Lecture 4: Search informed by lookahead heuristics: Greedy Search, A*

Mark Hasegawa-Johnson, January 2020

With some slides by Svetlana Lazebnik, 9/2016

Outline of lecture

1. Search heuristics
2. Greedy best-first search: minimum h(n)
3. A* optimal search: $f(n) = h(n) + g(n)$, where $h(n) \leq d(n)$

Review: DFS and BFS

- Depth-first search
  - LIFO: expand the deepest node (farthest from START)
  - Pro: only need to keep a small part of the search tree (space is $O\{bm\}$).
  - Con: not optimal, or even complete. Time is $O\{b^m\}$.
- Breadth-first search
  - FIFO: expand the shallowest node (closest to START)
  - Pro: complete and optimal. Time is $O\{b^d\}$
  - Con: no path is found until the best path is found. Space is $O\{b^d\}$.

Why don’t we just measure…

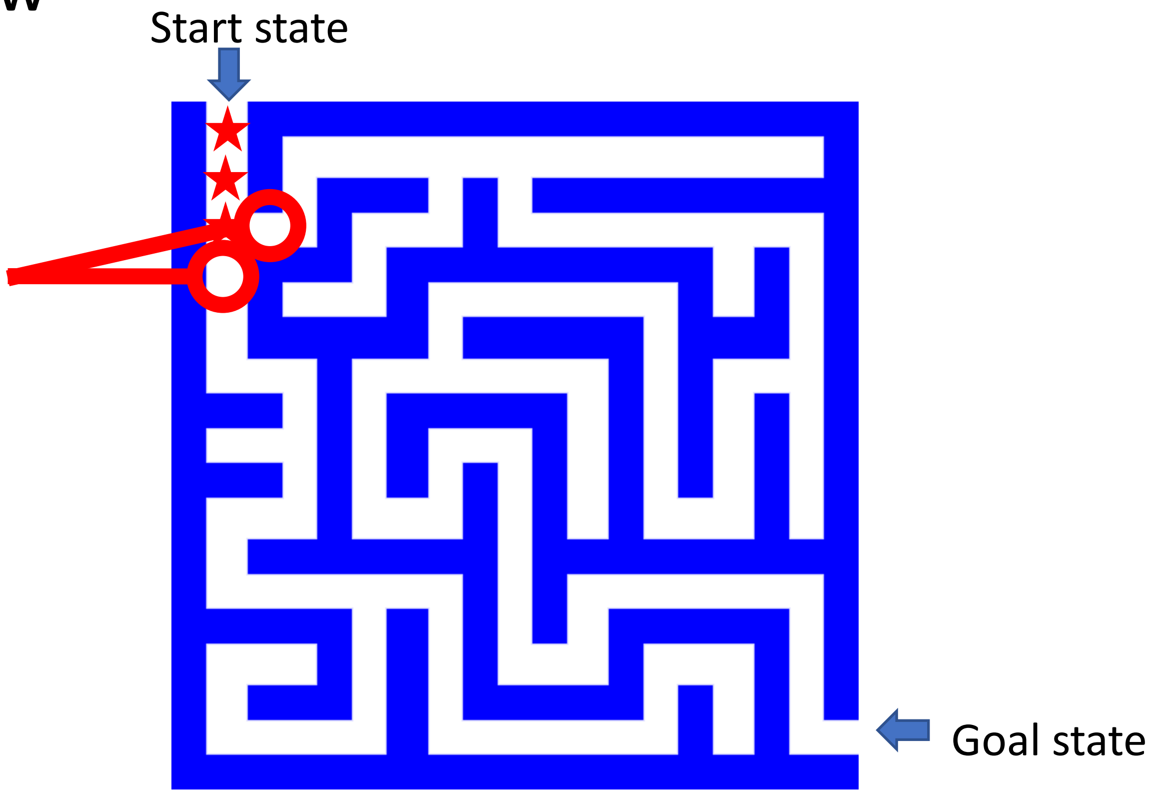
Instead of FARTHEST FROM START (DFS):

why not choose the node that’s CLOSEST TO GOAL?

Why not choose the node CLOSEST TO GOAL?

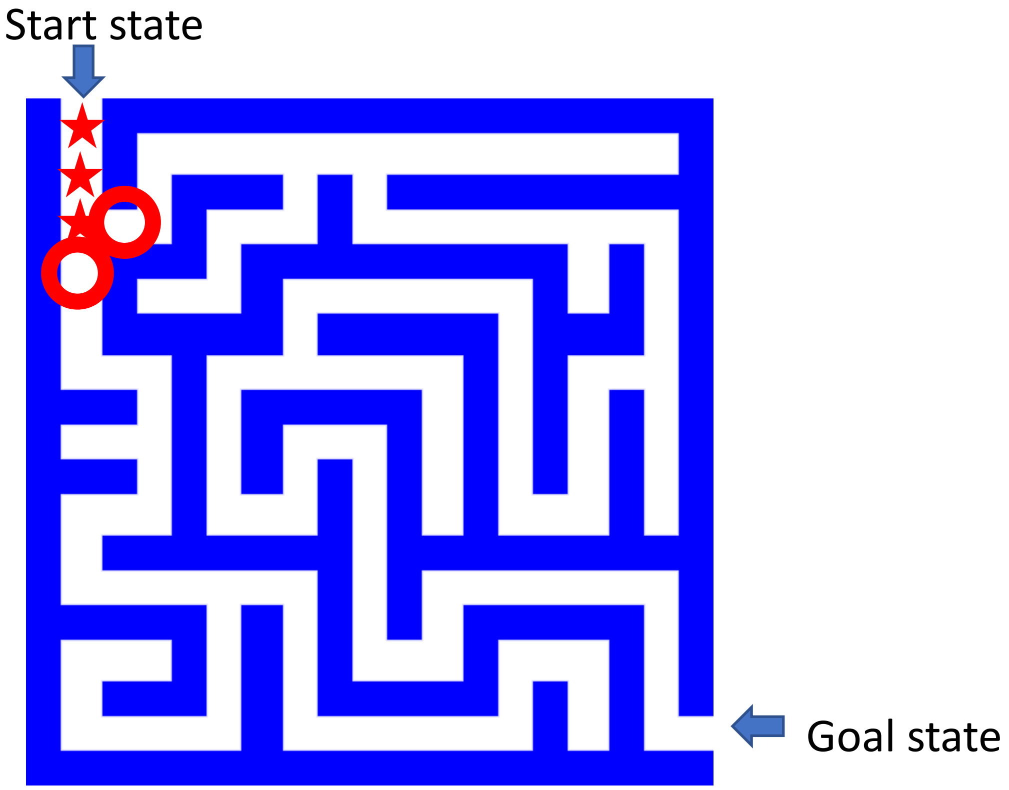
- Answer: because we don't know which node that is!!

- Example: which of these two is closest to goal?



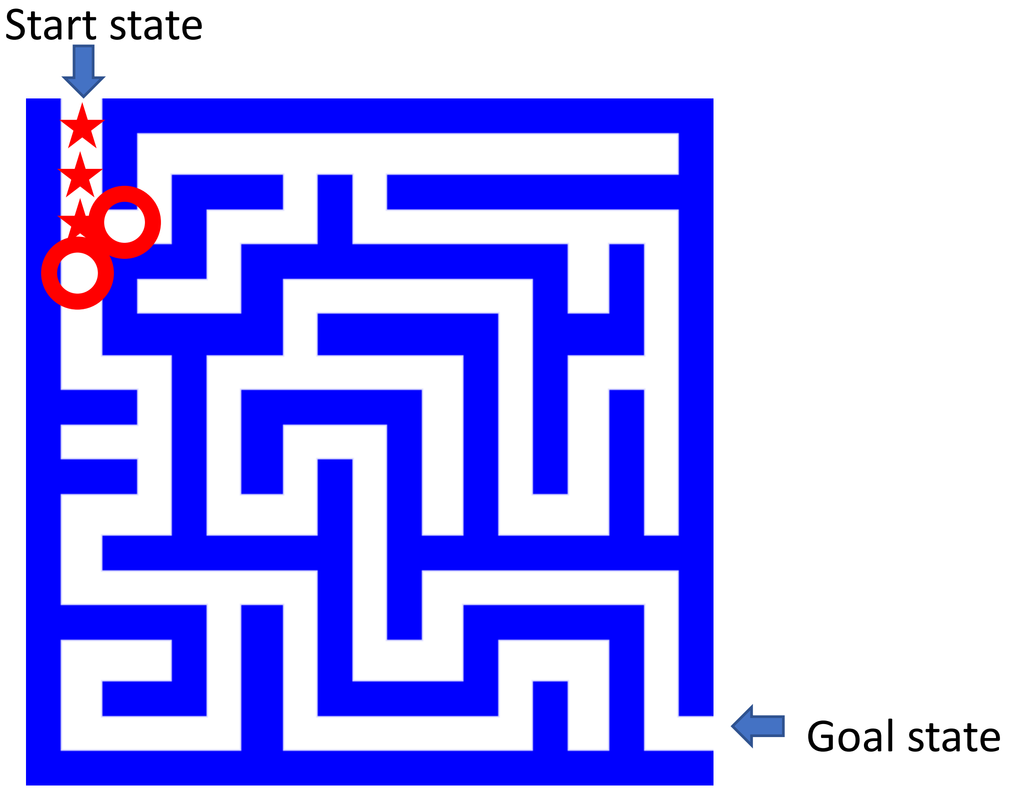
We don’t know which state is closest to goal

- Finding the shortest path is the whole point of the search
- If we already knew which state was closest to goal, there would be no reason to do the search
- Figuring out which one is closest, in general, is a complexity $O\{b^d\}$ problem.



Search heuristics: estimates of distance-to-goal

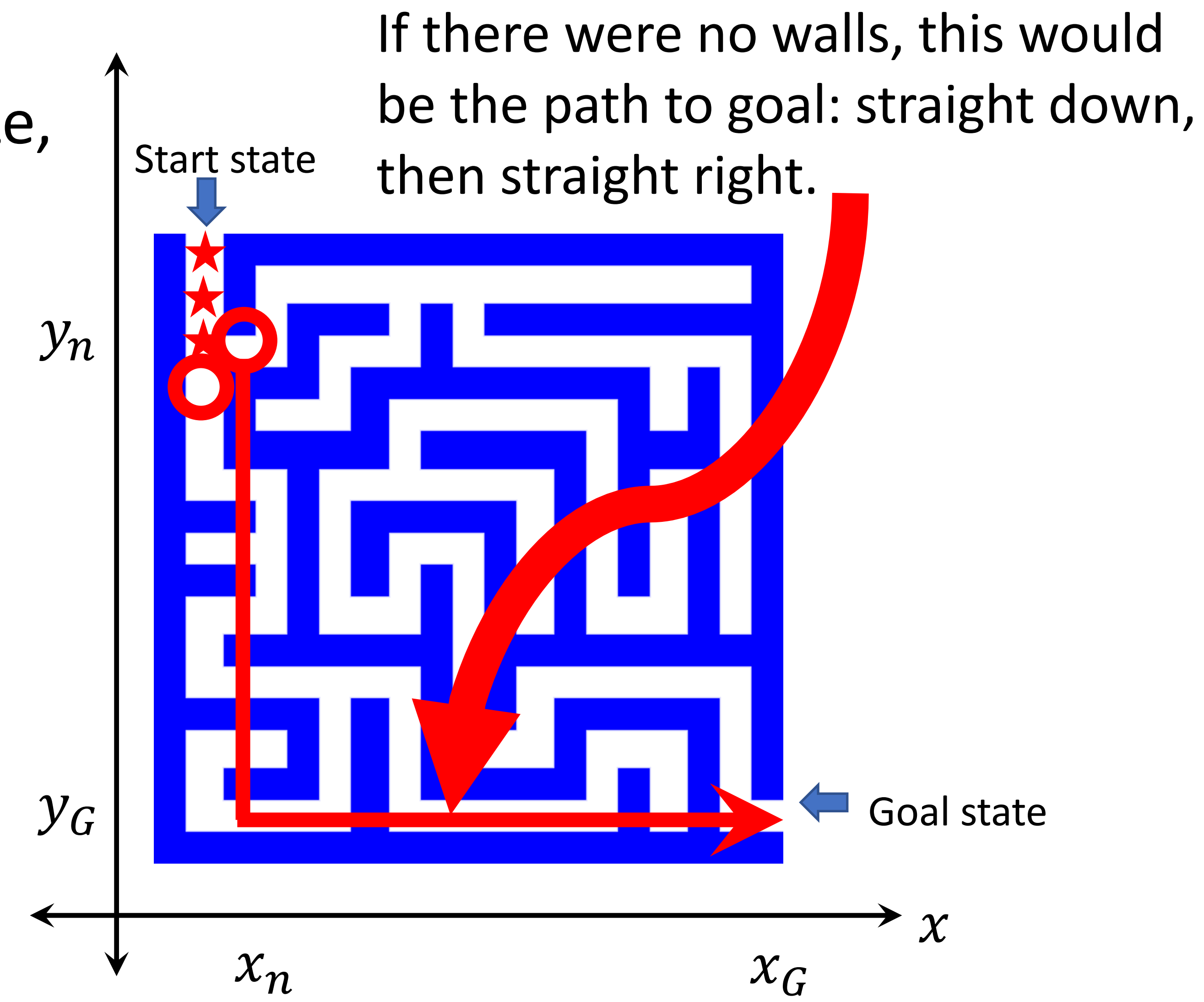
- Often, even if we don't know the distance to the goal, we can estimate it.
- This estimate is called a heuristic.
- A heuristic is useful if:
  1. **Accurate**: $h(n) \approx d(n)$, where $h(n)$ is the heuristic estimate, and $d(n)$ is the true distance to the goal
  2. **Cheap:** It can be computed in complexity less than $O\{b^d\}$



Example heuristic: Manhattan distance

If there were no walls in the maze, then the number of steps from position (x_n, y_n) to the goal position (x_G, y_G) would be

$$h(n) = |x_n - x_G| + |y_n - y_G|$$



Outline of lecture

1. Search heuristics
2. Greedy best-first search: minimum h(n)
3. A* optimal search: $f(n) = h(n) + g(n)$, where $h(n) \leq d(n)$

Greedy Best-First Search

Instead of FARTHEST FROM START (DFS):

why not choose the node whose

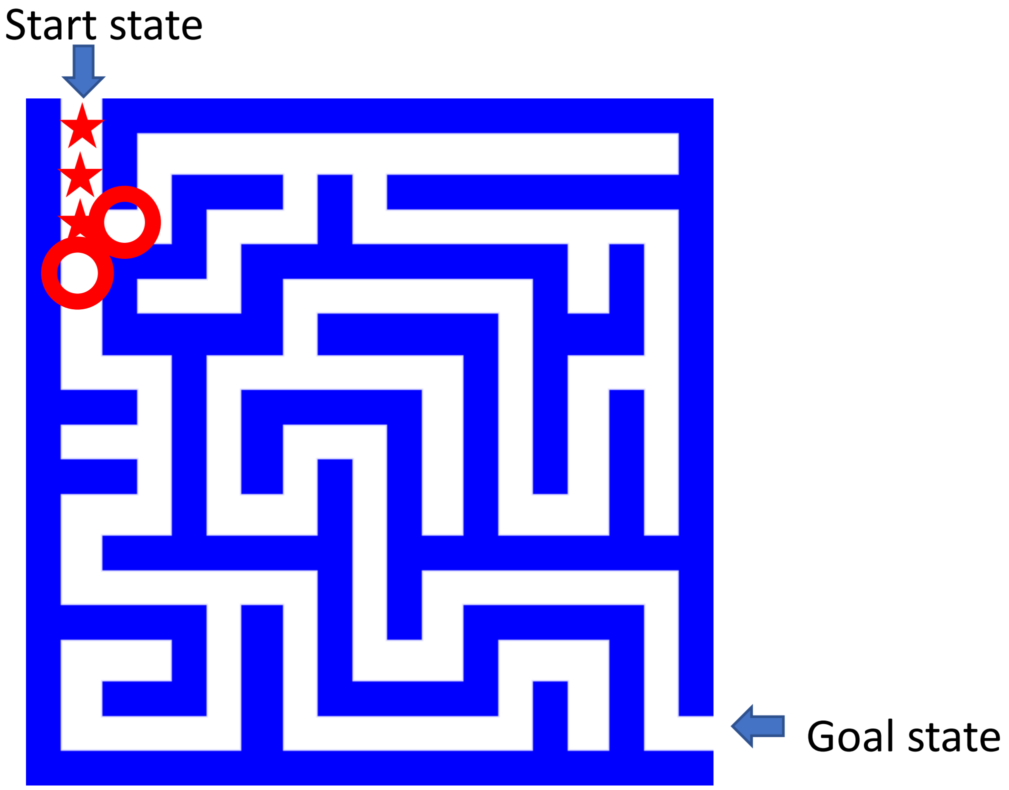
HEURISTIC ESTIMATE

indicates that it might be

CLOSEST TO GOAL?

Greedy Search Example

According to the Manhattan distance heuristic, these two nodes are equally far from the goal, so we have to choose one at random.

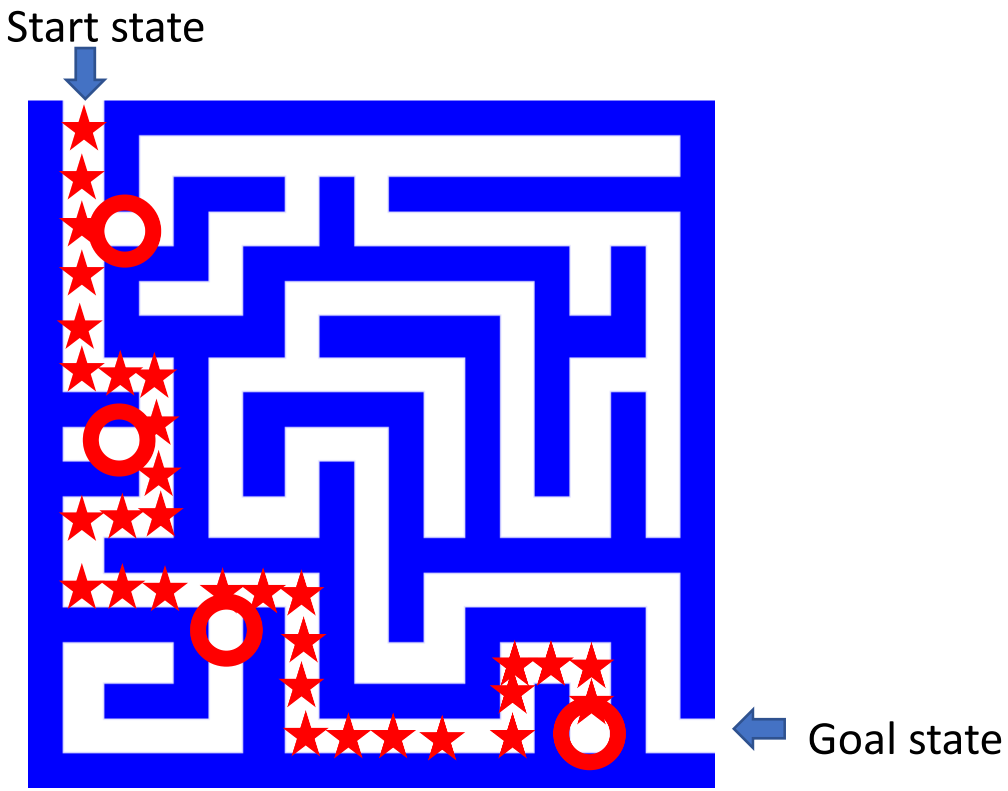


Greedy Search Example

If our random choice goes badly, we might end up very far from the goal.

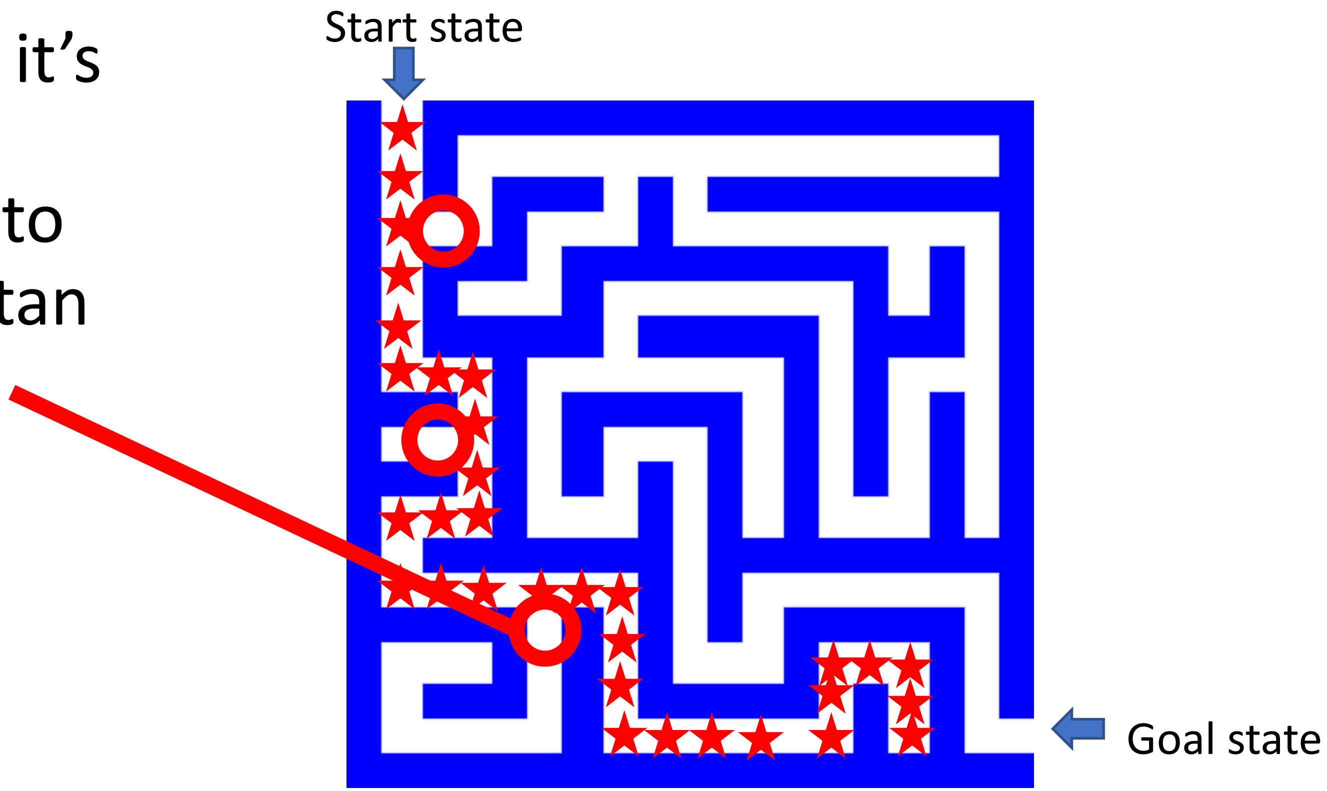
★ = states in the explored set

○ = states on the frontier



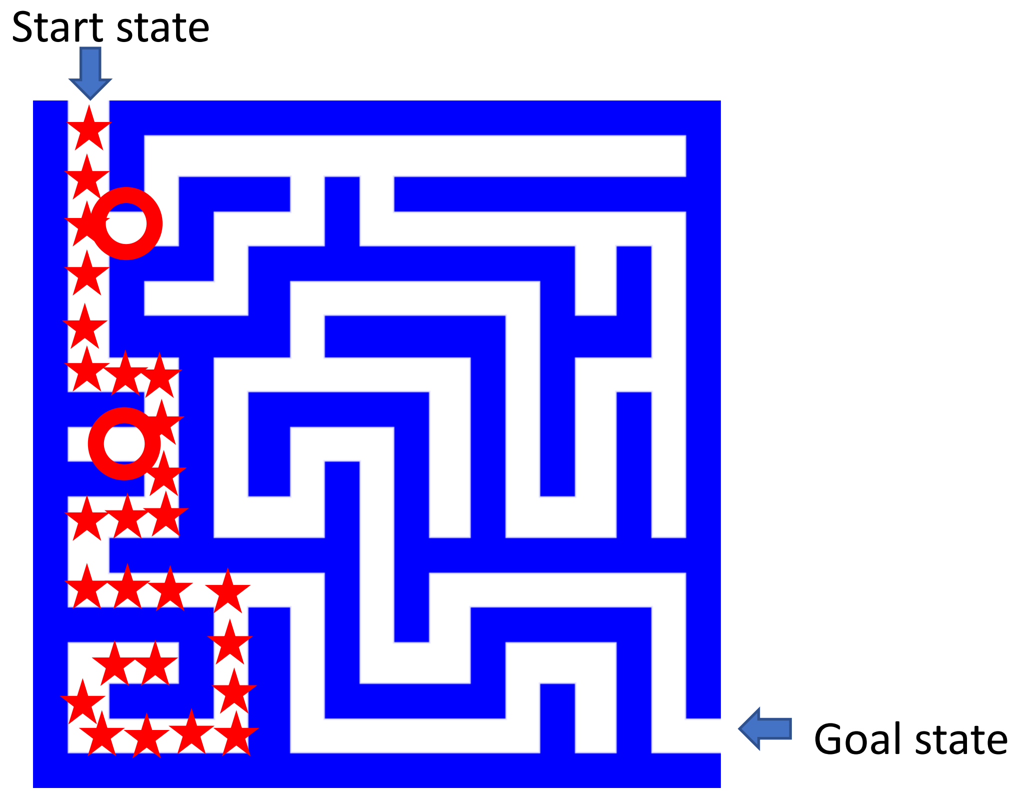
The problem with Greedy Search

Having gone down a bad path, it's very hard to recover, because now, the frontier node closest to goal (according to the Manhattan distance heuristic) is this one:



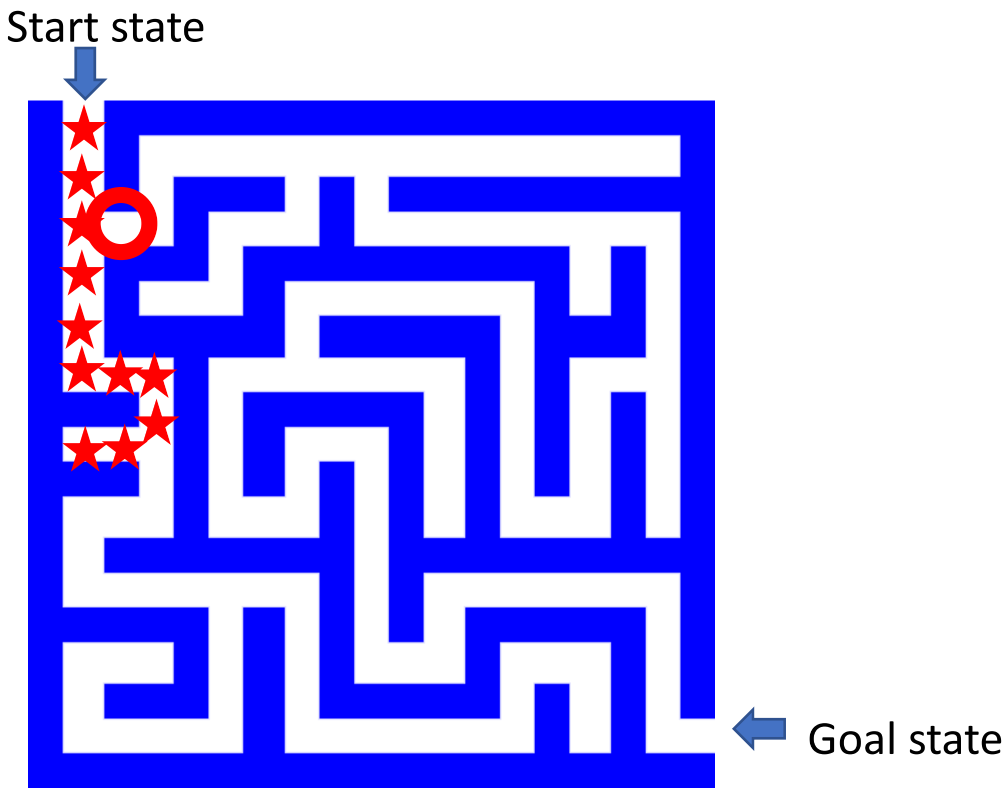
The problem with Greedy Search

That's not a useful path…



The problem with Greedy Search

Neither is that one…



What went wrong?

Outline of lecture

1. Search heuristics
2. Greedy best-first search: minimum h(n)
3. A* optimal search: $f(n) = h(n) + g(n)$, where $h(n) \leq d(n)$

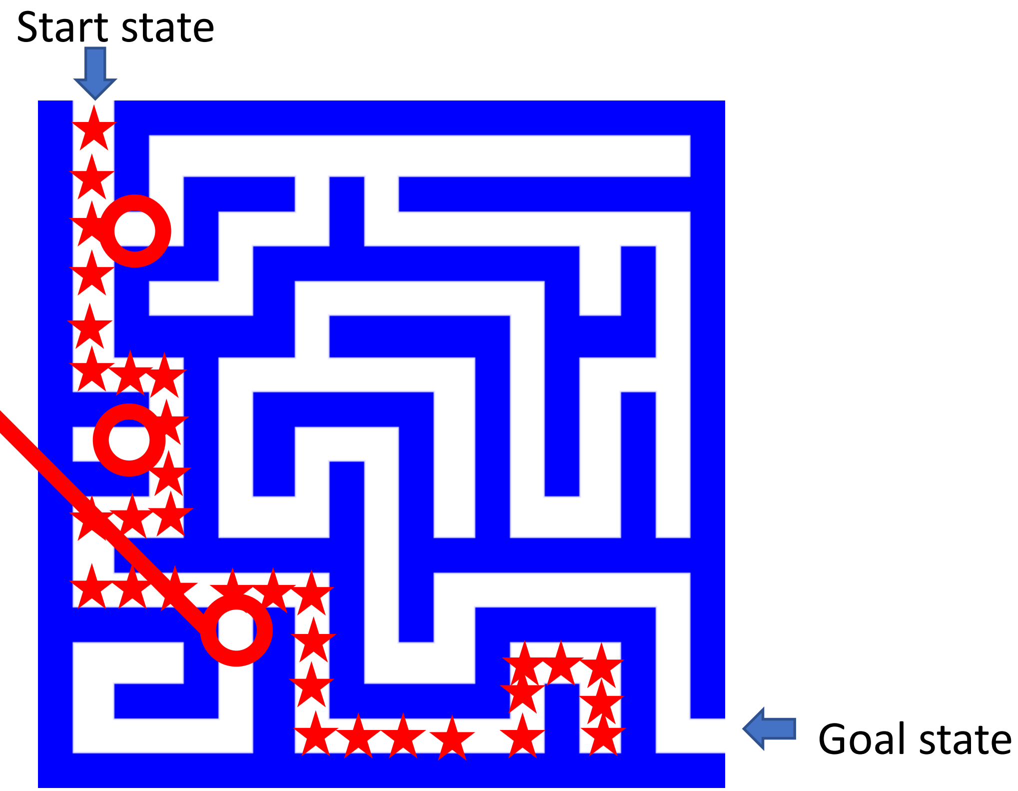
The problem with Greedy Search

Among nodes on the frontier, this one seems closest to goal (smallest $h(n)$, where $h(n) \leq d(n)$).

But it's also farthest from the start. Let's say $g(n)$ = total path cost so far.

So the total distance from start to goal, going through node n, is

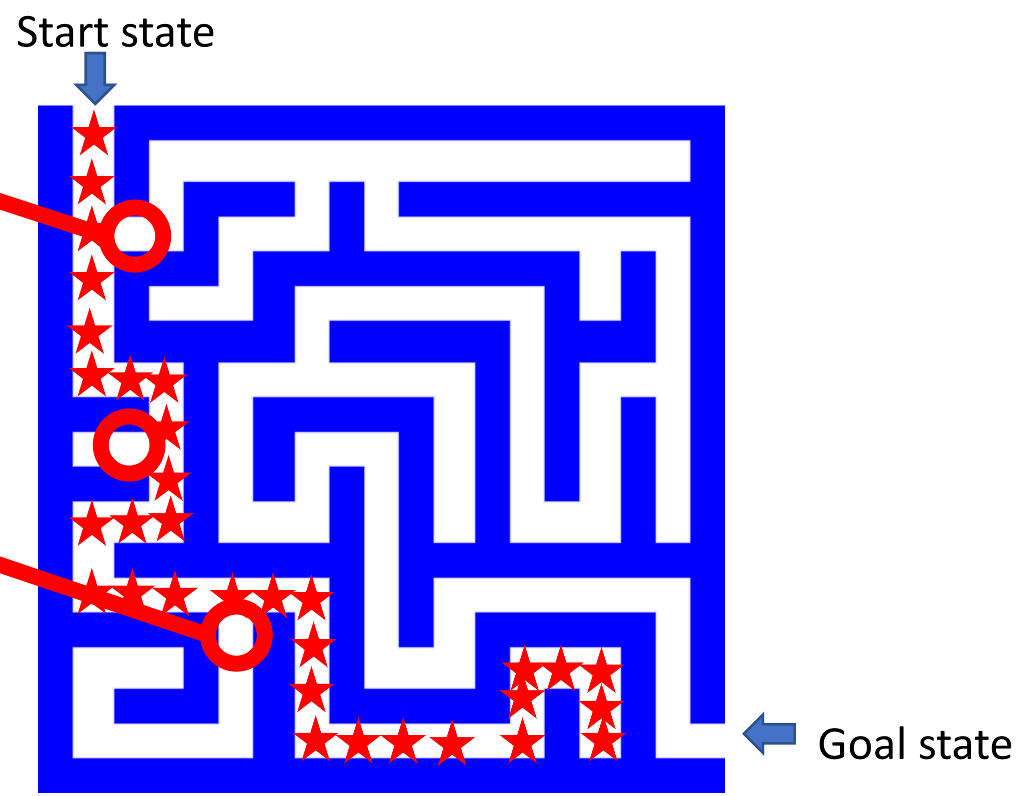
$$c(n) = g(n) + d(n) \geq g(n) + h(n)$$



The problem with Greedy Search

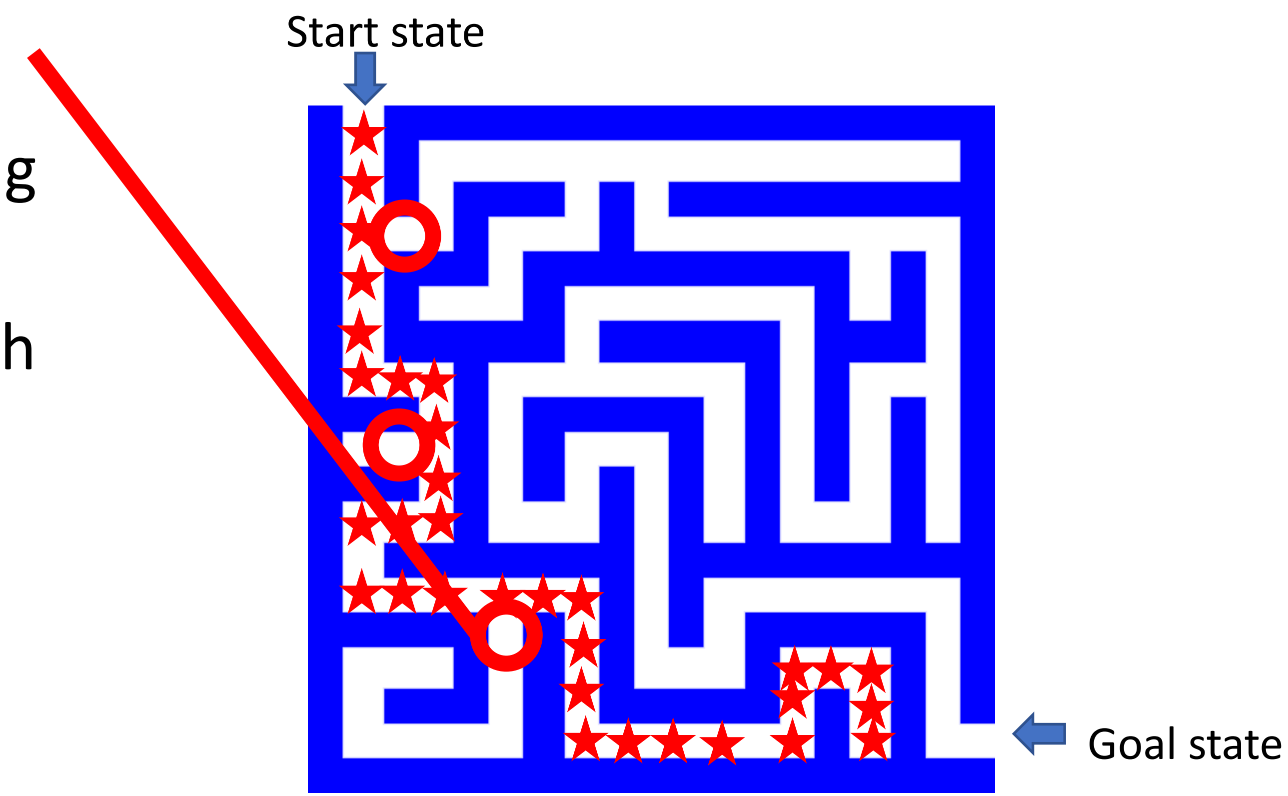
Of these three nodes, this one has the smallest $g(n) + h(n)$.

So if we want to find the lowest-cost path, then it would be better to try that node, instead of this one.



A* notation

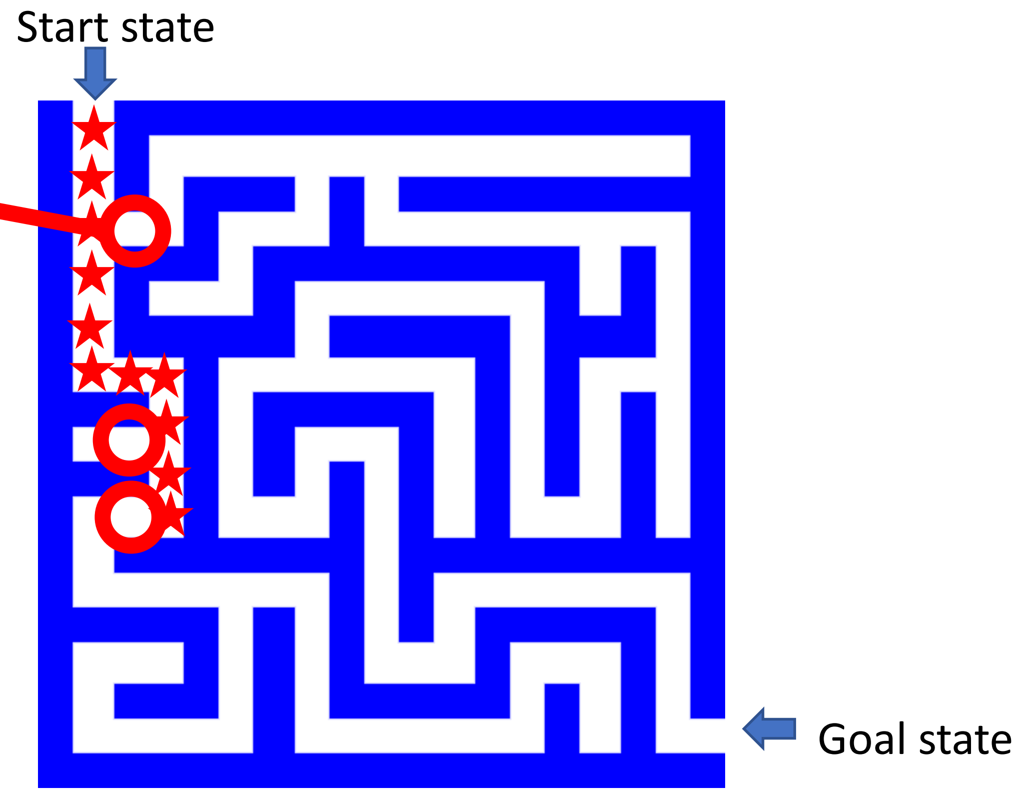
- $c(n)$ = **cost** of the total path (START,…,n,…,GOAL).
- $d(n)$ = **distance** of the remaining partial path (n,…,GOAL).
- $g(n)$ = **gone-already** on the path so far, (START,…,n).
- $h(n)$ = **heuristic,** $h(n) \leq d(n)$.



$$c(n) = g(n) + d(n) \geq g(n) + h(n)$$

Smart Greedy Search

In fact, let's back up. Already, at this point in the search, this node has the smallest $g(n) + h(n)$.



A* Search

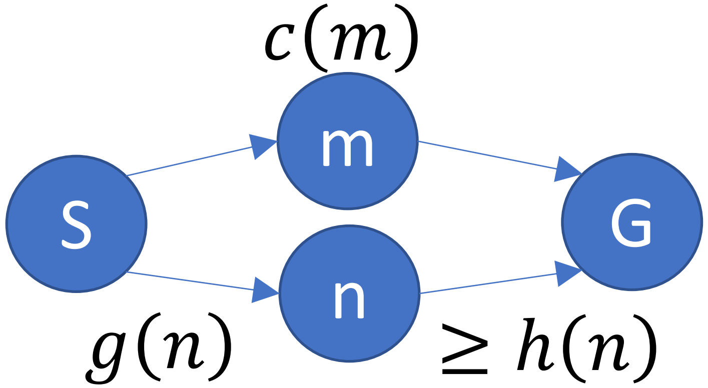
- Idea: avoid expanding paths that are already expensive
- The **evaluation function** $f(n)$ is the estimated total cost of the path through node n to the goal:

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

$g(n)$: cost so far to reach n (path cost)
$h(n)$: estimated cost from n to goal (heuristic)

- This is called A* search if and only if the heuristic, h(n), is **admissible**. The word "admissible" just means that $h(n) \leq d(n)$, and therefore, $f(n) \leq c(n)$.

Admissible heuristic



- Suppose we've found one path to G; the path goes through node m. Since we've calculated the whole path, we know its total path cost to be $c(m)$.
- For every other node, n, we don't know $c(n)$, but we know $f(n) = g(n) + h(n)$, and we know that

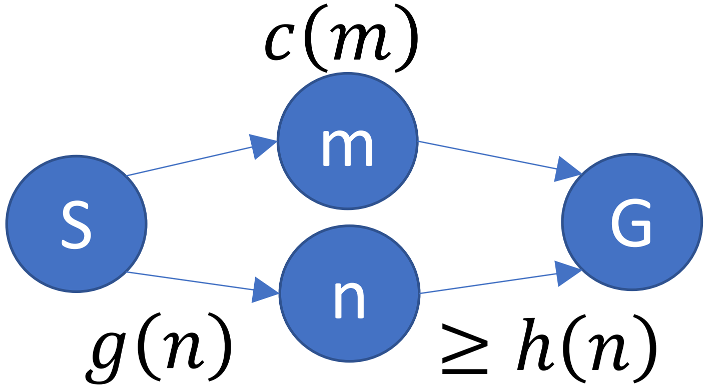
$$c(n) \geq f(n)$$

- Therefore we know that

$$IF \quad f(n) \geq c(m)$$
$$THEN \; c(n) \geq c(m)$$

- So if $f(n) \geq c(m)$ for every node n that's still in the frontier, then we know that m is the best path.

A* Search



Definition: A* SEARCH

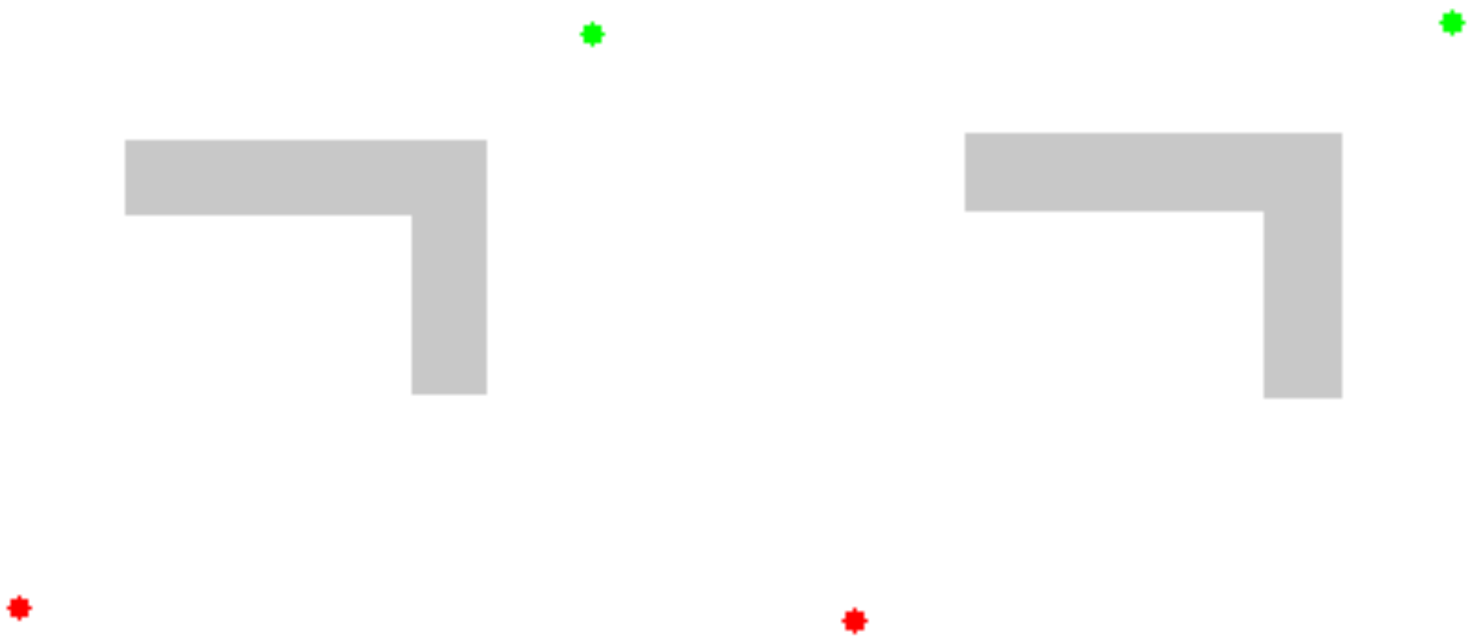
- If $h(n)$ is **admissible** $(d(n) \geq h(n))$, and
- if the frontier is a priority queue sorted according to $g(n) + h(n)$, then
- the FIRST path to goal uncovered by the tree search, path m, is guaranteed to be the SHORTEST path to goal

$(h(n) + g(n) \geq c(m)$ for every node n that is not on path $m)$

BFS vs. A* Search

The heuristic h(n)=Manhattan distance favors nodes on the main diagonal. Those nodes all have the same g(n)+h(n), so A* evaluates them first.

Note: Manhattan distance isn't an admissible heuristic if you can take diagonal steps. It must be using 8-direction Manhattan distance, or else Euclidean distance.



Source: [Wikipedia]