CSE 417 Autumn 2025

Lecture 12: Pathfinding

Glenn Sun

Logistics

- Quiz on Friday in class
  - Practice quiz posted on course website
  - Special concept check quiz for Wednesday
- HW 4 dates adjusted
  - New due date:
  - New resubmission date:

Wrapping up Prim’s algorithm

Prim's algorithm

1. **repeat $n - 1$ times**
2. Pick the cheapest edge that extends the current tree to a new vertex.

Remember only cheapest edge for every discovered vertex!

- Up to $O(n)$ vertices in data structure
- Operations that we do:
  - Pick cheapest vertex $O(n)$ times
  - Update the cost of vertices $O(m)$ times total

Priority queues

A priority queue is an abstract data type similar to a queue, but allows you to **get the highest priority elements first**.

It supports:

- **add** an element at a given priority
- **make** a whole priority queue from a given list with priorities
- **extract** the highest (minimum) priority element
- **peek** the highest (minimum) priority element (without deleting)
- **decrease the priority** of a given element to a new priority

Priority queues

	(last lecture) w/ array	(from 373) w/ binary heap	(just FYI) w/ Fibonacci heap
add(elt, prty)	$O(1)$	$O(\log n)$	$O(1)$
make(list)	$O(n)$	$O(n)$	$O(n)$
extractMin()	$O(n)$	$O(\log n)$	$O(\log n)$
peekMin()	$O(n)$	$O(1)$	$O(1)$
decrPriority(elt, prty)	$O(1)$	$O(\log n)$	$O(1)$ amortized

Running time of Prim's

As always, assuming $n \leq m \leq n^2$:

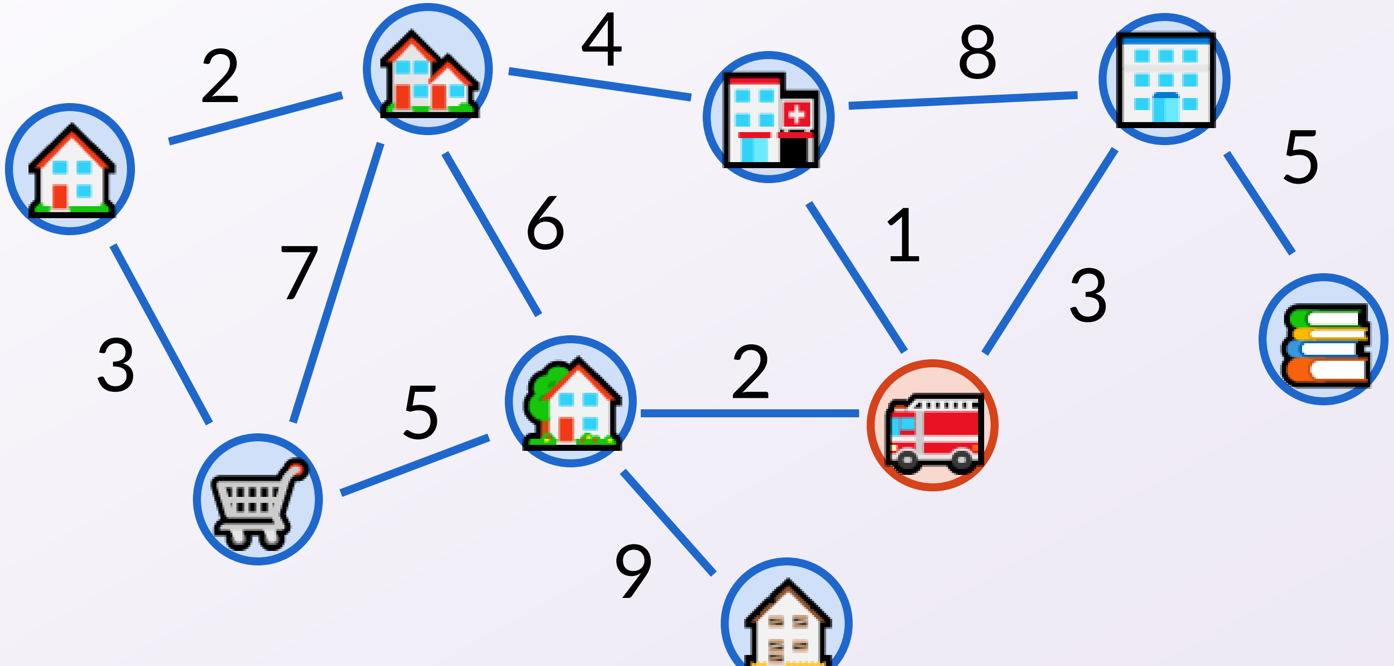
	w/ array	w/ binary heap	w/ Fibonacci heap
extractMin()	$O(n)$	$O(\log n)$	$O(\log n)$
decrPriority(elt, prty)	$O(1)$	$O(\log n)$	$O(1)$ amortized
Extract min $O(n)$ times and decrease priority $O(m)$ times			

Dijkstra’s algorithm

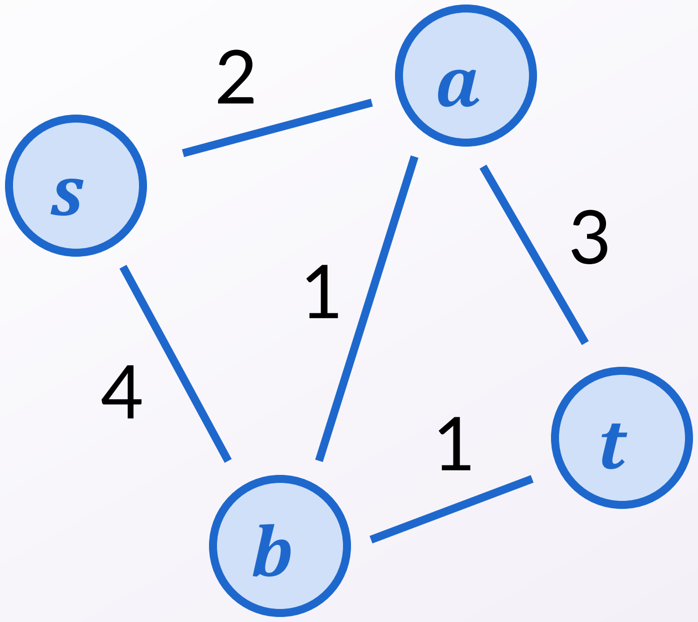
Emergency snow network

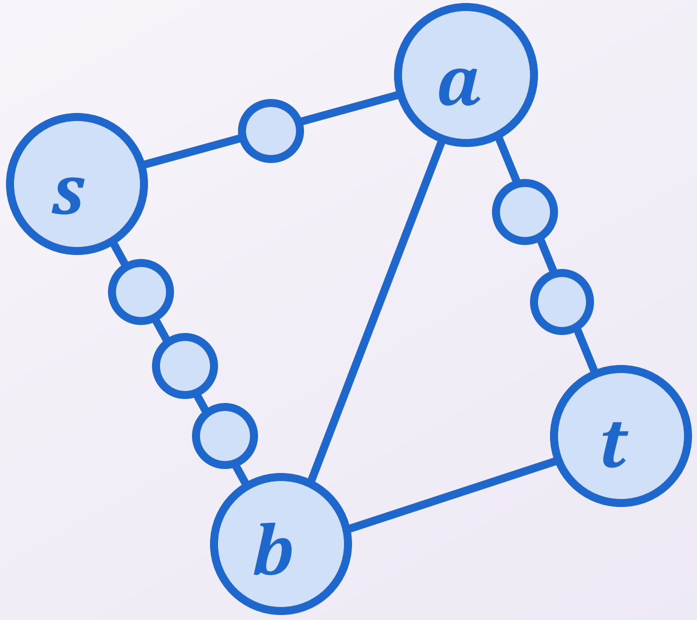
A city has a network of roads of various lengths.

Goal: Find a minimum length network to plow that ensures that the fire department can reach everyone as fast as possible.

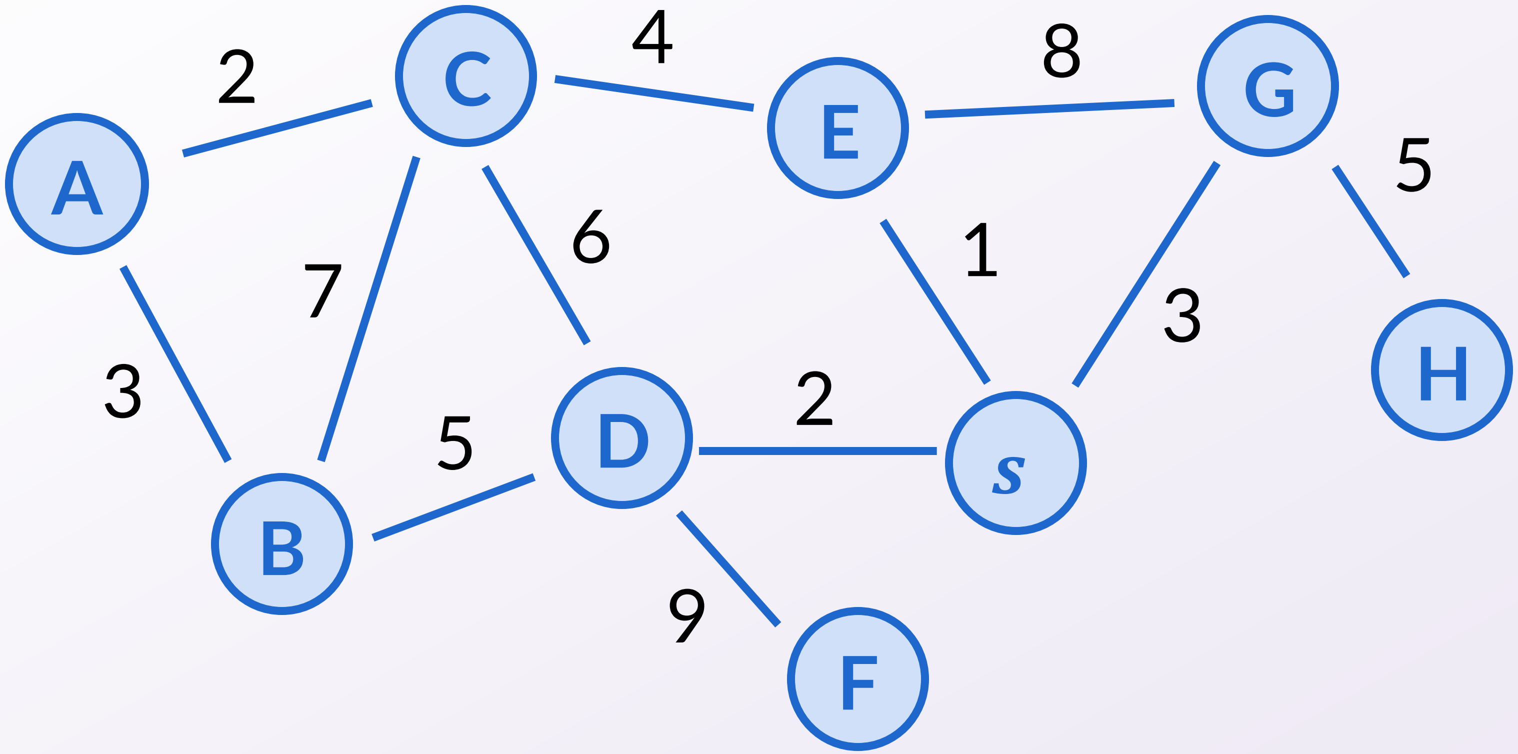


Dijkstra's algorithm is basically BFS





A larger example



Dijkstra's algorithm

1. Set $\mathbf{currPriority}[s] = 0$ and $\mathbf{currPriority}[x] = \infty$ for other x.
2. Make a priority queue Q with every vertex at its priority above.
3. **while** Q is not empty **do**
4. Get/remove the next vertex x from Q.
5. **for all** out-neighbors y of x **do**
6. **if** $\mathbf{currPriority}[y] > \mathbf{currPriority}[x] + \mathbf{dist}(x, y)$ **then**
7. Update $\mathbf{currPriority}[y] = \mathbf{currPriority}[x] + \mathbf{dist}(x, y)$.
8. Decrease the priority of y in Q to $\mathbf{currPriority}[y]$.
9. **return** $\mathbf{currPriority}$

Running time of Dijkstra’s

- Up to $\boldsymbol{n}$ “extract” operations (get each vertex at most once)
- Up to $\boldsymbol{m}$ “decrease priority” operations (look at each edge at most once)

Same as Prim’s algorithm!

- With $\boldsymbol{O(\log n)}$ “extract” and “decrease priority”: $\boldsymbol{O(m \log n)}$
- With Fibonacci heap ($\boldsymbol{O(1)}$ ”decrease priority”): $\boldsymbol{O(m + n \log n)}$

Shortest s-t path

Dijkstra's algorithm can be used to:

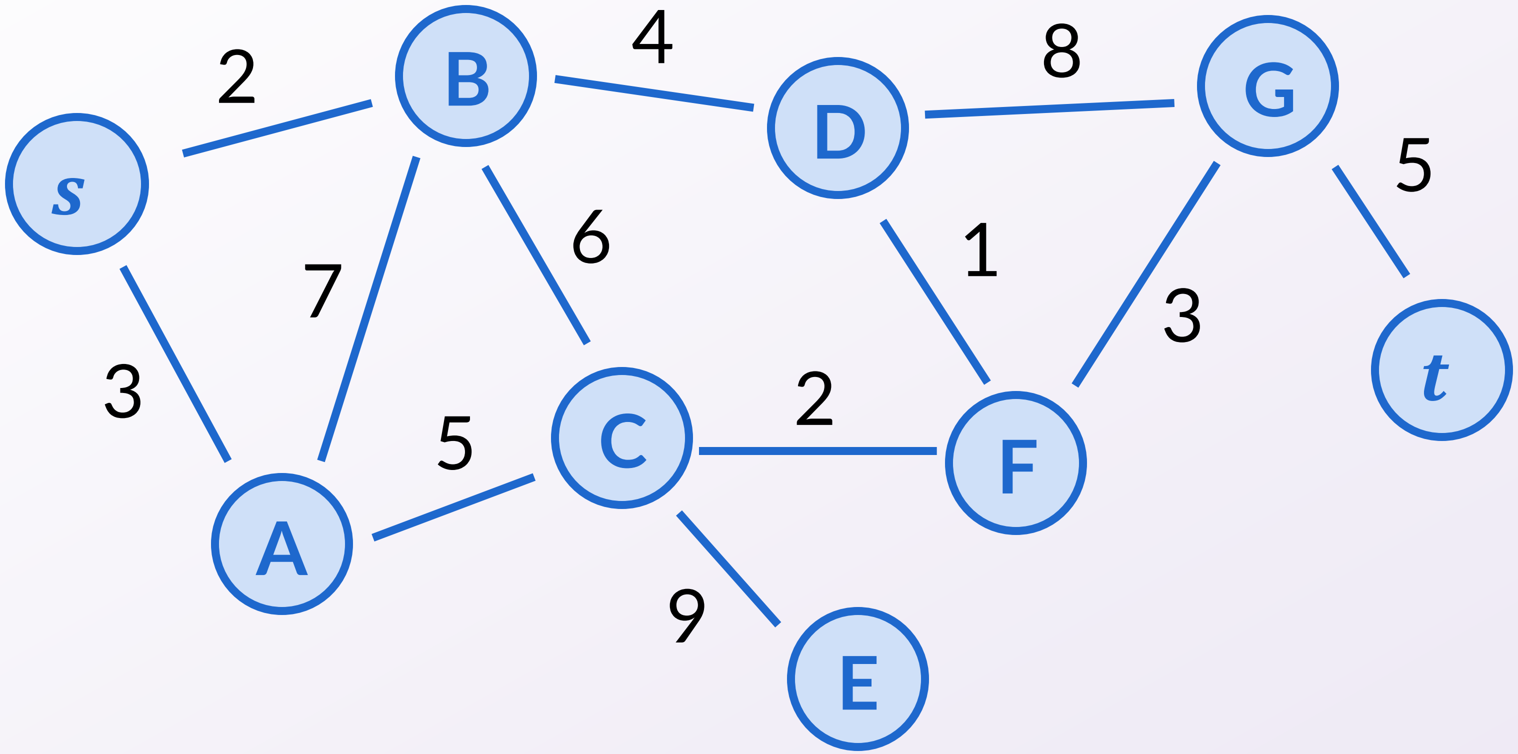
- Compute the **distance of every vertex** from a given point
- Compute the **distance between two points**
  - Just break upon finding the desired point
  - No asymptotic improvement in running time

Leaving crumbs to trace the solution

What if instead of finding the **minimum length** of a path from s to t, you were asked to find an **actual path** from s to t whose length is minimum?

Common strategy: leave "breadcrumbs" tell you where you came from in your search

Leaving crumbs to trace the solution



Leaving crumbs to trace the solution

1. Set $\mathbf{currPriority}[s] = 0$ and $\mathbf{currPriority}[x] = \infty$ for other x.
2. Make a priority queue Q with every vertex at its priority above.
3. **while** Q is not empty **do**
4.   Get/remove the next vertex x from Q.
5.   **for all** out-neighbors y of x **do**
6.   **if** $\mathbf{currPriority}[y] > \mathbf{currPriority}[x] + \mathbf{dist}(x, y)$ **then**
7.   Set $\mathbf{cameFrom}[y] = x$.
8.   Update $\mathbf{currPriority}[y] = \mathbf{currPriority}[x] + \mathbf{dist}(x, y)$.
9.   Decrease the priority of y in Q to $\mathbf{currPriority}[y]$.

Leaving crumbs to trace the solution

Now trace back to construct the path:

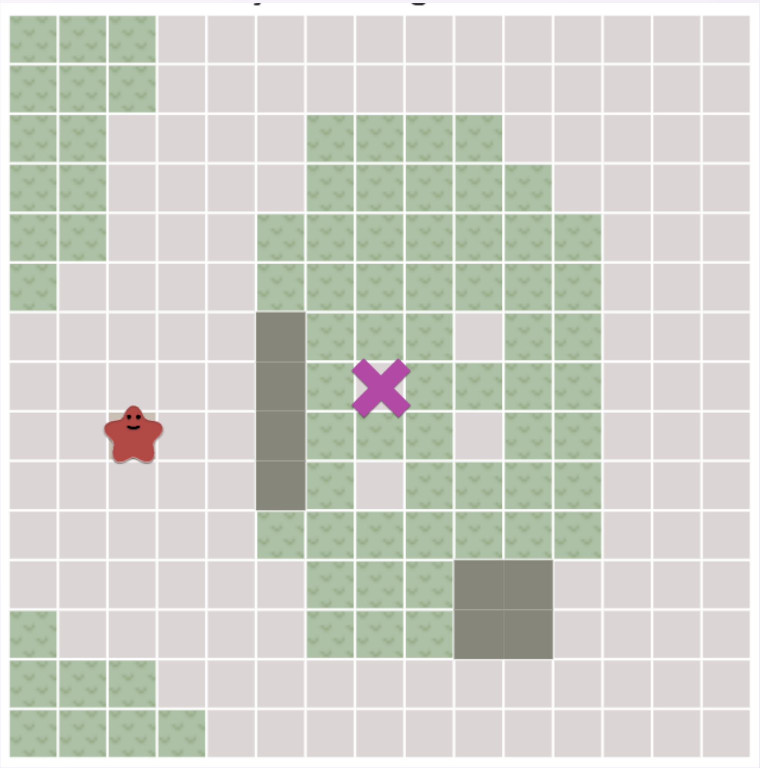
10. Let $v = t$ and initialize a list of vertices $\mathbf{path}$ to $[t]$.
11. **while** $v \neq s$ **do**
12.   Update $v = \mathbf{cameFrom}[v]$.
13.   Append v to $\mathbf{path}$.
14. **return** $\mathbf{path}$ in reverse order

A* search and heuristic algorithms

Pathfinding on a grid

Suppose a video character is moving through a 2D area represented by a grid of tiles:

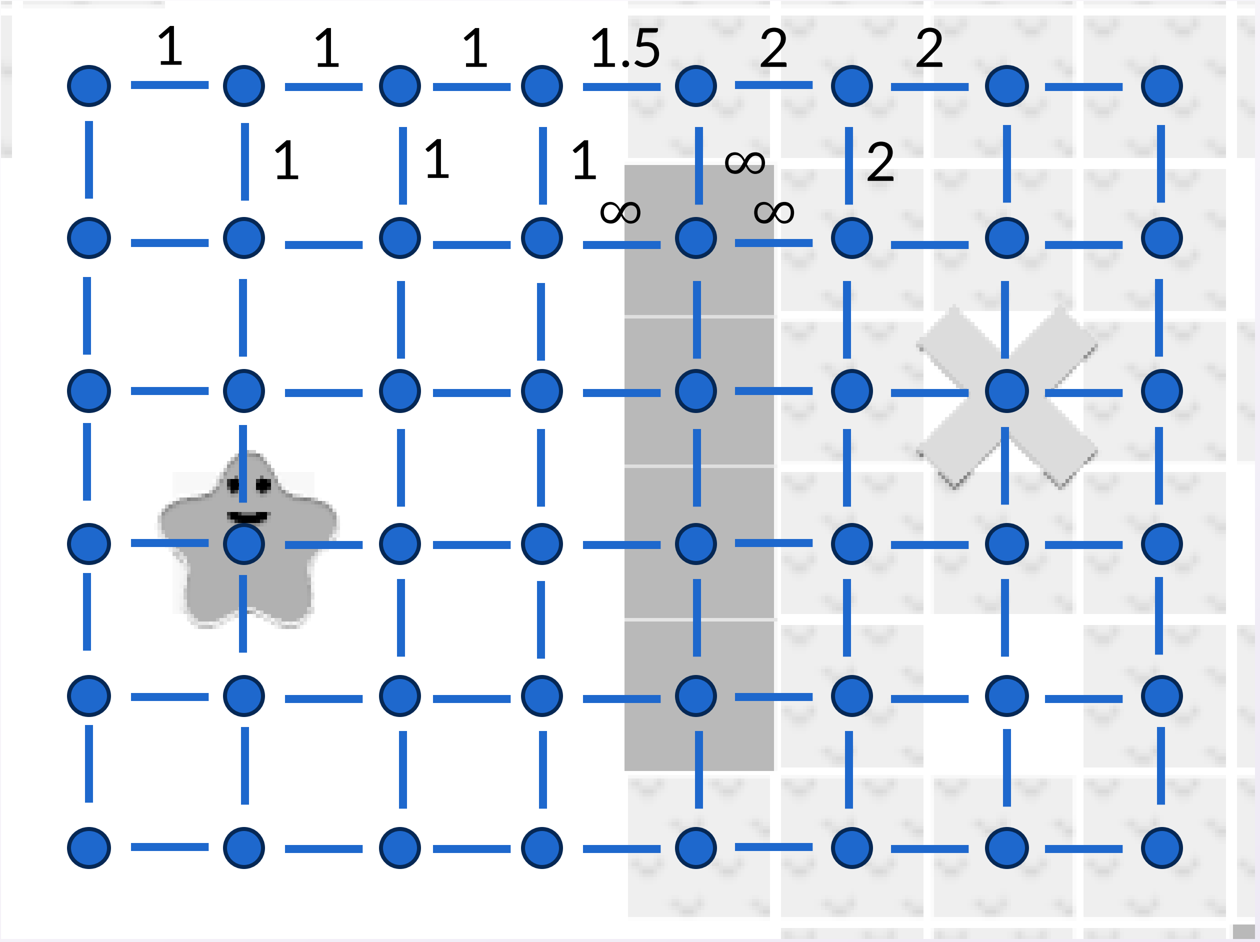
- Dirt tiles: fast to walk on, 1 sec/tile
- Tall grass: slow to walk through: 2 sec/tile
- Boulders: impassible objects
- Maybe more



graphics: [Amit Patel]

Pathfinding on a grid

Idea #1: Dijkstra's algorithm on a grid



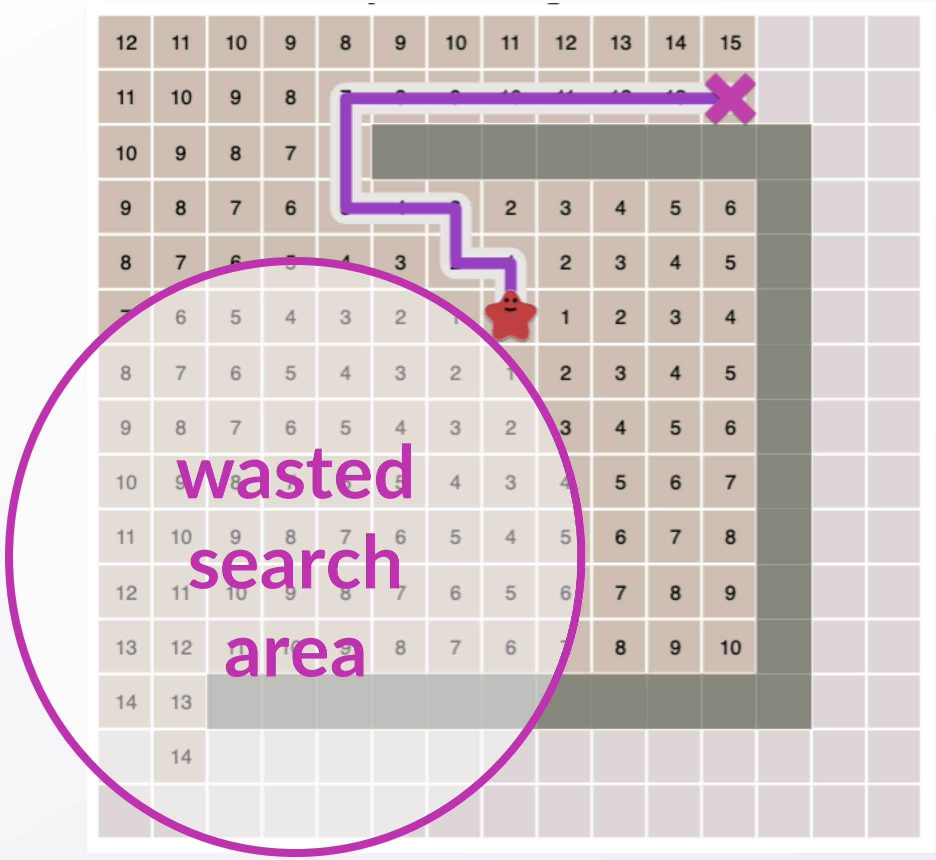
Dijkstra’s for pathfinding

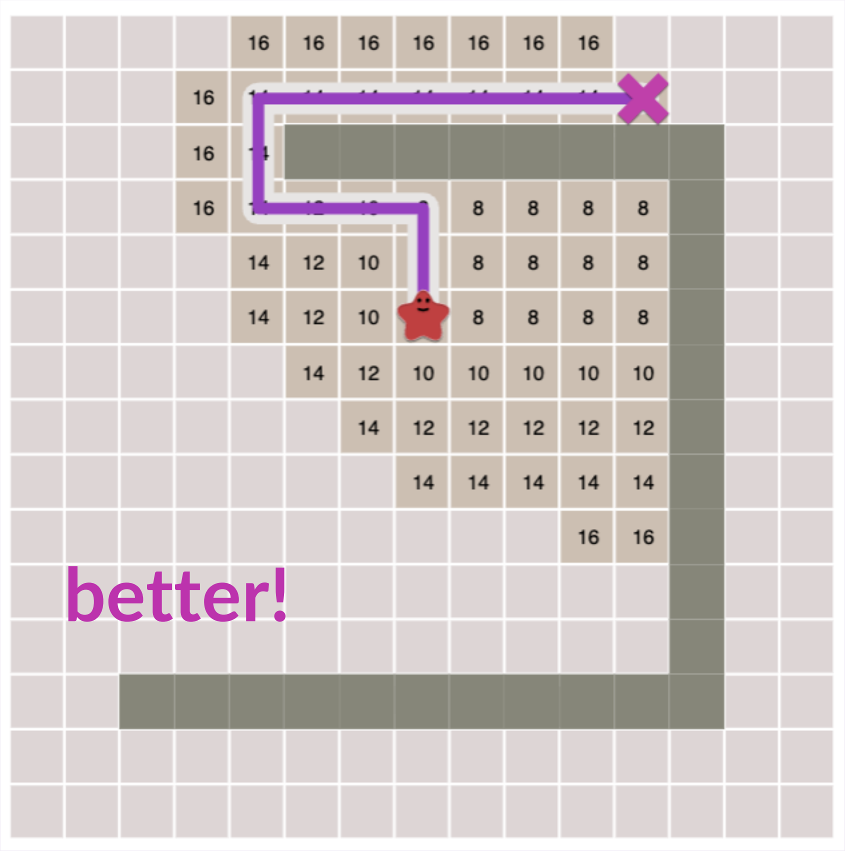
Use the “breadcrumbs” modification we discussed earlier.

Works okay! Guaranteed to find solution in $O(n)$ time, where n is the number of pixels in the grid.

Dijkstra's for pathfinding

Dijkstra's is not ideal though, because breadth-first is a bad search strategy in 2D.





A* search

Define a heuristic to quickly gauge how good a choice is:

$$\mathbf{heuristicDist}\big((x_1, y_1), (x_2, y_2)\big) = |x_1 - x_2| + |y_1 - y_2|$$

In Dijkstra's algorithm, replace:

$$\mathbf{currPriority}[y] = \mathbf{currPriority}[x] + \mathbf{dist}(x, y)$$

with

$$\mathbf{currPriority}[y] = \mathbf{currDistance}[x] + \mathbf{dist}(x, y) + \mathbf{heurDist}(y, t)$$

goal

A* search is not asymptotically better

Sometimes, algorithms are not designed to be asymptotically better. Small improvements for a particular application matter!

But is A* search still correct for shortest paths?

Theorem. If the heuristic only ever underestimates distance, A* search still produces the shortest path!

Final reminders

Fill out concept check for Wednesday review topics!

I have OH now-12:30pm:

- Meet at front of classroom, we'll walk over together
- CSE (Allen) 214 if you're coming later

Nathan has online OH 12–1pm:

- https://washington.zoom.us/my/nathanbrunelle