Reminders

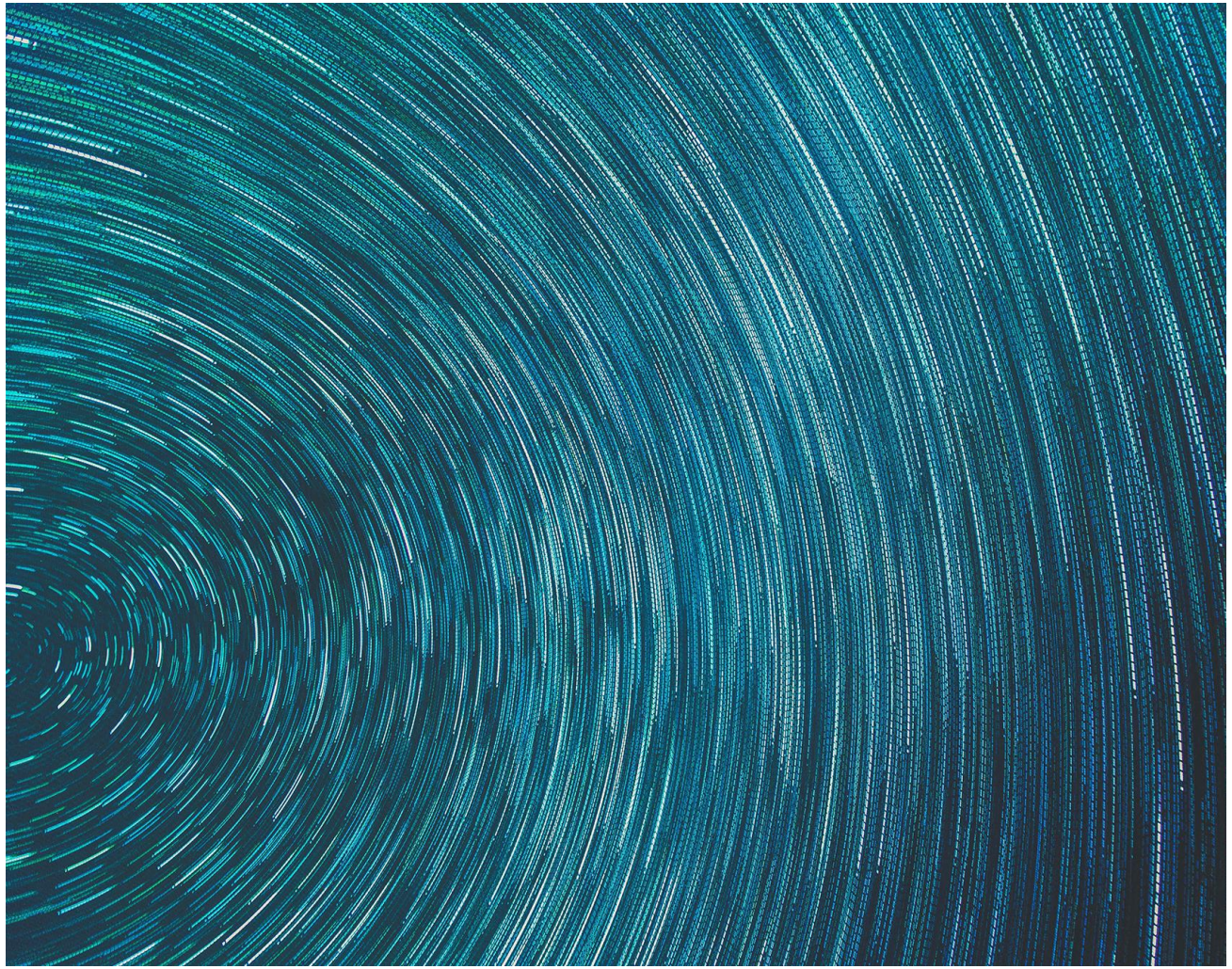
HW2 is due tonight before 11:59pm Eastern time.

HW3 has been released.

Register to vote: https://vote.gov

Online registration and mail-in request deadline:
Monday, October 19, 2020

Be a poll worker:
https://www.votespa.com/Resources/Pages/Be-a-Poll-Worker.aspx



A* search is Optimal

AIMA 3.5

Key: Admissibility

Inadmissible (pessimistic) heuristics break optimality by pushing good plans too far back on the frontier, which means they may never get expanded.

Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs. That means that the true best plan will always be expanded.

Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

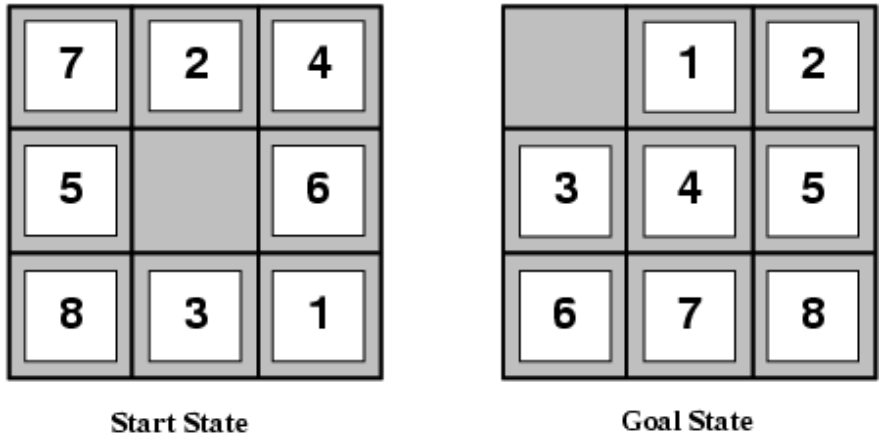
Admissible Heuristics

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

Is Manhattan Distance admissible?



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

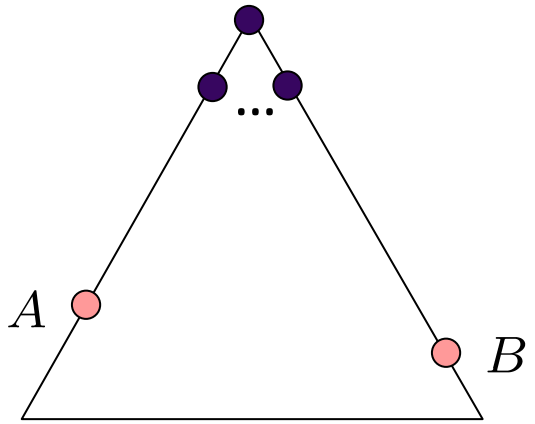
Optimality of A* Tree Search

Assume:

A is an optimal goal node

B is a suboptimal goal node

h is admissible



Claim:

A will exit the frontier before B

Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Optimality of A* Tree Search

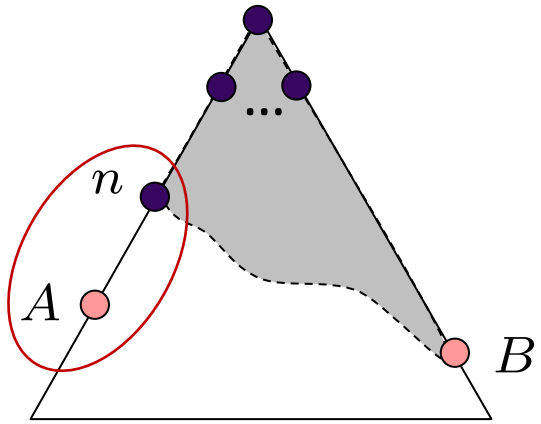
Proof:

Imagine B is on the frontier

Some ancestor *n* of A is on the frontier, too (maybe A!)

Claim: *n* will be expanded before B

- f(n) is less 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

Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Optimality of A* Tree Search

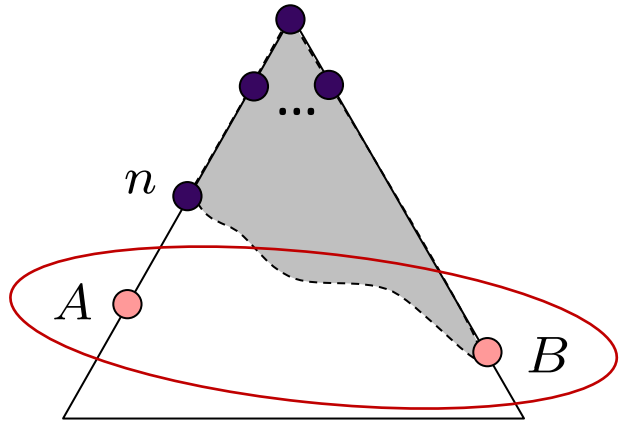
Proof:

Imagine B is on the frontier

Some ancestor *n* of A is on the frontier, too (maybe A!)

Claim: *n* will be expanded before B

- f(n) is less or equal to f(A)
- f(A) is less than f(B)



$g(A) < g(B)$ B is suboptimal

$f(A) < f(B)$ h = 0 at a goal

Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Optimality of A* Tree Search

Proof:

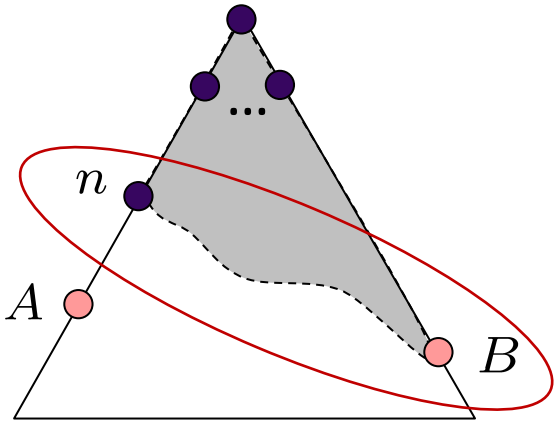
- Imagine B is on the frontier
- Some ancestor *n* of A is on the frontier, too (maybe A!)
- Claim: *n* will be expanded before B

    f(n) is less or equal to f(A)

    f(A) is less than f(B)

    n expands before B

- All ancestors of A expand before B
- A expands before B
- A* search is optimal



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

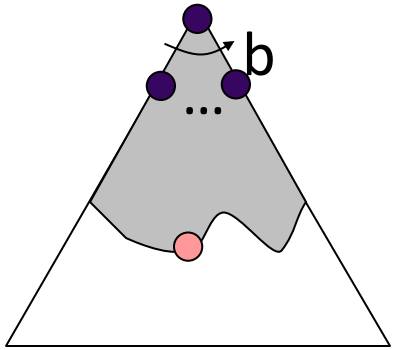
Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Properties of A*

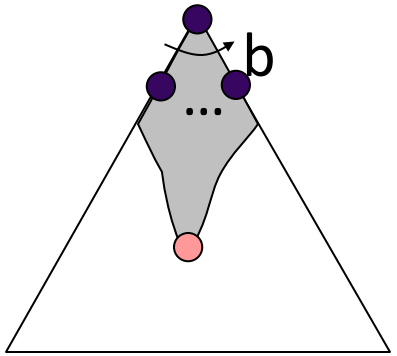
Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

Properties of A*

Uniform-Cost



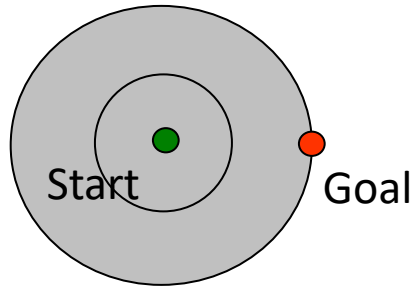
A*



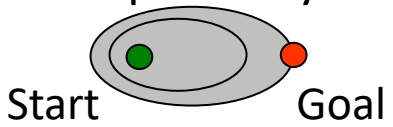
Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

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



Slide credit: Dan Klein and Pieter Abbeel
http://ai.berkeley.edu

A* Applications

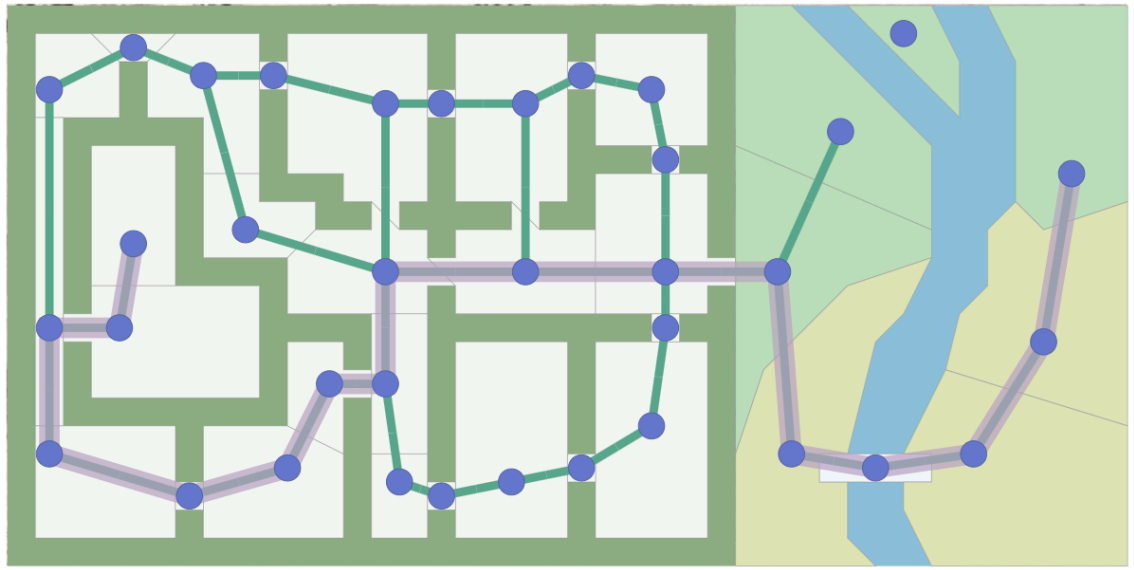
Pathing / routing problems (A* is in your GPS!)

Video games

Robot motion planning

Resource planning problems

…



Supplemental Reading

I recommend this A* tutorial by Amit Patel of Red Blob Games

https://www.redblobgames.com/pathfinding/a-star/introduction.html

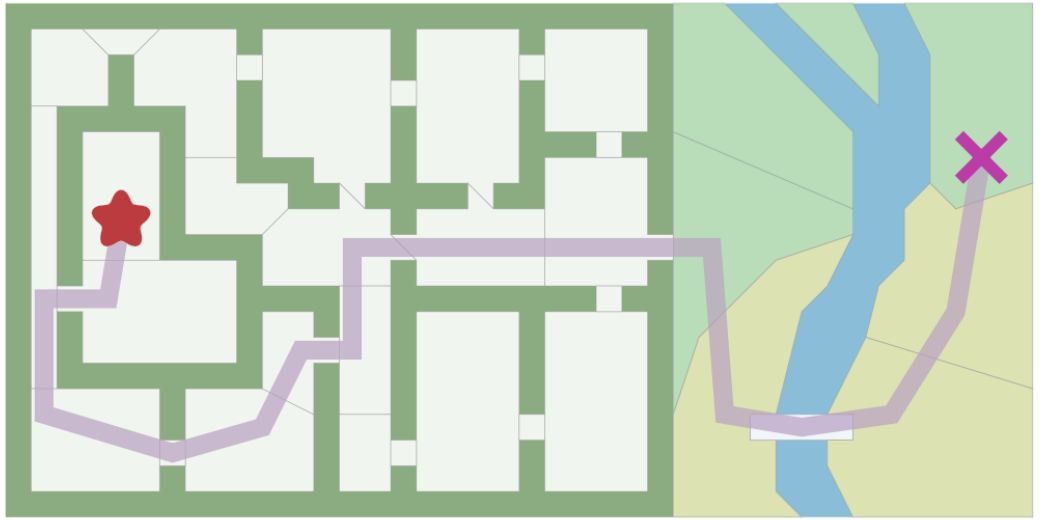
Introduction to the A* Algorithm

from Red Blob Games

Home Blog Links Twitter About Search

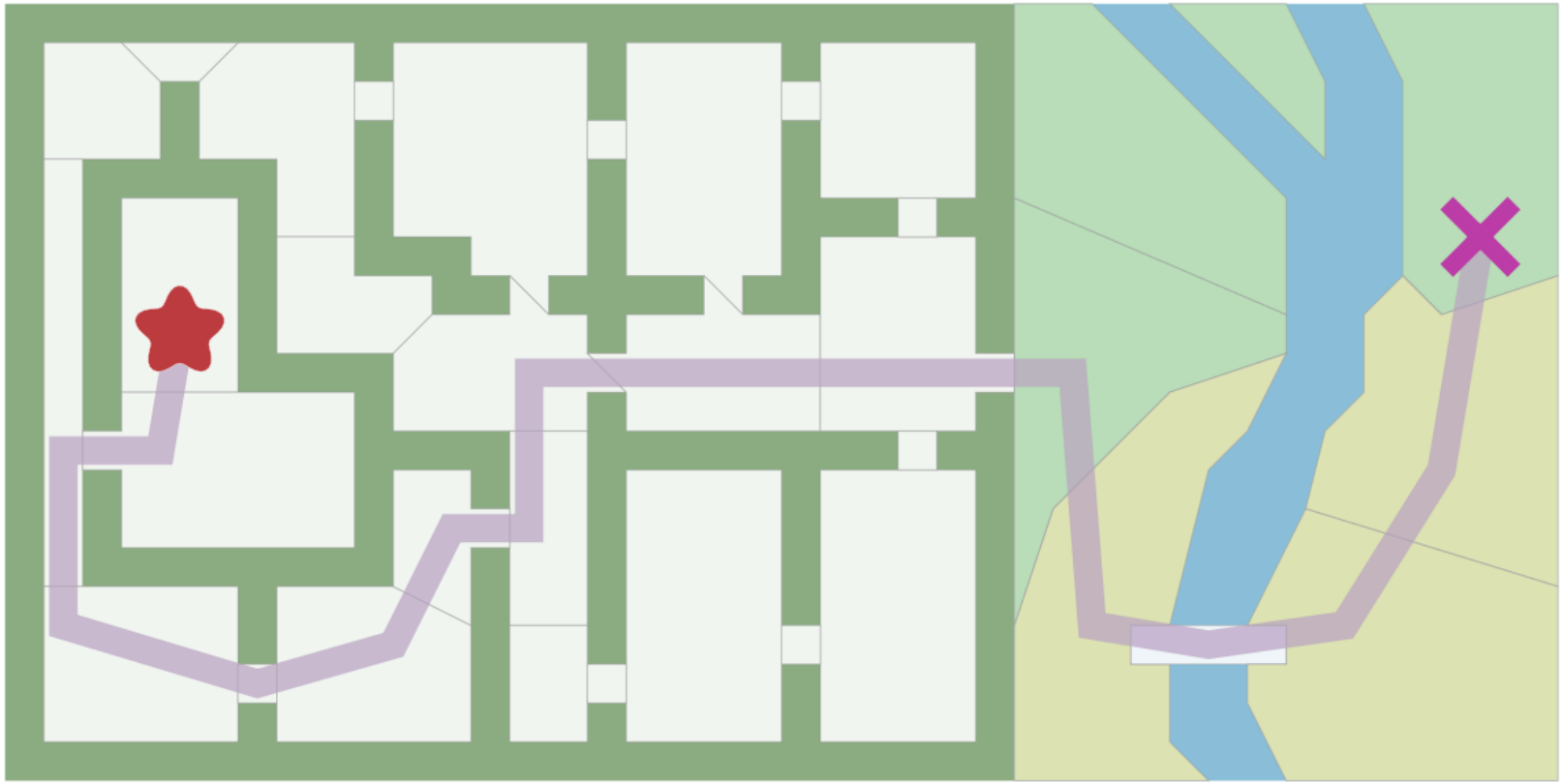
Created 26 May 2014, updated Aug 2014, Feb 2016, Jun 2016

In games we often want to find paths from one location to another. We're not only trying to find the shortest distance; we also want to take into account travel time. Move the blob (start point) and cross (end point) to see the shortest path.



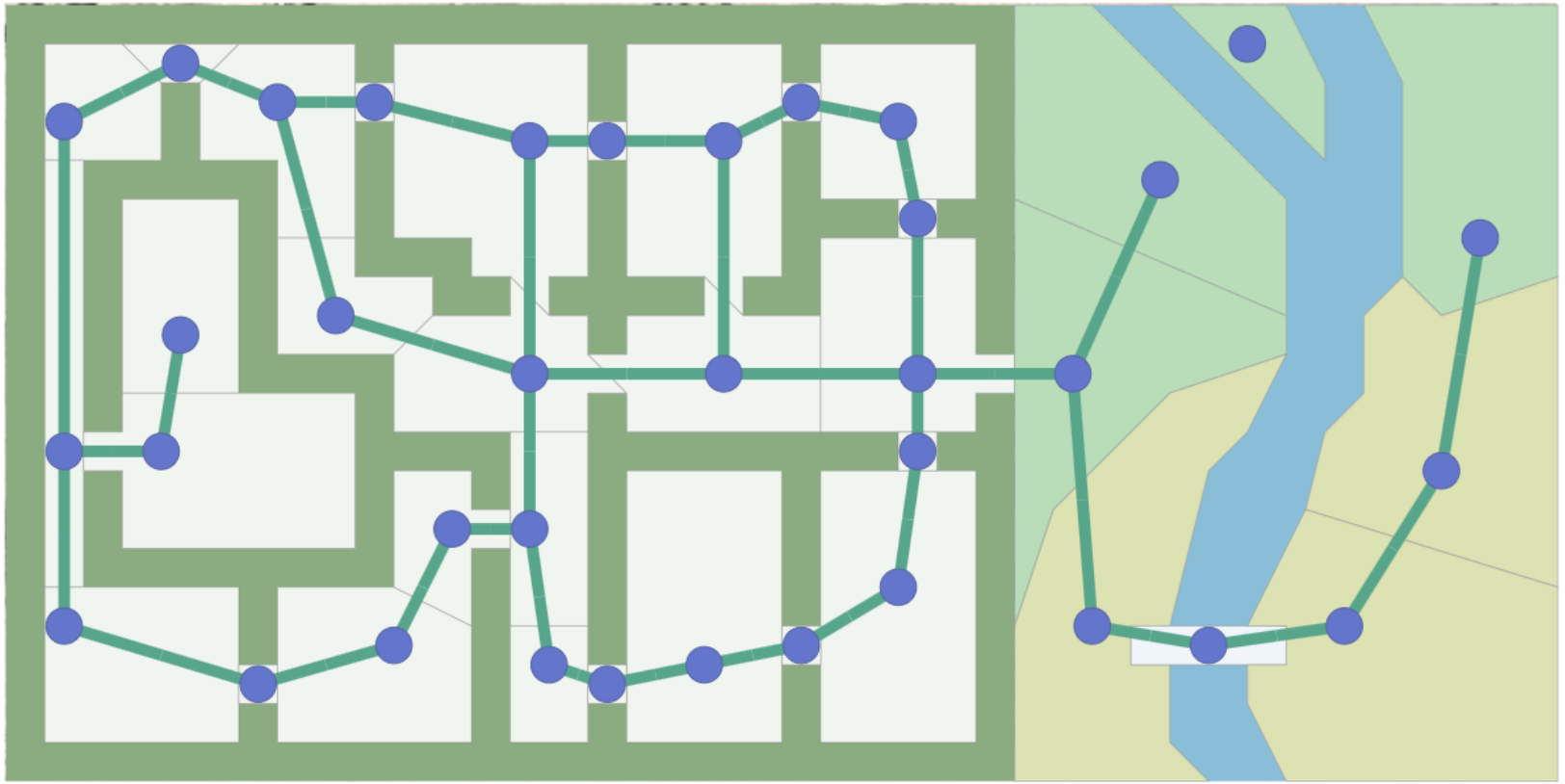
To find this path we can use a *graph search* algorithm, which works when the map is represented as a graph. **A*** is a popular choice for graph search. **Breadth First Search** is the simplest of the graph search algorithms, so let's start there, and we'll work our way up to A*.

Pathfinding in Games



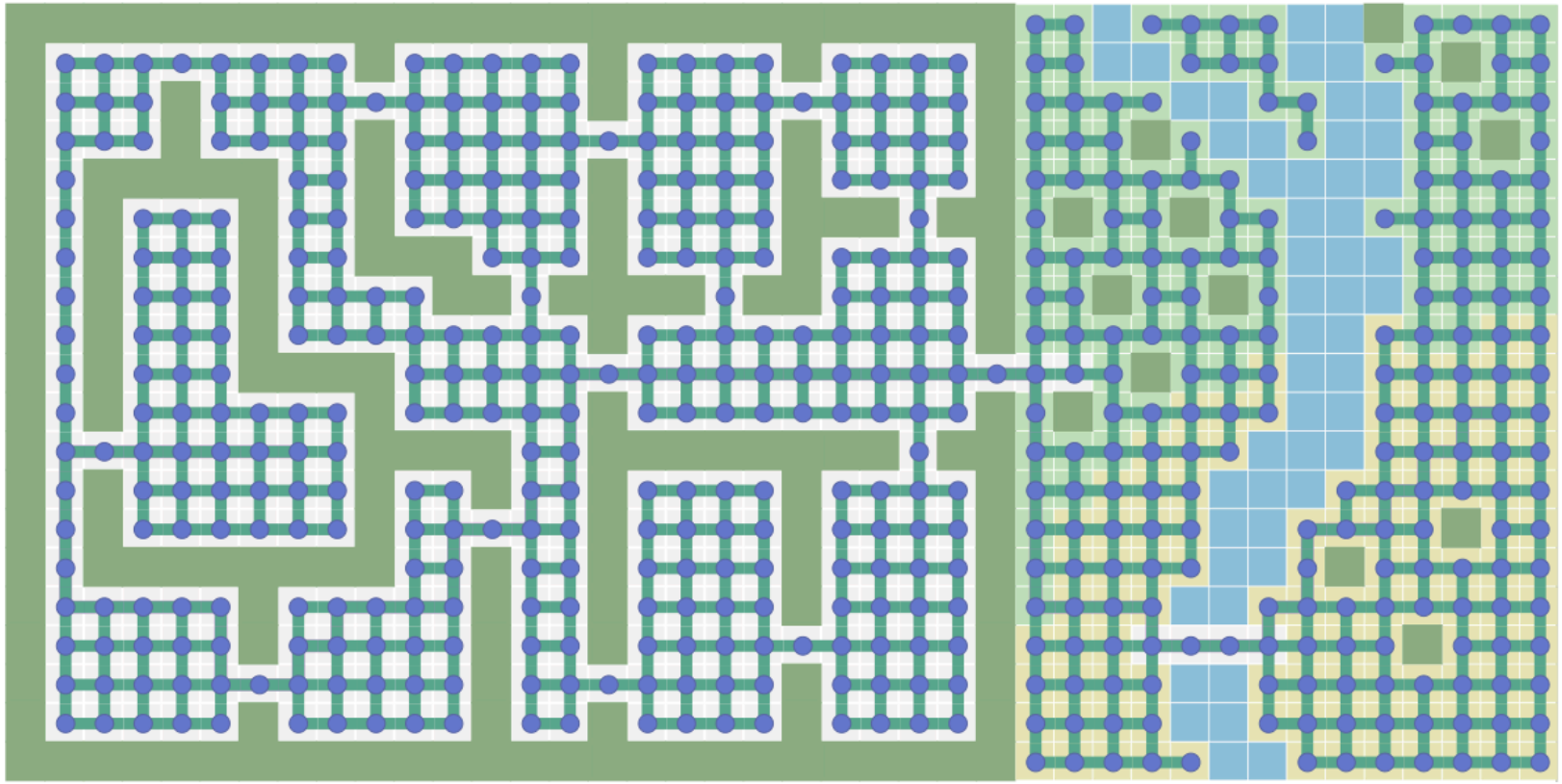
https://www.redblobgames.com/pathfinding/a-star/introduction.html

Pathfinding in Games



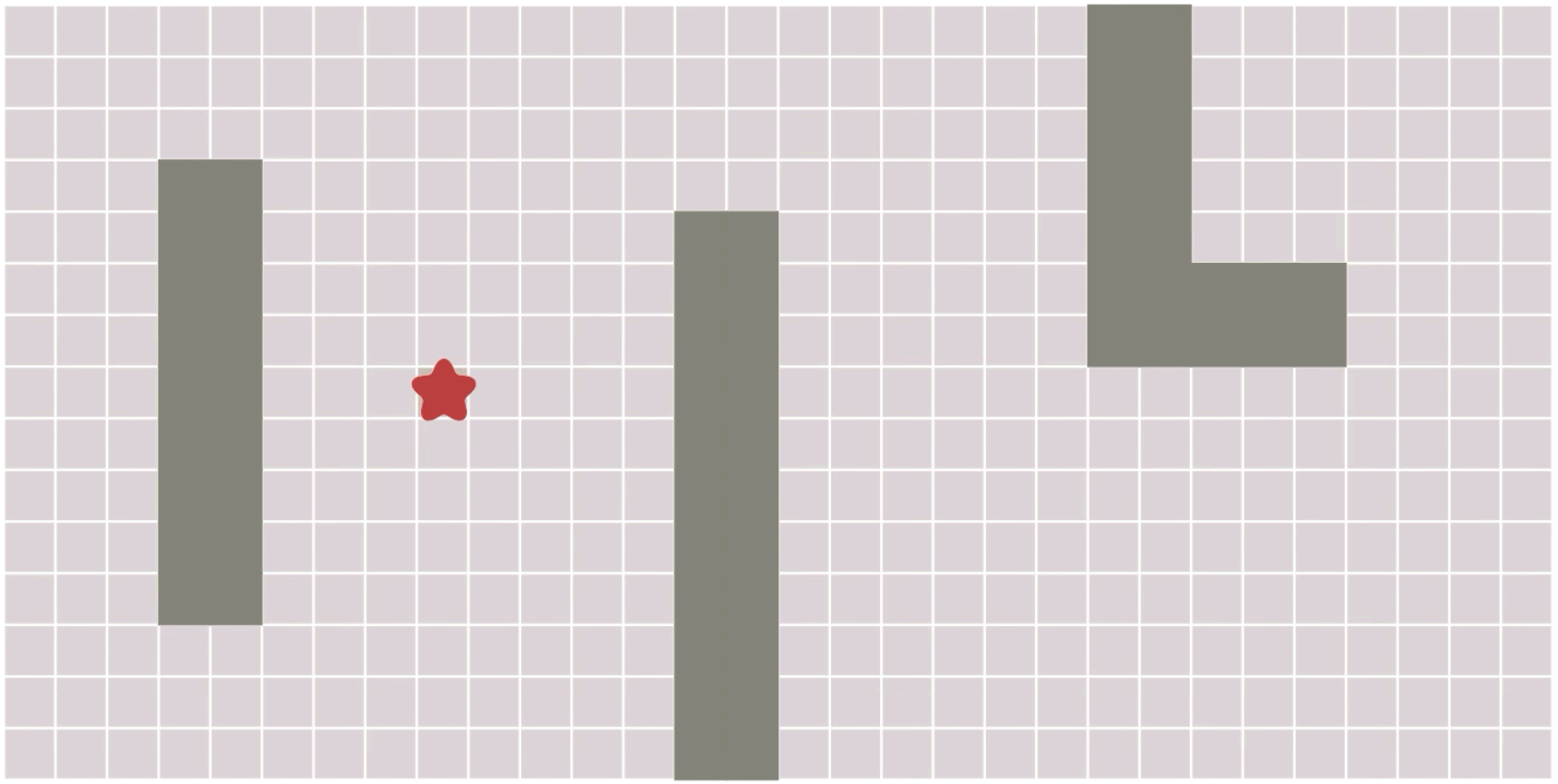
https://www.redblobgames.com/pathfinding/a-star/introduction.html

Pathfinding in Games



https://www.redblobgames.com/pathfinding/a-star/introduction.html

Breadth First Search



https://www.redblobgames.com/pathfinding/a-star/introduction.html

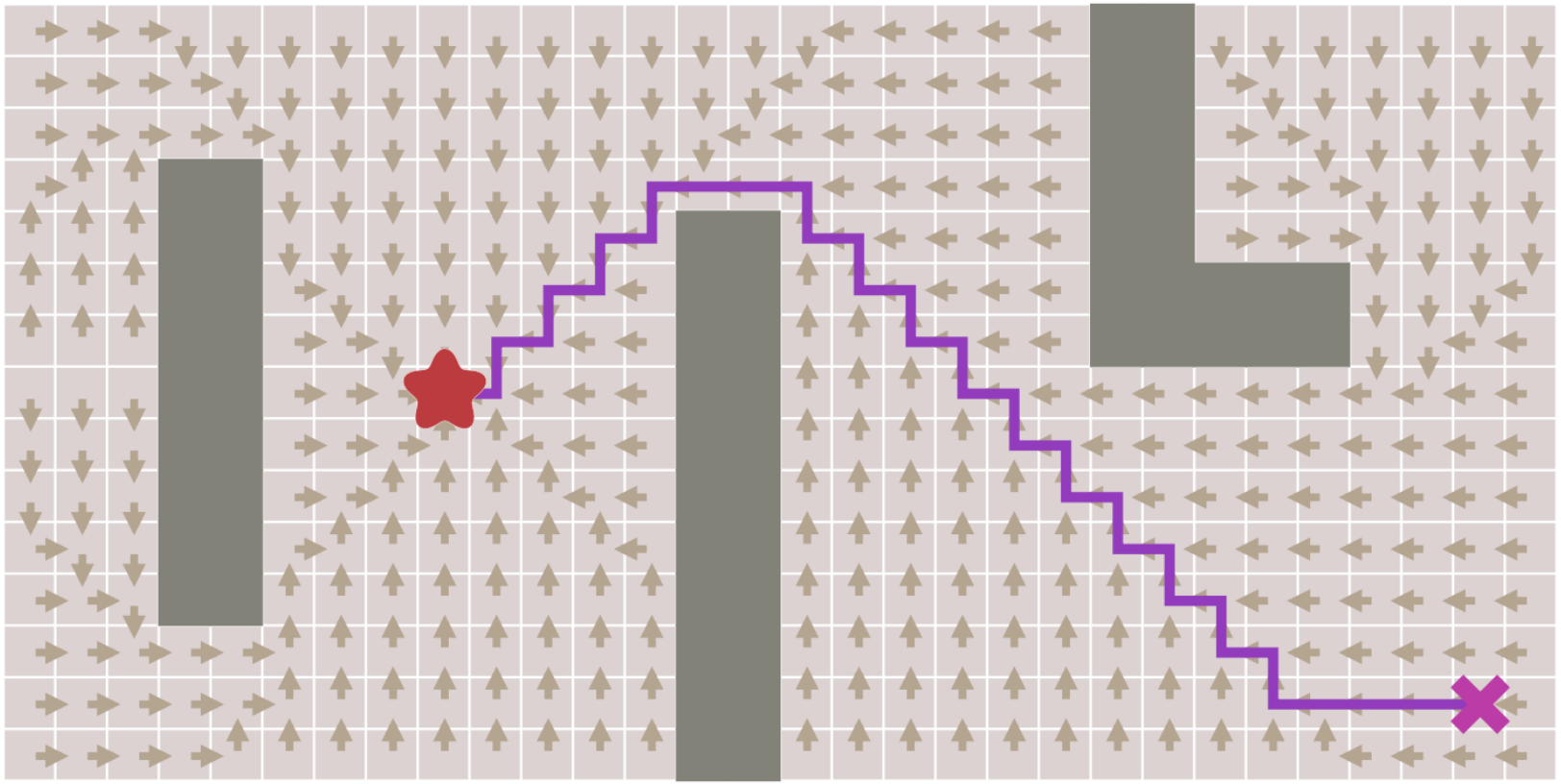
BFS in 10 lines of Python

```
frontier = Queue()
frontier.put(start)
visited = {}
visited[start] = True

while not frontier.empty():
    current = frontier.get()
    for next in graph.neighbors(current):
        if next not in visited:
            frontier.put(next)
            visited[next] = True
```

https://www.redblobgames.com/pathfinding/a-star/introduction.html

Finding the shortest path



https://www.redblobgames.com/pathfinding/a-star/introduction.html

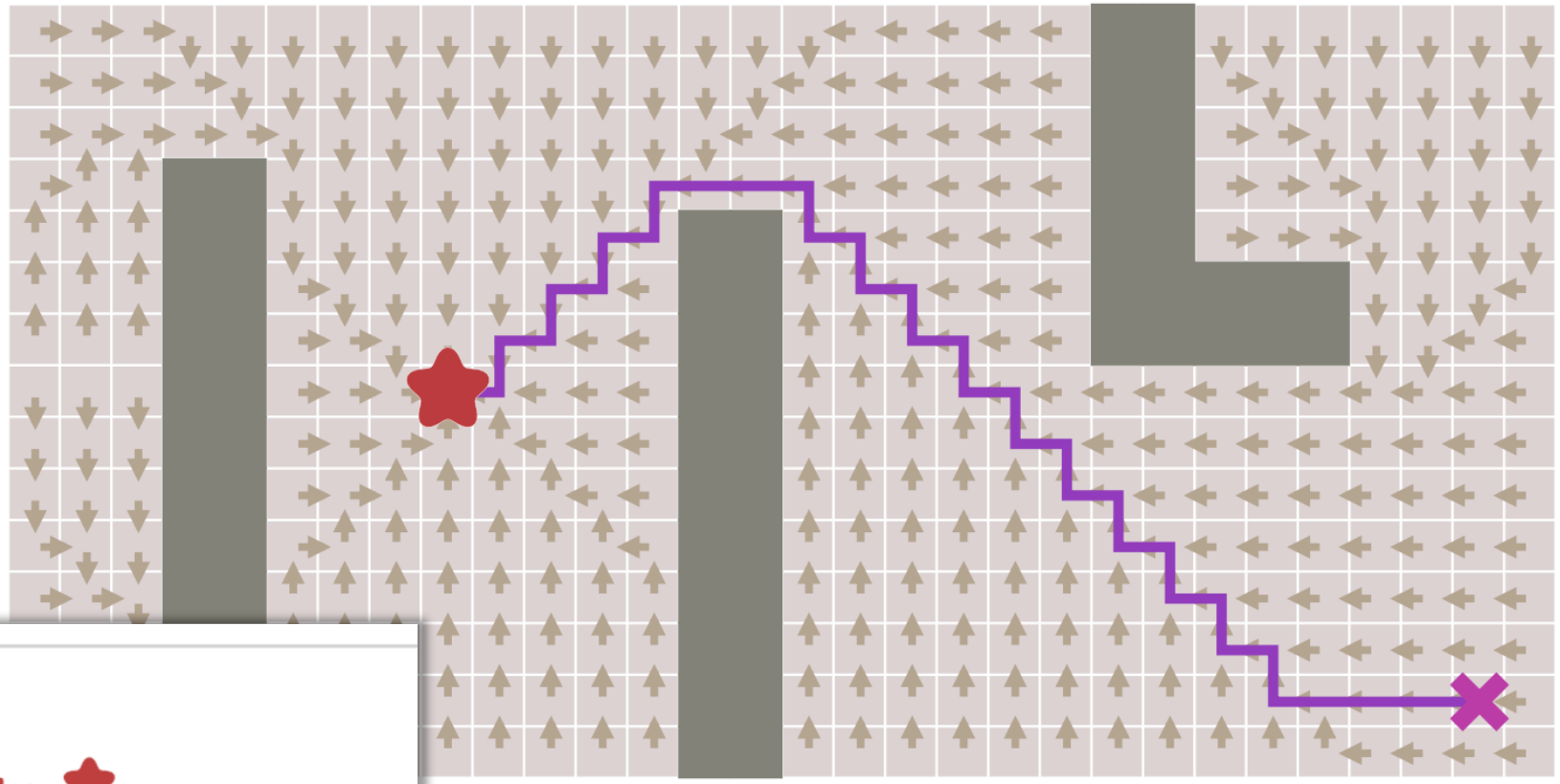
Finding the shortest path

```
frontier = Queue()
frontier.put(start)
came_from = {}
came_from[start] = None

while not frontier.empty():
    current = frontier.get()
    for next in graph.neighbors(current):
        if next not in came_from:
            frontier.put(next)
            came_from[next] = current
```

https://www.redblobgames.com/pathfinding/a-star/introduction.html

Finding the shortest path



```
current = goal
path = []
while current != start:
    path.append(current)
    current = came_from[current]
path.append(start) # optional
path.reverse() # optional
```