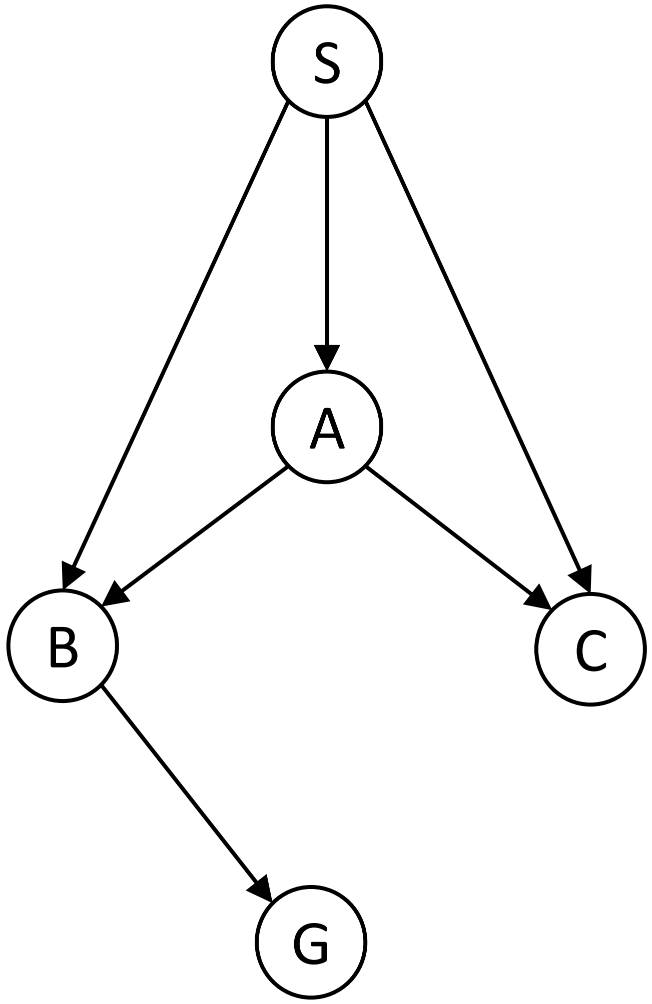
Warm-up: DFS Graph Search

Why is the answer S->B->G, not S->A->B->G?

After all, we were doing DFS and breaking ties alphabetically.



Warm-up: DFS Graph Search

Why is the answer S->B->G, not S->A->B->G?

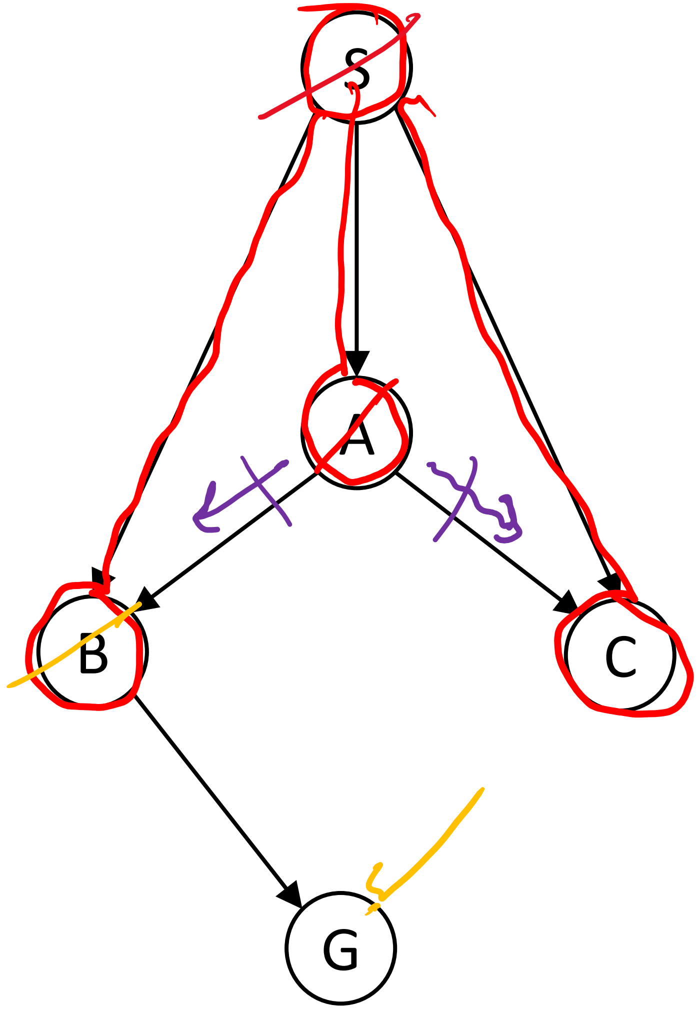
After all, we were doing DFS and breaking ties alphabetically.

Exp: S A B

Front: ~~S~~ ~~S-A~~ S-B S-C

~~S-A-B~~ ~~S-A-C~~

~~S-B-G~~



Plan

Last time

- Tree search vs graph search

Today

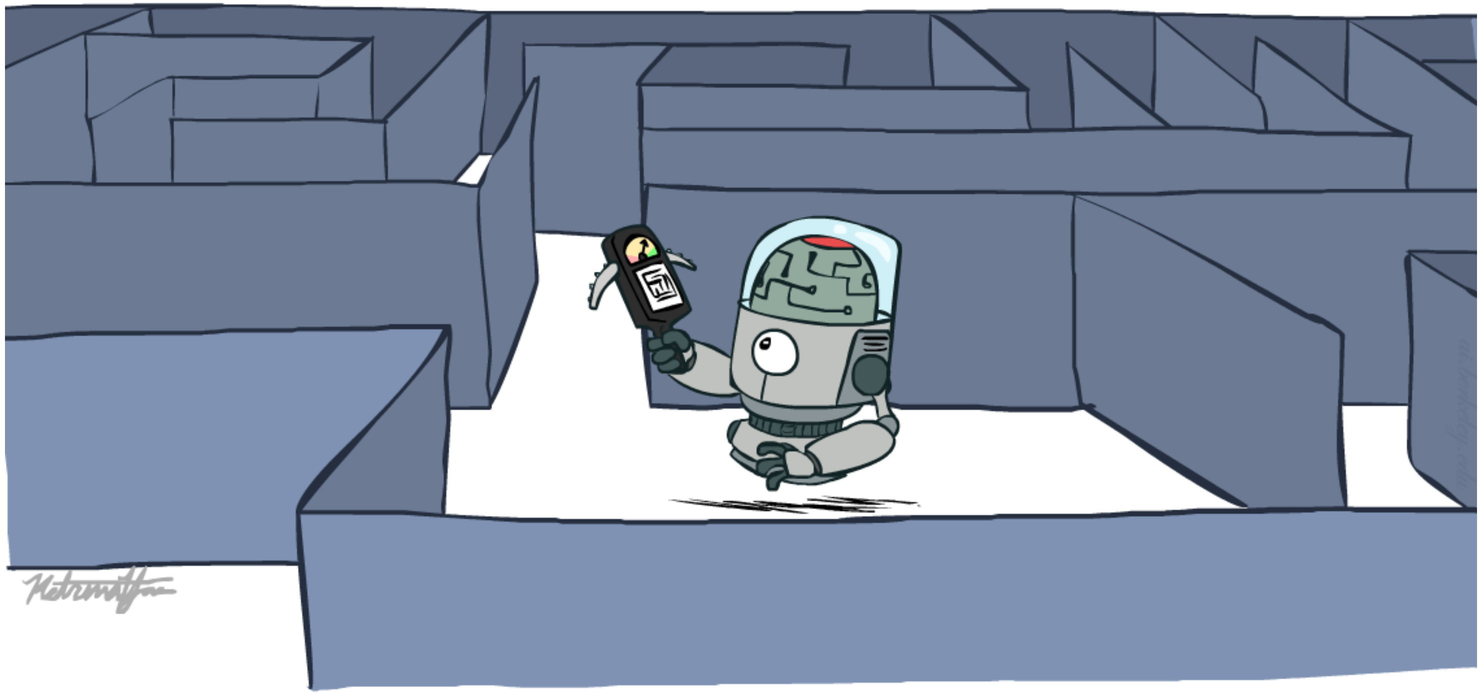
- Uniform cost search
- Heuristics
- Greedy search
- A* search
  - Optimality
- [More on heuristics]

Uniform Cost Search

Back to Lecture 2 slides

AI: Representation and Problem Solving

Informed Search

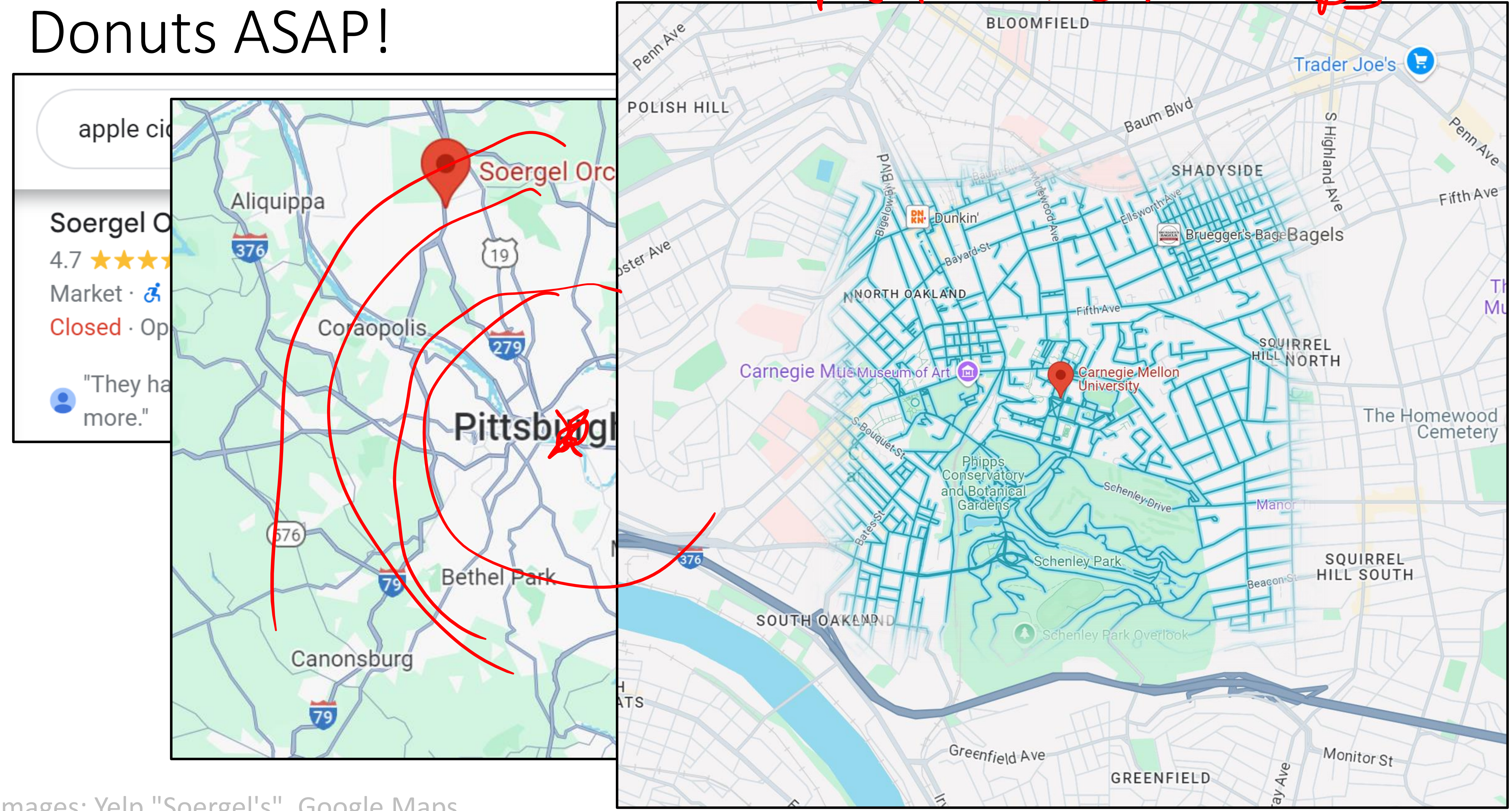


Instructor: Pat Virtue

Slide credits: CMU AI, http://ai.berkeley.edu

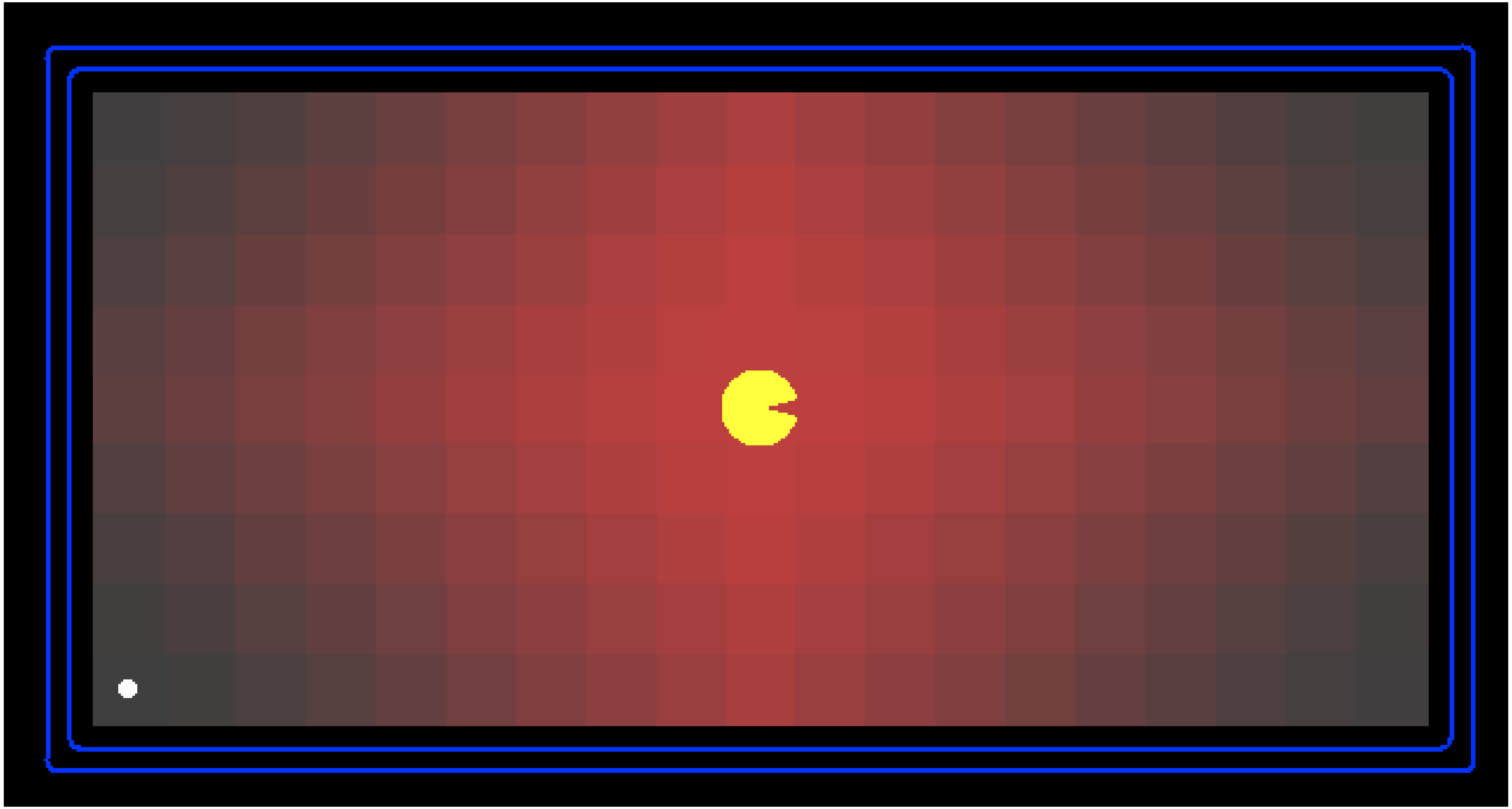
Donuts ASAP!

Open Street Maps

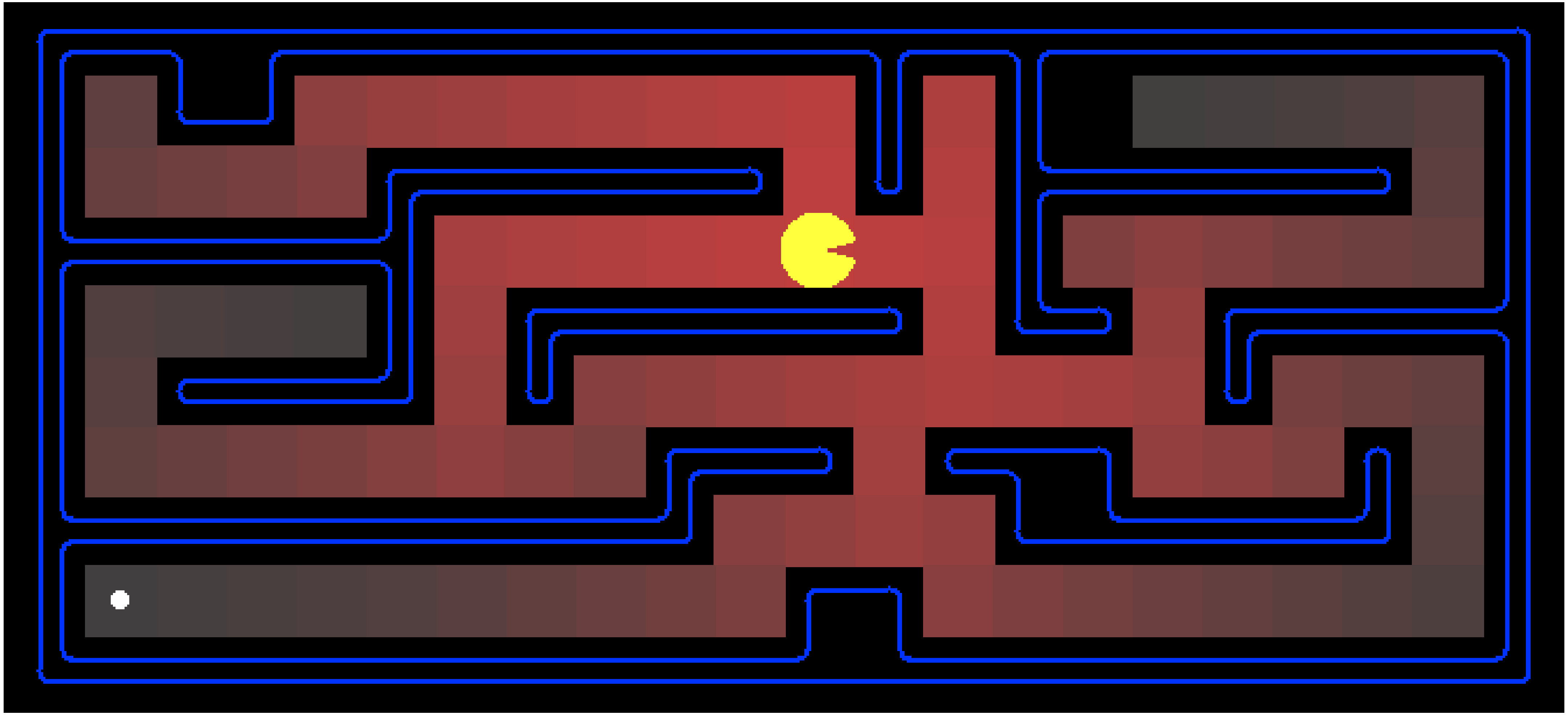


Images: Yelp "Soergel's", Google Maps

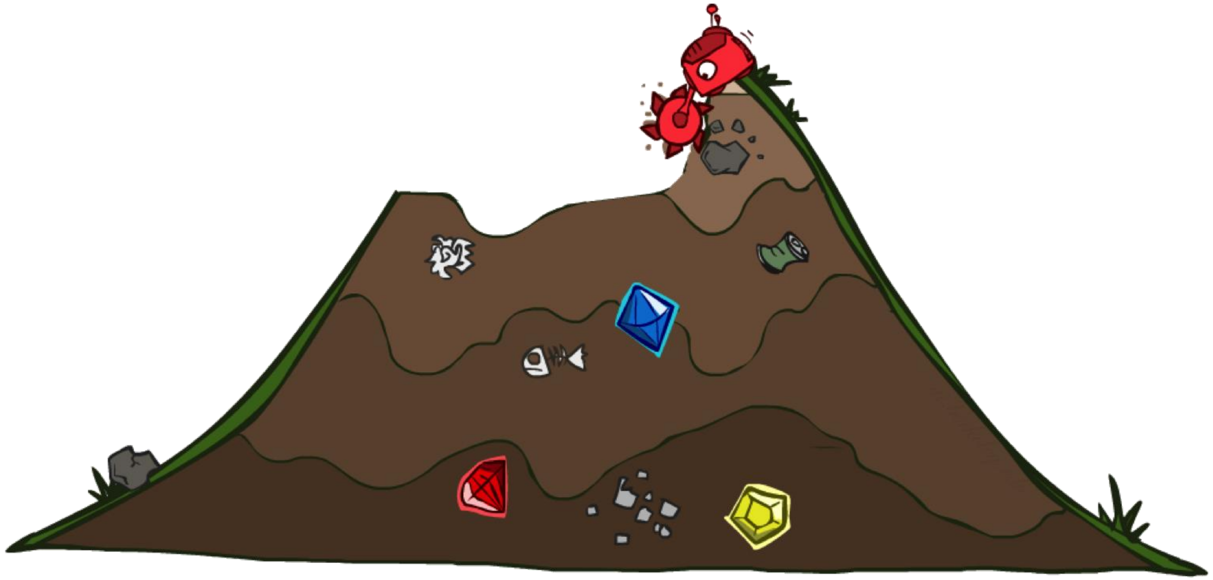
Demo Contours UCS Empty



Demo Contours UCS Pacman Small Maze



Uninformed vs Informed Search



Today

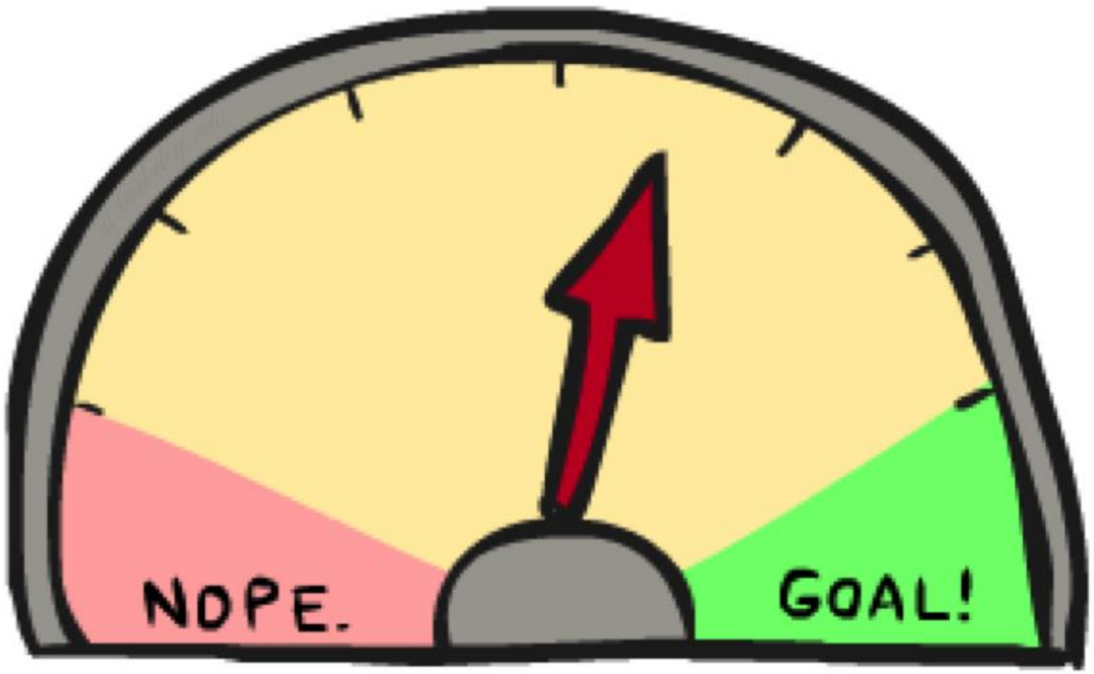
Informed Search

- Heuristics
- Greedy Search
- A* Search

$h(x)$

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

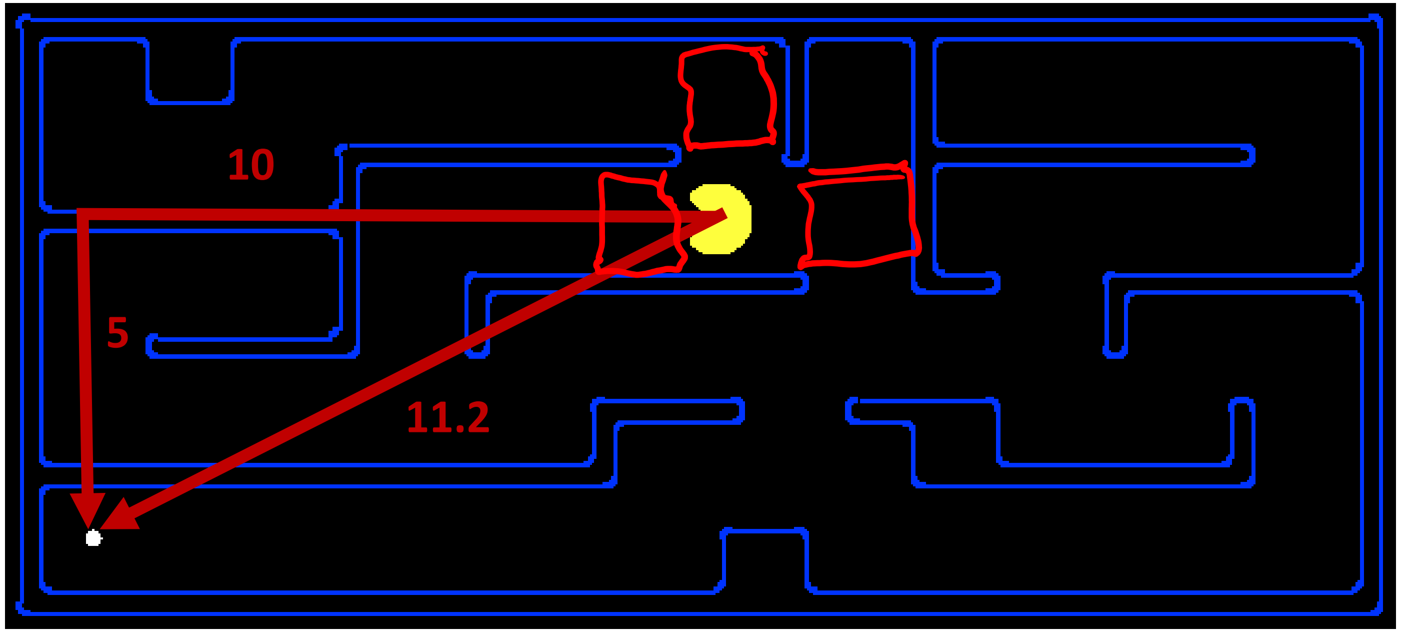
Informed Search

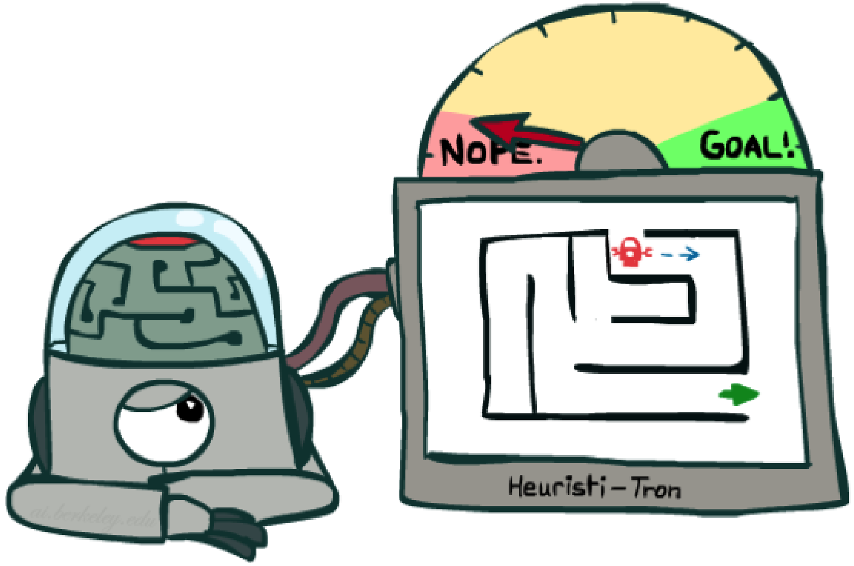


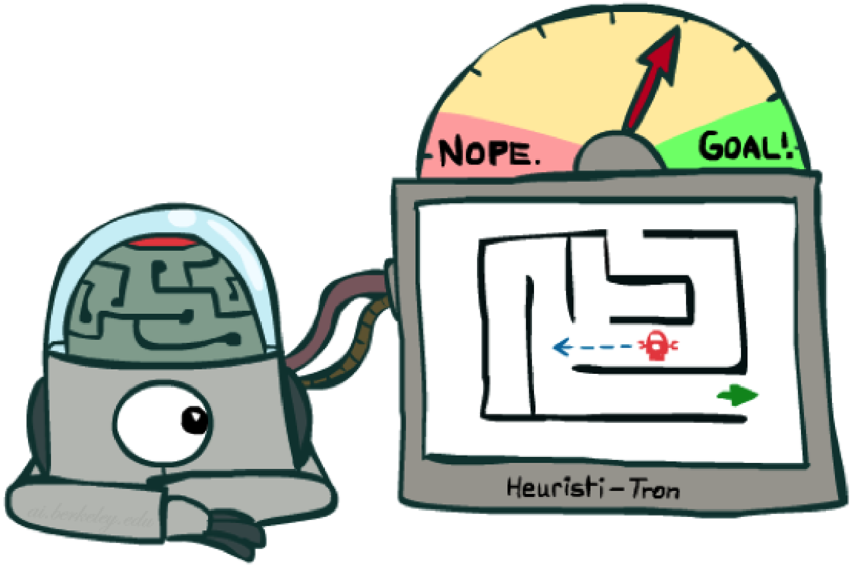
Search Heuristics

A heuristic is:

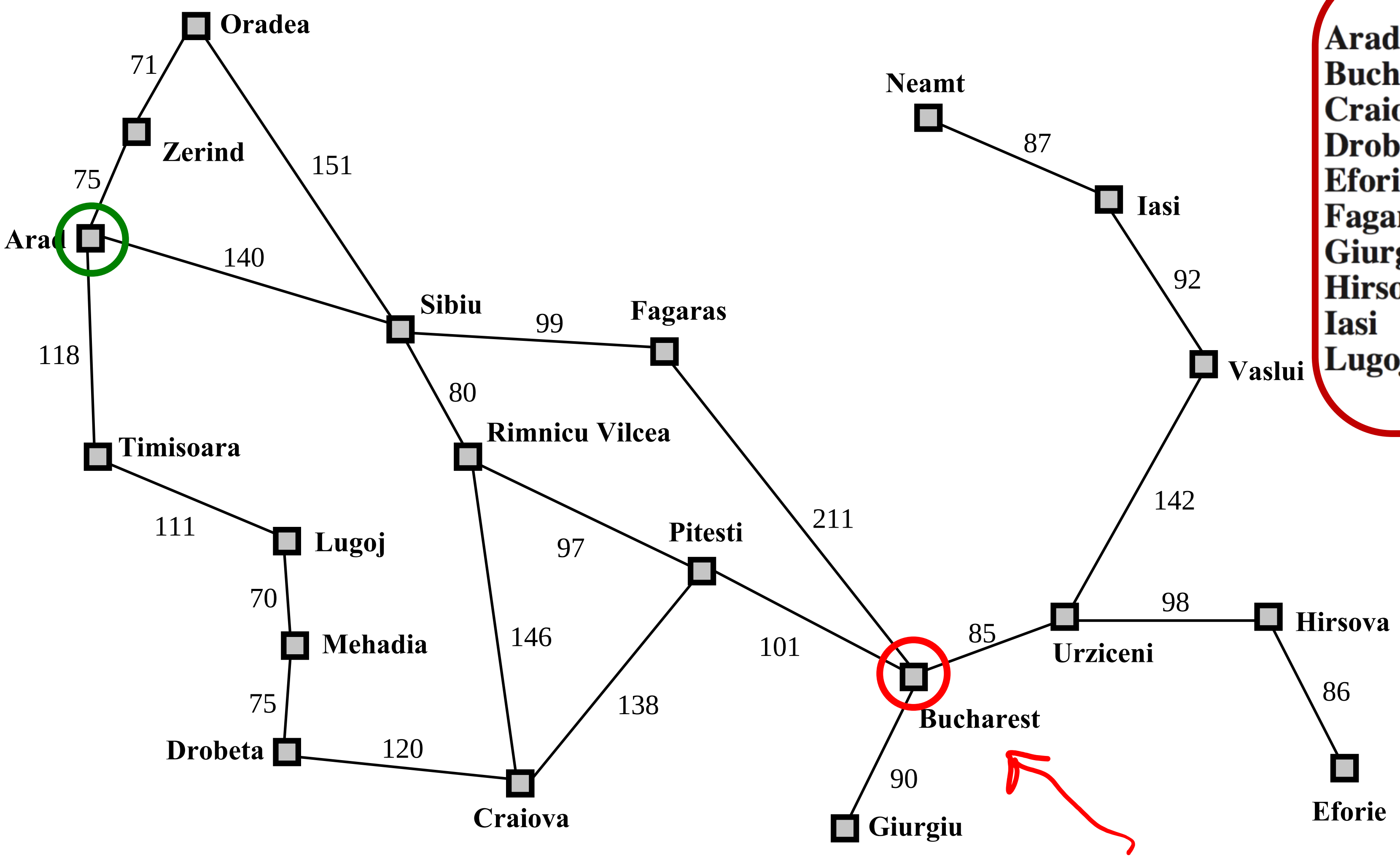
- A function that *estimates* how close a state is to a goal
- Designed for a particular search problem
- Examples: Manhattan distance, Euclidean distance for pathing







Example: Euclidean distance to Bucharest

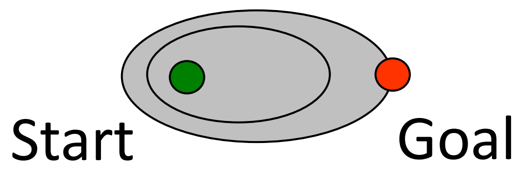


Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

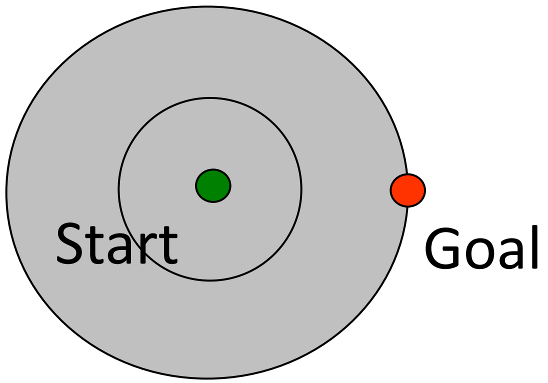
h(state) → value

Effect of heuristics

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



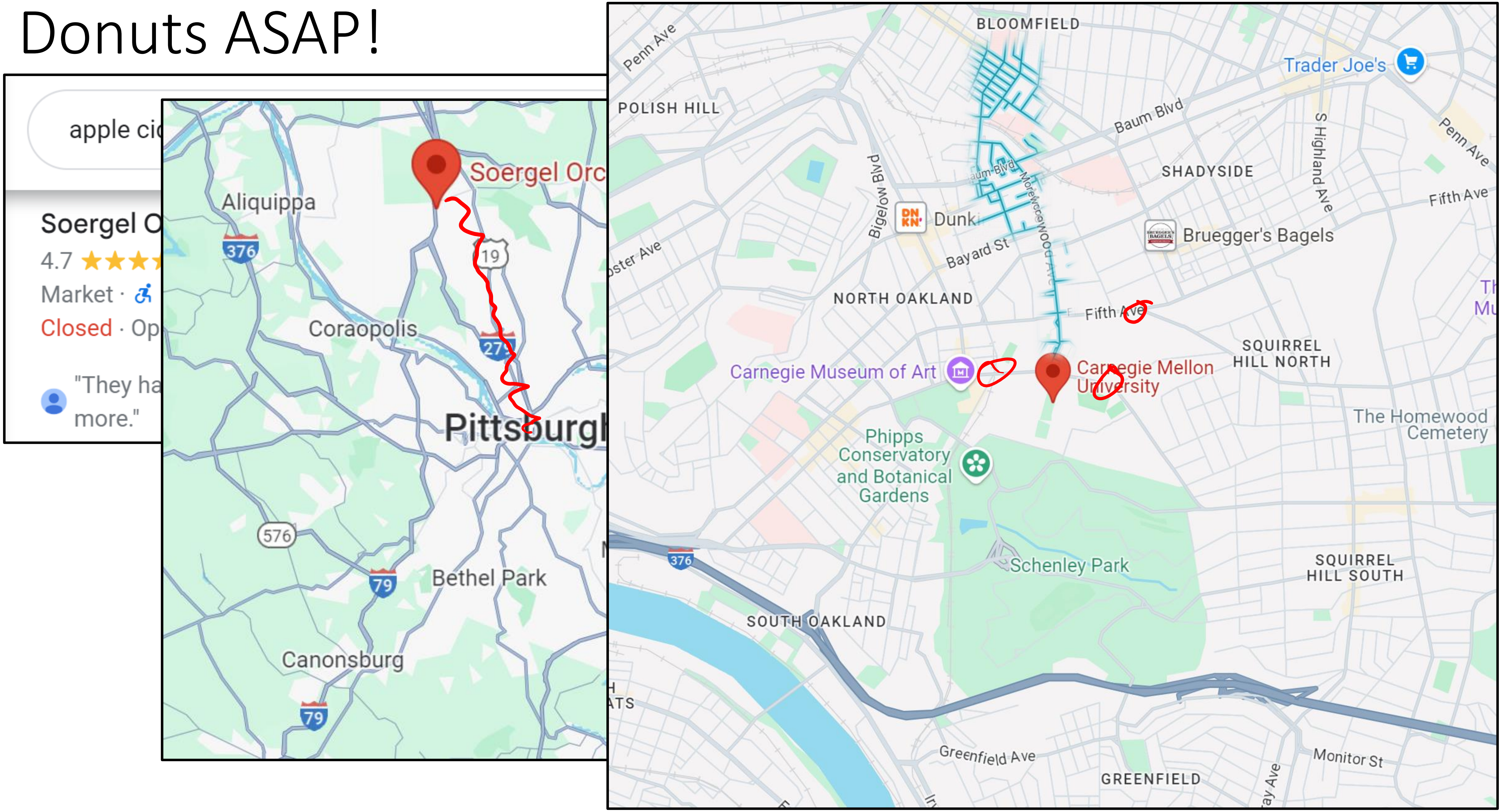
Informed



Uninformed

Greedy Search

Donuts ASAP!



apple ci
Soergel O
4.7
Market ·
Closed · Op
"They ha
more."
Soergel Orc
Aliquippa
376
19
Coraopolis
279
Pittsburgh
576
79
Bethel Park
Canonsburg
BLOOMFIELD
Penn Ave
POLISH HILL
Trader Joe's
Baum Blvd
SHADYSIDE
S Highland Ave
Penn Ave
Fifth Ave
Bigelow Blvd
Dunk
Morewood Ave
Brueggers Bagels
Bayard St
NORTH OAKLAND
Fifth Ave
SQUIRREL HILL NORTH
Carnegie Museum of Art
Carnegie Mellon University
The Homewood Cemetery
Phipps Conservatory and Botanical Gardens
376
Schenley Park
SQUIRREL HILL SOUTH
SOUTH OAKLAND
Greenfield Ave
Monitor St
GREENFIELD

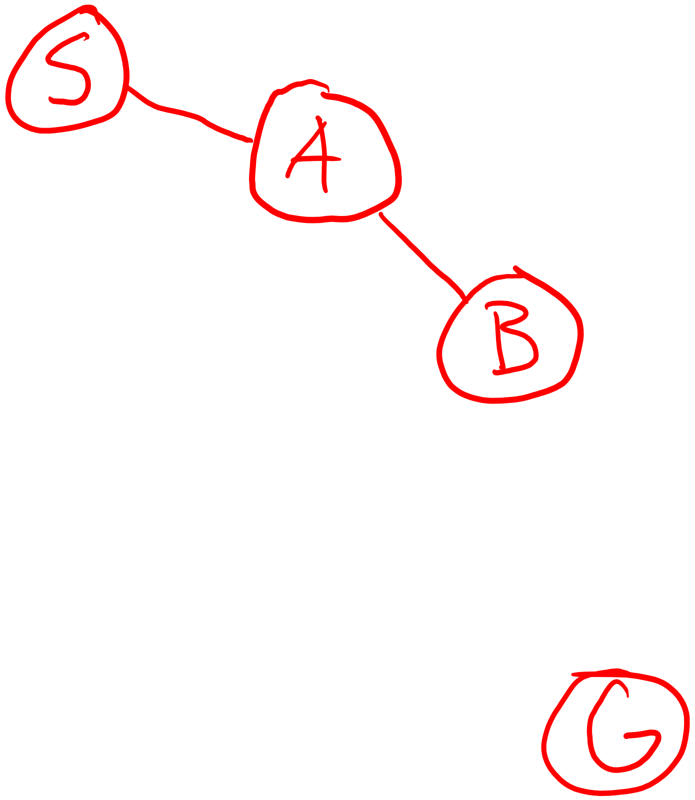
A* Search

A* Search

UCS

Greedy

A*



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

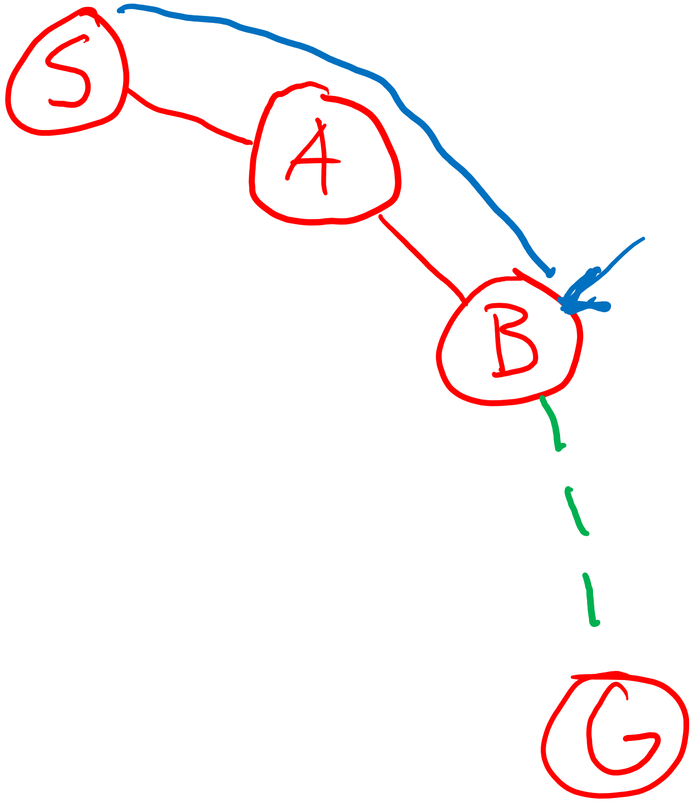
A* Search

$$f(S\text{-}A\text{-}B) = g(S\text{-}A\text{-}B) + h(B)$$

UCS

Greedy

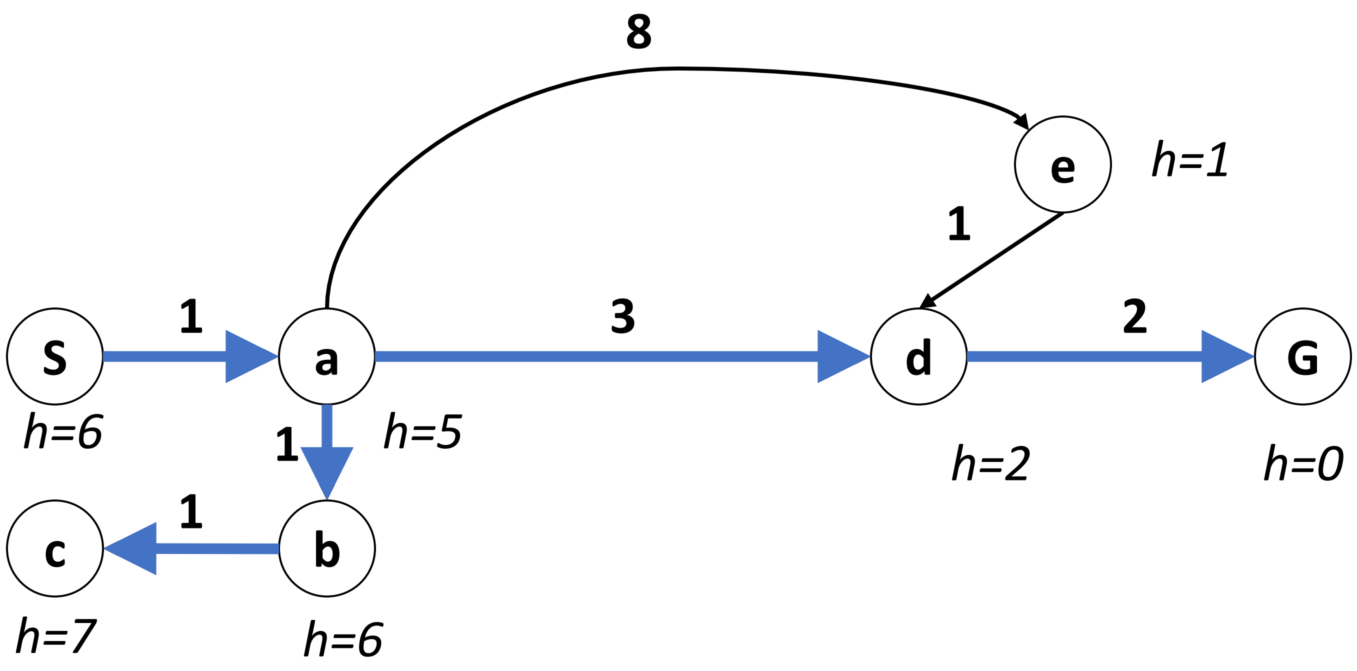
A*

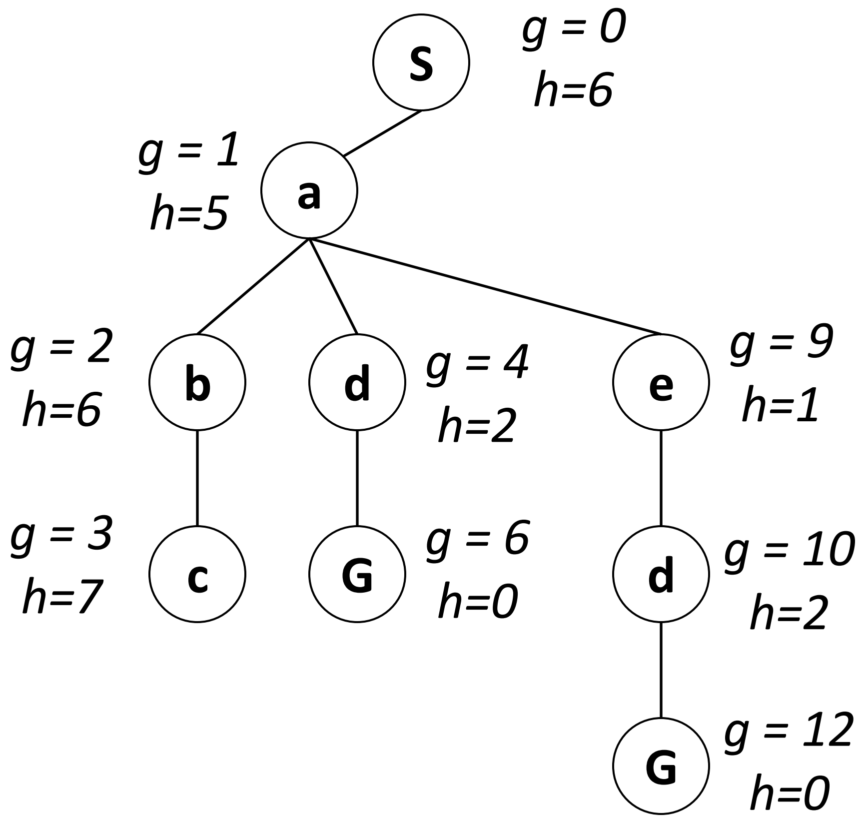


$$f(x) = g(x) + h(x)$$

Combining UCS and Greedy

Uniform-cost orders by path cost, or *backward cost* $g(n)$

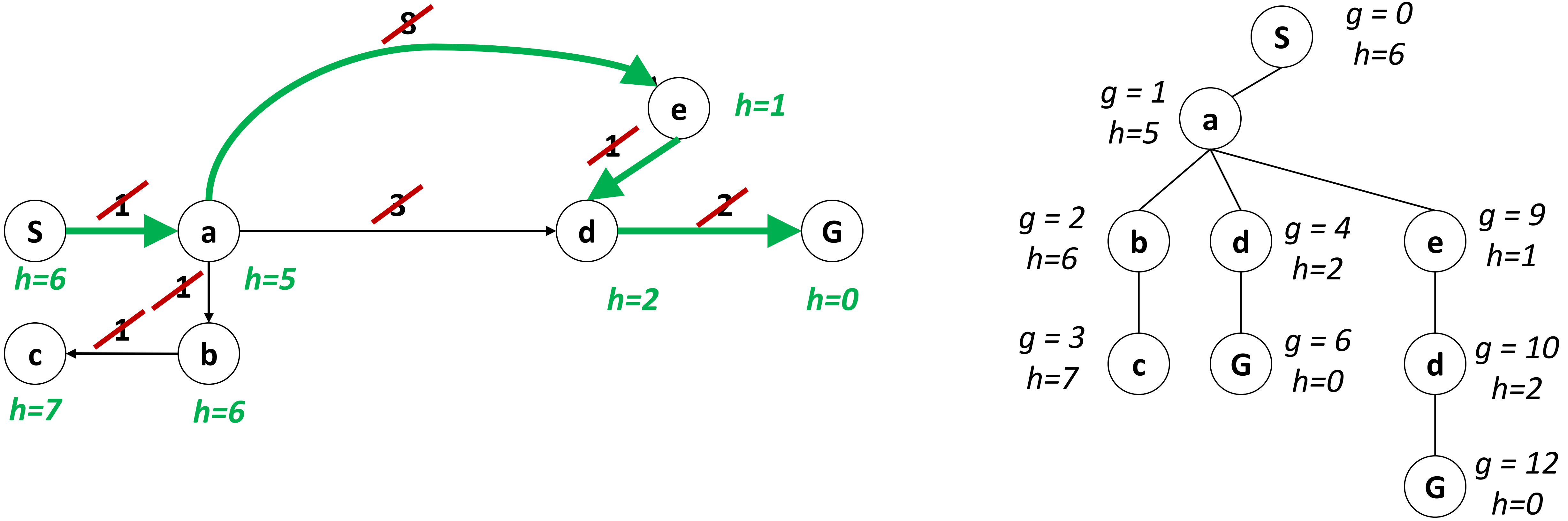




Example: Teg Grenager

Combining UCS and Greedy

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

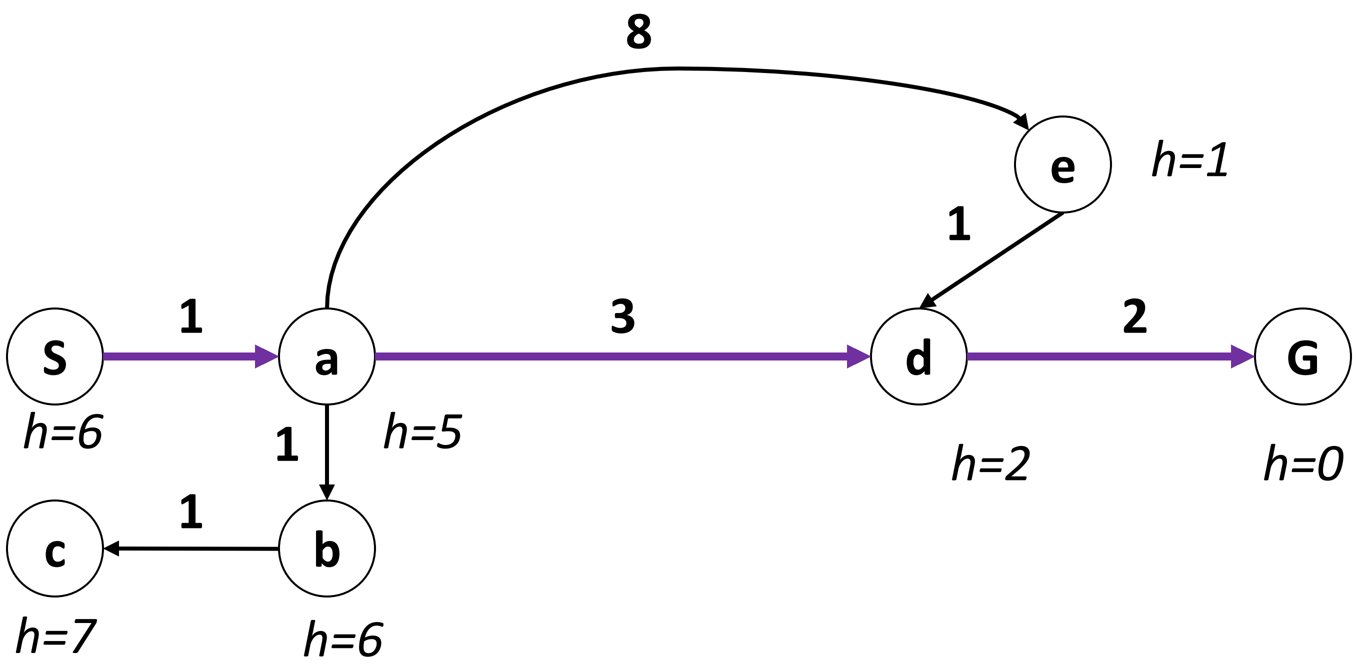


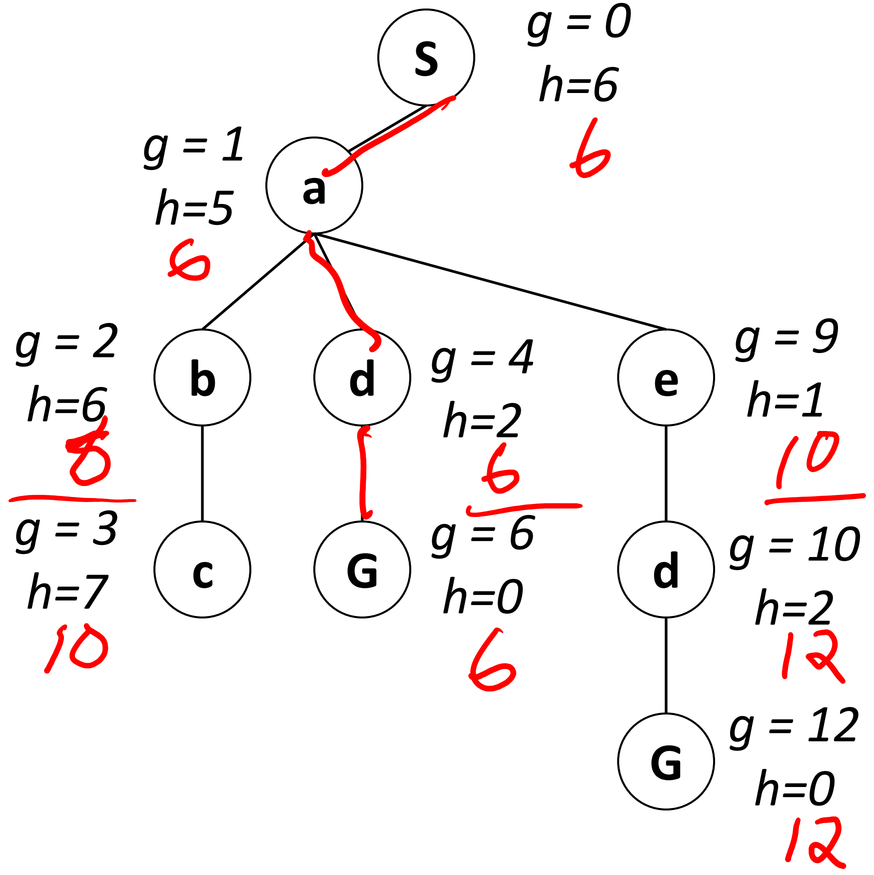
Example: Teg Grenager

Combining UCS and Greedy

Uniform-cost orders by path cost, or *backward cost* $g(n)$

Greedy orders by goal proximity, or *forward cost* $h(n)$

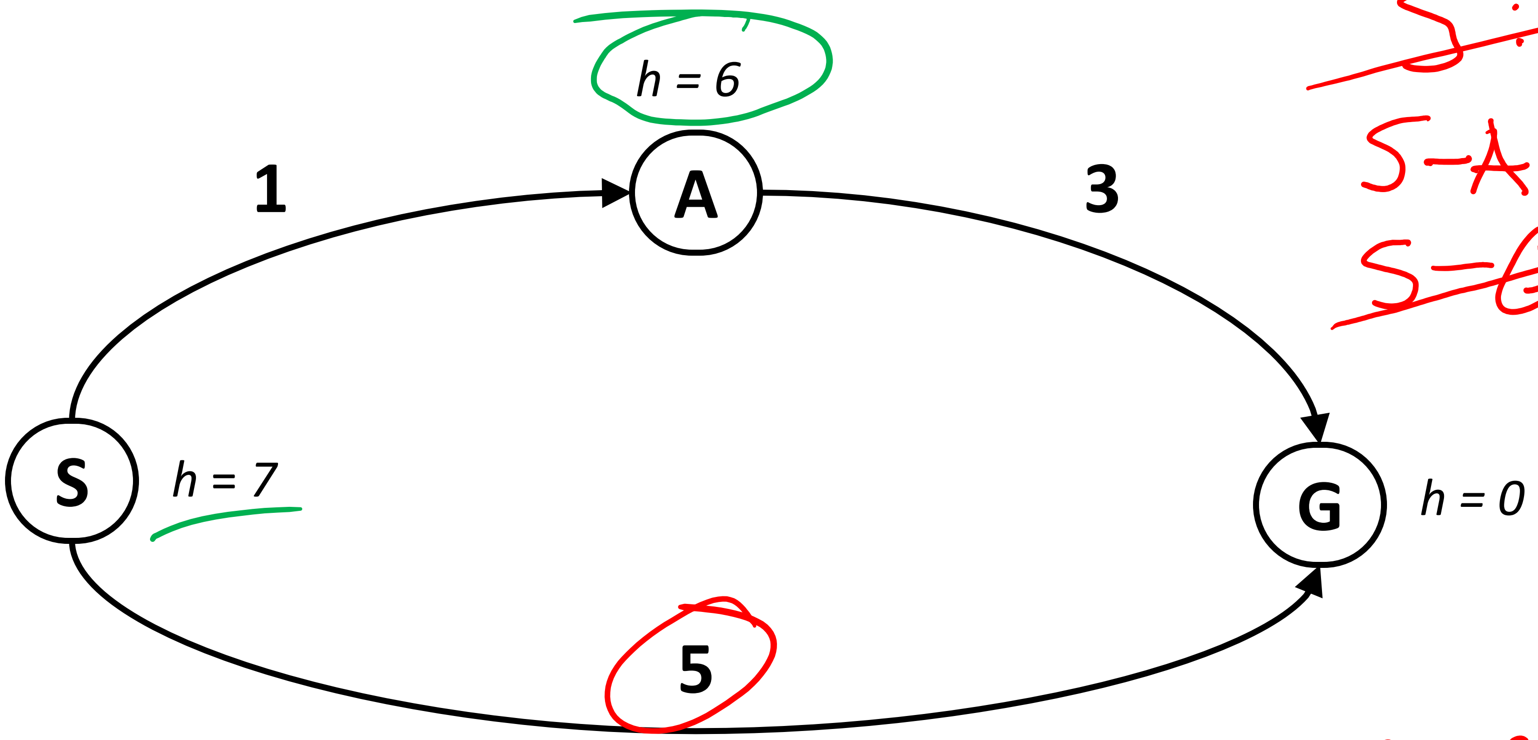




A* Search orders by the sum: $\boxed{f(n) = g(n) + h(n)}$

Example: Teg Grenager

Is A* Optimal?



Frontier

~~S : 0+7~~

S–A: 1+6

~~S–G: 5+0~~

G ✓

S–G : 5

What went wrong?

Estimated good goal cost > ***Actual*** future cost!

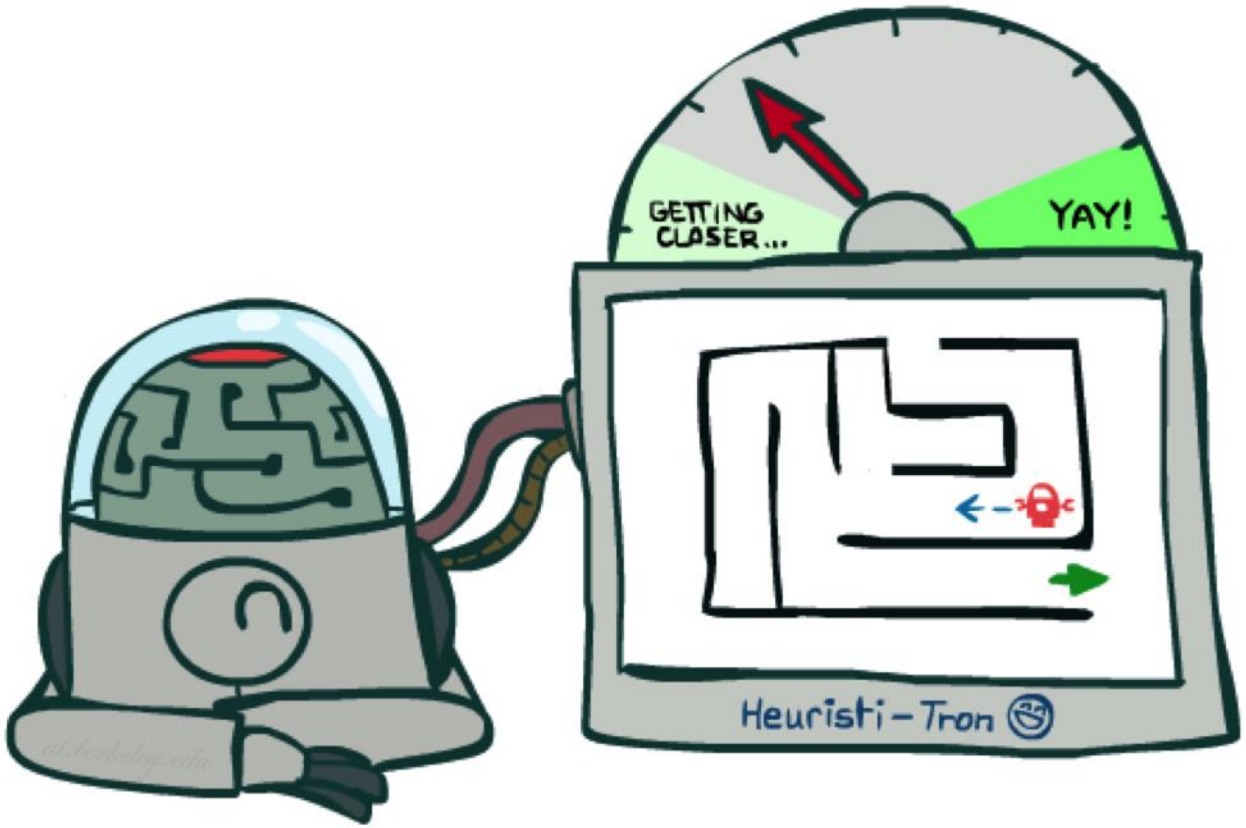
We need estimates to be less than actual costs!

The Price is Wrong…

Closest bid without going over…

https://www.youtube.com/watch?v=9B0ZKRurC5Y

Admissible Heuristics



Admissible Heuristics

$f = g + h$

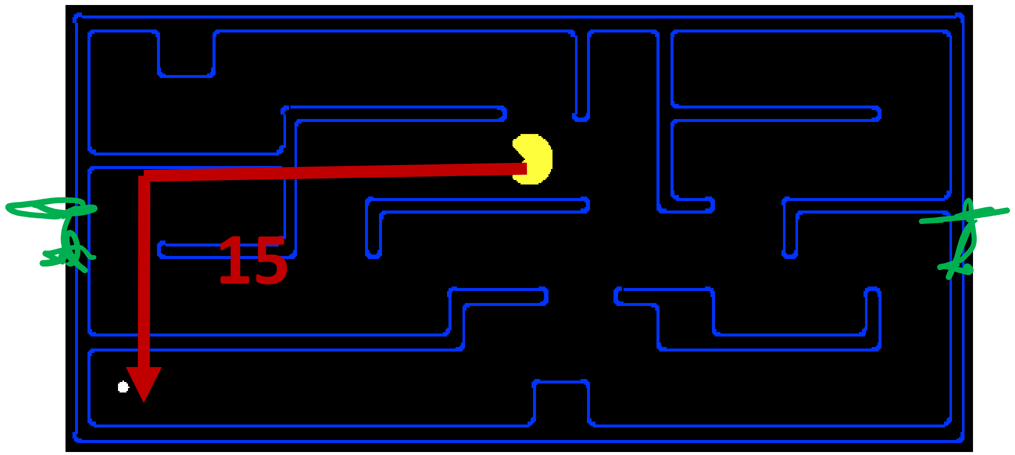
A heuristic *h* is *admissible* (optimistic) if:

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

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

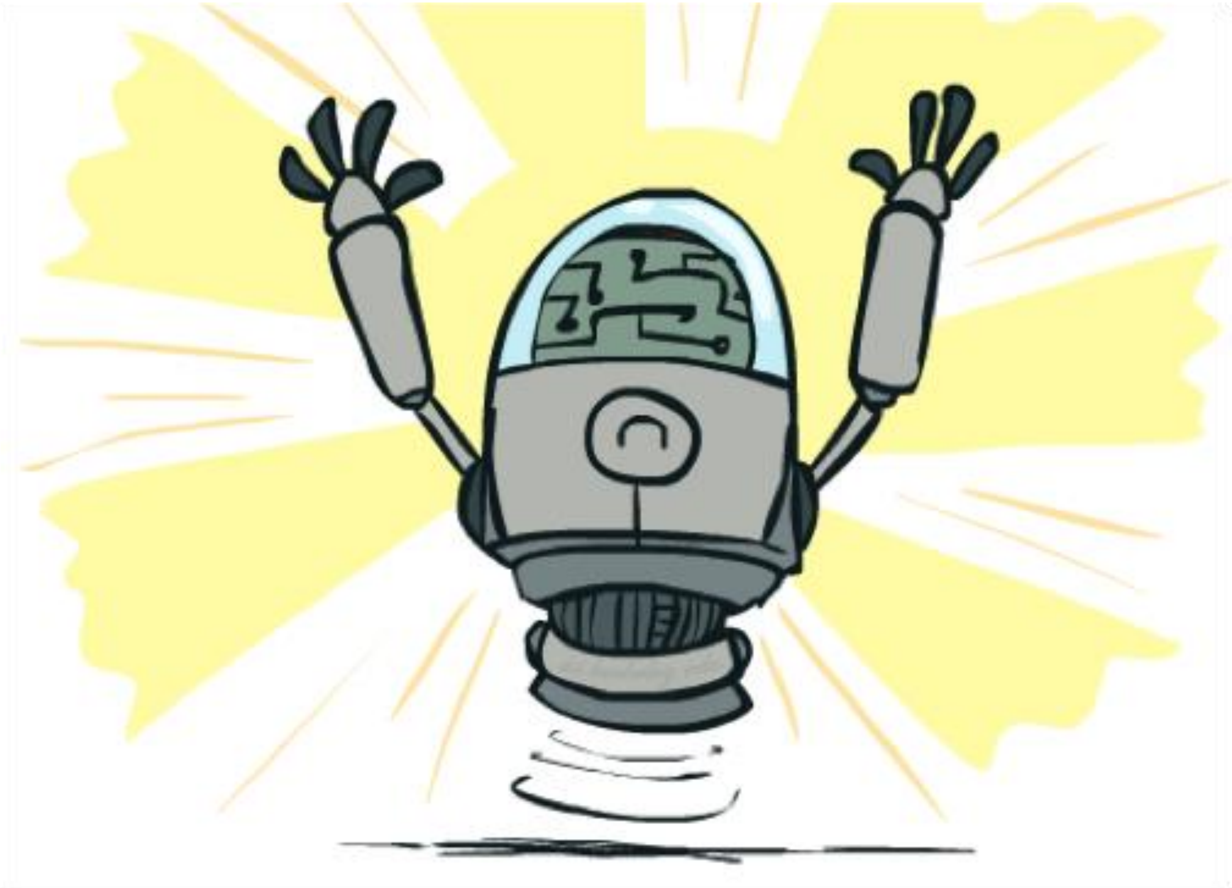
$h(x) = 0$

Example:



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

Optimality of A* Tree Search



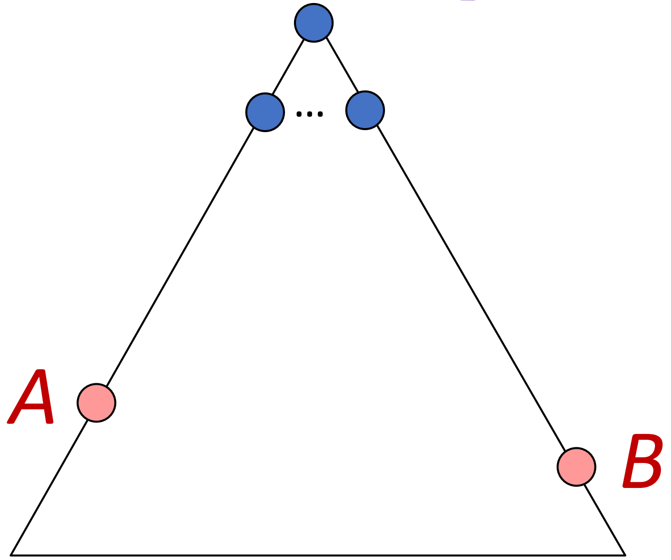
Optimality of A* Tree Search

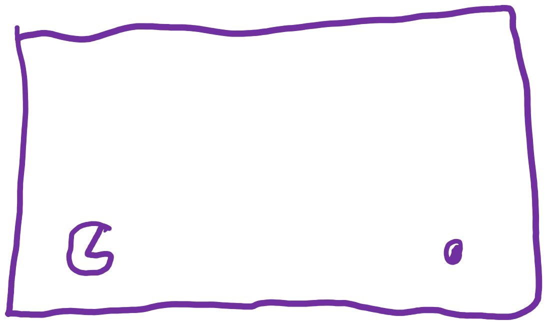
Assume:

A is an optimal goal node

B is a suboptimal goal node

h is admissible

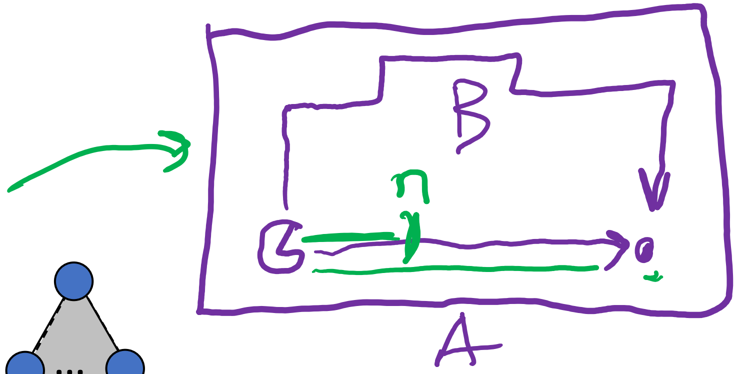




Claim:

A will be chosen for exploration (popped off the frontier) before *B*

Optimality of A* Tree Search: Blocking



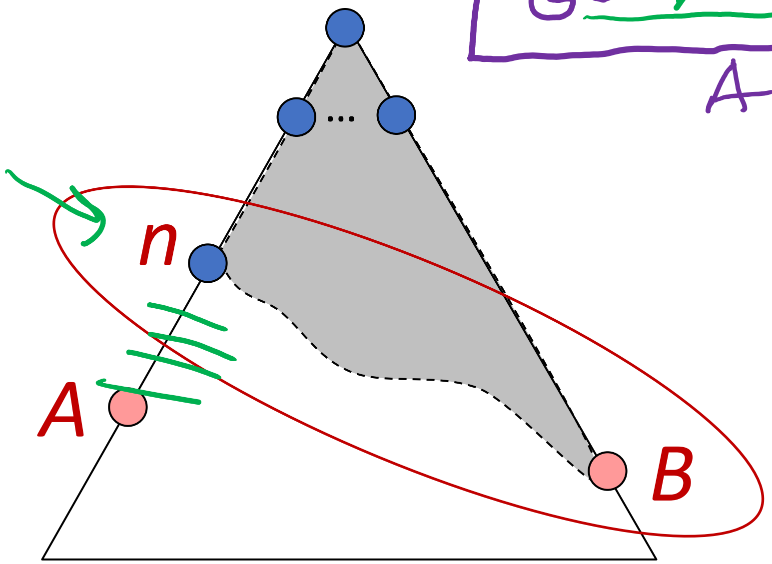
Proof:

Imagine *B* is on the frontier

Some ancestor *n* of *A* is on the frontier, too
(Maybe the start state; maybe *A* itself!)

Claim: *n* will be explored before *B*

1.
2.
3.

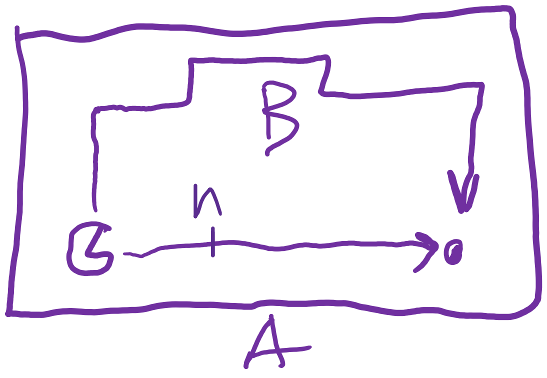


All ancestors of *A* are explored before *B*

A is explored before *B*

A* search is optimal

Optimality of A* Tree Search: Blocking



Proof:

Imagine *B* is on the frontier

Some ancestor *n* of *A* is on the frontier, too
(Maybe the start state; maybe *A* itself!)

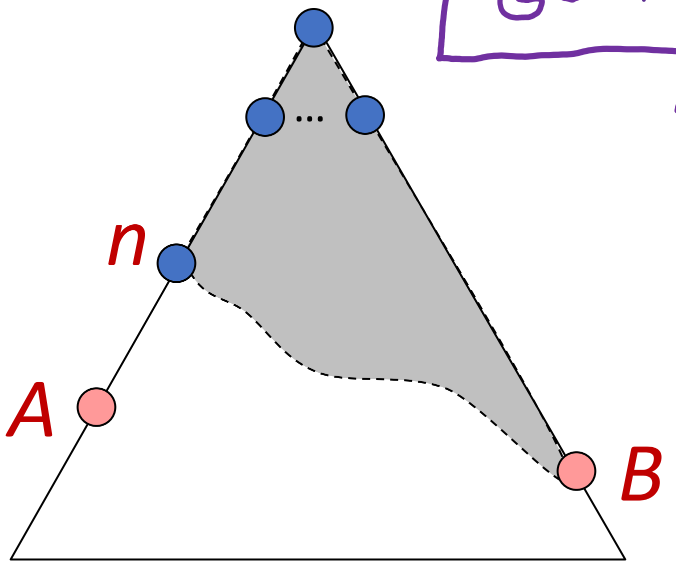
Claim: *n* will be explored before *B*

1. $f(n) \leq f(A)$ ← TODO
2. $f(A) < f(B)$ ← TODO
3. $f(n) \leq f(A) < f(B)$ then *n* is explored before *B*

All ancestors of *A* are explored before *B*

A is explored before *B*

A* search is optimal



Optimality of A* Tree Search: Blocking

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

$h(x) \leq h^*(x)$

Proof:

Imagine *B* is on the frontier

Some ancestor *n* of *A* is on the frontier, too
(Maybe the start state; maybe *A* itself!)

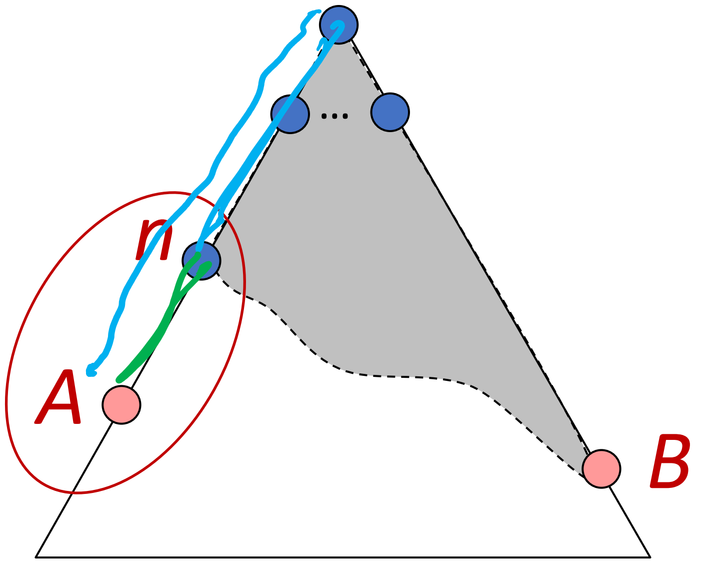
Claim: *n* will be explored before *B*

1. $f(n) \leq f(A)$
2. $f(A) < f(B)$
3. $f(n) \leq f(A) < f(B)$ the…

All ancestors of *A* are explored be…

A is explored before *B*

A* search is optimal



$f(n) = g(n) + h(n)$	Definition of *f*-cost
$f(n) \leq g(n) + h^*(n)$	Admissibility of *h*
$f(n) \leq g(A)$ $+ h(A)$	n on optimal path to *A*
$f(n) \leq f(A)$	*h* = 0 at a goal

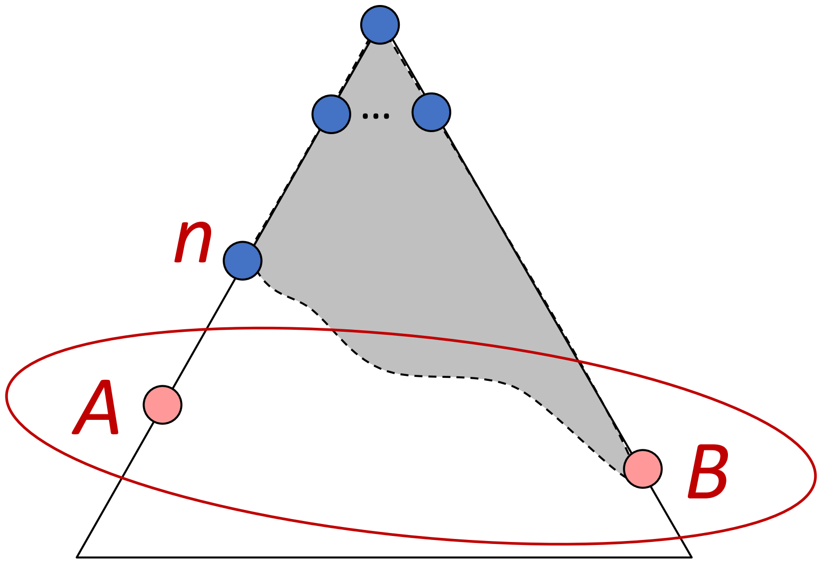
Optimality of A* Tree Search: Blocking

$f(x) = g(x) + h(x)$
$h(x) \leq h^*(x)$

Proof:

Imagine *B* is on the frontier

Some ancestor *n* of *A* is on the frontier, too
(Maybe the start state; maybe *A* itself!)



Claim: *n* will be explored before *B*

1. $f(n) \leq f(A)$
2. $f(A) < f(B)$
3. $f(n) \leq f(A) < f(B)$ th

All ancestors of *A* are explored befo

A is explored before *B*

A* search is optimal

$f(A) = g(A) + h(A)$	Def. of $f(x)$
$= g(A)$	$h = 0$ at a goal
$f(B) = g(B)$	Similarly for *B*
$g(A) < g(B)$	Suboptimality of *B*
$f(A) < f(B)$	

Optimality of A* Tree Search: Blocking

$f(x) = g(x) + h(x)$
$h(x) \leq h^*(x)$

Proof:

Imagine *B* is on the frontier

Some ancestor *n* of *A* is on the frontier, too
(Maybe the start state; maybe *A* itself!)

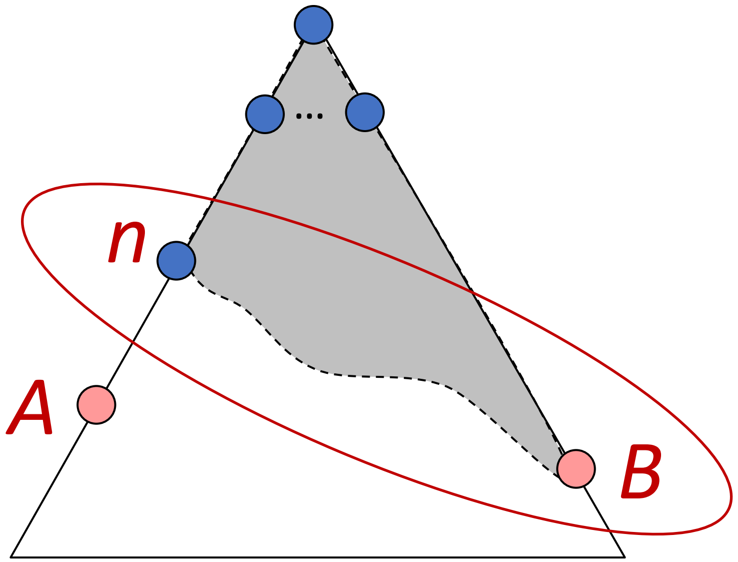
Claim: *n* will be explored before *B*

1. $f(n)$ is less than or equal to $f(A)$
2. $f(A)$ is less than $f(B)$
3. $\underline{f(n)} \leq f(A) < \underline{f(B)}$ then *n* is explored before *B*

All ancestors of *A* are explored before *B*

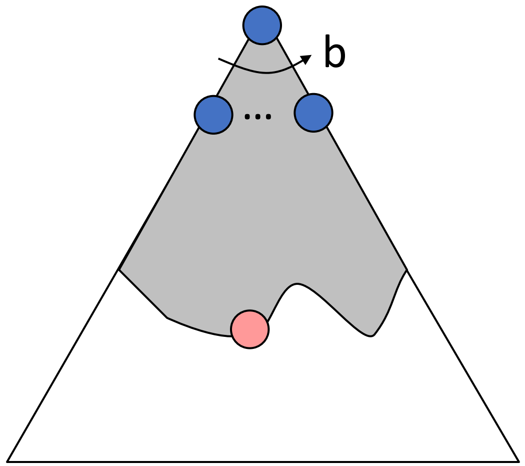
A is explored before *B*

A* search is optimal

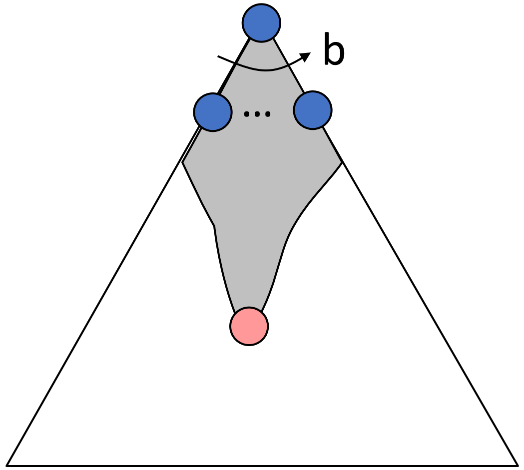


UCS vs A* Contours

Uniform-Cost

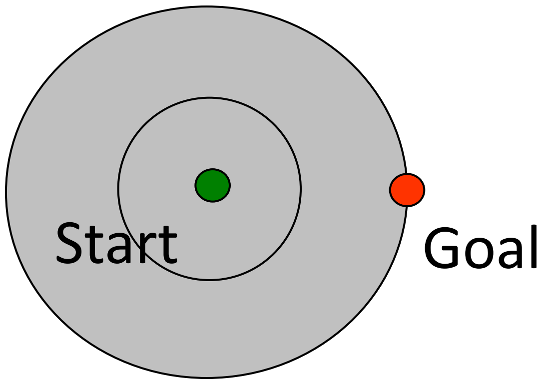


A*

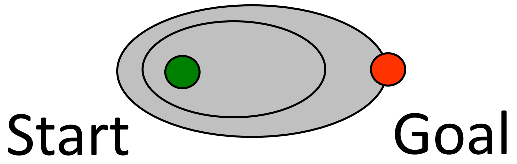


UCS vs A* Contours

Uniform-cost expands equally in all "directions"



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



Comparison

Greedy

Uniform Cost

A*

A* Search Algorithms

A* Tree Search

- Same tree search algorithm as before but with a frontier that is a priority queue using priority $f(n) = g(n) + h(n)$

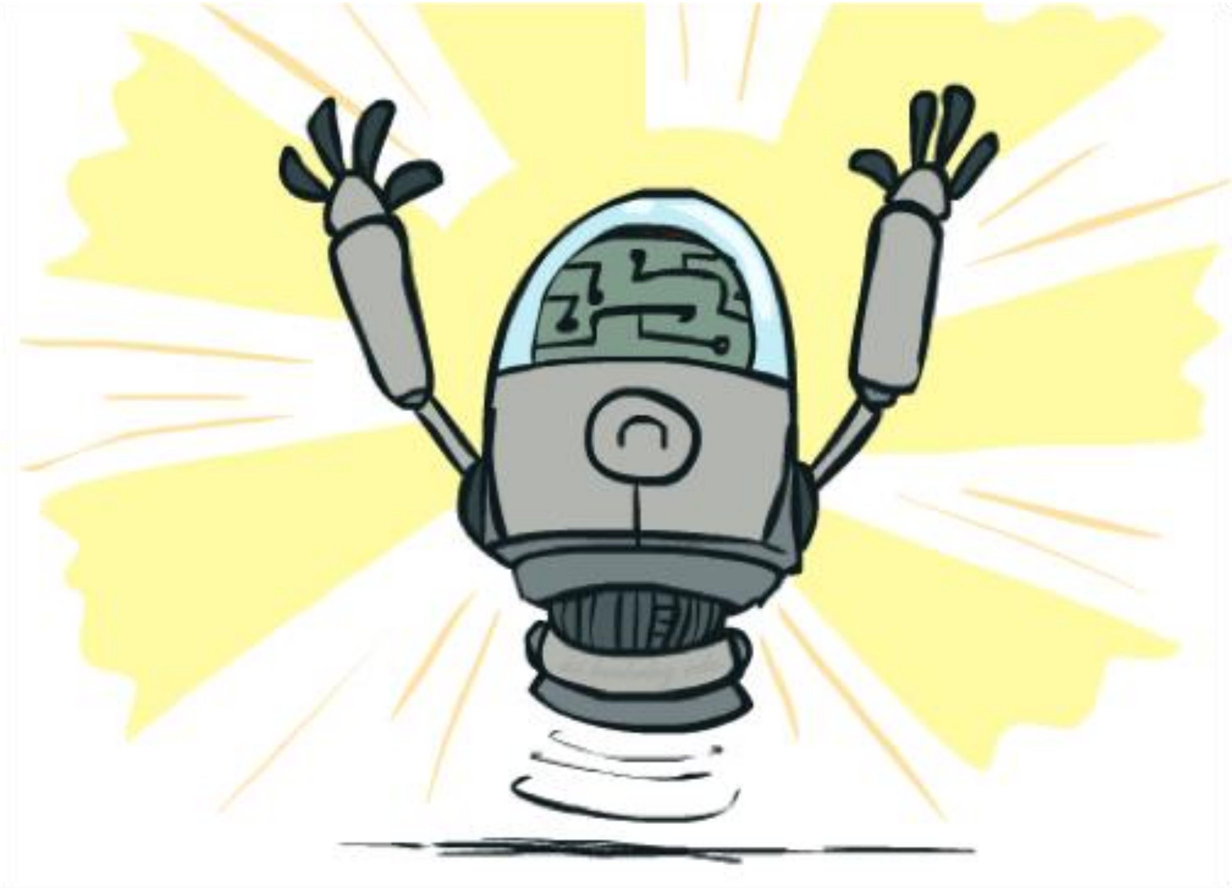
A* Graph Search

- Same **UCS** graph search algorithm but with a frontier that is a priority queue using priority $f(n) = g(n) + h(n)$

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
    initialize the explored set to be empty
    initialize the frontier as a priority queue using g(n) as the priority
    add initial state of problem to frontier with priority g(S) = 0
    loop do
        if the frontier is empty then
            return failure
        choose a node and remove it from the frontier
        if the node contains a goal state then
            return the corresponding solution
        add the node state to the explored set
        for each resulting child from node
            if the child state is not already in the frontier or explored set then
                add child to the frontier
            else if the child is already in the frontier with higher g(n) then
                replace that frontier node with child
```

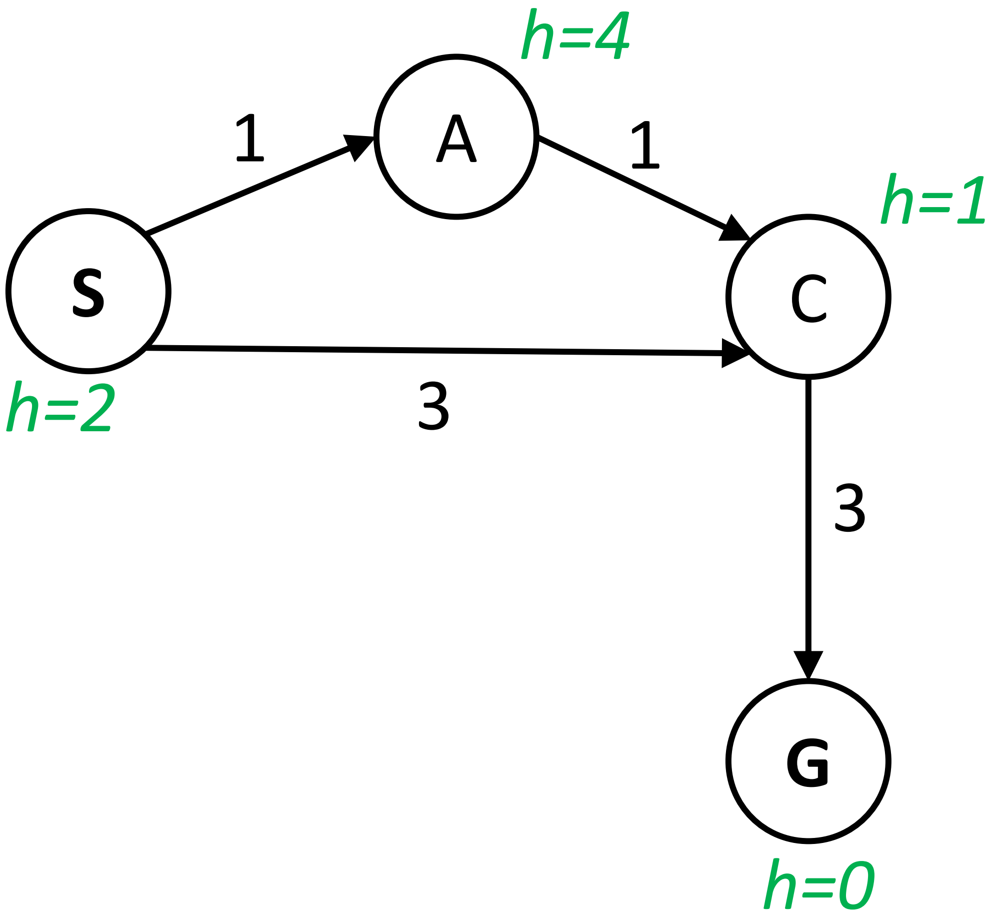
```
function A-STAR-SEARCH(problem) returns a solution, or failure
    initialize the explored set to be empty
    initialize the frontier as a priority queue using f(n) = g(n) + h(n) as the priority
    add initial state of problem to frontier with priority f(S) = 0 + h(S)
    loop do
        if the frontier is empty then
            return failure
        choose a node and remove it from the frontier
        if the node contains a goal state then
            return the corresponding solution
        add the node state to the explored set
        for each resulting child from node
            if the child state is not already in the frontier or explored set then
                add child to the frontier
            else if the child is already in the frontier with higher f(n) then
                replace that frontier node with child
```

Optimality of A* Graph Search

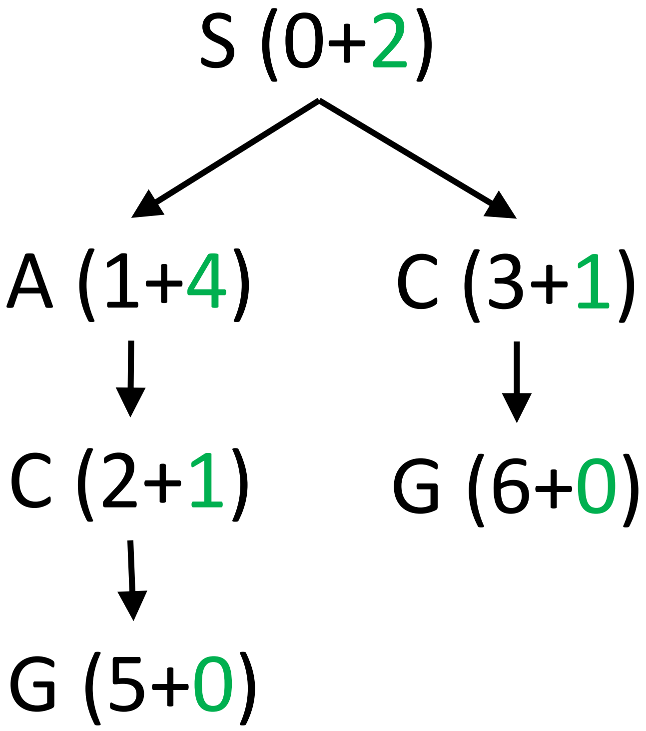


A* Tree Search

State space graph



Search tree



Frontier

~~S (0+2)~~

~~S-A (1+4)~~ 5

~~S-C (3+1)~~ 4

S-C-G (6+0)

~~S-A-C (2+1)~~ 3

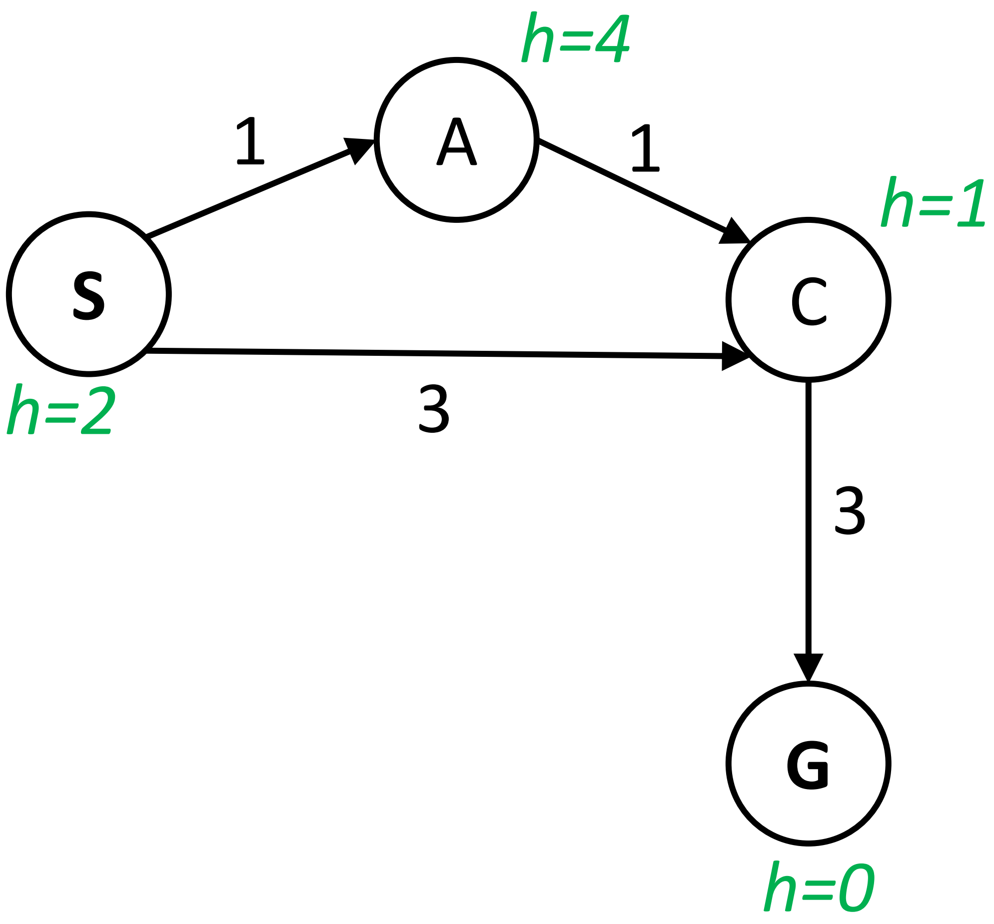
~~S-A-C-G (5+0)~~

Result: S-A-C-G cost 5

Correct!

A* Graph Search

What paths does A* graph search consider during its search?

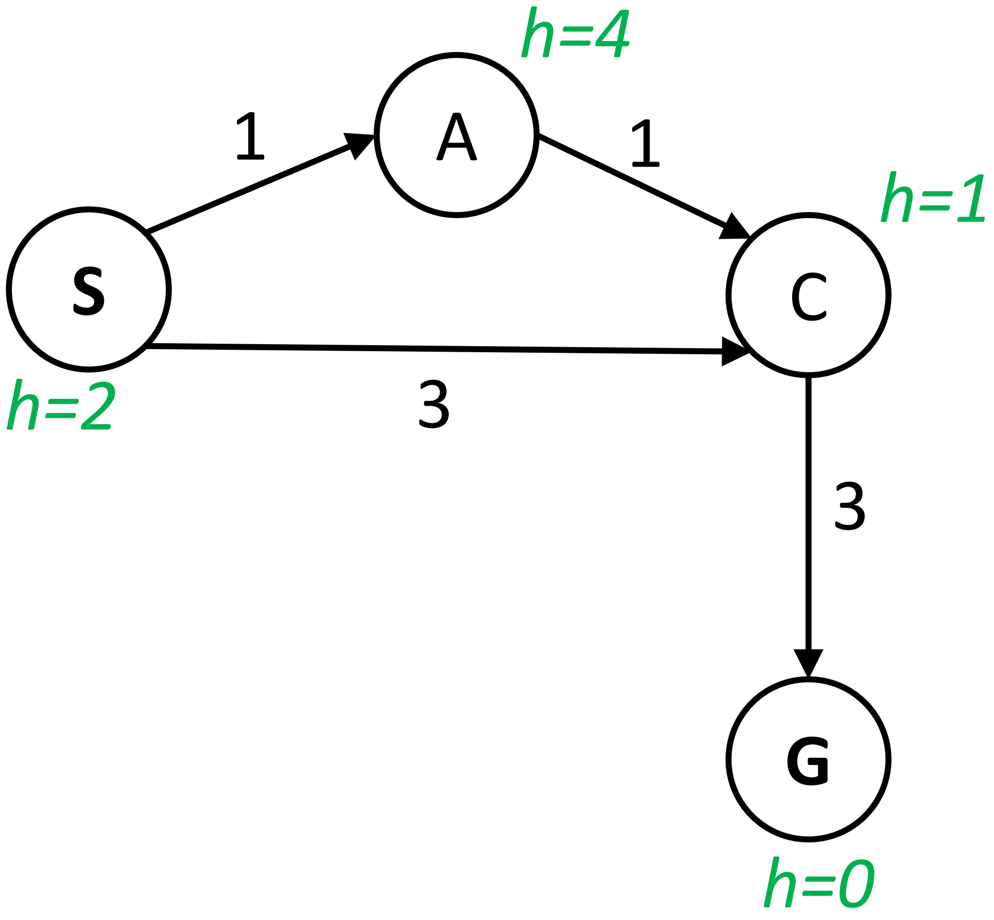


Frontier

Explored

Poll 1: A* Graph Search

What paths does A* graph search consider during its search?
(What does your work for the frontier look like?)



A) ~~S~~, ~~S-A~~, ~~S-C~~, S-C-G

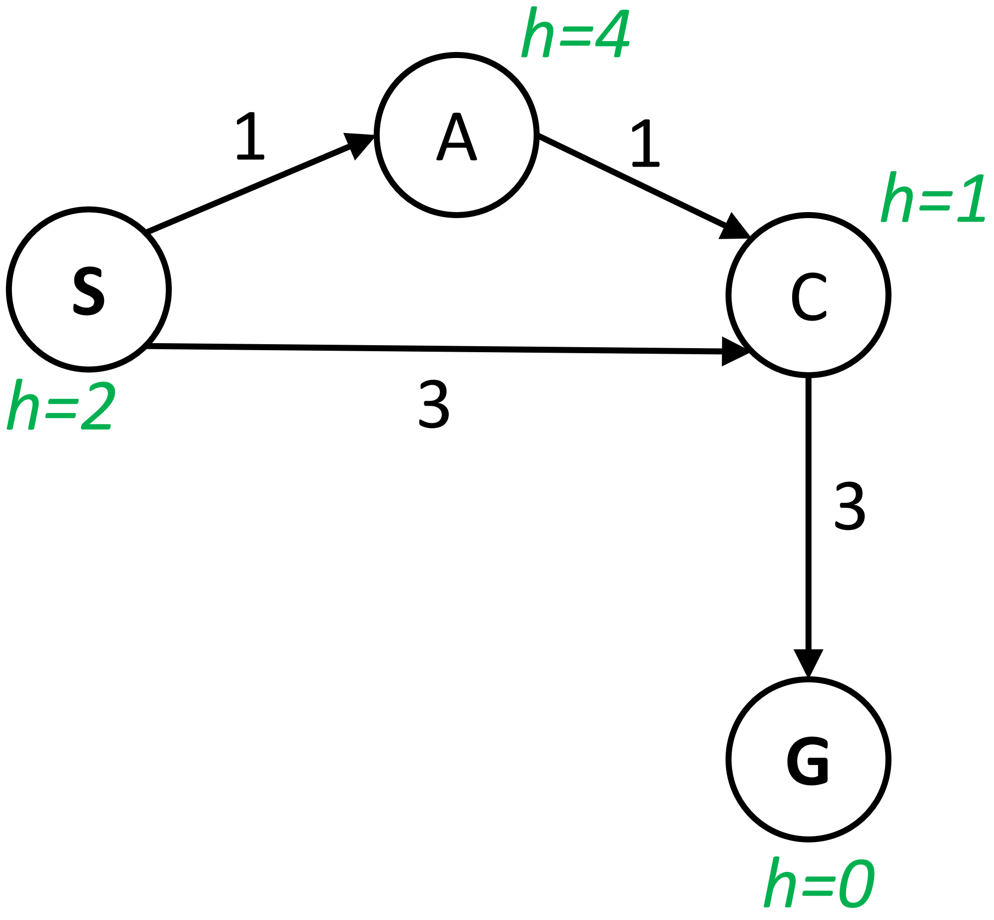
B) ~~S~~, ~~S-A~~, S-C, ~~S-A-C~~, S-C-G

C) ~~S~~, ~~S-A~~, ~~S-A-C~~, S-A-C-G

D) ~~S~~, ~~S-A~~, ~~S-C~~, ~~S-A-C~~, S-A-C-G

Poll 1: A* Graph Search

What paths does A* graph search consider during its search?



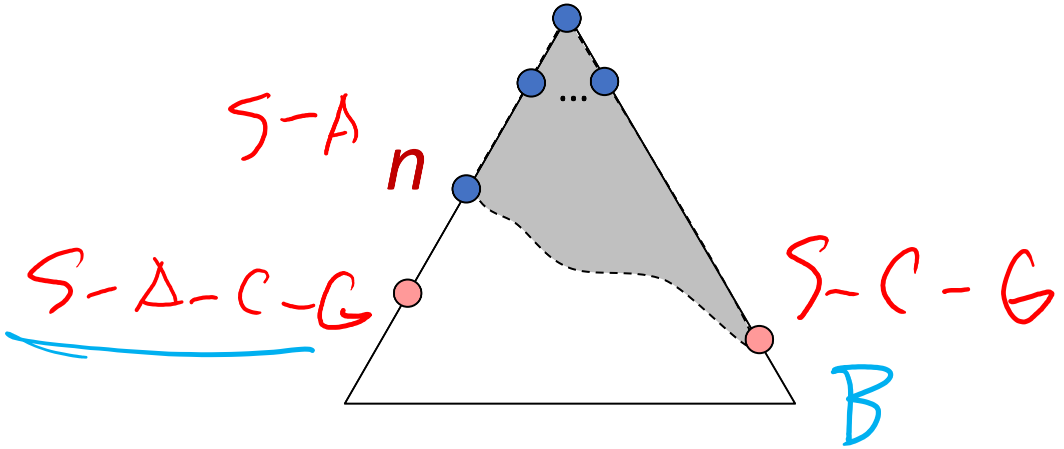
A) ~~S~~, ~~S-A~~, ~~S-C~~, S-C-G

B) ~~S~~, ~~S-A~~, S-C, ~~S-A-C~~, S-C-G

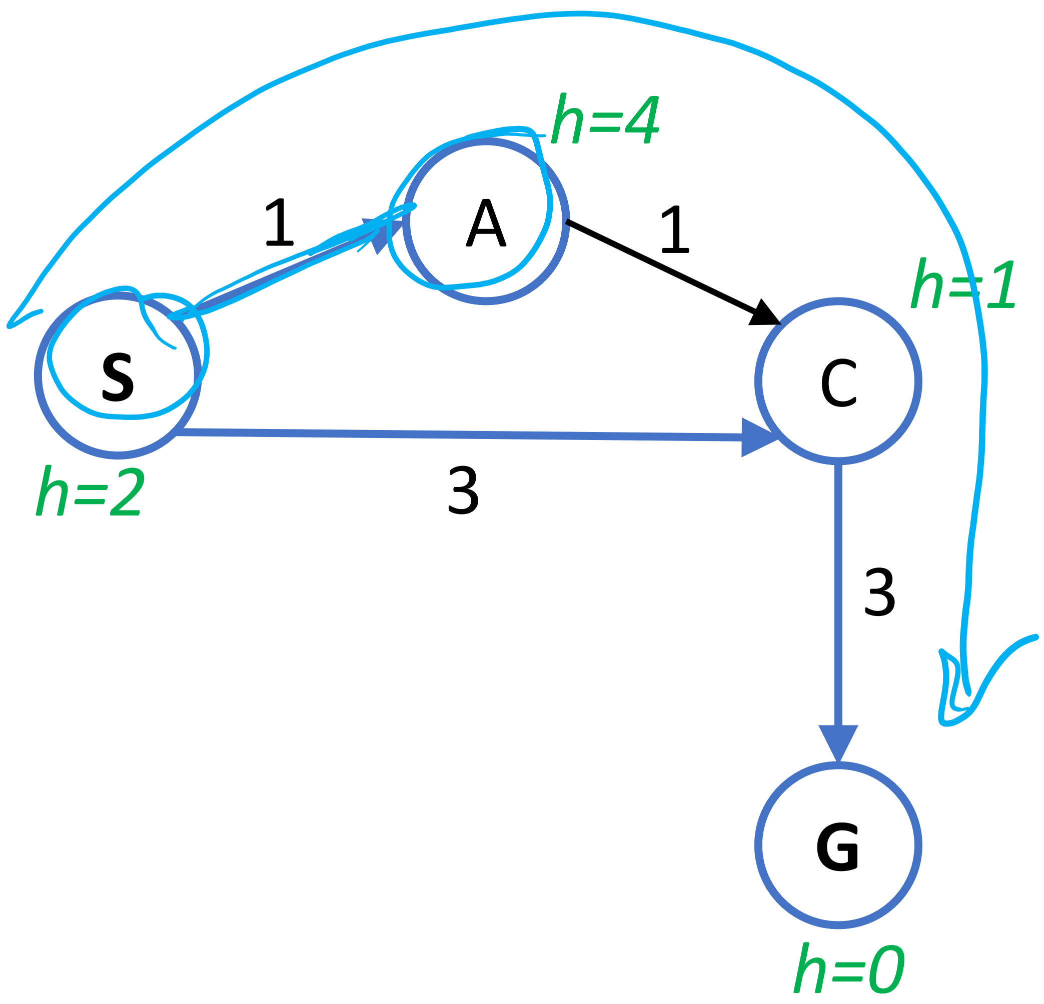
C) ~~S~~, ~~S-A~~, ~~S-A-C~~, S-A-C-G

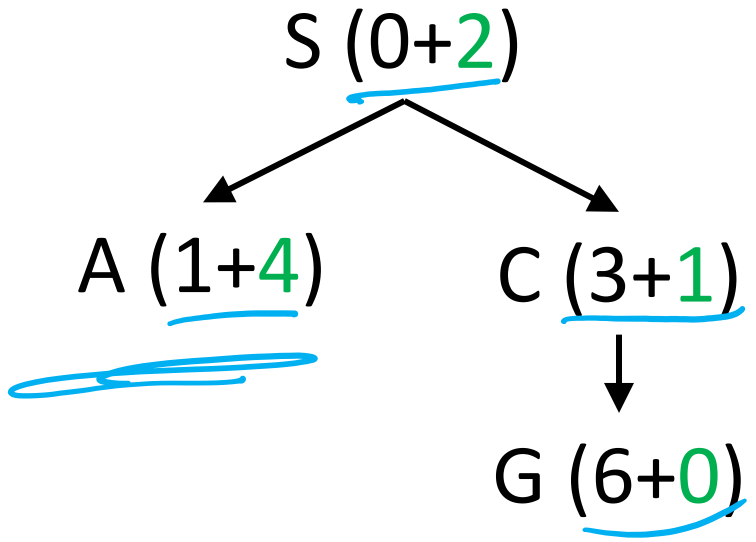
D) ~~S~~, ~~S-A~~, ~~S-C~~, ~~S-A-C~~, S-A-C-G

A* Graph Search Gone Wrong?



State space graph

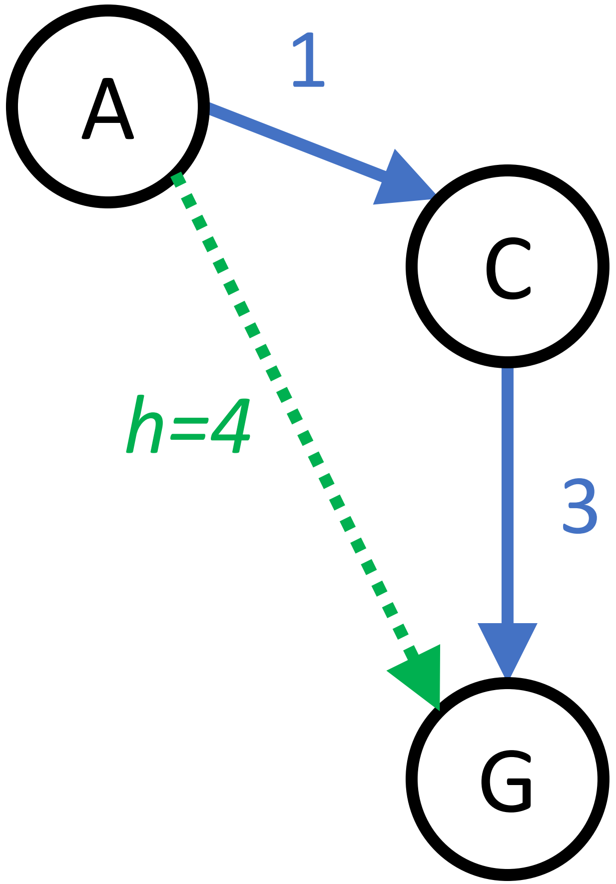




Simple check against explored set blocks C

S-A-C never gets considered

Admissibility of Heuristics



Main idea: Estimated heuristic values ≤ actual costs

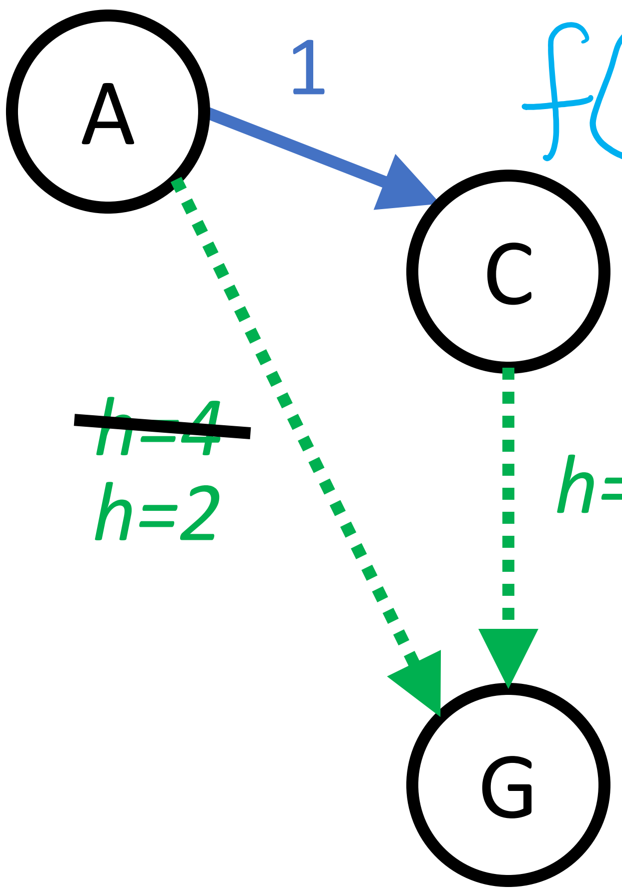
- Admissibility:

  heuristic value ≤ actual cost to goal

  h(A) ≤ actual cost from A to G

Consistency of Heuristics

f(A): 0+4



Main idea: Estimated heuristic costs ≤ actual costs

- Admissibility:

  heuristic cost ≤ actual cost to goal

  h(A) ≤ actual cost from A to G

- Consistency:

  "heuristic step cost" ≤ actual cost for each step

  h(A) – h(C) ≤ cost(A to C)

  triangle inequality

  h(A) ≤ cost(A to C) + h(C)

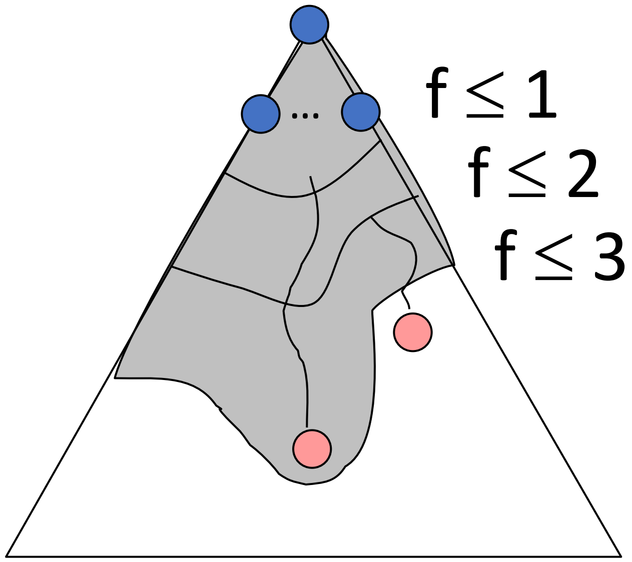
Consequences of consistency:

- The f value along a path never decreases
- 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 explored before nodes that reach s suboptimally
- Result: A* graph search is optimal



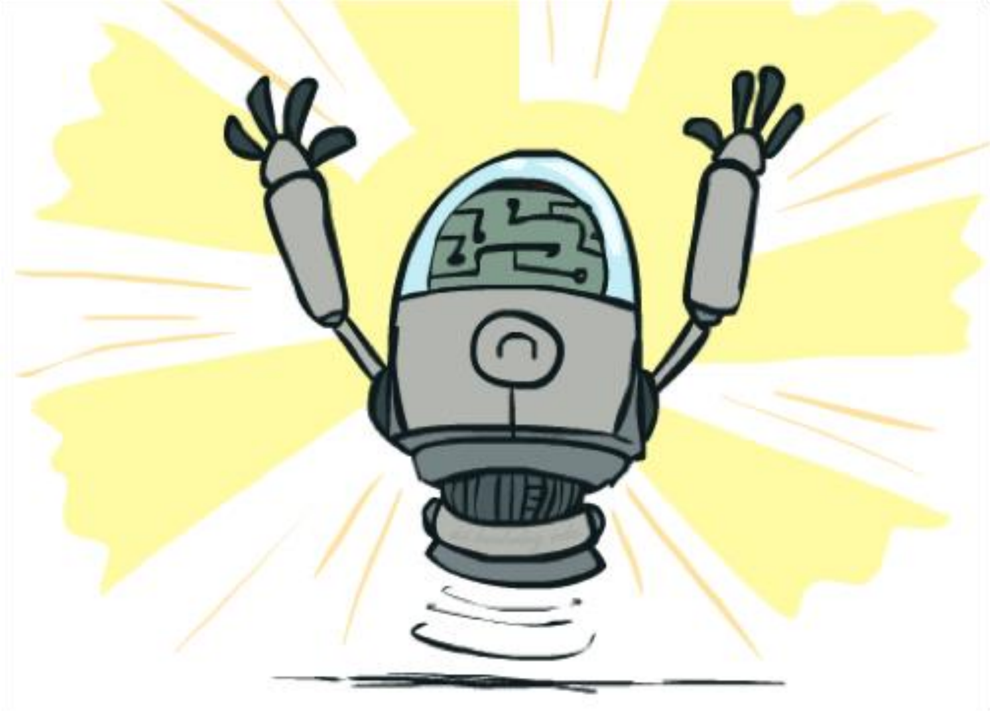
Optimality

Tree search:

- A* is optimal if heuristic is admissible
- UCS is a special case (h = 0)

Graph search:

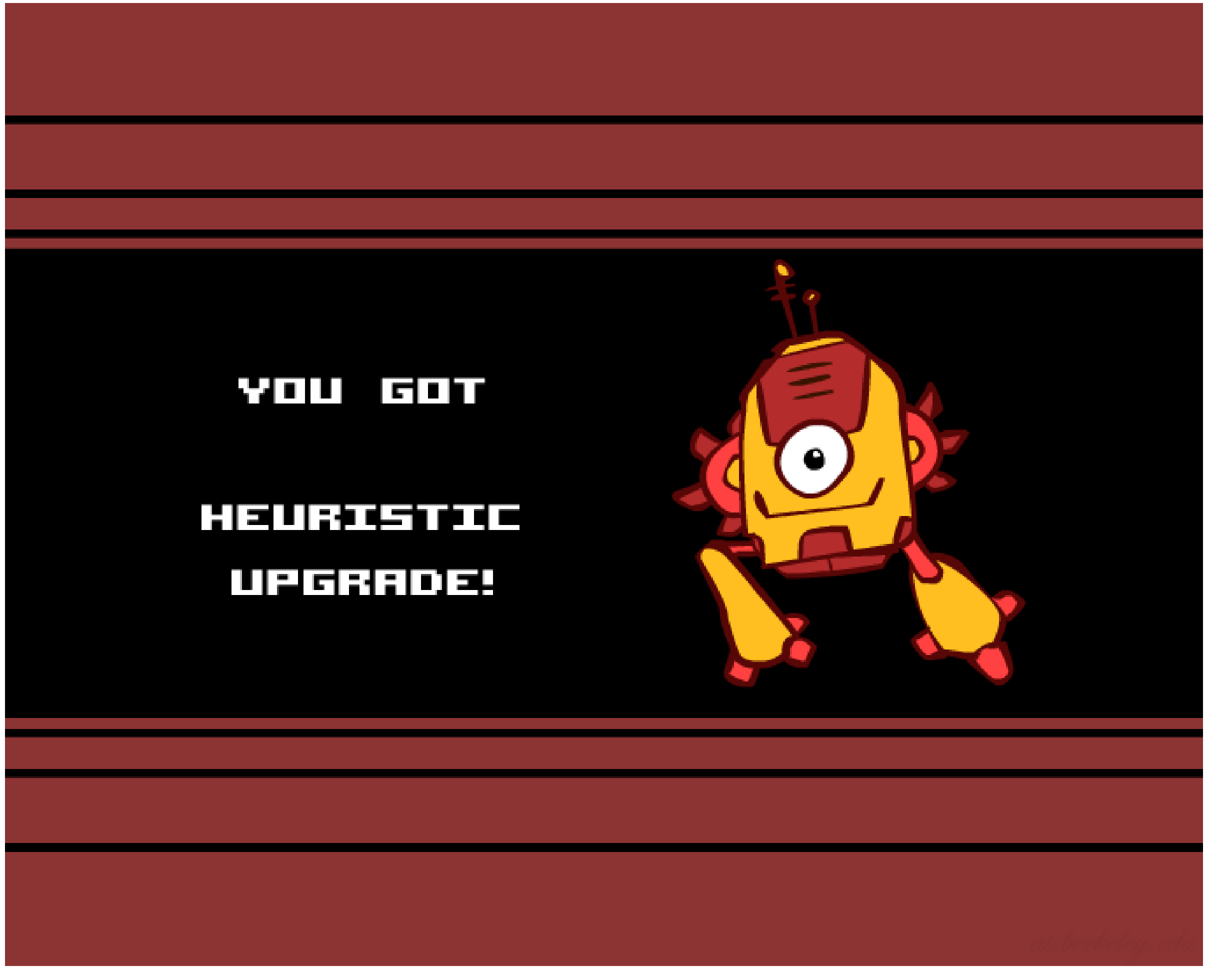
- A* optimal if heuristic is consistent
- UCS optimal (h = 0 is consistent)



Consistency implies admissibility

In general, most natural admissible heuristics tend to be consistent, especially if from relaxed problems

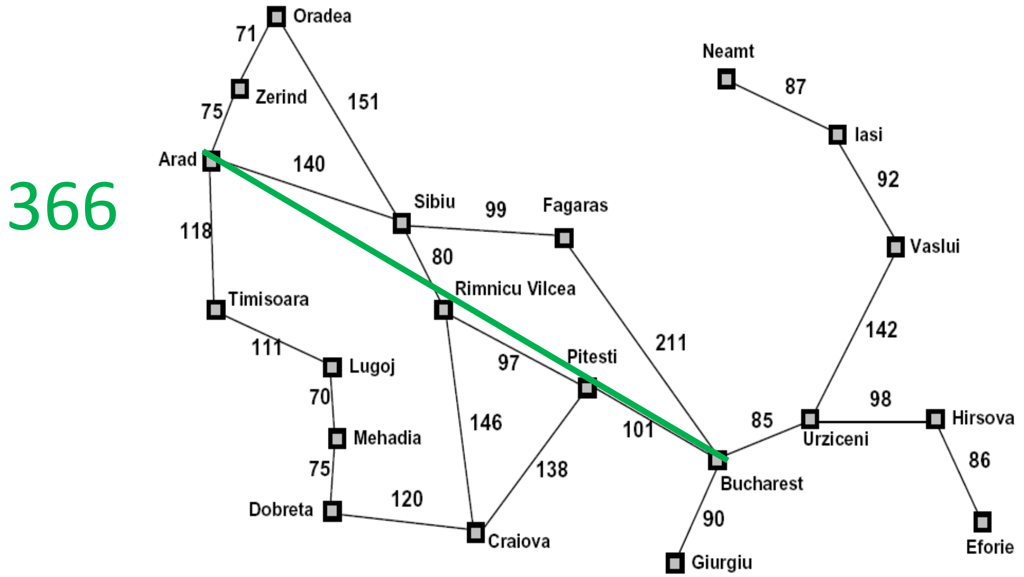
Creating Heuristics

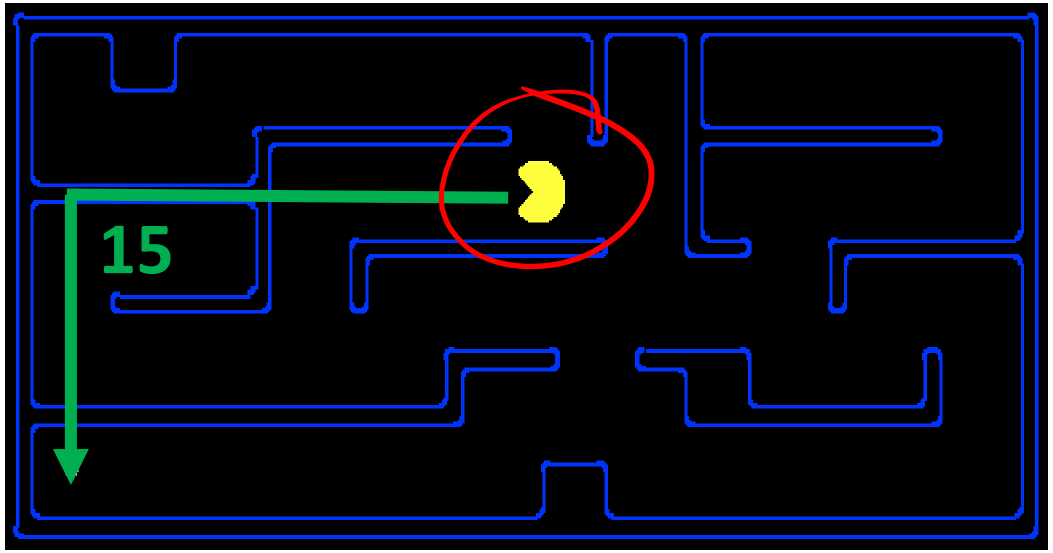


Creating Admissible Heuristics

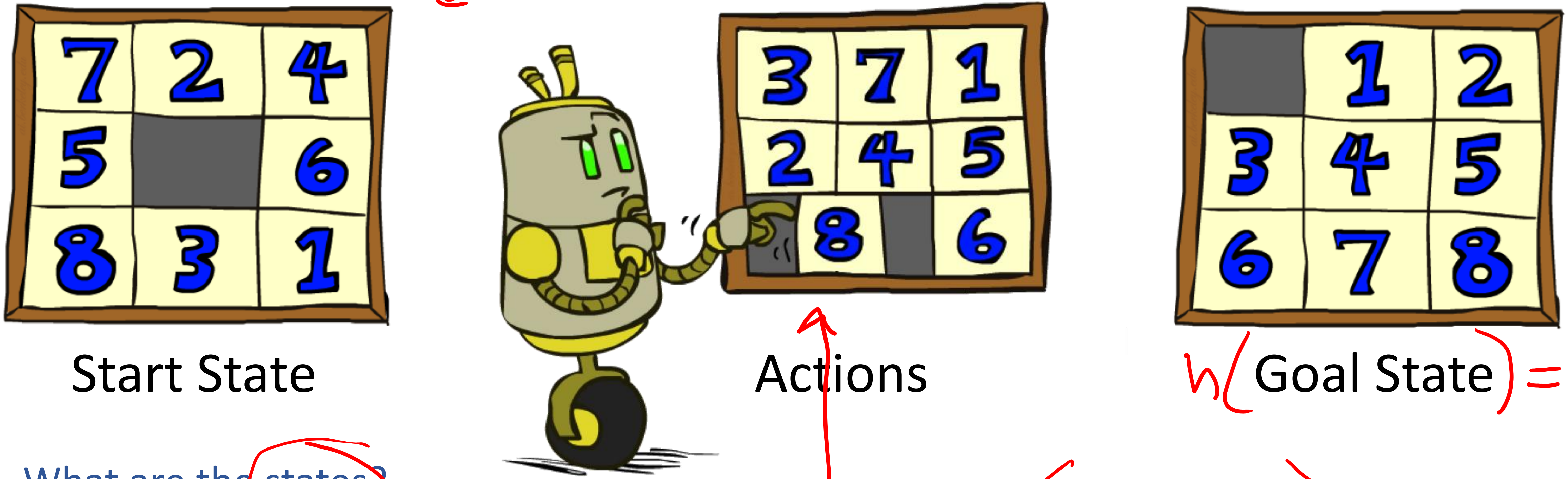
Most of the work in solving hard search problems optimally is in coming up with admissible heuristics

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





Example: 8 Puzzle



Start State

Actions

$h($Goal State$) = 0$

$cost(s, a, s') = 1$

What are the states?

How many states?

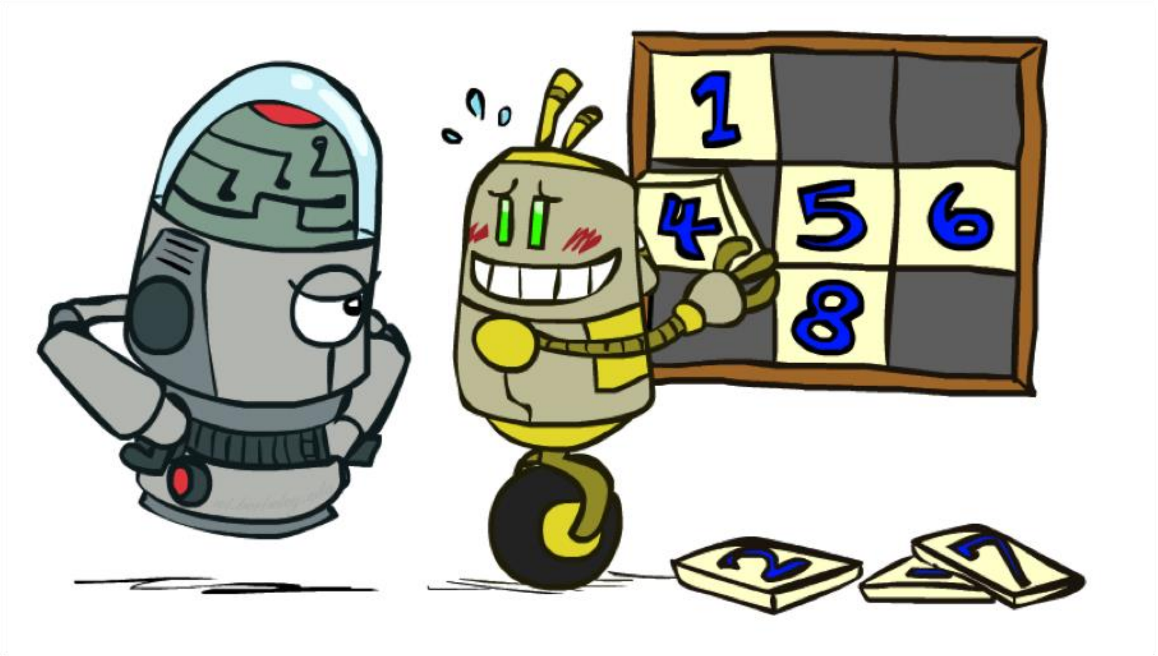
What are the actions?

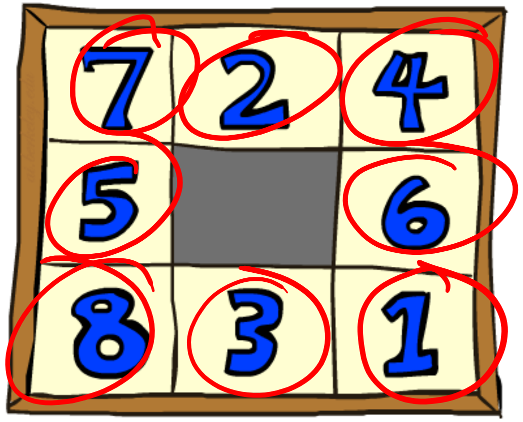
How many actions from the start state?

What should the step costs be?

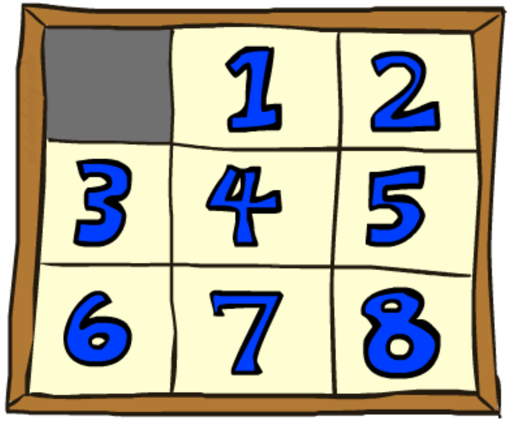
8 Puzzle I

- Heuristic: Number of tiles misplaced
- Why is it admissible?
- h(start) = 8
- This is a *relaxed-problem* heuristic





Start State



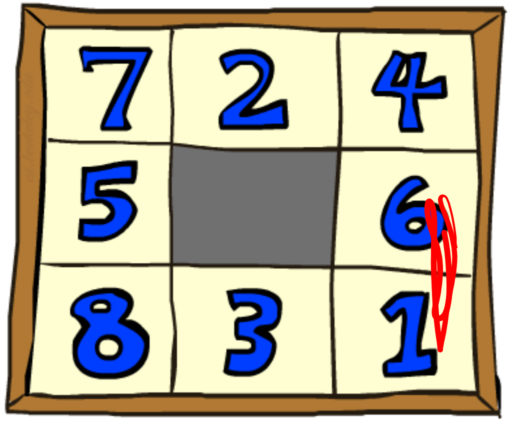
Goal State

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

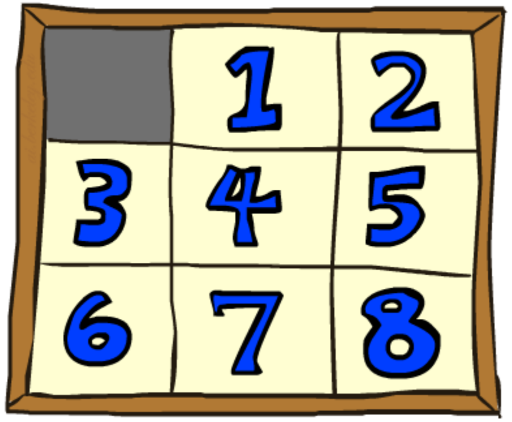
Statistics from Andrew Moore

8 Puzzle II

What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?



Start State



Goal State

Total *Manhattan* distance

Why is it admissible?

h(start) = 3 + 1 + 2 + … = 18

	Average nodes expanded when the optimal path has… 3×10^6		
UCS	…4 steps	…8 steps	…12 steps
A*TILES	13	39	227
A*MANHATTAN	12	25	73

Combining heuristics

Dominance: $h_a \geq h_c$ if

$$\boxed{\forall n} \; h_a(n) \geq h_c(n)$$

- Roughly speaking, larger is better as long as both are admissible
- The zero heuristic is pretty bad (what does A* do with h=0?) ← UCS
- 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:

$$h(n) = \max(h_a(n), h_b(n))$$

- Max of admissible heuristics is admissible and dominates both!

A*: Summary

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