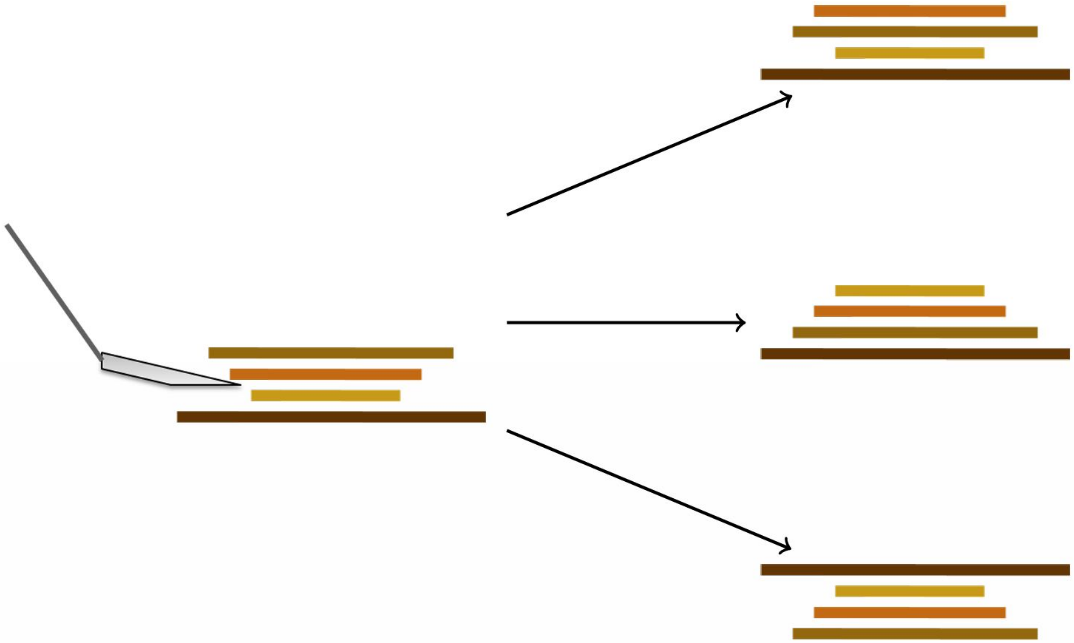
Informed Search

CSE 4617: Artificial Intelligence

Isham Tashdeed
Lecturer, CSE

Search Problems

- State?
- Start state?
- Goal test?
- Successor Function?
- Cost?
- Total number of states?

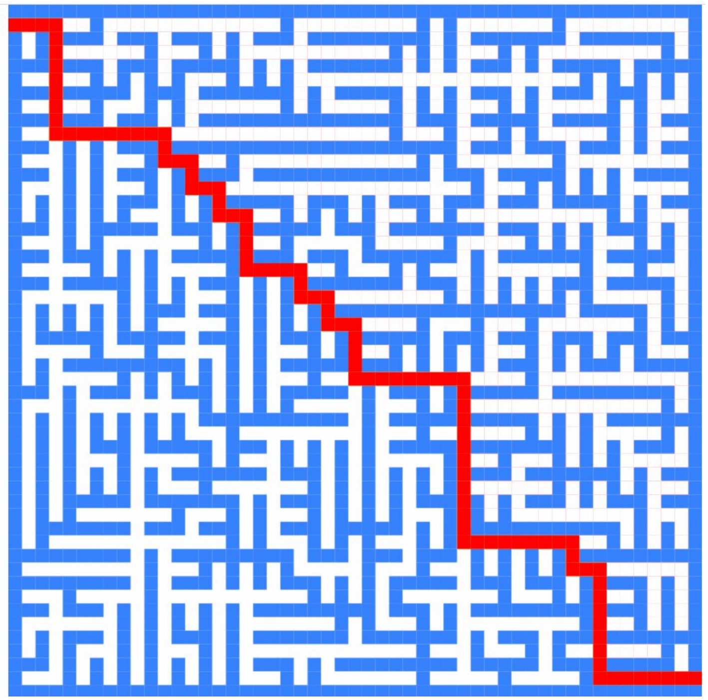


General Tree Search

```
TREE-SEARCH (problem, strategy) → returns a solution or failure
    initialize tree search using the initial state of the problem
    loop do:
        if there are no candidates for expansion:
            then return failure

        choose a leaf node according to strategy
        if chosen node contains a goal state:
            then return the solution
        else:
            Expand the node & add the resulting nodes to the search tree
    end
```

Issues with Search

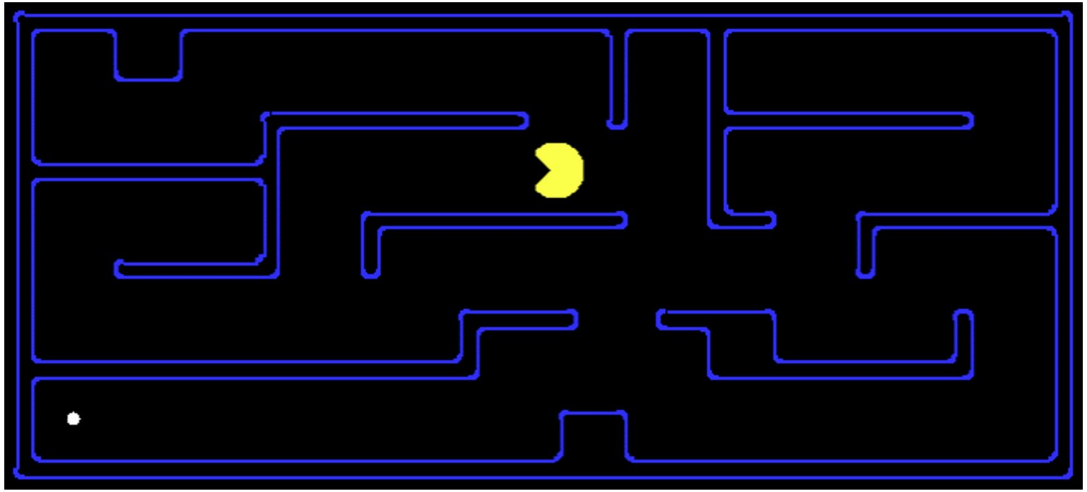
- It explores in every "direction" as long as the costs are lower
- No information about where the goal state is located



How could we infuse the idea of where the goal is located?

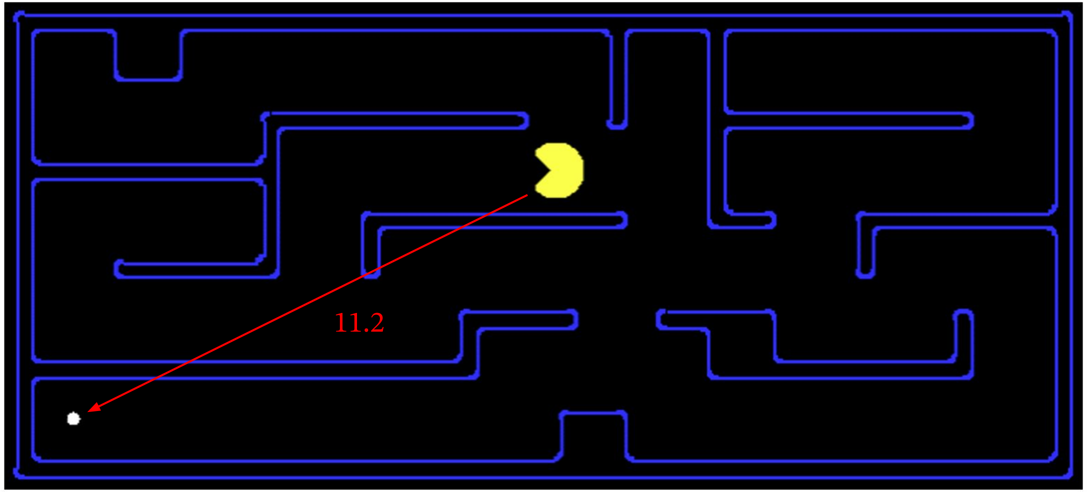
Search Heuristics

- A function that *estimates* how close a state is to the goal state
- Designed for a specific problem → A heuristic is not universal
- Gives an idea of the cost needed to reach the goal from current state



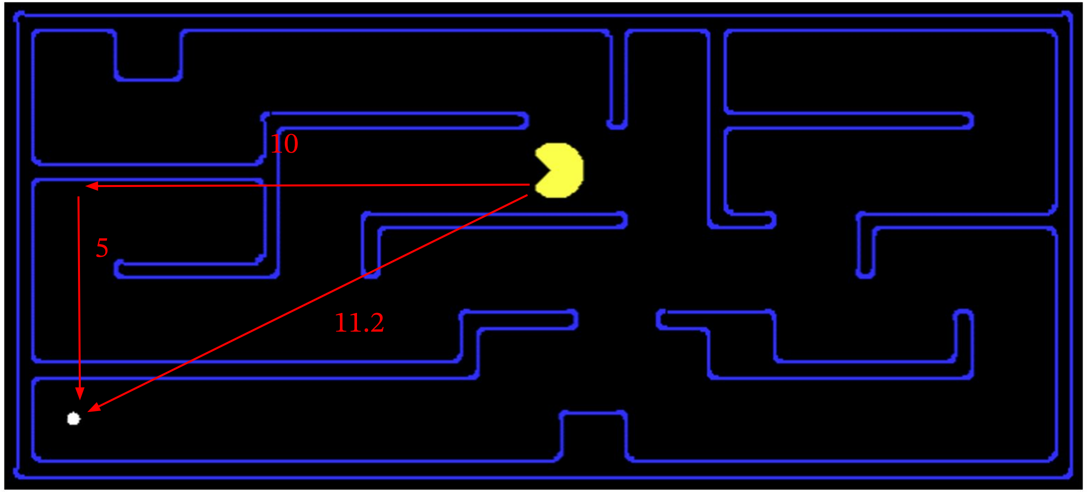
Search Heuristics

- A function that *estimates* how close a state is to the goal state
- Designed for a specific problem → A heuristic is not universal
- Gives an idea of the cost needed to reach the goal from current state

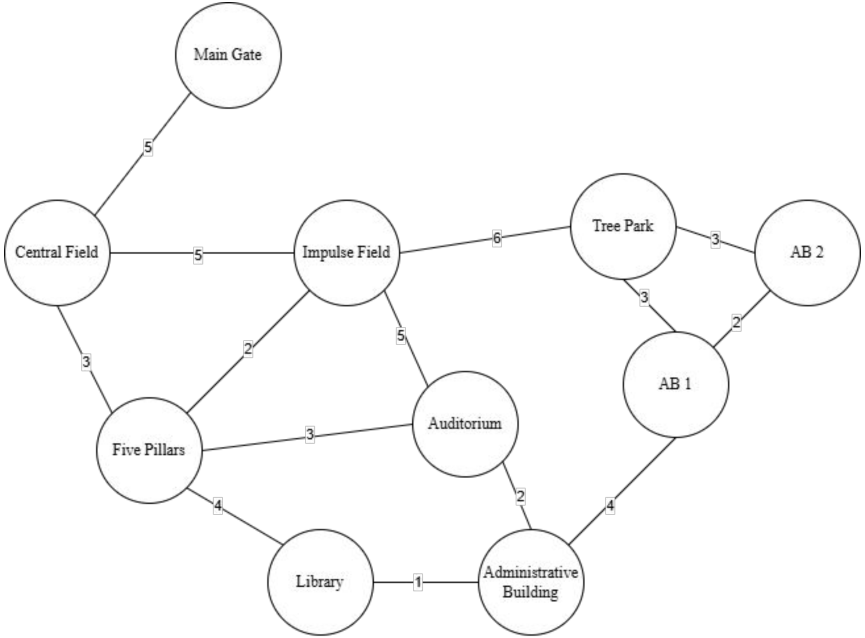


Search Heuristics

- A function that *estimates* how close a state is to the goal state
- Designed for a specific problem → A heuristic is not universal
- Gives an idea of the cost needed to reach the goal from current state

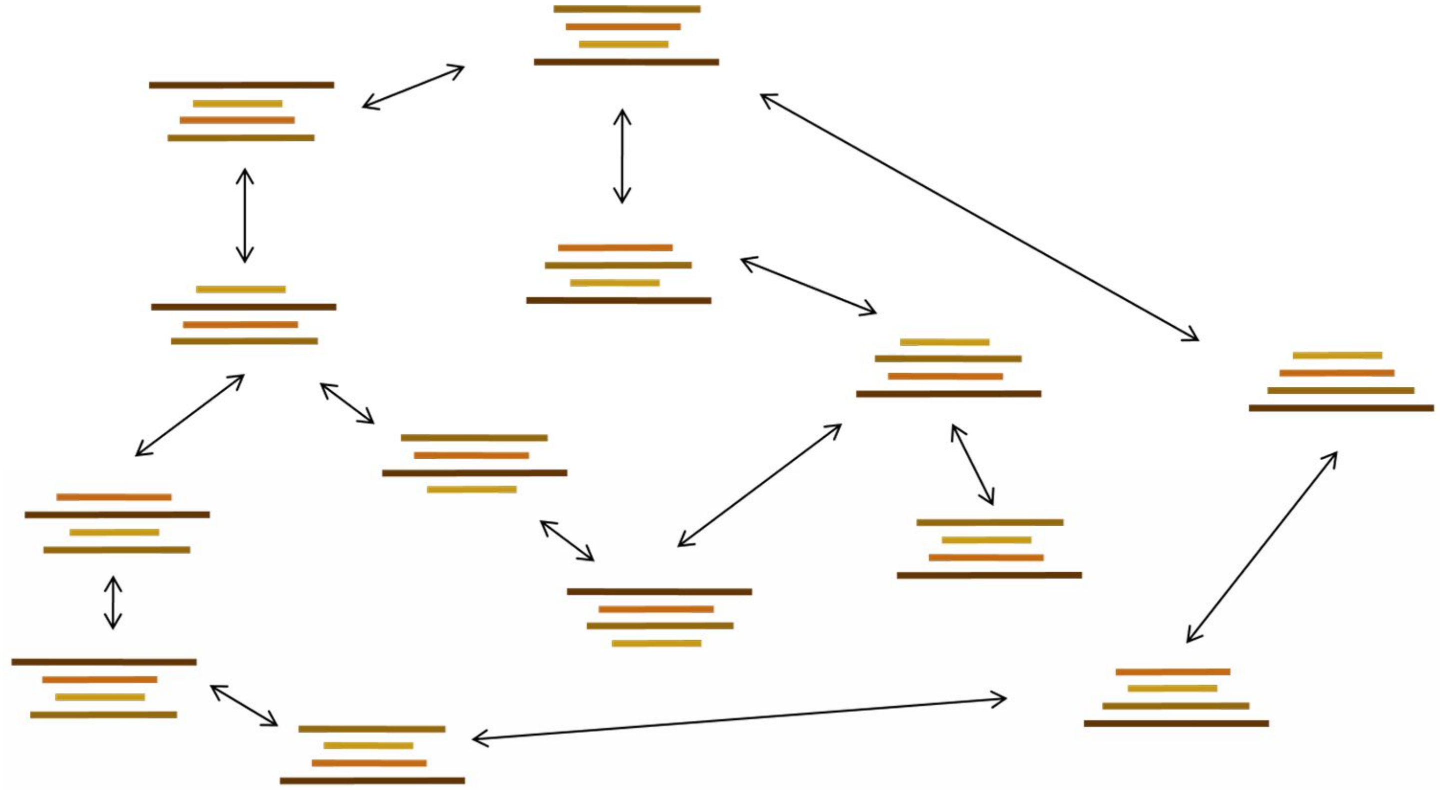


Search Heuristics

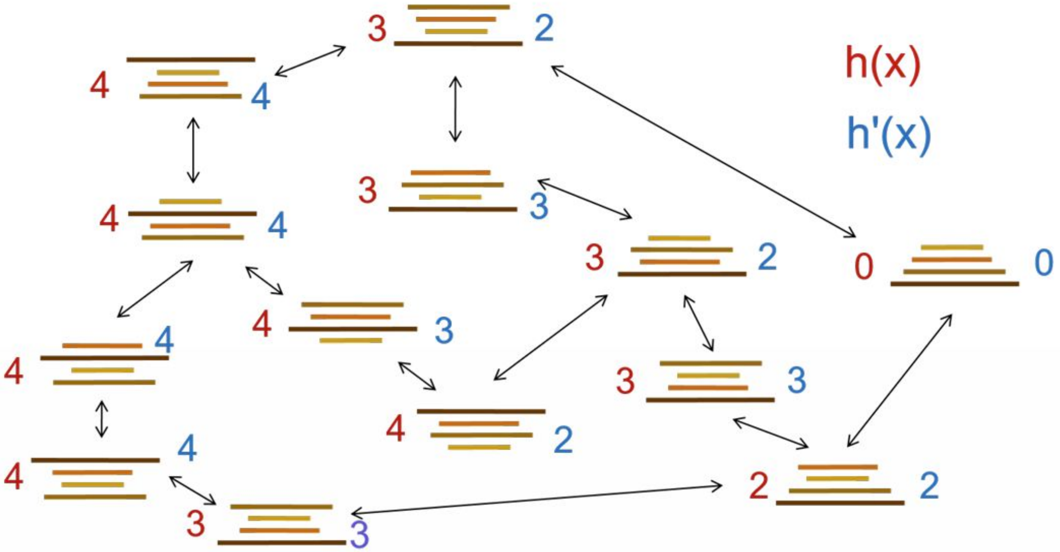


- What could be a good heuristic for this problem?
- What even is a *good* heuristic?

Search Heuristics



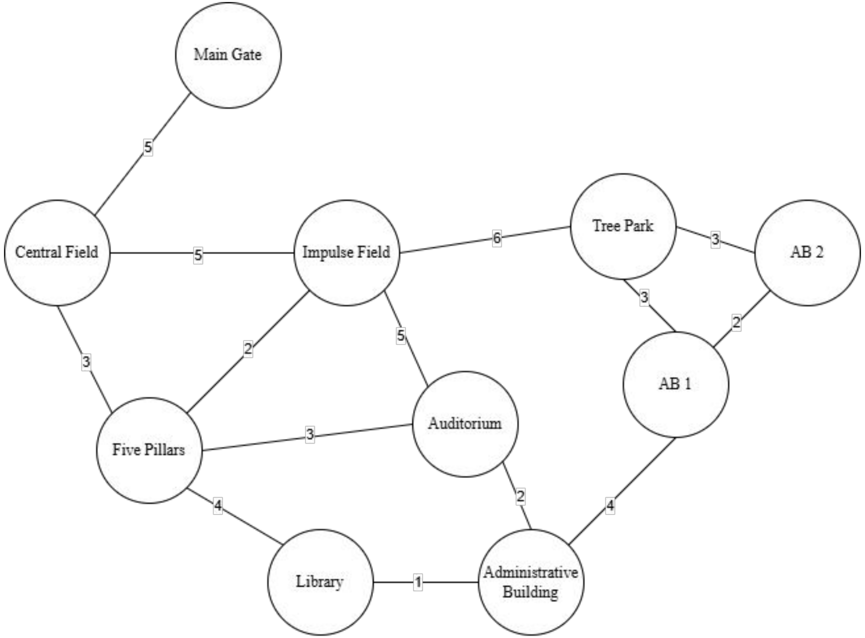
Search Heuristics



$h(x)$ = The ID of the largest pancake that is still out of place
$h'(x)$ = The number of the incorrectly placed pancakes

Greedy Search

Greedy Search



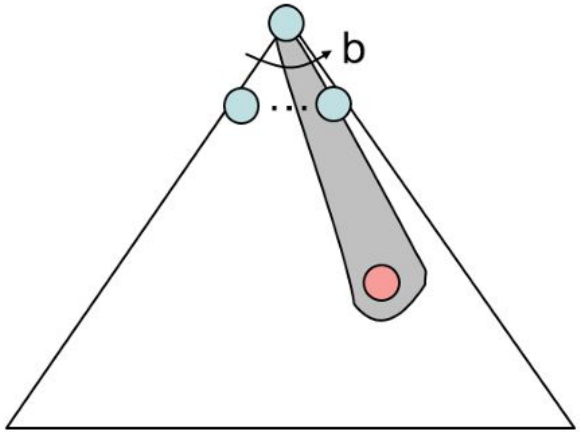
- Expand the node that takes you closer to the goal
- Heuristic → Estimate of cost to
- Which nodes will be expanded?
- How is it different from the other search methods?
- What could be an apparent issue?

Greedy Search

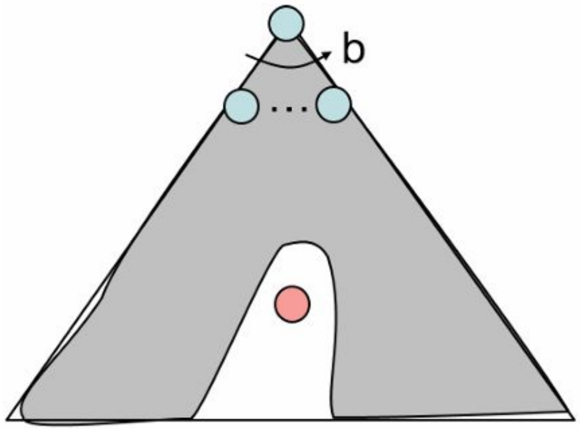
```
Greedy_Search (problem) → returns a solution or failure
    initialize tree search using the initial state of the problem
    loop do:
        if there are no candidates for expansion:
            then return failure

        choose the node with the least heuristic cost
        if chosen node contains a goal state:
            then return the solution
        else:
            Expand the node & add the resulting nodes to the search tree
    end
```

Greedy Search Properties

- Is it complete?
    - Only if *m* is finite
- Is it optimal?
    - No
    - Finds the solution from heuristic cost
- Time complexity?
    - At worst case, needs to process the whole tree
    - $O(b^m)$
- Space complexity?
    - Similar to DFS
    - $O(b^m)$



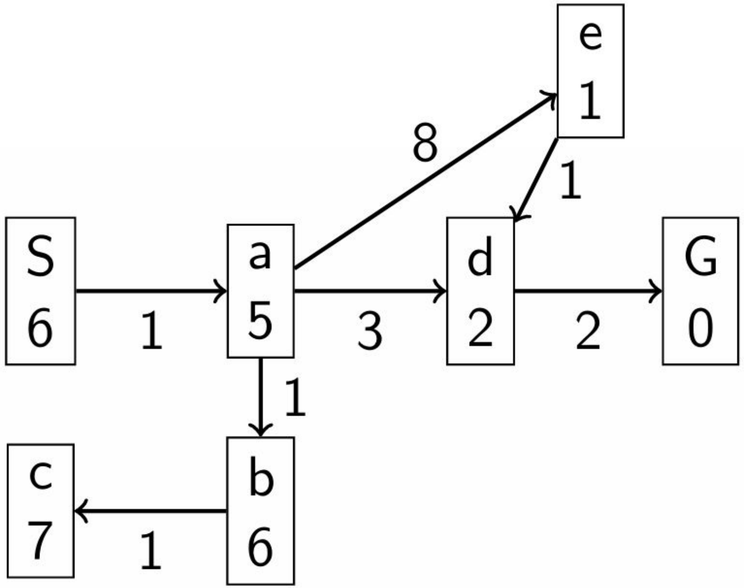
Takes you straight towards a possibly wrong goal

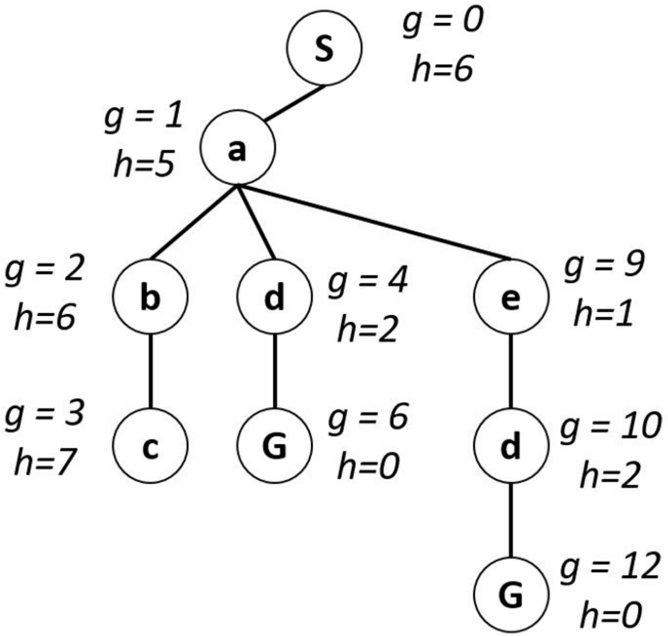


Badly-guided DFS

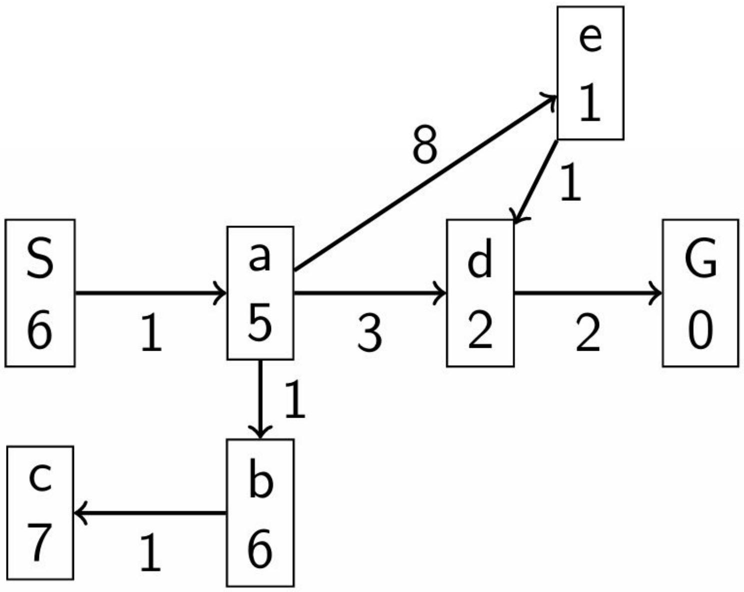
A* Search

A* Search


e
1
8
1
S
6
1
a
5
3
d
2
2
G
0
1
c
7
1
b
6


S
g = 0
h=6
g = 1
h=5
a
g = 2
h=6
b
d
g = 4
h=2
e
g = 9
h=1
g = 3
h=7
c
G
g = 6
h=0
d
g = 10
h=2
G
g = 12
h=0

A* Search

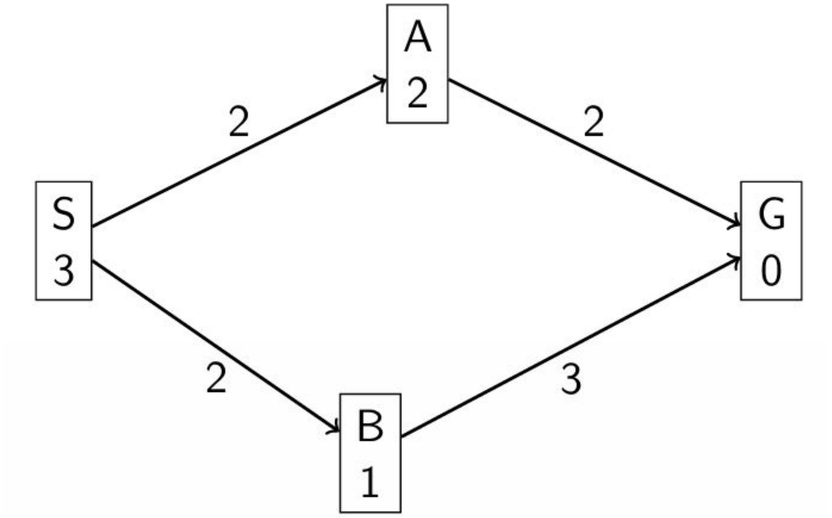


- UCS → Orders by path cost, *g(n)*
- Greedy → Orders by goal proximity, *h(n)*
- A* Search → Orders by:

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

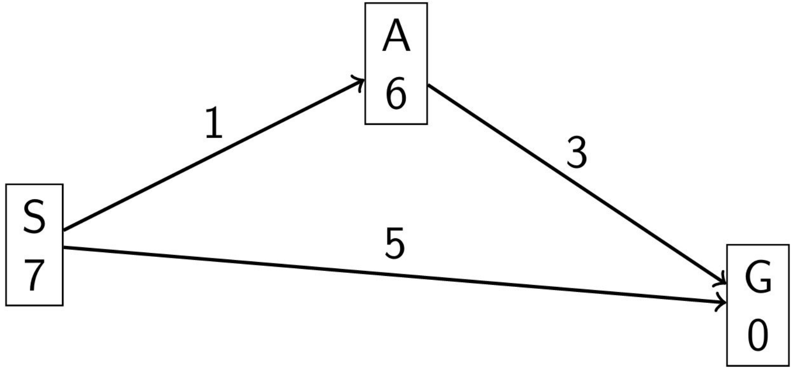
- Combining both backward cost and forward cost
- Best of both worlds

When should A* Stop?



- Do we stop when we have the goal state in our fringe?
- We should only stop when the goal state has been dequeued from the fringe

Is A* Optimal?

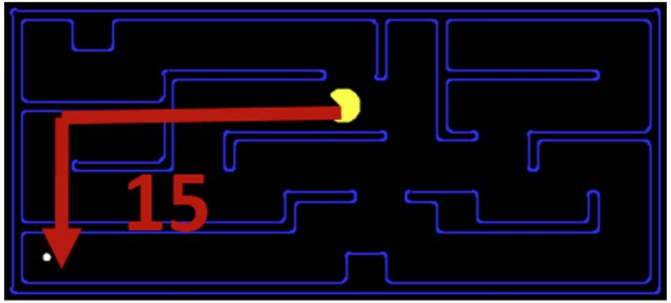


- Will A* always choose the most optimal path? → The path with the least cost
- A* will fail if
  - Actual bad goal cost < estimated good goal cost
- We need estimates to be less than the actual cost
- If your heuristic is too pessimistic, it traps good pans on the fringe

Admissible Heuristics

Admissible Heuristics

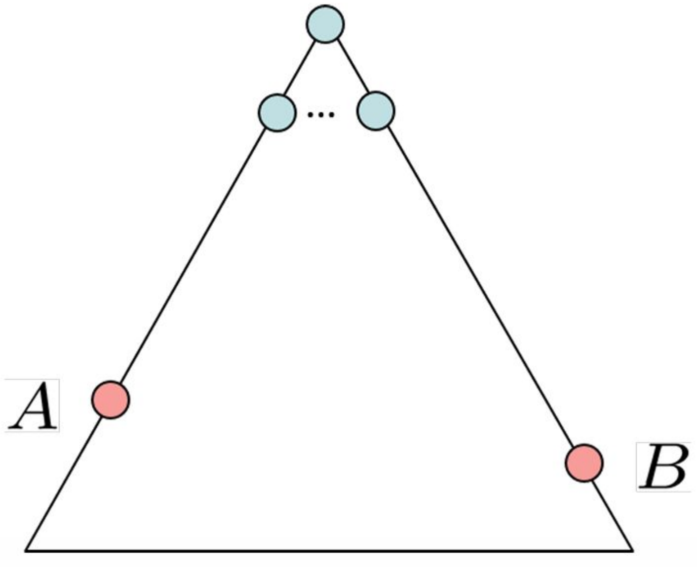
- A heuristic h is admissible if: $0 \leq h(n) \leq h^*(n)$
- $h^*(n)$ is the true cost to a nearest goal
- Creating an admissible heuristics is the most important thing you would need to do if you want to run A* search!



Is A* Optimal?

Assume:

- *A* is an optimal goal node
- *B* is a suboptimal goal node
- *h(n)* is an admissible heuristic



If A* is optimal, *A* will expand before *B*

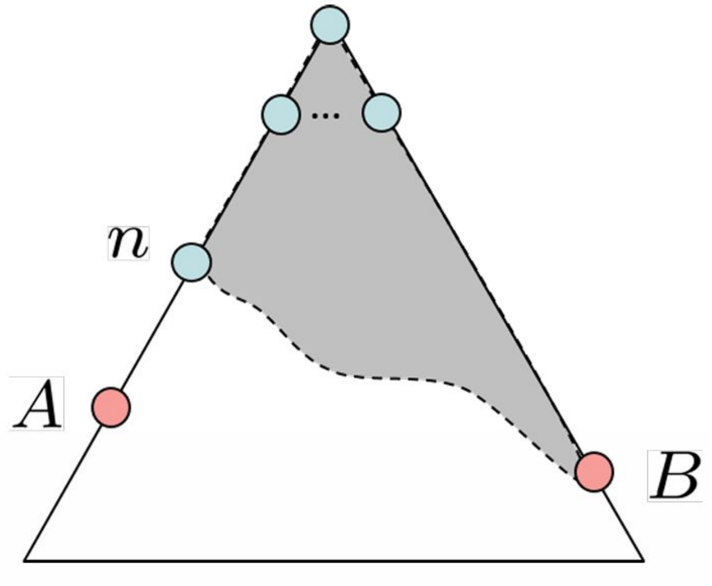
Is A* Optimal?

Imagine:

- B is on the fringe
- Some ancestor n of A is also on the fringe

$f(n) \leq f(A) \rightarrow$ Why?

- $f(n) = g(n) + h(n)$
- $f(n) \leq g(A) \rightarrow$ Admissible heuristic
- $g(A) = f(A) \rightarrow h(A) = 0$ at goal



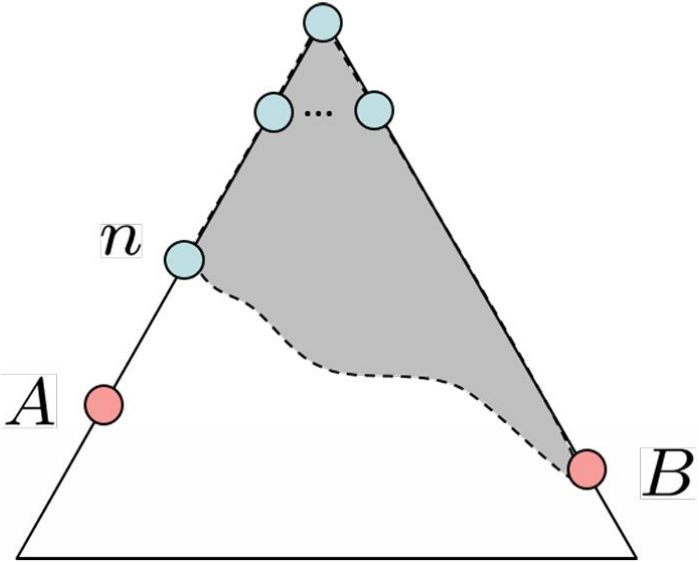
Is A* Optimal?

$f(A) < f(B) \rightarrow$ Why?

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

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

- All ancestors of A will expand before B
- A expands before B



How to Design Admissible Heuristics?

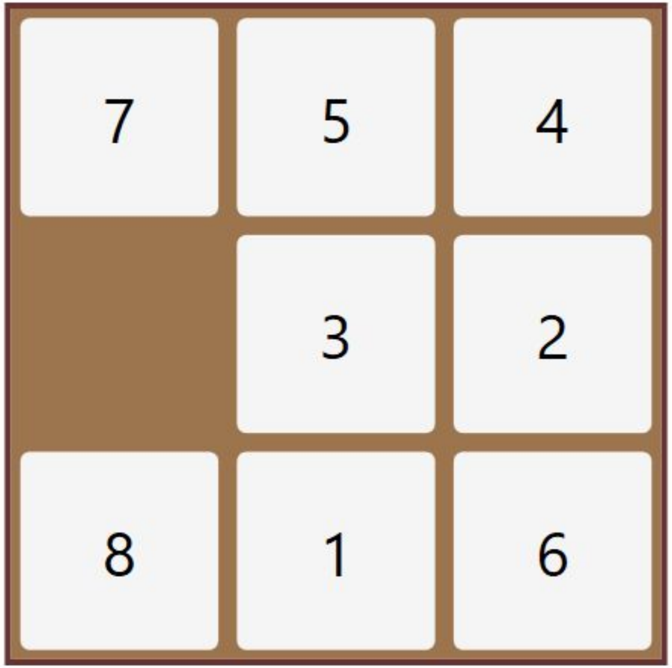
How to Design Admissible Heuristics?

Think of the 8 puzzle problem

- States? → Tile configurations
- How many possible states? → 9!
- Start state? → Any random configuration
- Goal test? → Check if all tiles are in the *correct* places
- Successor? → Move the empty piece in 4 directions
- Cost? → Number of moves

What could be an admissible heuristic for this problem?

Let's relax the problem! → Getting rid of constraints

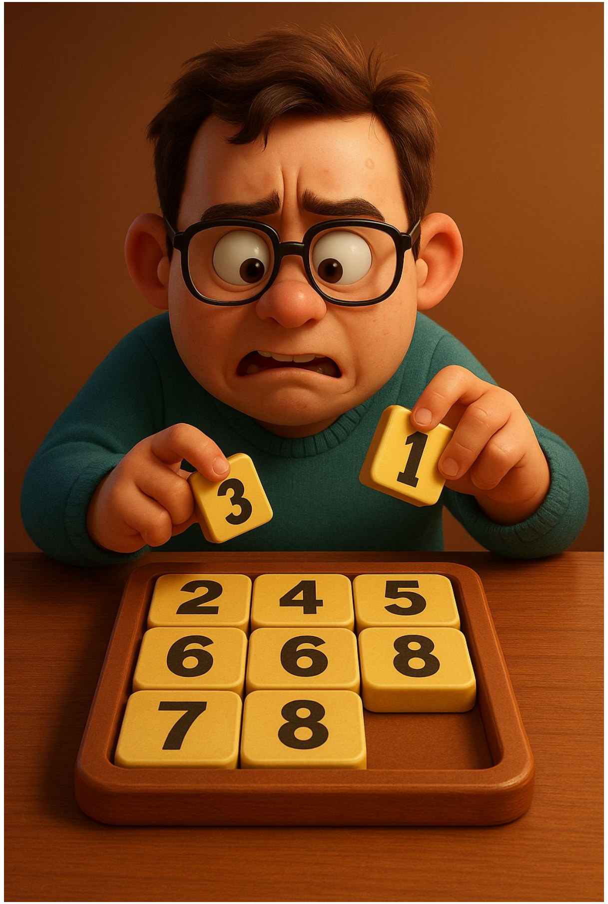


How to Design Admissible Heuristics?

Imagine that you could just pick up and place the tiles however you wanted!

- You would only need to change the tiles that are not in the correct position to begin with
- Number of misplaced tiles → Could be a heuristic
- Is it admissible? → It is an underestimate of the actual cost needed to reach the goal

Sometimes, making the problem easier by relaxing some constraints can help you design an admissible heuristic



How to Design Admissible Heuristics?

What could be a *better* heuristic → A heuristic that is closer to the actual cost

What if you could move the tiles as you wished, without the other tiles colliding with each other?

- Manhattan distance → Could be a heuristic
- Is it admissible? → It is an underestimate of the actual cost needed to reach the goal

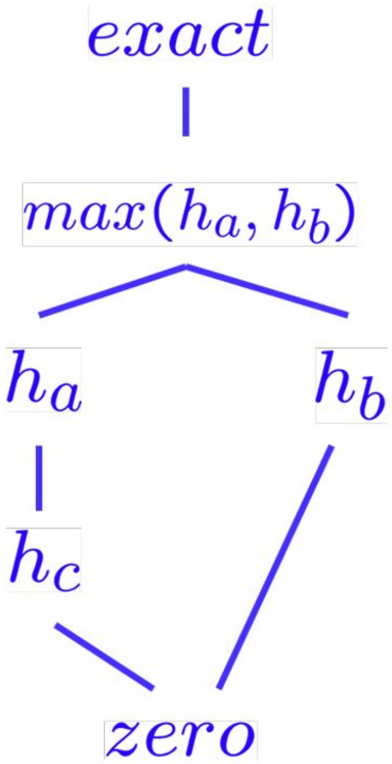
AI Failed to generate my vision! :(

Trade-off between Heuristics

- Trivial heuristics
  - Bottom of lattice is the zero
  - Top of lattice is the exact cost
- Dominance: $h_a \geq h_c$ if:
  - For all $n \rightarrow h_a(n) \geq h_c(n)$

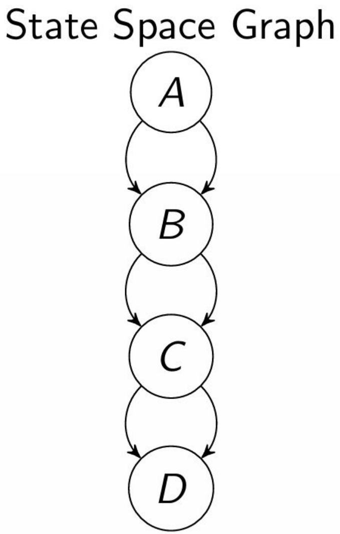
Heuristics can form a semi-lattice

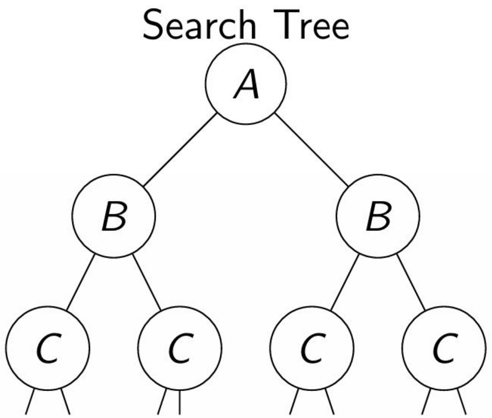
- Max of admissible heuristics is admissible
  - $h(n) = \max(h_a(\mathrm{n}), h_b(\mathrm{n}))$



Graph Search

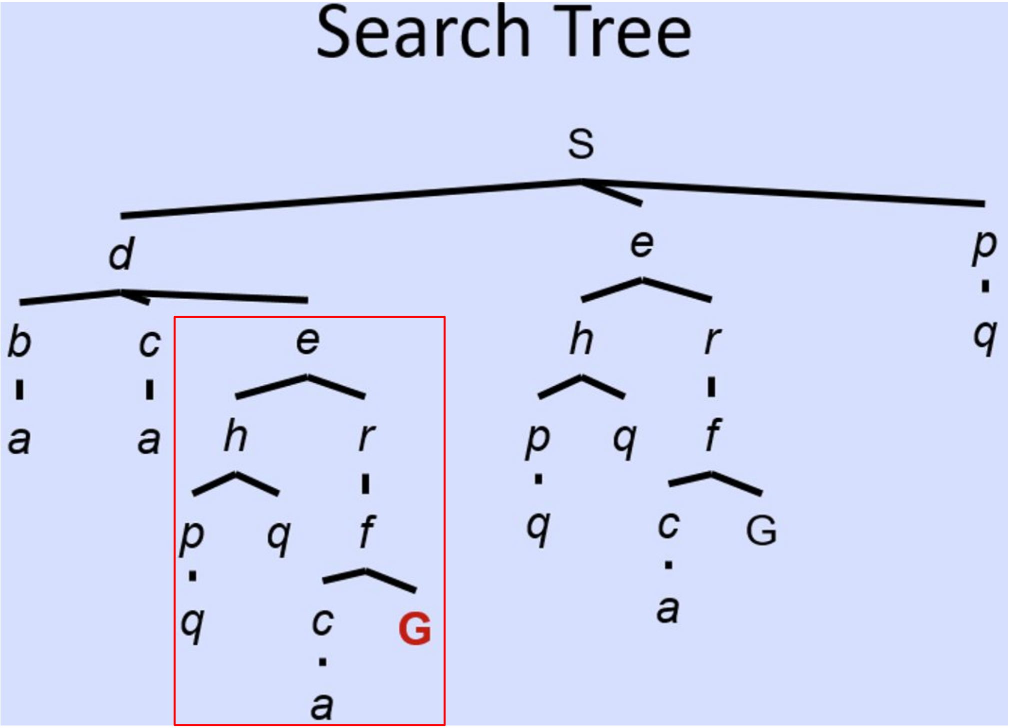
Graph Search





Tree search requires extra work as repeated states can cause exponentially more work!

Graph Search



The subtree below node *e* is repeated, expanding it again is useless → Why?

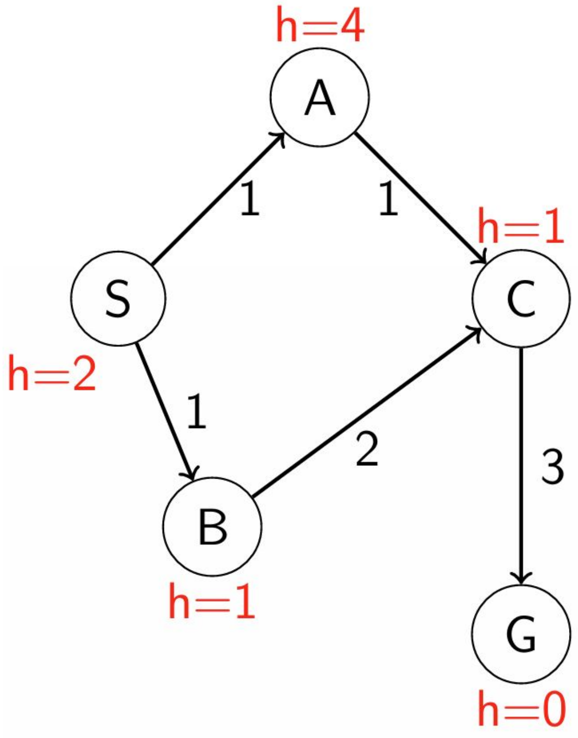
- What if we never expand a state twice?
- Tree Search + History of expanded states
- Before expanding a node, check if it had been expanded before
- Keep the history as a **set**, not list

Is it complete? Is it optimal?

Is Graph Search Optimal?

- Running A* Graph Search, we get the suboptimal path as we don't expand *C* twice
- We must ensure that when we first expand a node, we were there following the most optimal path

We need *consistent* heuristics



Consistency of Heuristics

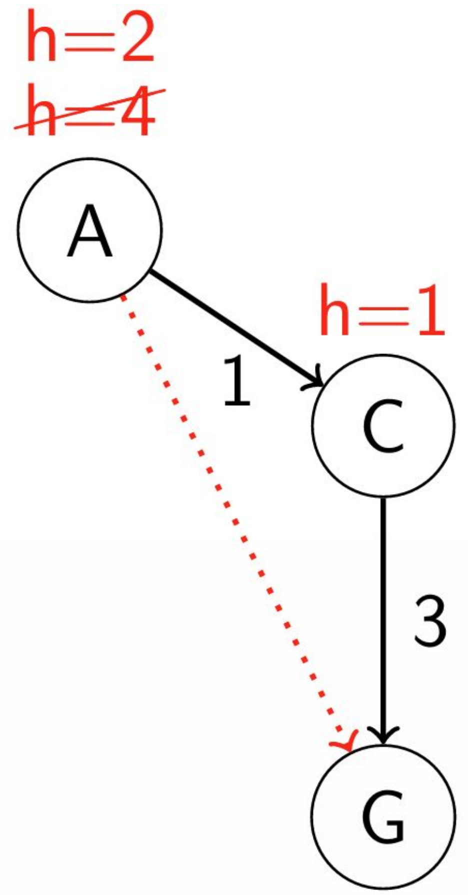
Estimated heuristics cost ≤ Actual cost

- Admissibility → $h(A) \leq$ Actual cost from A to G
- Consistency → Heuristic "arc" cost ≤ actual cost of each arc
  - $h(A) - h(C) \leq \text{cost}(A \text{ to } C)$
  - $h(A) \leq h(C) + \text{cost}(A \text{ to } C)$
- $f(n)$ should keep increasing as we keep exploring deeper nodes

$$h(A) \leq \text{cost}(A \text{ to } C) + h(C)$$

$$g(A) + h(A) \leq g(A) + \text{cost}(A \text{ to } C) + h(C)$$

$$f(A) \leq f(C)$$



Additional Resources

- Bill Gates and his Pancake Problem
- A* Search: How Your Map Applications Find Shortest Routes
- Pathfinder-Algorithm-Visualization→ Shameless self plug

Quiz-1 Syllabus

- Russell & Norvig
  - Chapter 1
  - Chapter 2
  - Chapter 3
- Poole & Mackworth
  - Chapter 1
  - Chapter 2
  - Chapter 3

Thank you