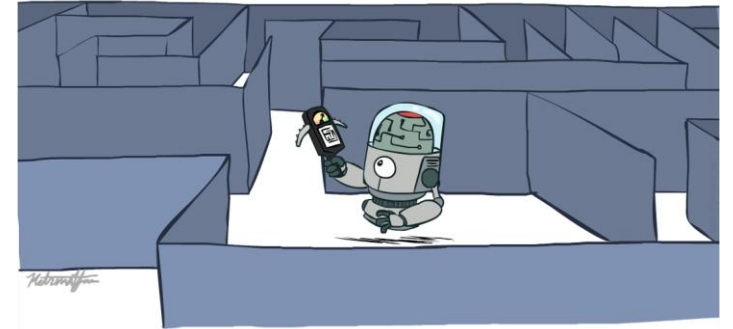


Informed Search

CSE 473: Introduction to Artificial Intelligence



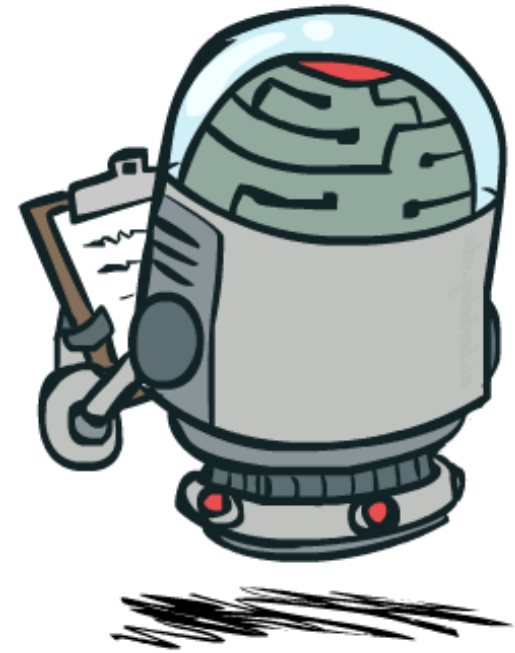
Administrivia

ANNOUNCEMENTS

- Past quizzes, exams available on course website [resources tab](#)

ASSIGNMENTS

- **Homework 1** due Monday (10/12)
- **Project 1** due next Monday (10/19)



Search

FOUNDATIONS



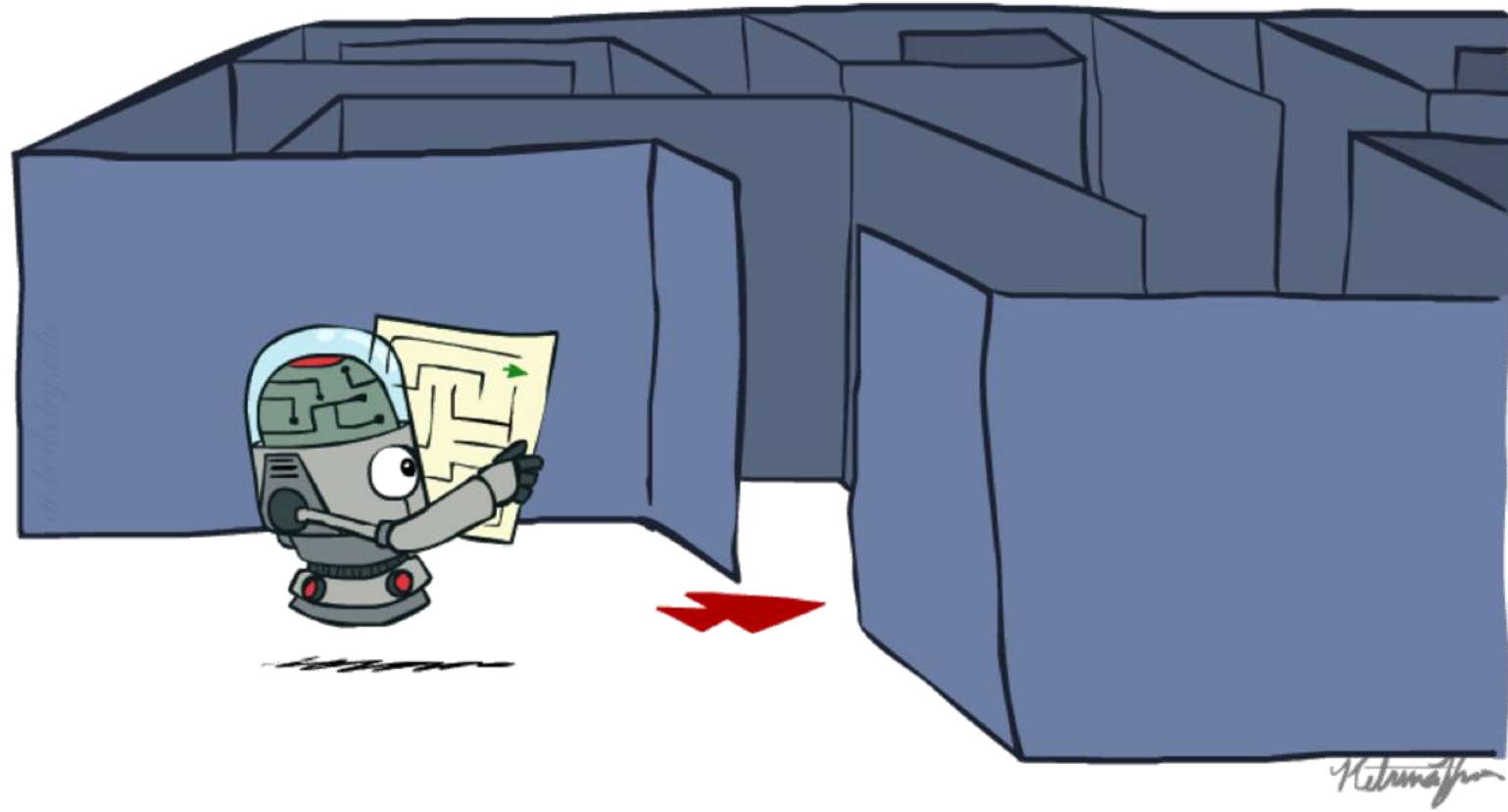
CSE 473

Breadth-First Search

4

Pacman Wayfinding

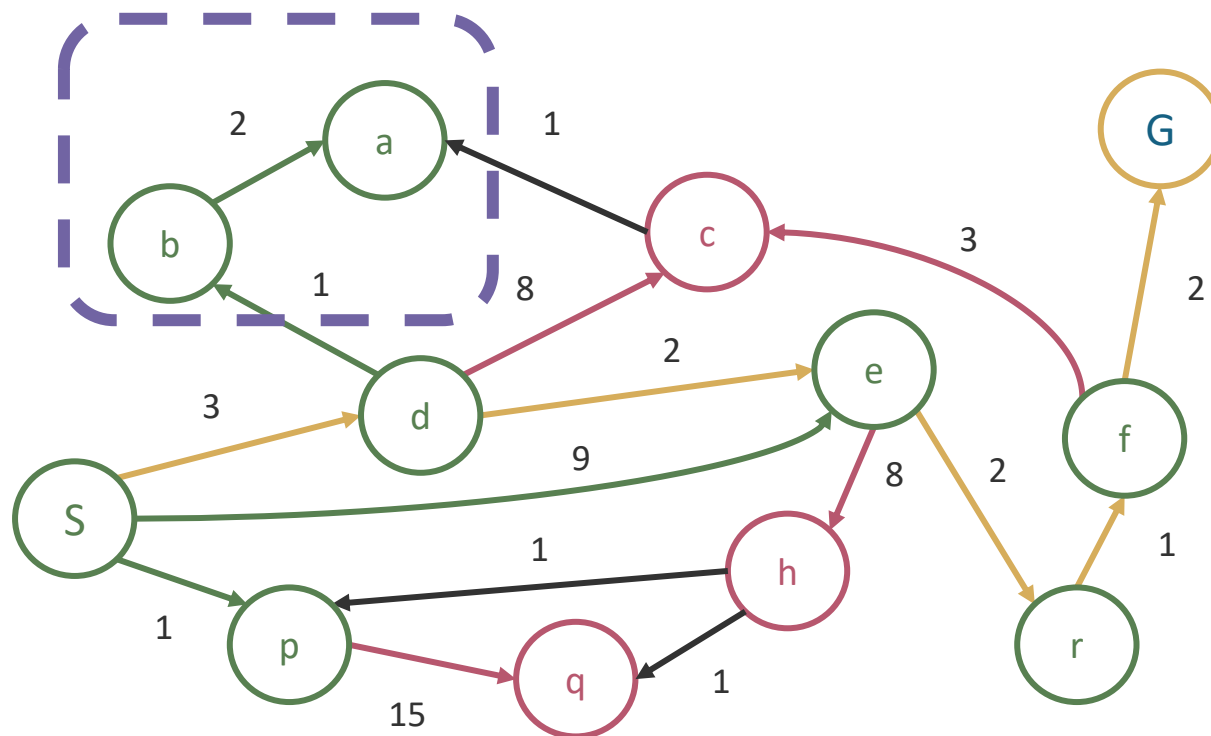
EXAMPLE



What's Next?

CONTEXT

dead end
can we avoid it?

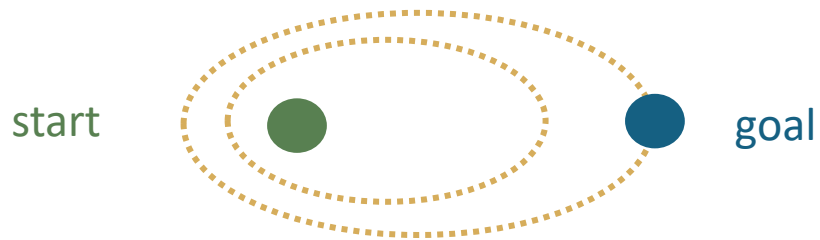


Directed Search

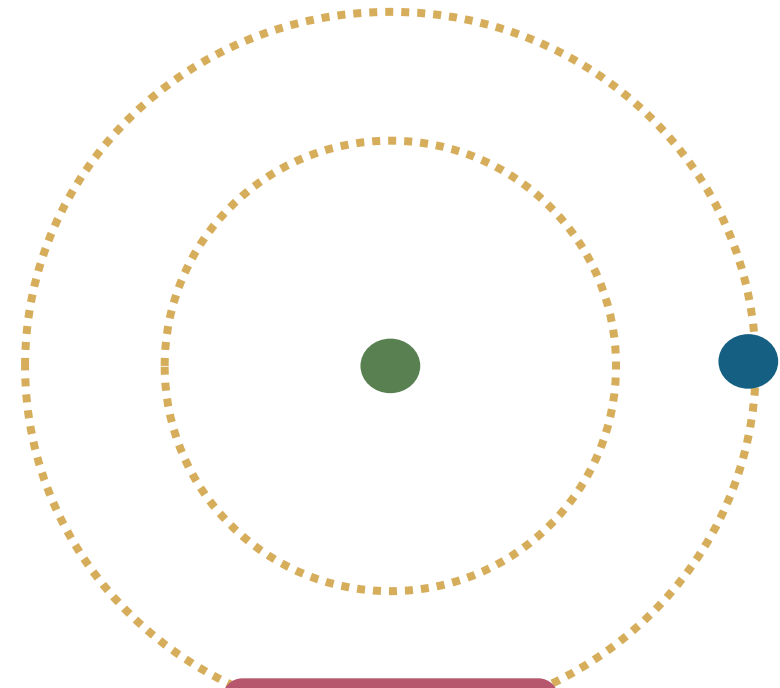


CONTEXT

Guide search in the *direction* of the goal, not in all directions



Informed

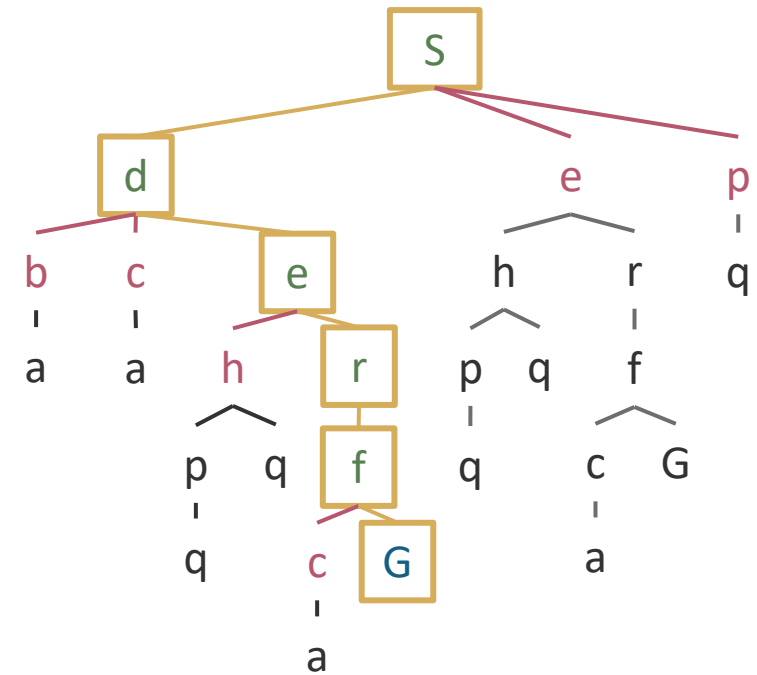
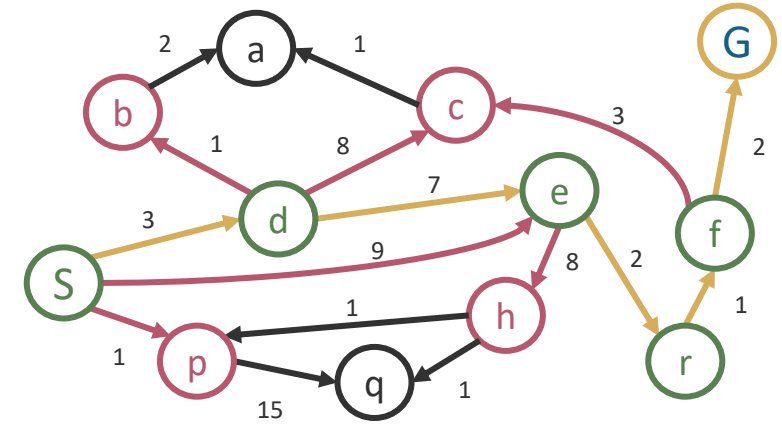


Uninformed

Greedy Search

FOUNDATIONS

- Strategy
 - Expand the node you *think* is closest to the goal
- Implementation
 - Use **heuristics** to estimate forward cost

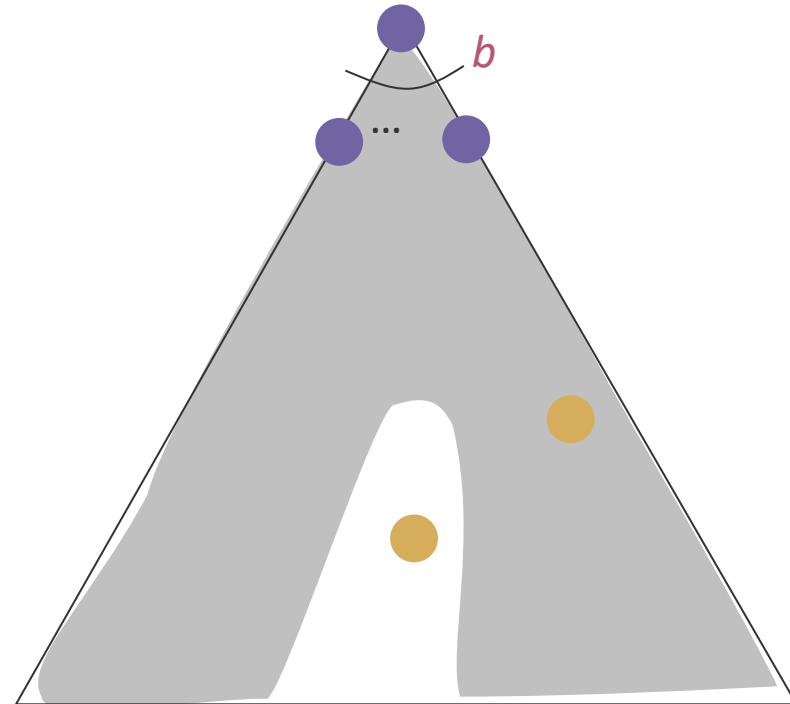


more directed, but not optimal

Properties of Greedy Search

DEFINITION

- Strategy
 - Expand the node you *think* is closest to the goal
 - Use **heuristics** to estimate forward cost
- Common Case
 - Expand straight to the (suboptimal) goal
- Worst Case
 - Search around optimal goal
 - Like a poorly-guided DFS

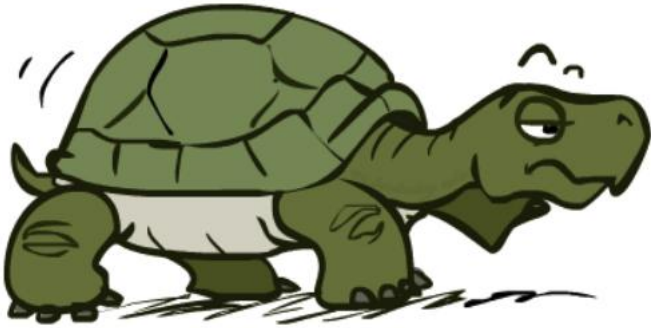


Questions?

We're getting closer, though...

Combining UCS and Greedy Search

CONTEXT



Uniform Cost Search



Greedy Search



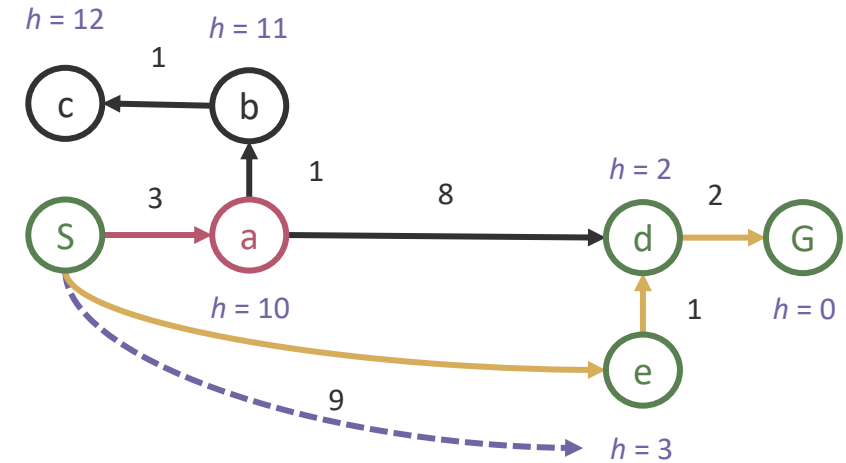
A* Search

A* Search

FOUNDATIONS

- Goal
 - Avoid unnecessary searching (and backtracking)
- Strategy
 - Expand the node most likely to be on the **optimal path**
 - Combine **UCS** and **Greedy Search**
- Implementation
 - $g(n)$ = cost from root to n
 - $h(n)$ = heuristic estimate of cost from n to goal
 - Use **priority queue** with $f(n) = g(n) + h(n)$

Key: good **heuristic**



Heuristics

DEFINITION

- We want to know the **forward cost $h^*(n)$** from n to the goal
 - If we know (and can quickly calculate) $h^*(n)$, our search is straightforward
 - ...but that doesn't happen (often) in the 'real world'...
- Instead, let's model a **simplified forward cost**
- A **heuristic** is a function that estimates proximity to goal

$$h(n) \approx h^*(n)$$

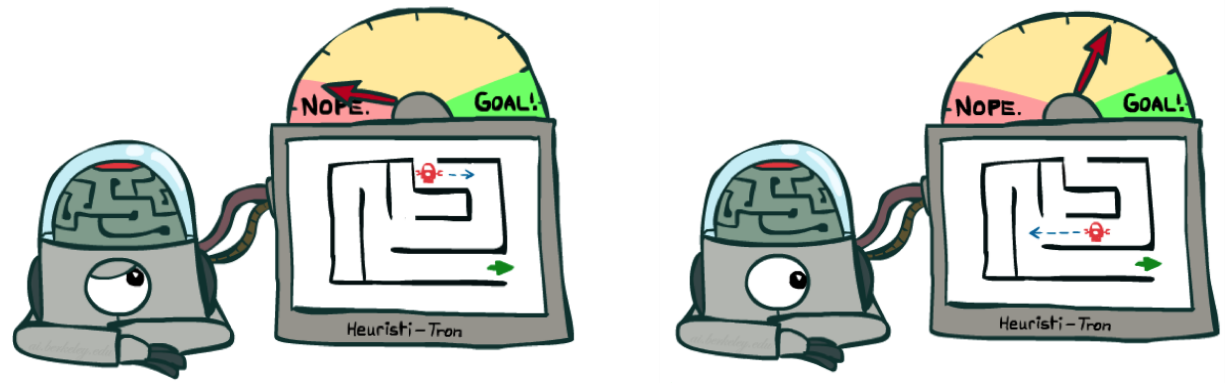
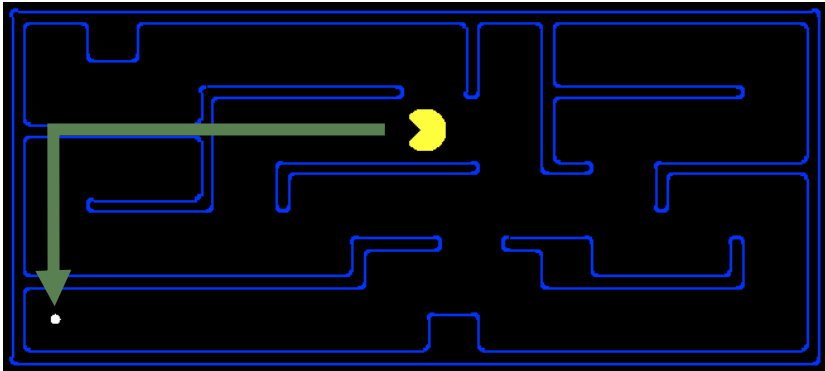
- Designed for the specific search problem
- Easy to calculate
- An *underestimate* of the true forward cost

Pacman Pathing

EXAMPLE

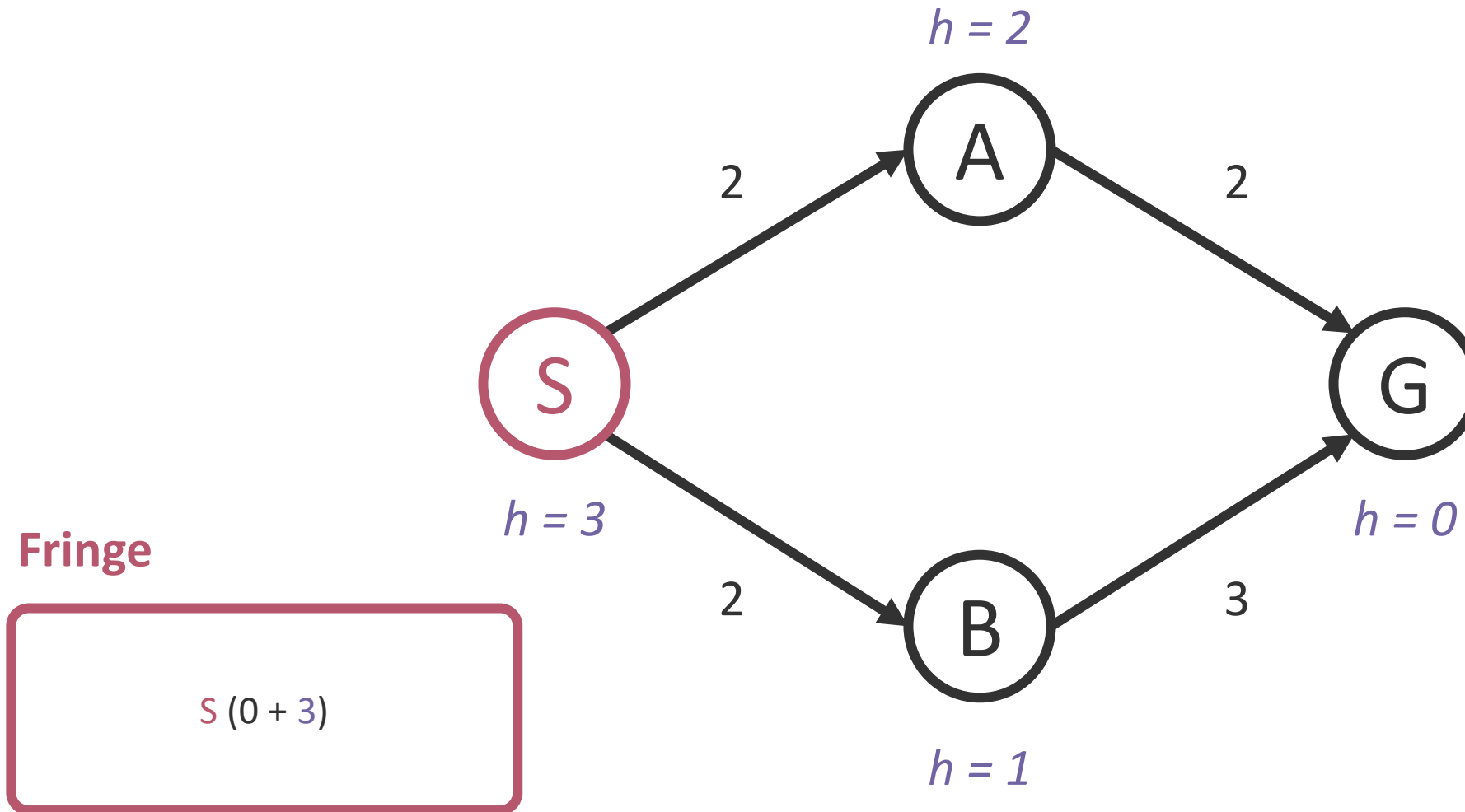
For Pacman maze navigation, **Manhattan distance** is one possible heuristic

- $\text{manhattan}(n) = |\Delta x| + |\Delta y|$
- Manhattan distance cannot overestimate the cost to get to the goal
- It likely is an underestimate (walls!)



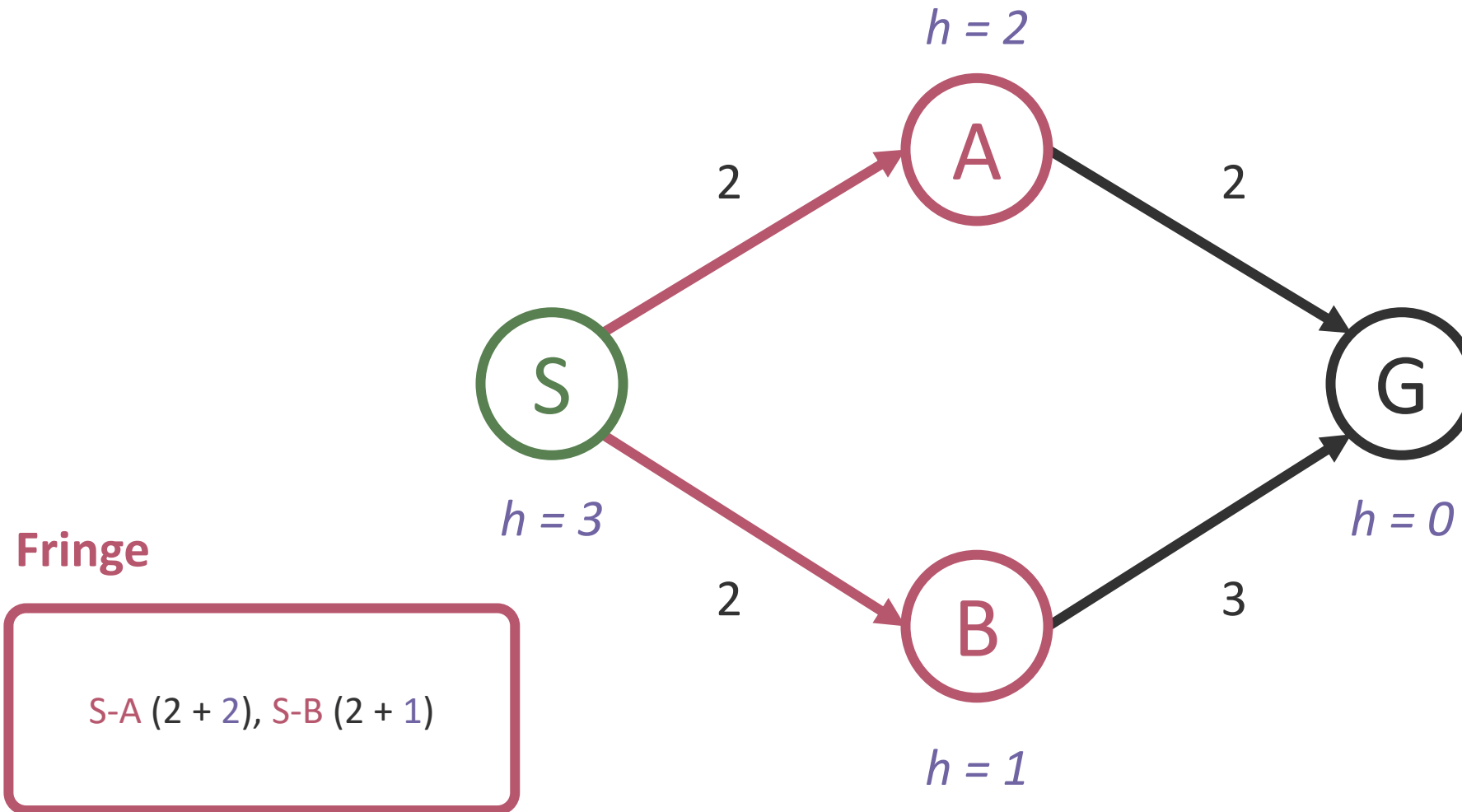
When to Dequeue Goal States? (1)

FOUNDATIONS



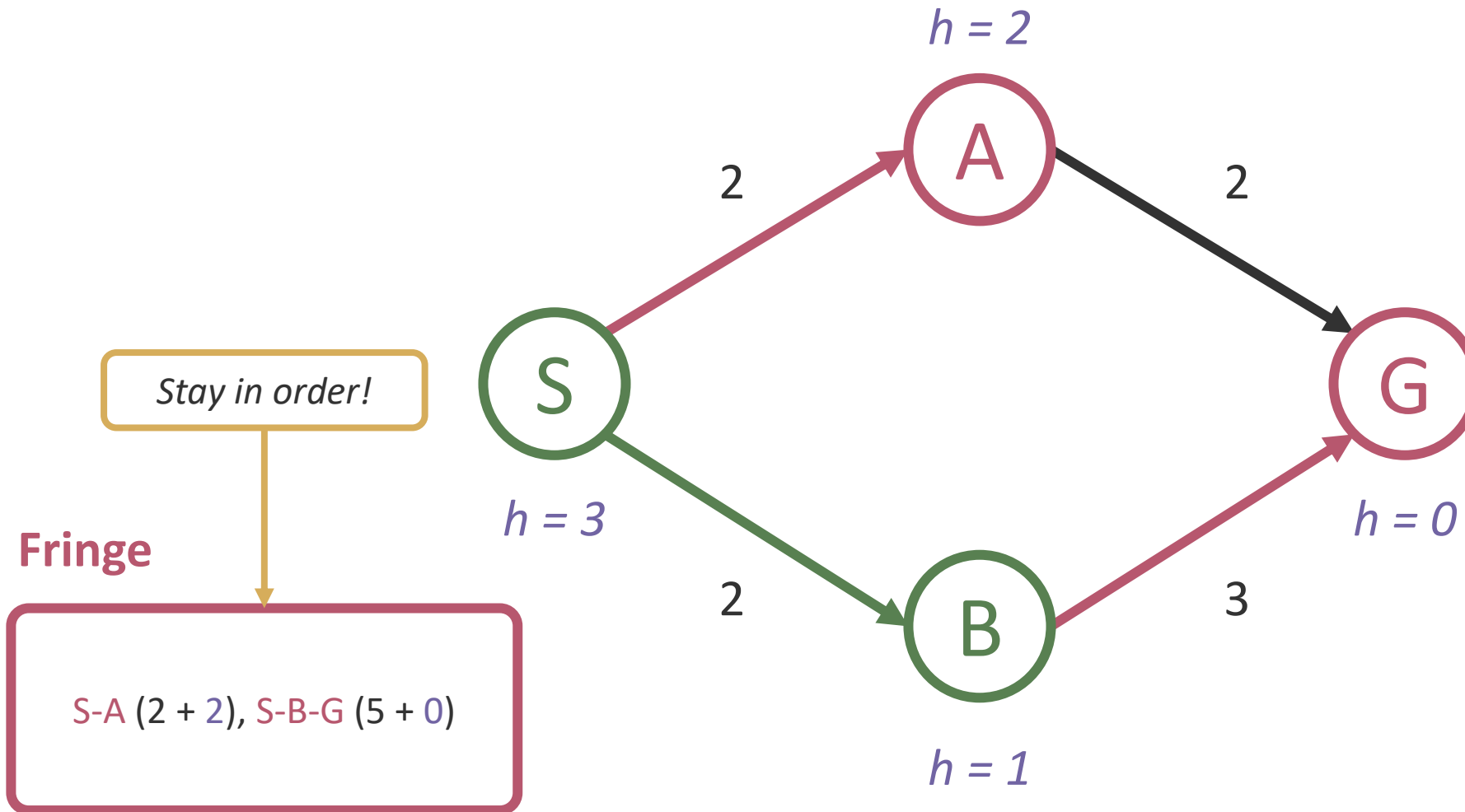
When to Dequeue Goal States? (2)

FOUNDATIONS



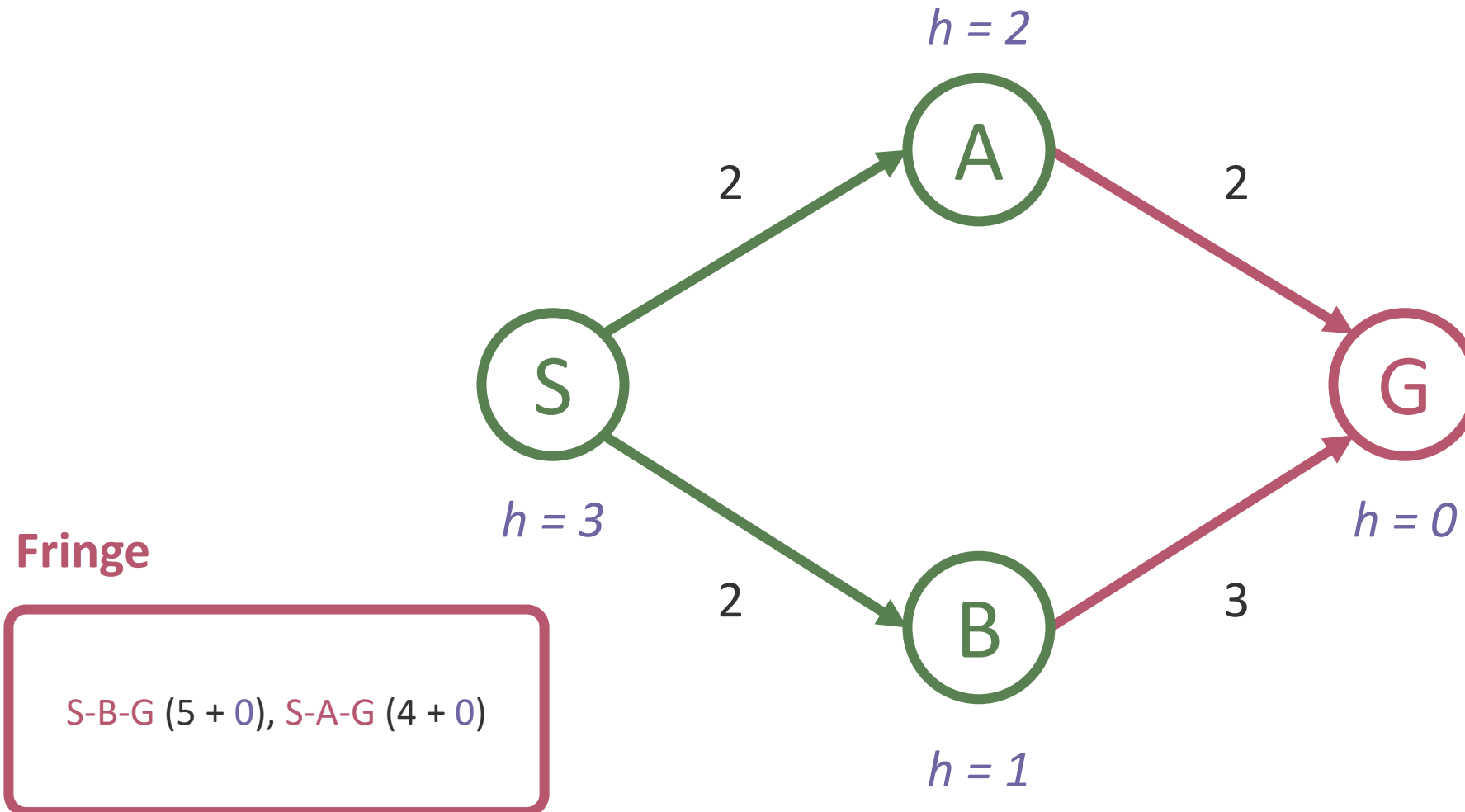
When to Dequeue Goal States? (3)

FOUNDATIONS



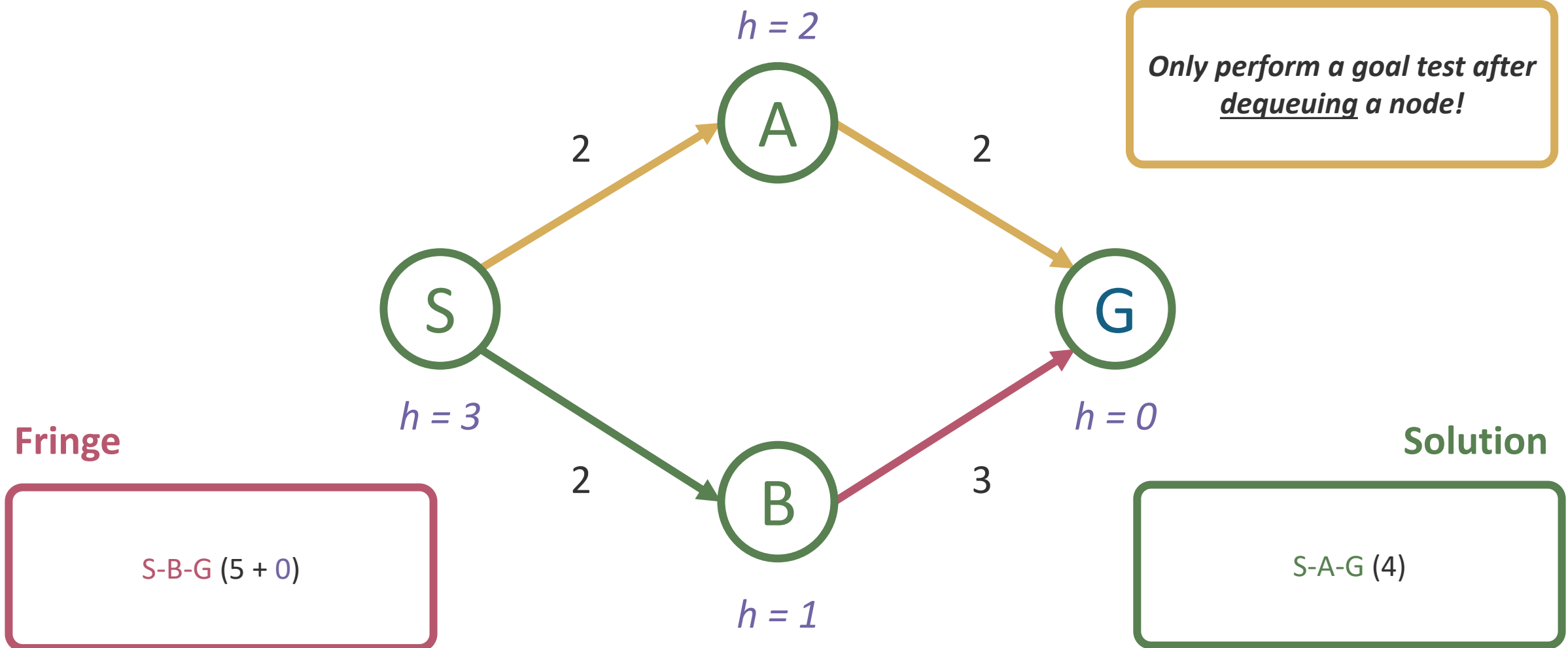
When to Dequeue Goal States? (4)

FOUNDATIONS



When to Dequeue Goal States? (5)

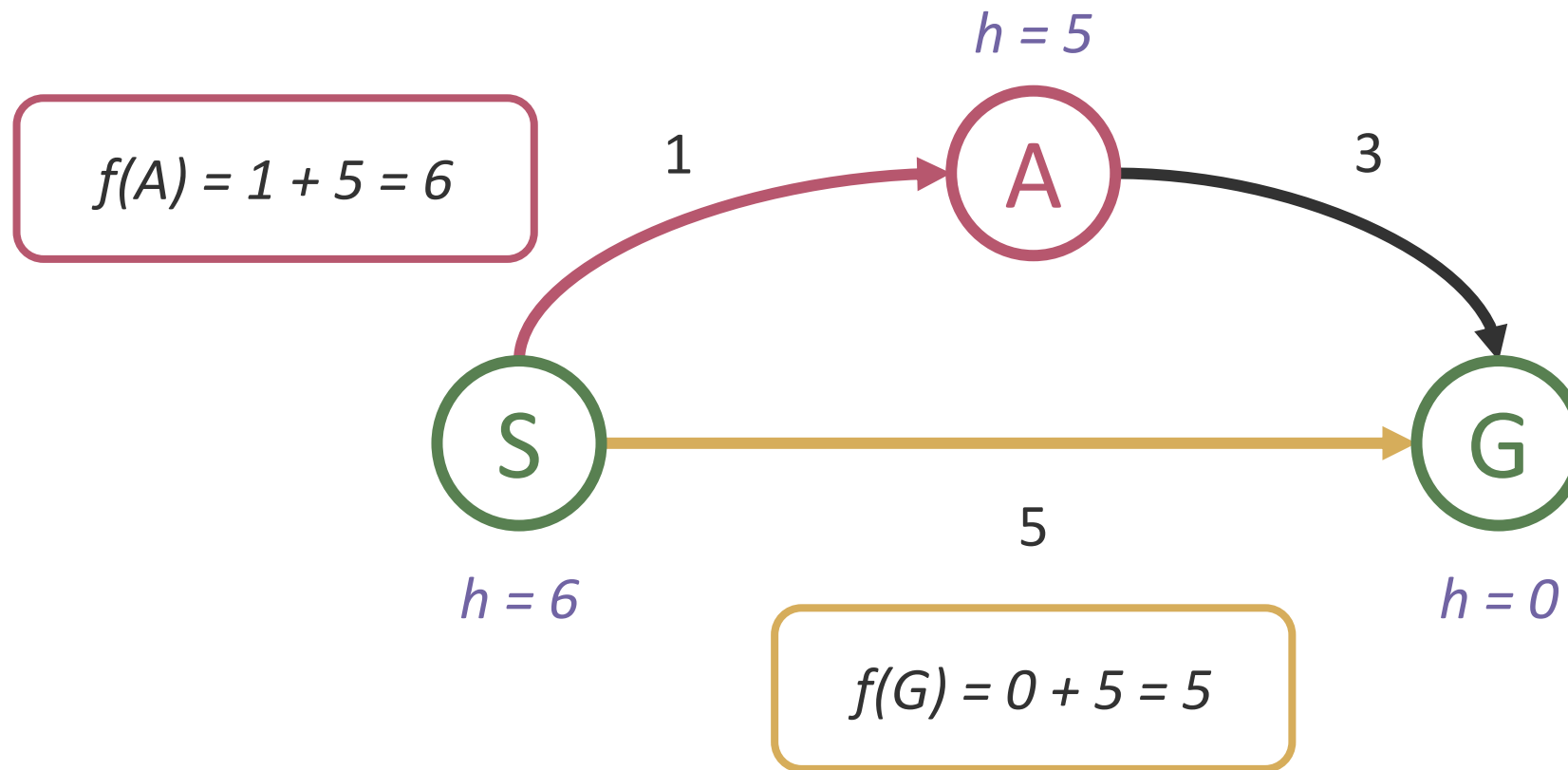
FOUNDATIONS



Questions?

Horrible Heuristics

CONTEXT



What went wrong?

- Heuristic made bad path seem better than good path
- Need $h(n) \leq h^*(n)$

Admissibility

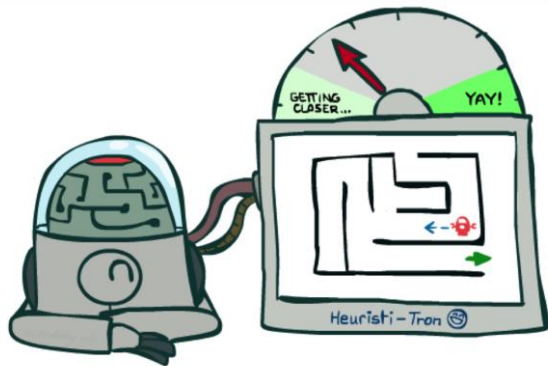
DEFINITION

Heuristics must be *underestimates* of the true cost for A* to be optimal

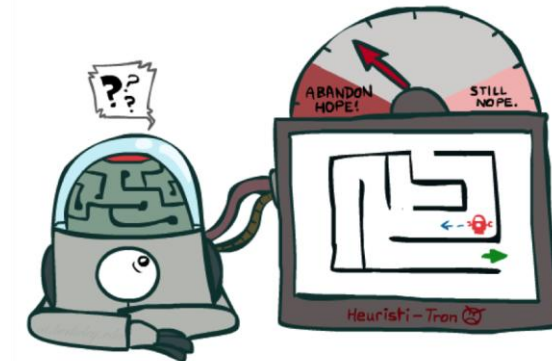
- An **admissible heuristic** is optimistic about cost to goal

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

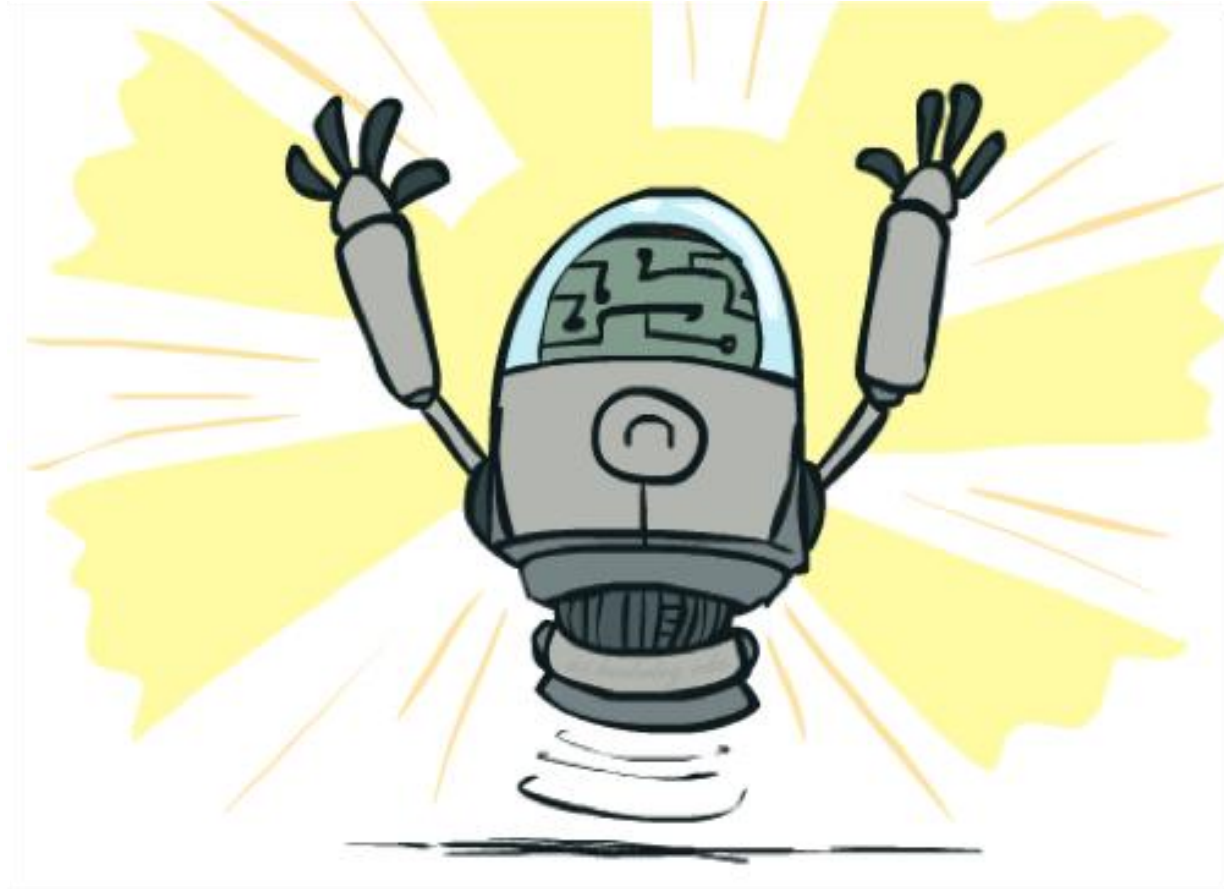
$h^*(n)$ is the true cost from n to the goal



Admissible heuristics
slow down bad paths
but never outweigh
the true cost



Inadmissible heuristics
trap good
plans on the fringe,
breaking optimality

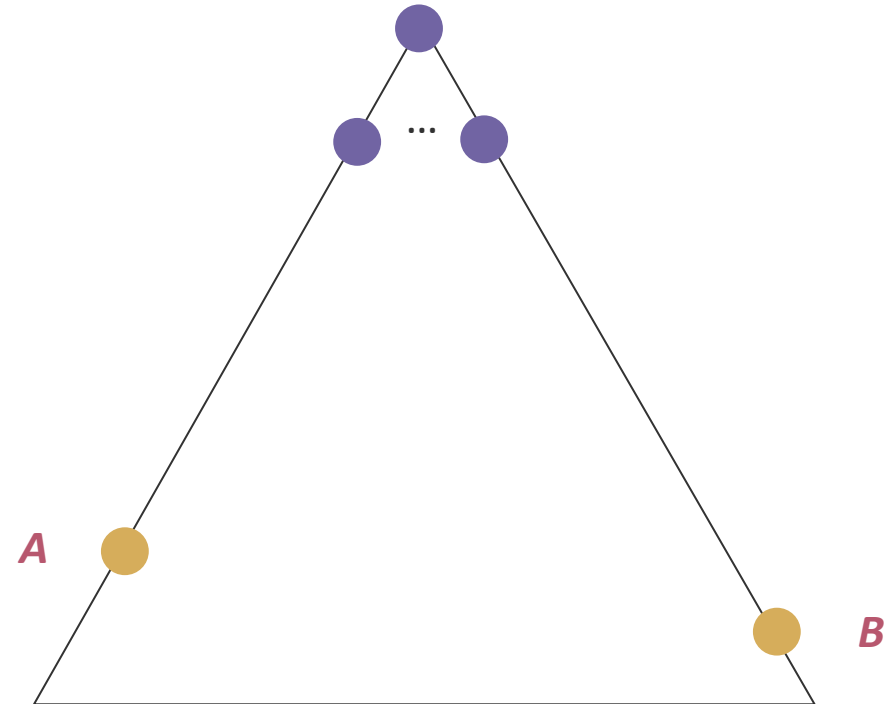


Claim: A* Search is Optimal!

Optimality of A* Tree Search

PROOF

- 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

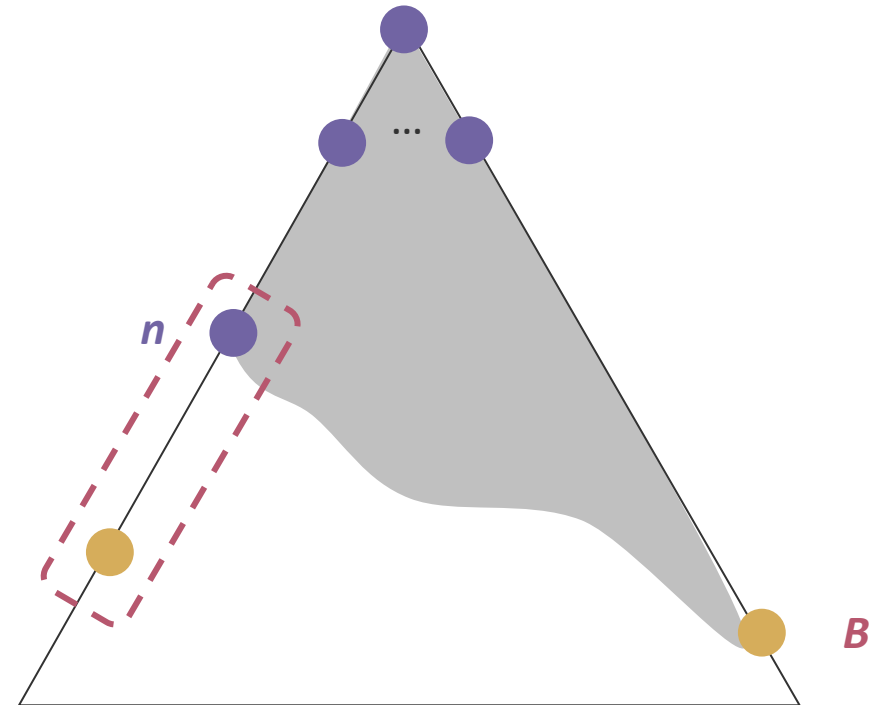


Optimality of A* Tree Search (1)

PROOF

- Proof
 - Let B be on the fringe
 - Some ancestor n of A is on the fringe, too
 - (maybe A itself)
- Claim
 - n will be expanded before B
- Proof
 - $f(n) \leq f(A)$

$$\begin{aligned} f(n) &= g(n) + h(n) \\ f(n) &\leq g(A) \text{ because } h \text{ is admissible} \\ g(A) &= f(A) \\ \rightarrow f(n) &\leq f(A) \end{aligned}$$



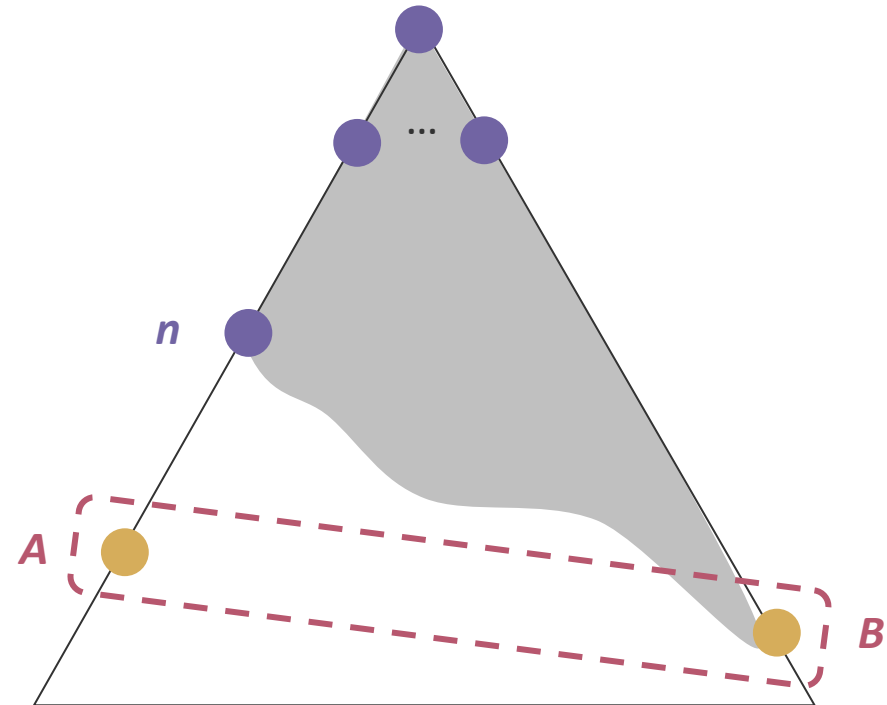
Optimality of A* Tree Search (2)

PROOF

- Proof
 - Let B be on the fringe
 - Some ancestor n of A is on the fringe, too
 - (maybe A itself)
- Claim
 - n will be expanded before B

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

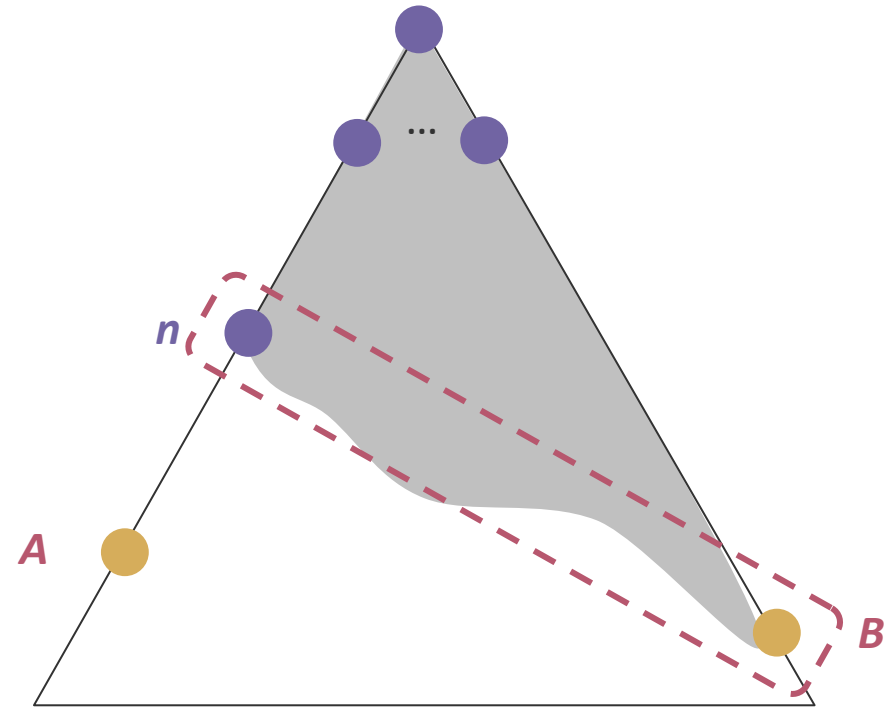
$g(A) < g(B)$ B is suboptimal
 $\rightarrow f(A) < f(B)$ $h = 0$ at goal



Optimality of A* Tree Search (3)

PROOF

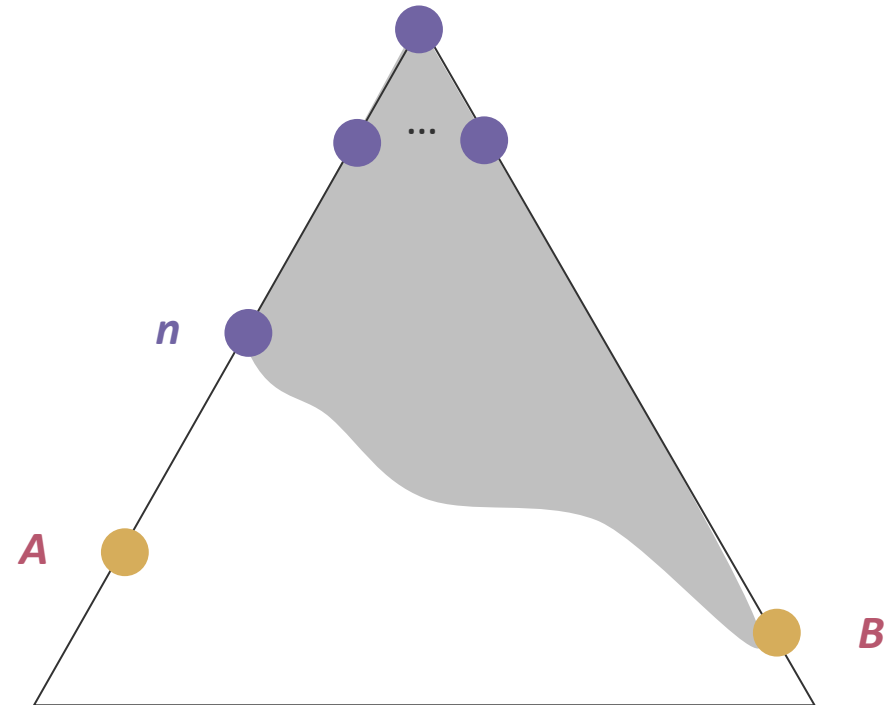
- Proof
 - Let B be on the fringe
 - Some ancestor n of A is on the fringe, too
 - (maybe A itself)
- Claim
 - n will be expanded before B
- Proof
 - $f(n) \leq f(A)$
 - $f(A) \leq f(B)$
 - $\rightarrow f(n) \leq f(A) < f(B)$



Optimality of A* Tree Search (4)

PROOF

- Proof
 - Let B be on the fringe
 - Some ancestor n of A is on the fringe, too
 - (maybe A itself)
- Proof
 - $\rightarrow f(n) \leq f(A) < f(B)$
- Conclusion
 - All ancestors of A are expanded before B
 - Therefore A itself is expanded before B
 - Therefore **A* tree search is optimal**

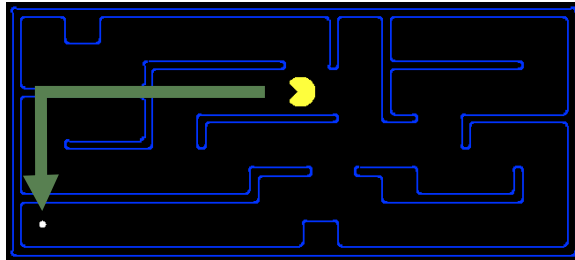


Questions?

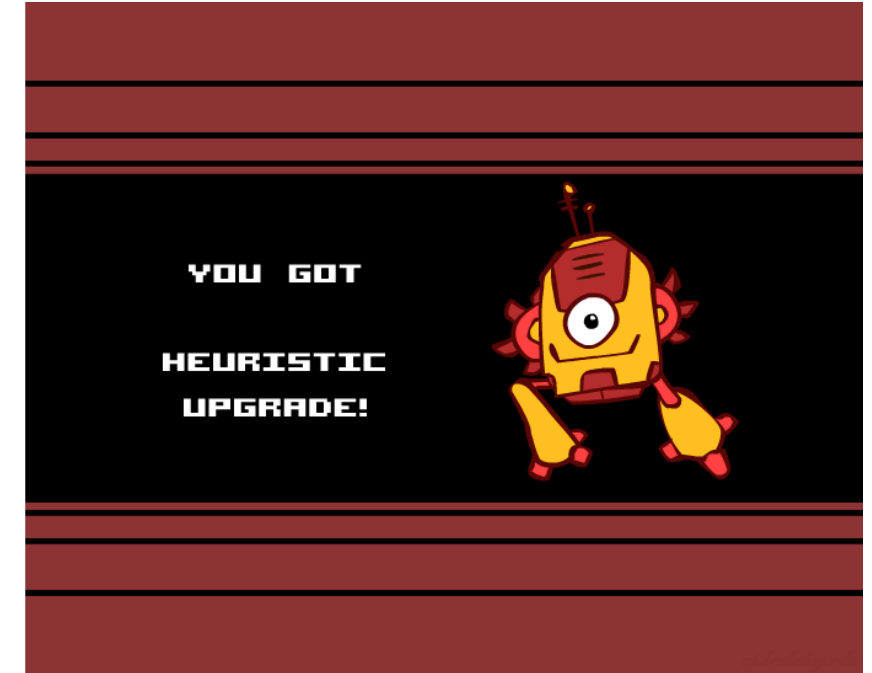
Creating Heuristics

CONTEXT

- Most of the work in solving hard search problems involves finding **admissible heuristics**
- Often, admissible heuristics are solutions to *relaxed problems* where new actions are available



- Inadmissible heuristics can be useful too

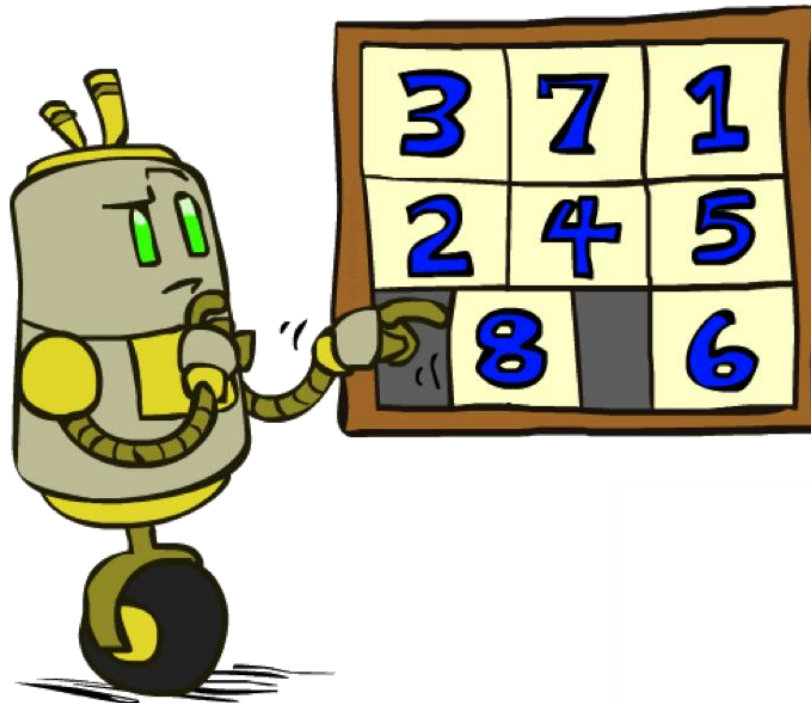


8 Puzzle

EXAMPLE

7	2	4
5		6
8	3	1

Start State



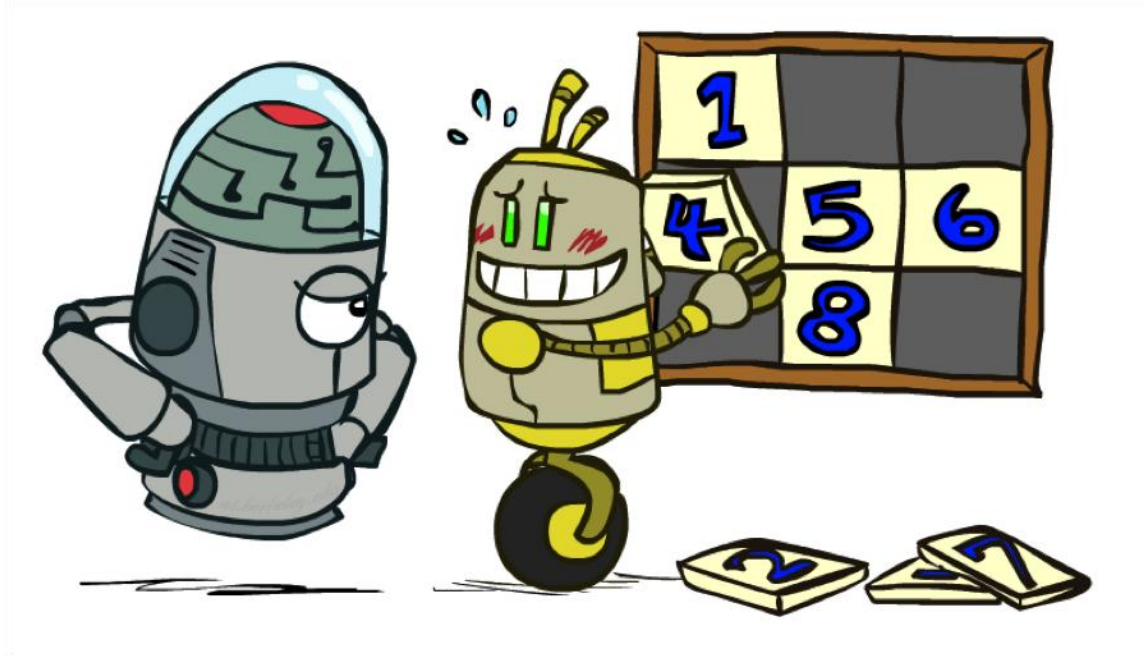
	1	2
3	4	5
6	7	8

Goal State

8 Puzzle: Heuristic

EXAMPLE

- **Heuristic:** number of misplaced tiles
 - *Admissible?* Yes, because putting tiles in correct position requires at least as many moves



Start State

7	2	4
5		6
8	3	1

Goal State

	1	2
3	4	5
6	7	8

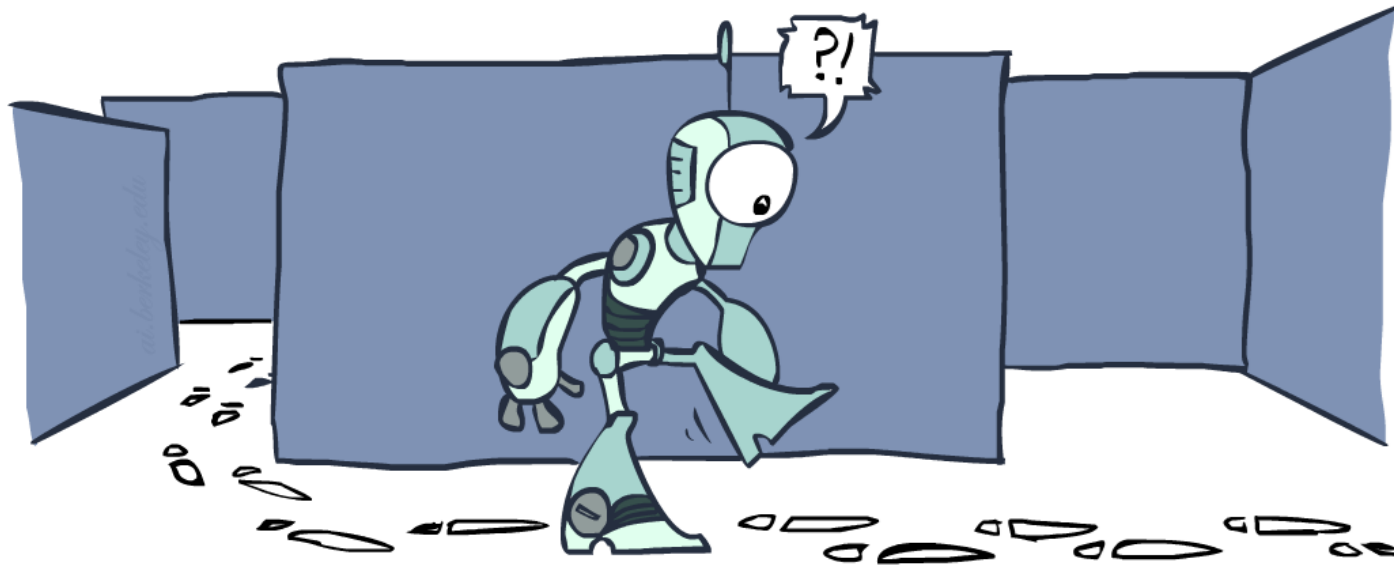
General Graph Search



ALGORITHM

```
function GRAPH-SEARCH(problem, strategy) returns solution or failure
  closed ← empty set
  initialize fringe using initial state of problem
  loop do
    if fringe empty then return failure
    node ← remove node from fringe
    if node is goal state then return corresponding solution
    if node not in closed then
      expand node and add resulting nodes to fringe
      add node to closed
  end
```

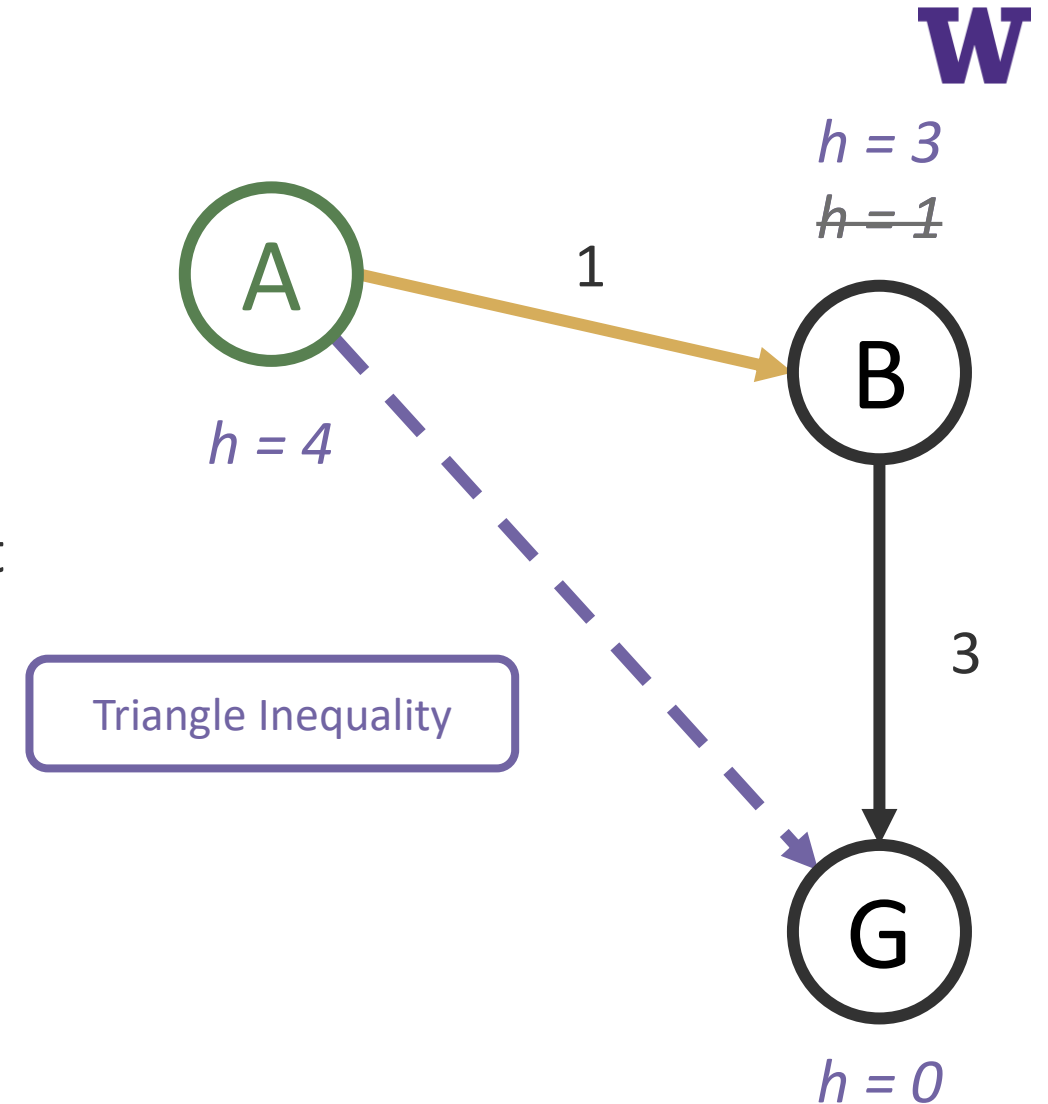
Is A^* Graph Search Optimal?



Consistency

DEFINITION

- We want heuristic costs $\leq$ actual costs
 - **Admissibility** guarantees heuristic cost $\leq$ actual cost to goal
- We also want heuristic “arc” cost $\leq$ actual arc cost
 - **Consistency** guarantees this
$$h(A) - h(B) \leq \text{cost}(A, B)$$
$$h(A) \leq \text{cost}(A, B) + h(B)$$
- Consequences
 - f value along a path never decreases
 - A* graph search is optimal
 - Consistency implies admissibility



Questions?

Search Algorithms

- Depth First – memory efficient, good for solutions at maximum depth
- Breadth first – optimal
- Uniform cost search – optimal if cost varies
 - (BFS is just UCS with cost of always 1)
- A* search – adds heuristic to guide search direction
 - (UCS is A* with a heuristic of 0!)
- Graph searches maintain a list of expanded nodes and don't re-expand
 - Implementing graph searches can prevent cycles from occurring.
- How does all this relate to Dijkstra's algorithm?

that's it for today!

SUMMARY

- Informing the direction of our search with heuristics
- Consequences of heuristic admissibility and consistency

UPCOMING

- Multi-Agent Search

REMINDERS

- Hand-in **Practice Problem**
- Do **Homework 1**
- Do **Project 1**