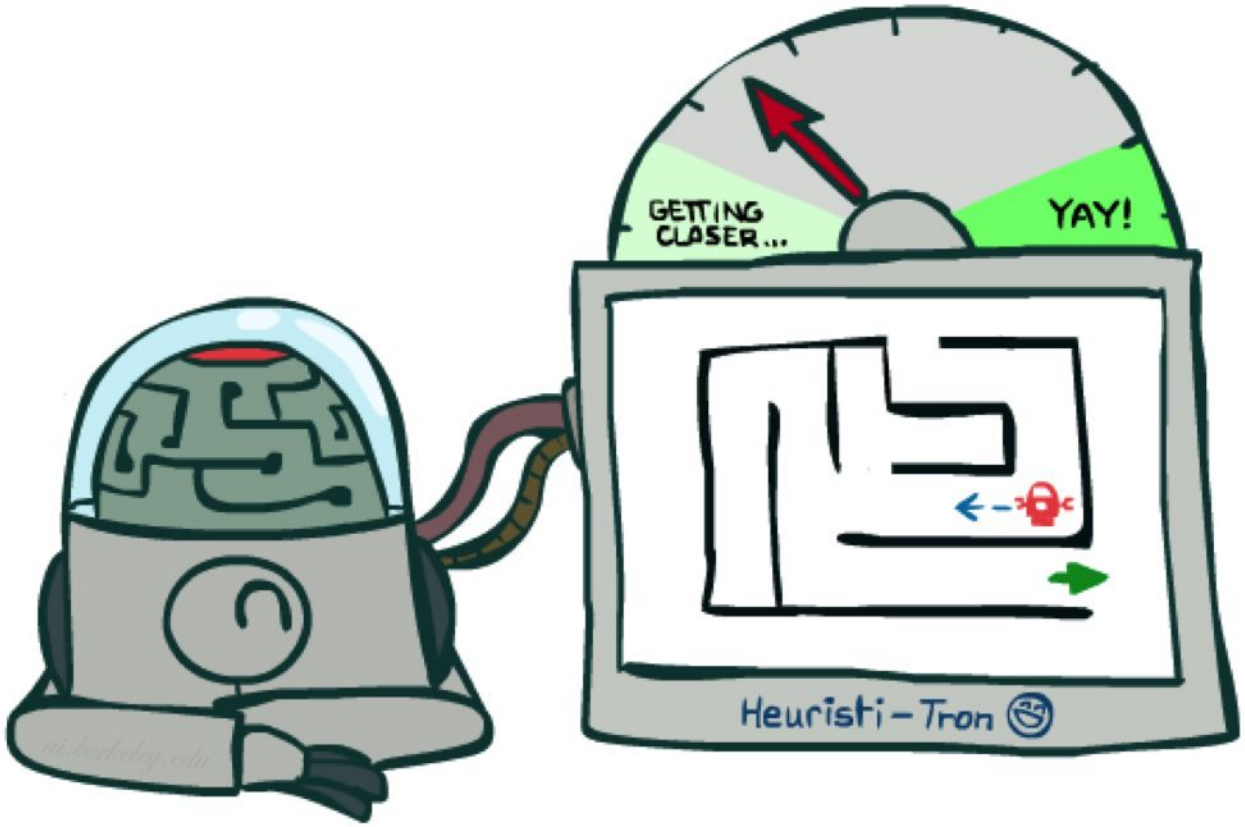
Is A* Optimal?



What went wrong?

- ***Actual*** bad solution cost < ***estimated*** good solution cost
- We need estimates to be less than actual costs!

Admissible Heuristics



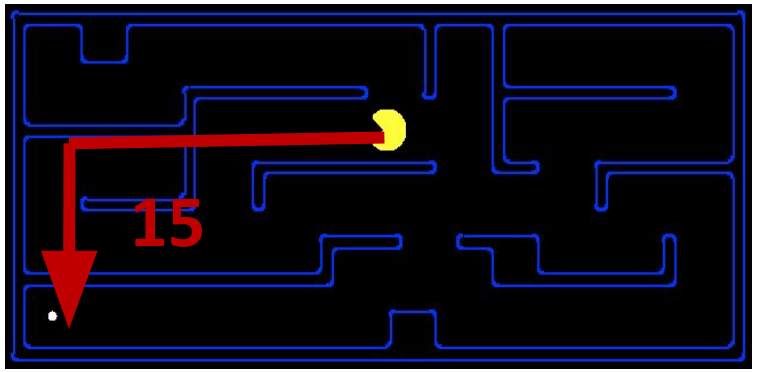
Admissible Heuristics

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

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

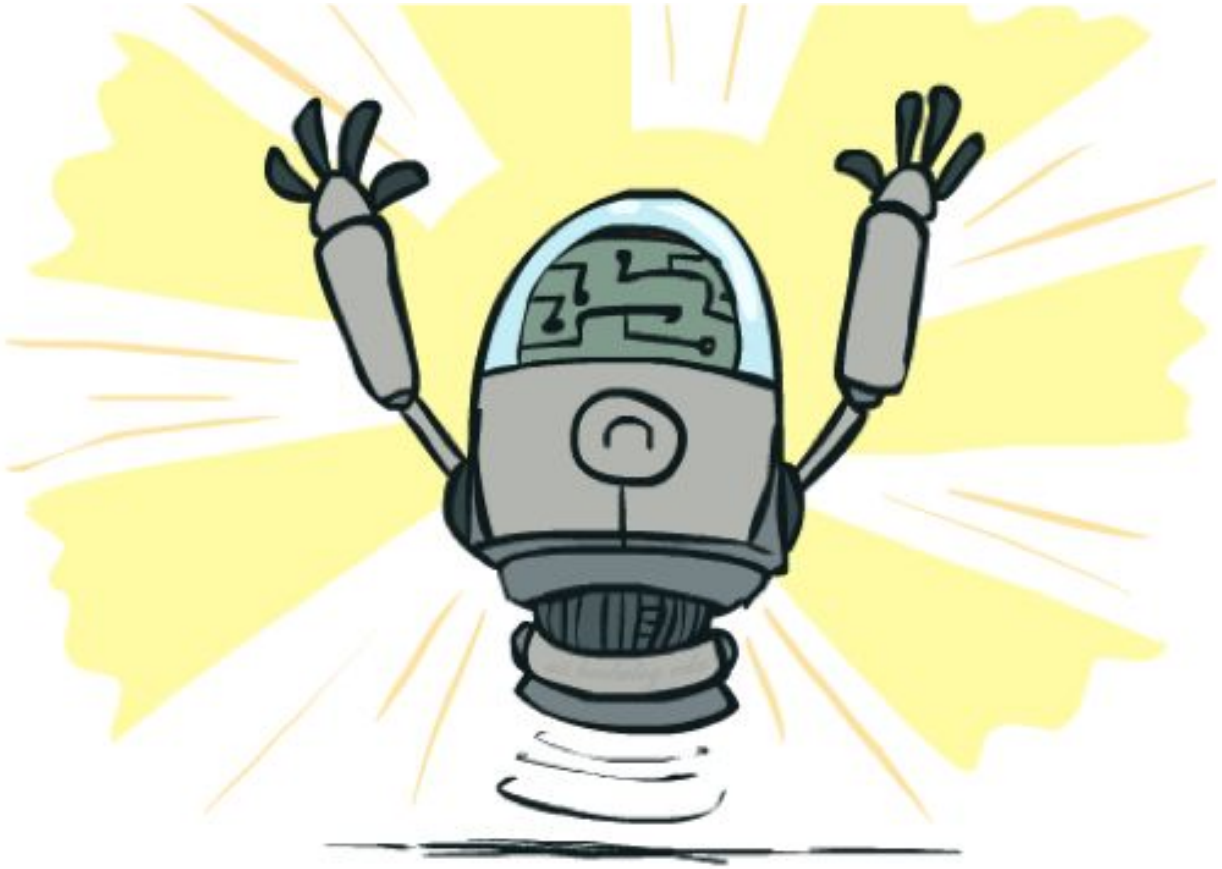
where $h^*(n)$ is the true cost to a nearest goal

- Example:



- Finding good, cheap admissible heuristics is the key to success

Optimality of A* Tree Search



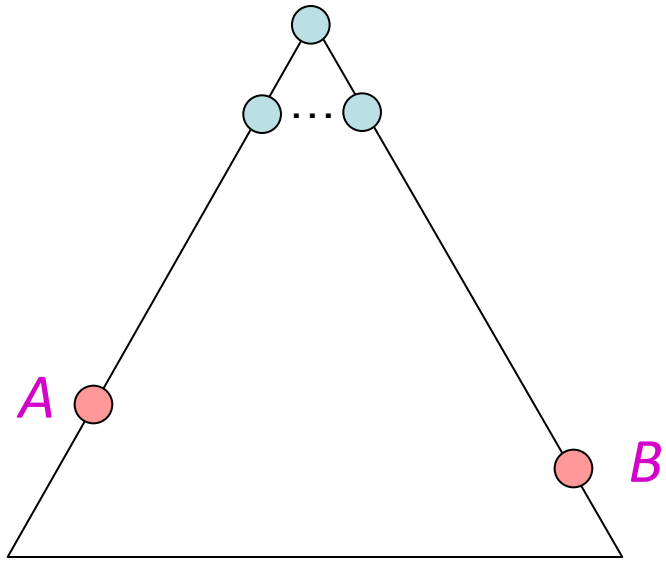
Optimality of A* Tree Search

Assume:

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

Claim:

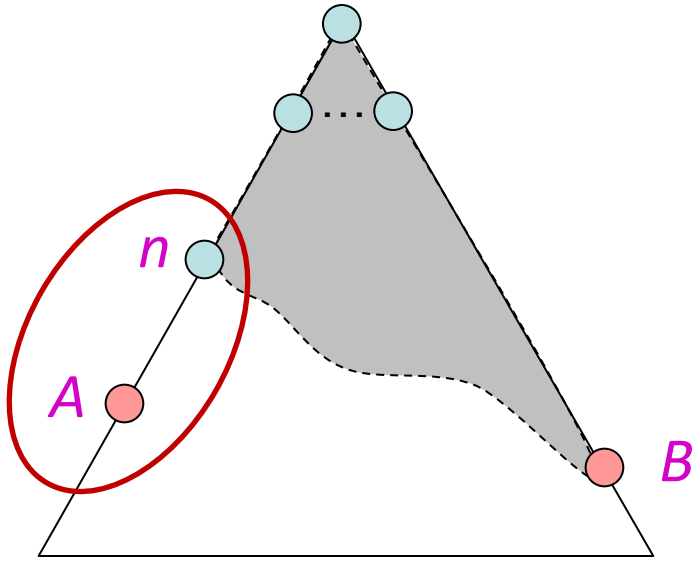
- *A* will be chosen for expansion before *B*



Optimality of A* Tree Search: Blocking

Proof:

- Imagine *B* is on the frontier
- Some ancestor *n* of *A* is on the frontier, too (maybe *A* itself!)
- Claim: *n* will be expanded before *B*
  1. $f(n)$ is less than or equal to $f(A)$



$f(n) = g(n) + h(n)$ Definition of *f*-cost

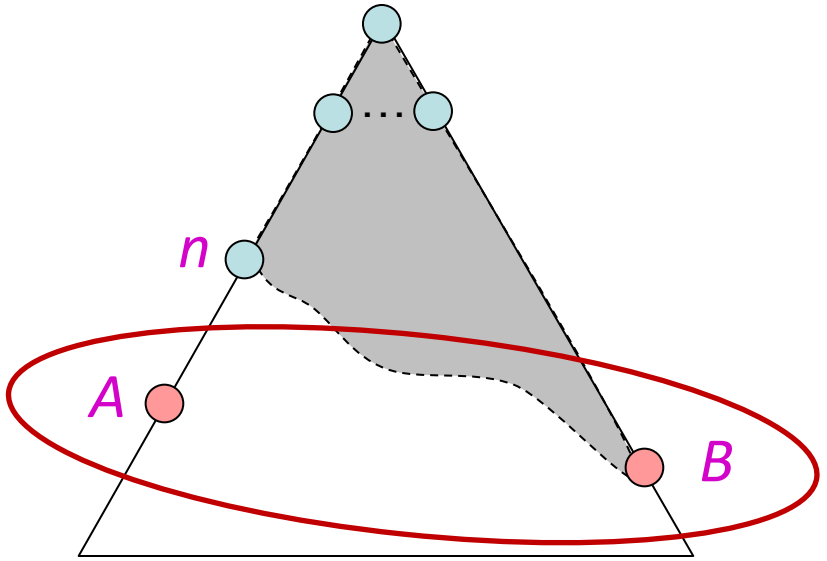
$f(n) \leq g(A)$ Admissibility of *h*

$g(A) = f(A)$ *h* = 0 at a goal

Optimality of A* Tree Search: Blocking

Proof:

- Imagine *B* is on the frontier
- Some ancestor *n* of *A* is on the frontier, too (maybe *A* itself!)
- Claim: *n* will be expanded before *B*
  1. $f(n)$ is less than or equal to $f(A)$
  2. $f(A)$ is less than $f(B)$



$g(A) < g(B)$

$f(A) < f(B)$

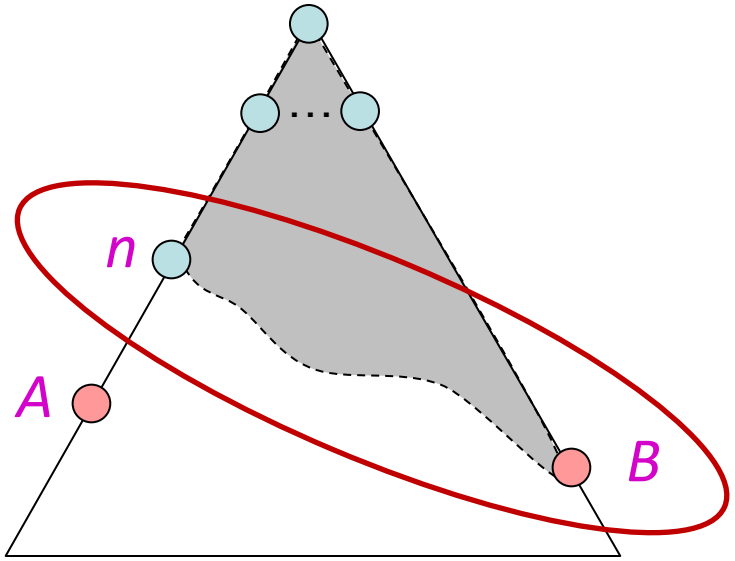
Suboptimality of *B*

$h = 0$ at a goal

Optimality of A* Tree Search: Blocking

Proof:

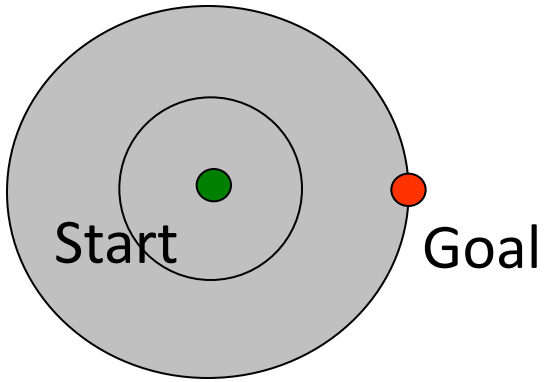
- Imagine *B* is on the frontier
- Some ancestor *n* of *A* is on the frontier, too (maybe *A* itself!)
- Claim: *n* will be expanded before *B*
  1. $f(n)$ is less than or equal to $f(A)$
  2. $f(A)$ is less than $f(B)$
  3. *n* is expanded before *B*
- All ancestors of *A* are expanded before *B*
- *A* is expanded before *B*
- **A* tree search is optimal**



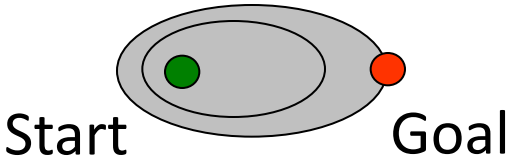
$$f(n) \leq f(A) < f(B)$$

UCS vs A* Contours

- Uniform-cost expands equally in all "directions"



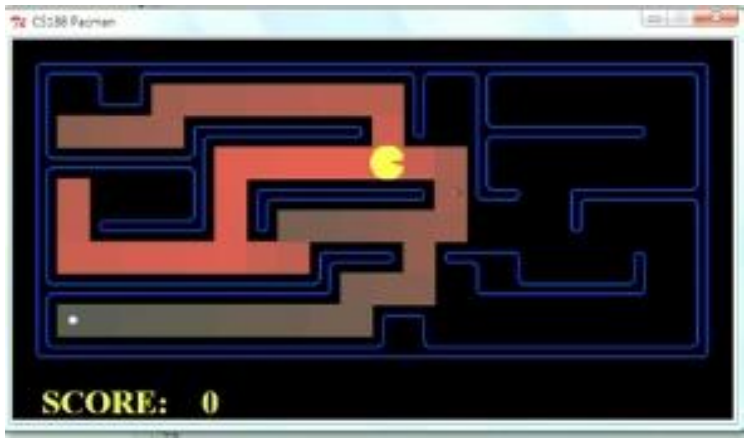
- A* expands mainly toward the goal, but does hedge its bets to ensure optimality



Comparison

Greedy (h)

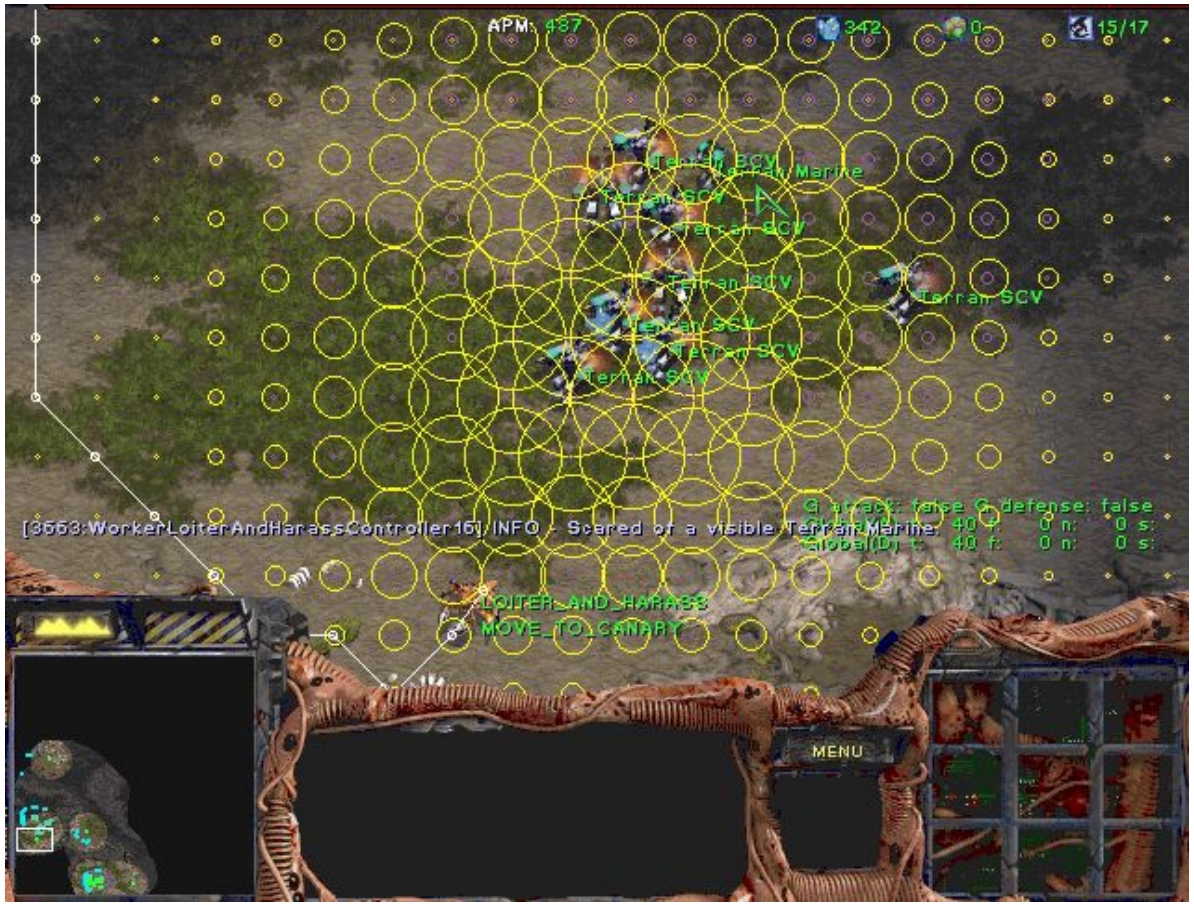
Uniform Cost (g)



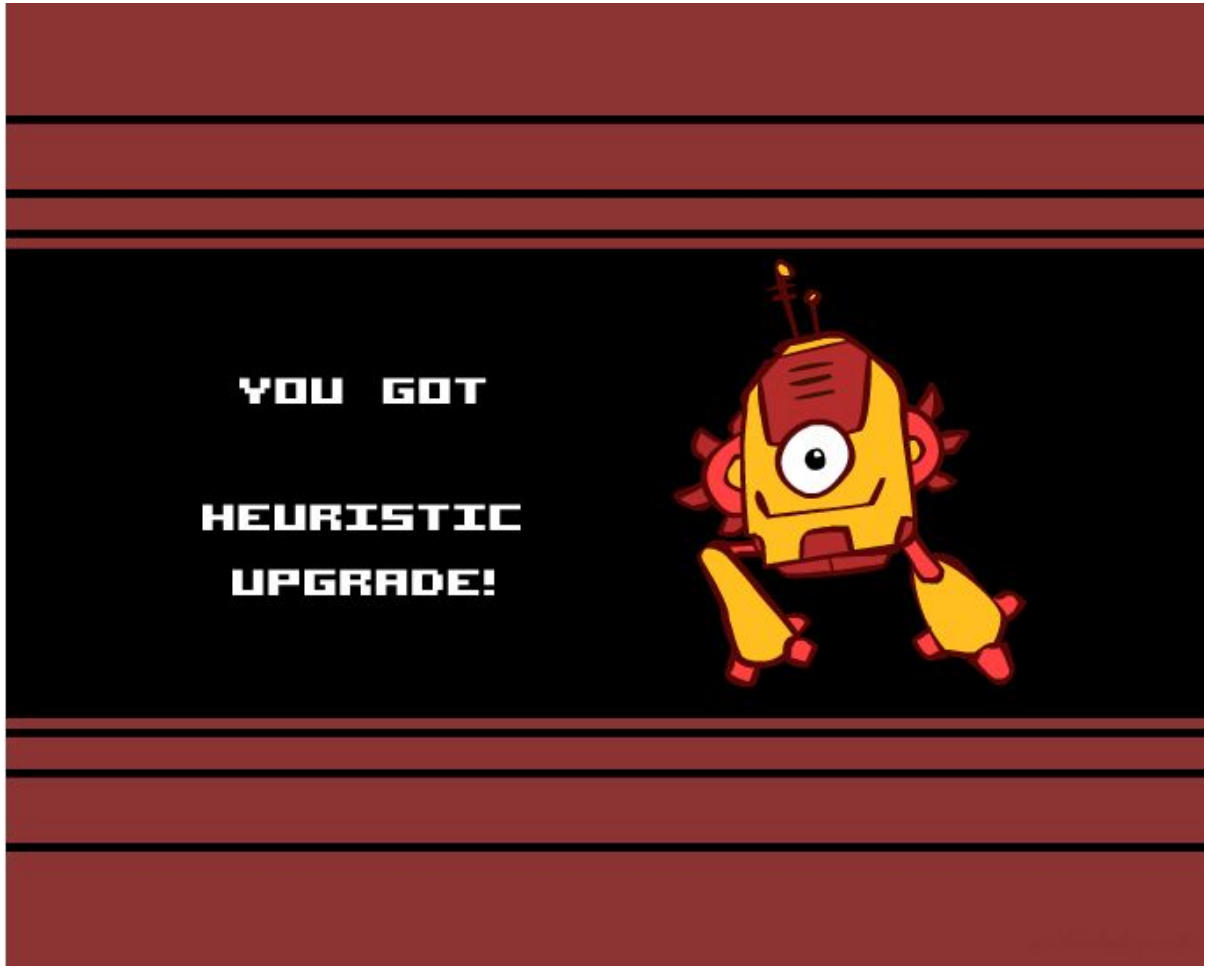
A* (g+h)

A* Applications

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



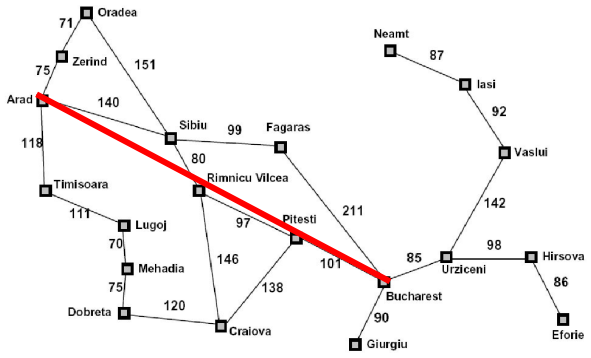
Creating Heuristics

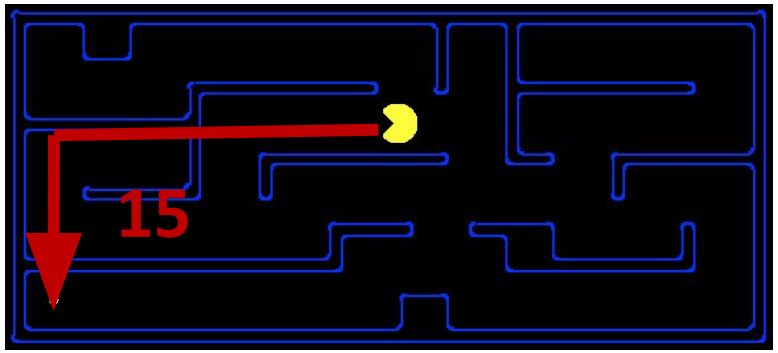


Creating Admissible Heuristics

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

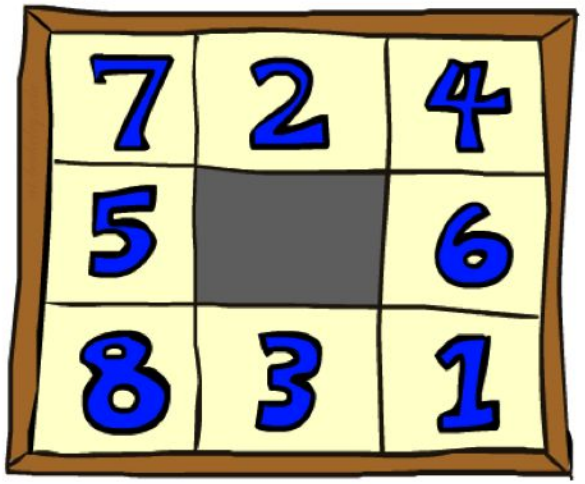
366



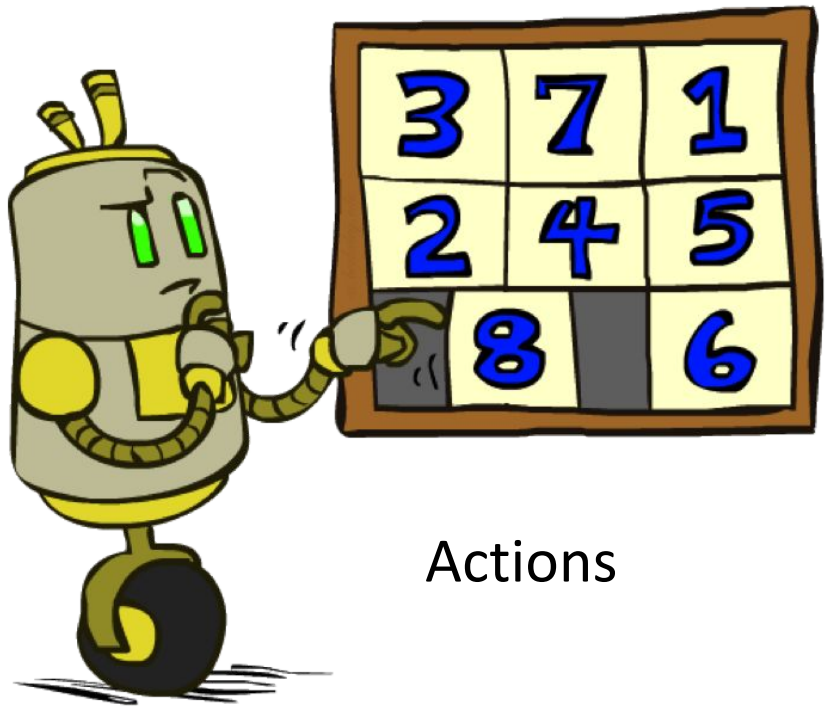


- Problem P_2 is a relaxed version of P_1 if $A_2(s) \supseteq A_1(s)$ for every s
- Theorem: $h_2^*(s) \leq h_1^*(s)$ for every s, so $h_2^*(s)$ is admissible for P_1

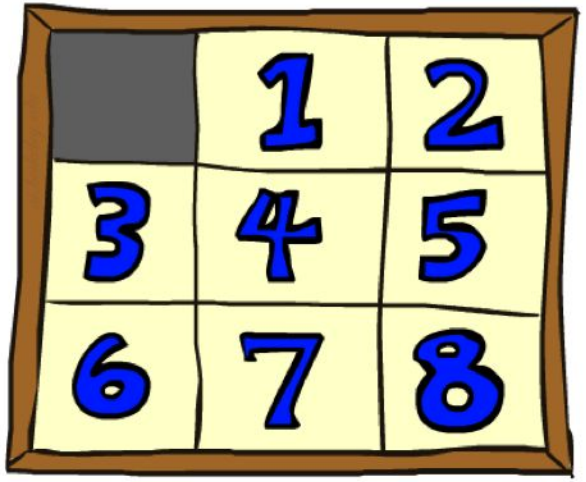
Example: 8 Puzzle



Start State



Actions

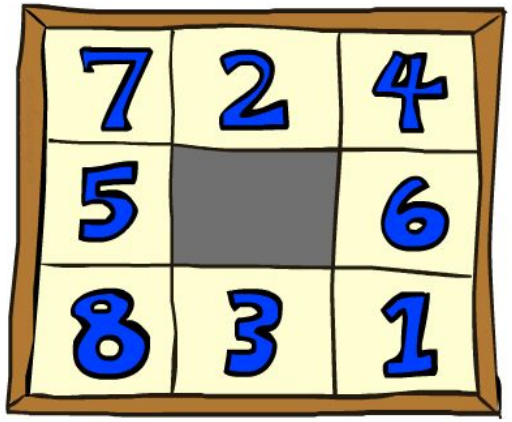


Goal State

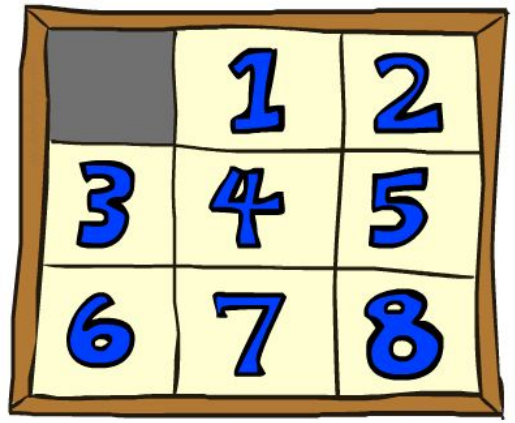
- What are the states?
- How many states?
- What are the actions?
- What are the step costs?

8 Puzzle I

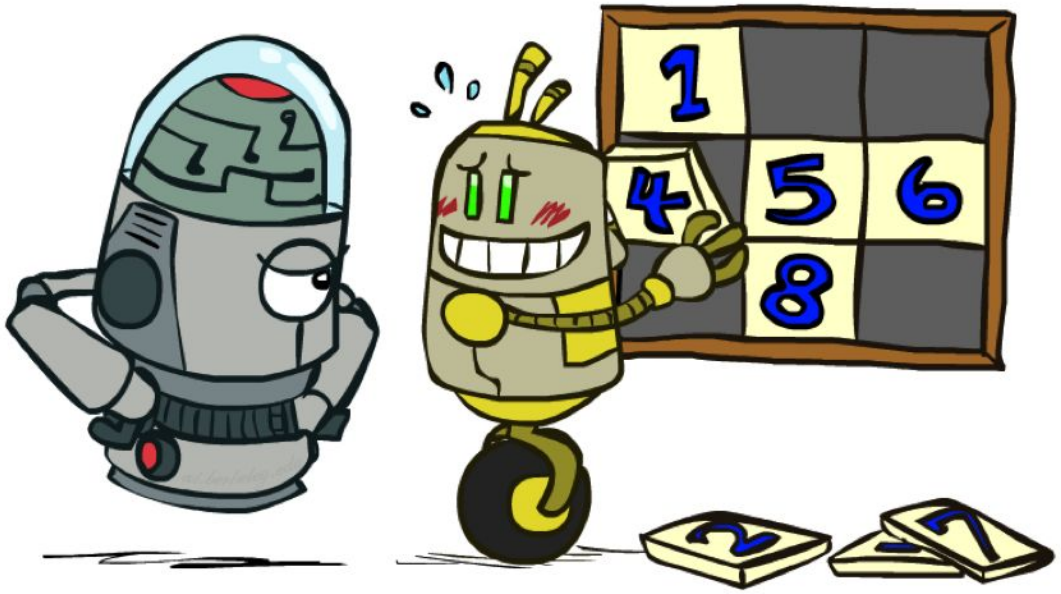
- Heuristic: Number of tiles misplaced
- Why is it admissible?
- h(start) = 8



Start State



Goal State

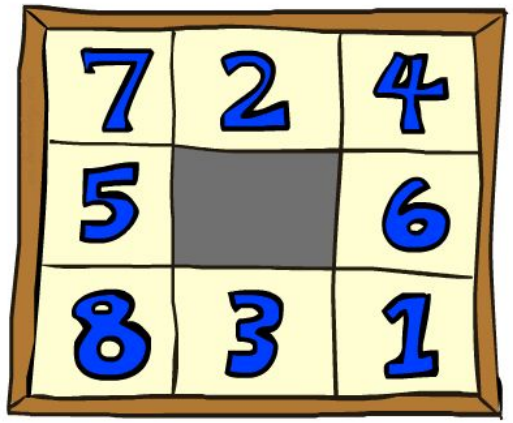


	Average nodes expanded when the optimal path has…		
	…4 steps	…8 steps	…12 steps
UCS	112	6,300	3.6×10^6
A*TILES	13	39	227

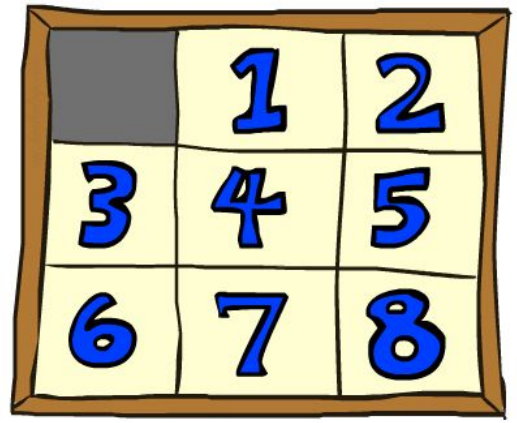
Statistics from Andrew Moore

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
A*TILES	13	39	227
A*MANHATTAN	12	25	73

Combining heuristics

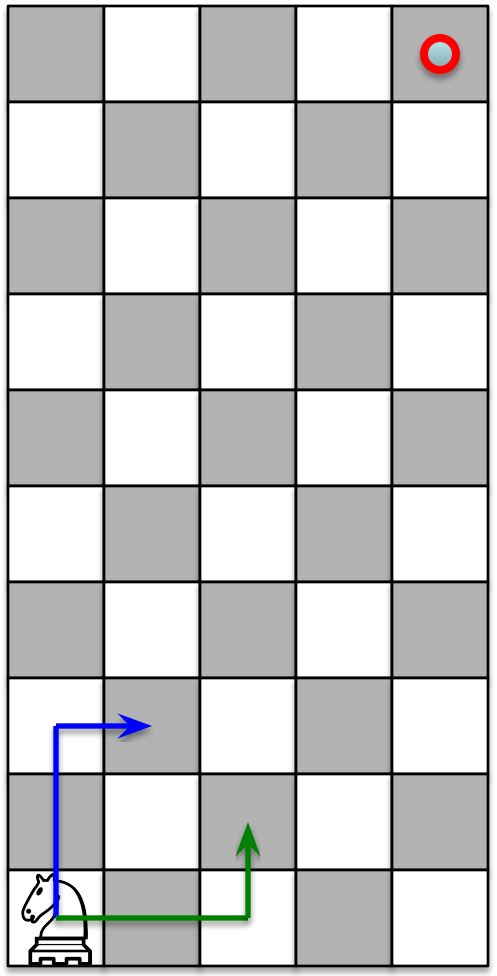
- Dominance: $h_1 \geq h_2$ if

  $$\forall n \;\; h_1(n) \geq h_2(n)$$

  - Roughly speaking, larger is better as long as both are admissible
  - The zero heuristic is pretty bad (what does A* do with h=0?)
  - The exact heuristic is pretty good, but usually too expensive!
- What if we have two heuristics, neither dominates the other?
  - Form a new heuristic by taking the max of both:

    $$h(n) = \max(\, h_1(n), h_2(n))$$

  - Max of admissible heuristics is admissible and dominates both!
  - Example: number of knight's moves to get from A to B
    - h1 = (Manhattan distance)/3 (rounded up to correct parity)
    - h2 = (Euclidean distance)/√5 (rounded up to correct parity)
    - h3 = (max x or y shift)/2 (rounded up to correct parity)

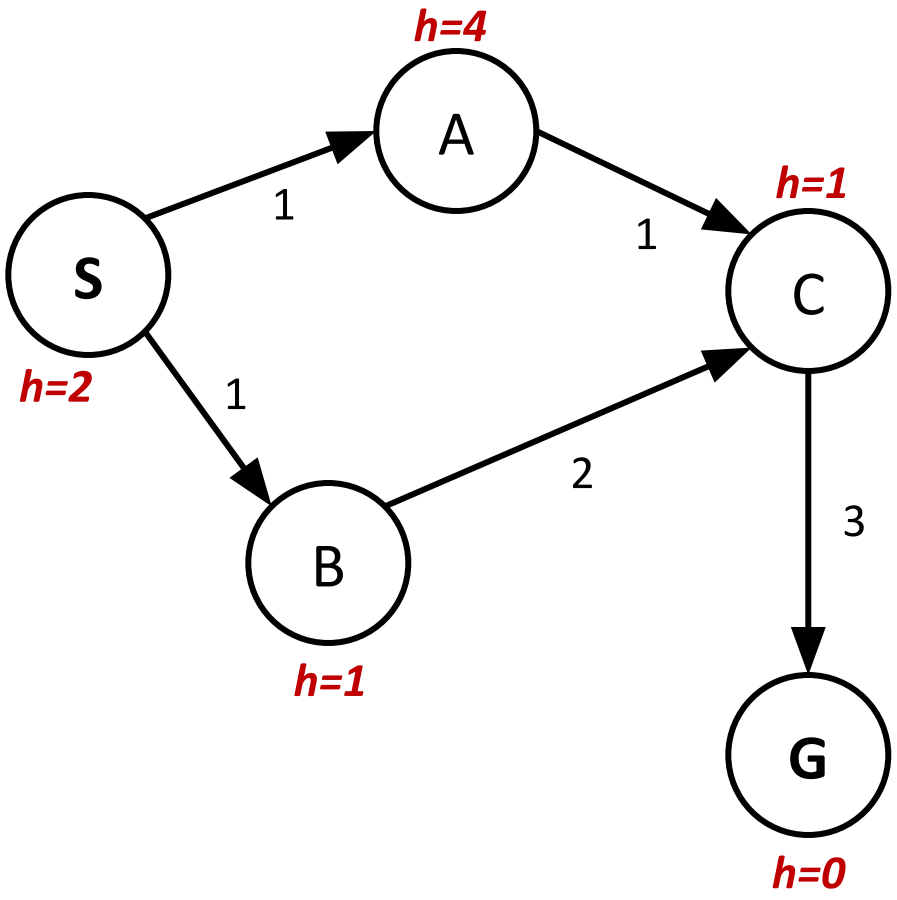


Optimality of A* Graph Search

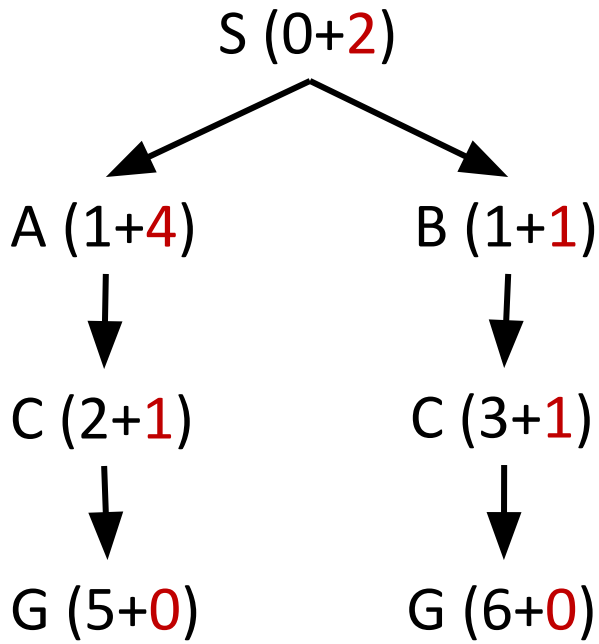
This part is a bit fiddly, sorry about that

A* Graph Search Gone Wrong?

State space graph



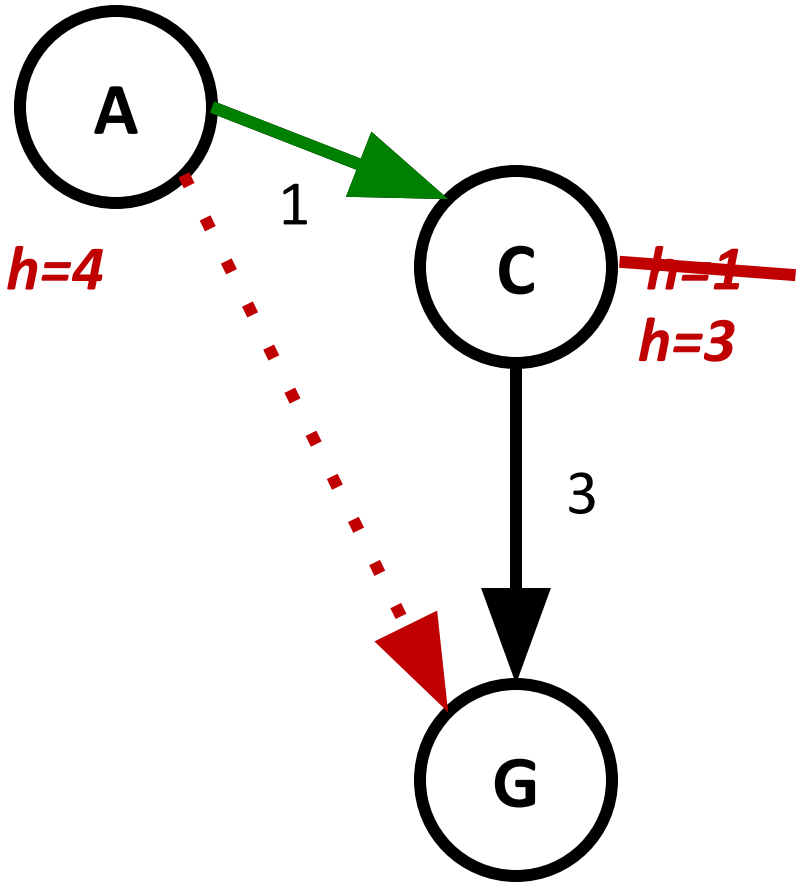
Search tree



Simple check against expanded set blocks C

Fancy check allows new C if cheaper than old but requires recalculating C's descendants

Consistency of Heuristics



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

    $h(A) \leq h^*(A)$
  - Consistency: heuristic "arc" cost ≤ actual cost for each arc

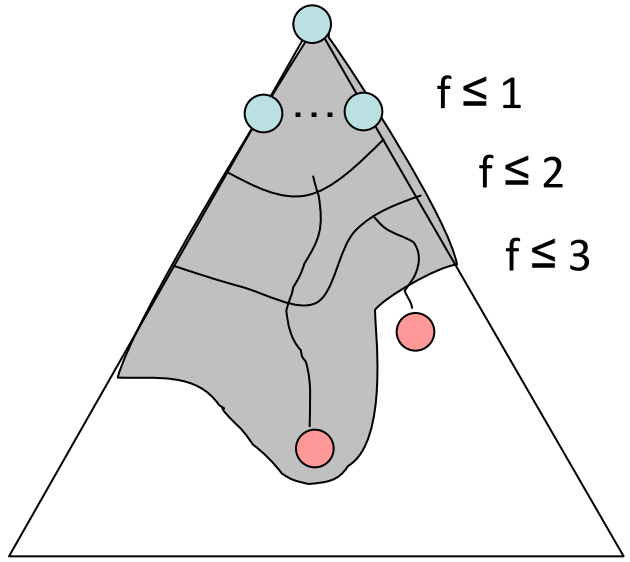
    $h(A) - h(C) \leq c(A,C)$

  or $h(A) \leq c(A,C) + h(C)$ (triangle inequality)
- Consequences of consistency:
  - The *f* value along a path never decreases:

    $h(A) \leq c(A,C) + h(C) \Rightarrow g(A) + h(A) \leq g(A) + c(A,C) + h(C)$
  - A* graph search is optimal

Optimality of A* Graph Search

- Sketch: consider what A* does with a consistent heuristic:
  - Fact 1: In tree search, A* expands nodes in increasing total f value (f-contours)
  - Fact 2: For every state s, nodes that reach s optimally are expanded before nodes that reach s suboptimally
  - Result: A* graph search is optimal



Optimality

- Tree search:
  - A* is optimal if heuristic is admissible
- Graph search:
  - A* optimal if heuristic is consistent
- Consistency implies admissibility
- Most natural admissible heuristics tend to be consistent, especially if from relaxed problems