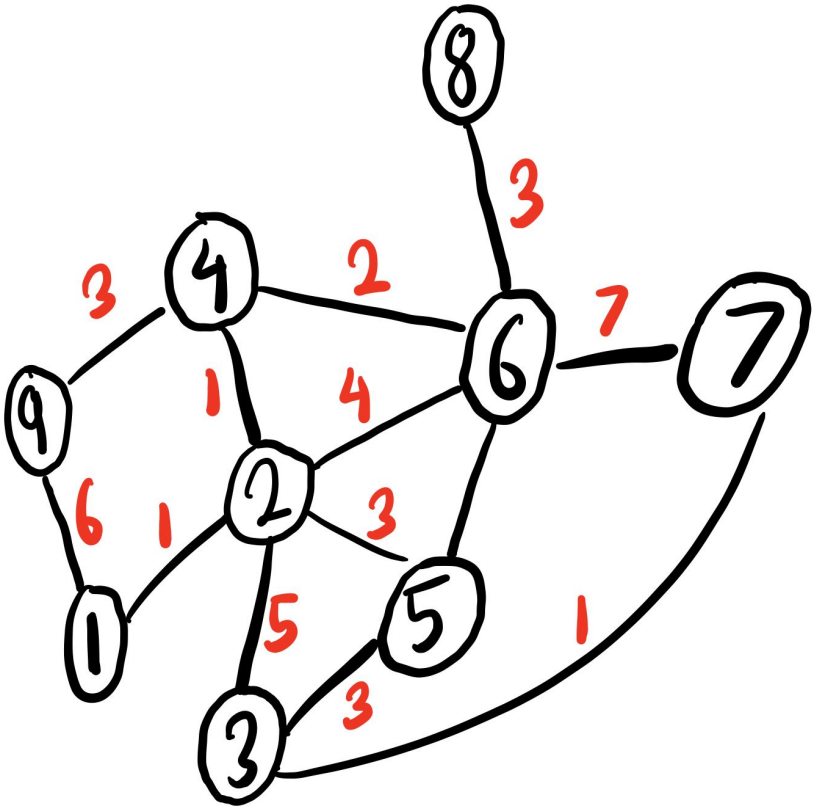
Dijkstra's Algorithm

August 7th, 2021 (Class #5)

Weighted Graphs



Every edge has a weight!

The Shortest Path Problem… Again!

weights are positive

Input: A weighted graph and a start node (START).

Output: Shortest length paths from START to other nodes.

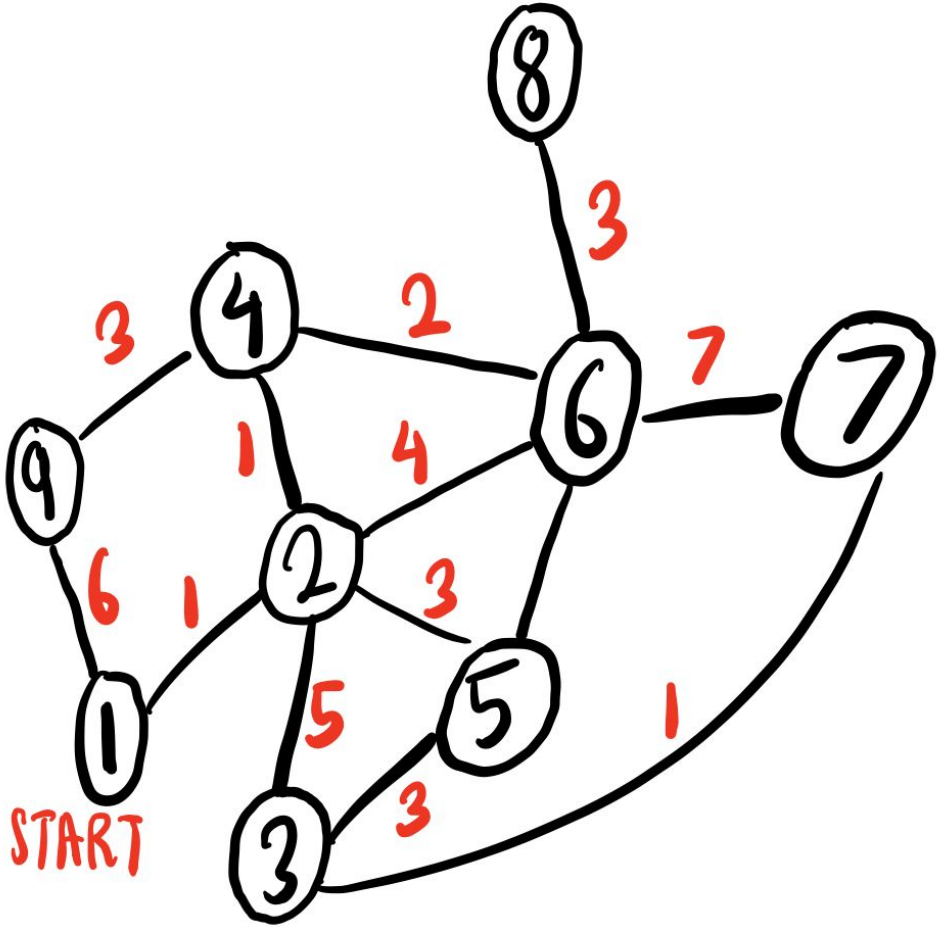
Let's try some algorithms...

Brute force?

Breadth first search?

(from class 3!)

Ideas?

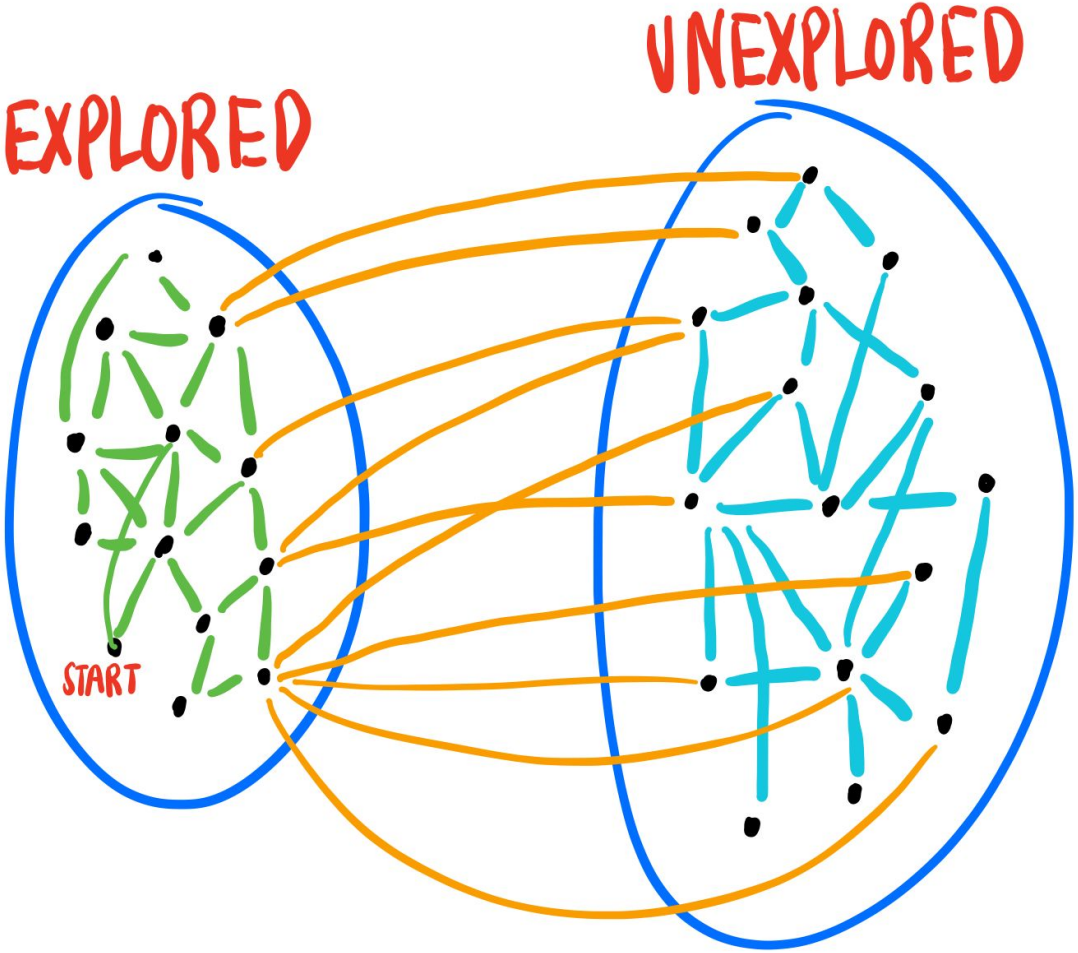


Graph Traversal

Remember: A graph traversal algorithm explores nodes one-by-one.

We need to specify: which unexplored node do we explore next?

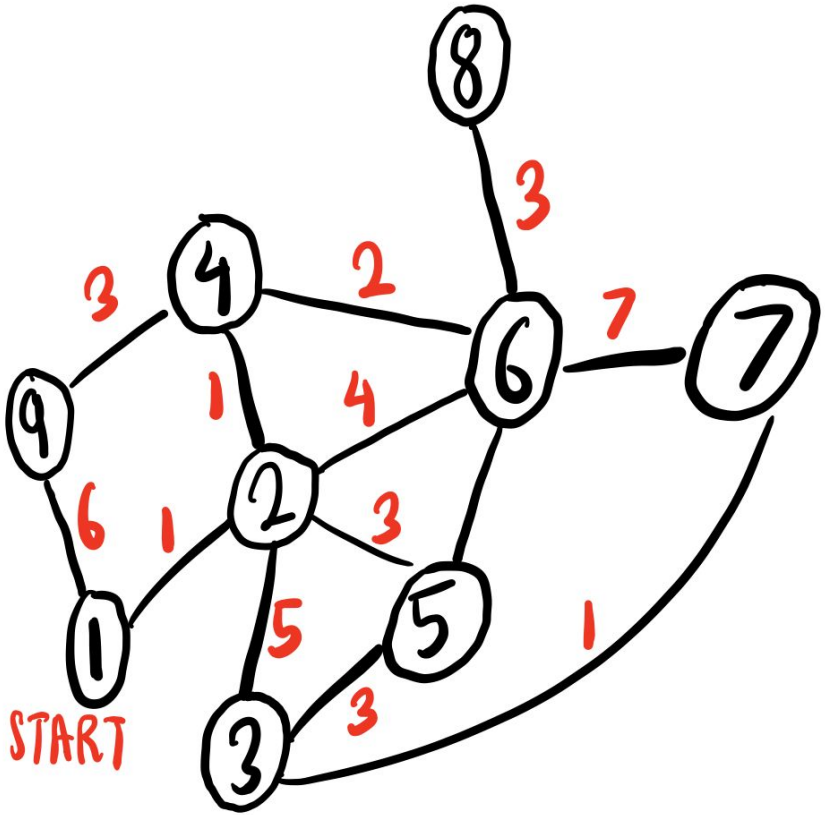
Which edge do we traverse next?



Goal

We want to guarantee that we get to every node by the shortest length path possible.

Let's discover Dijkstra's Algorithm!



5 minute break

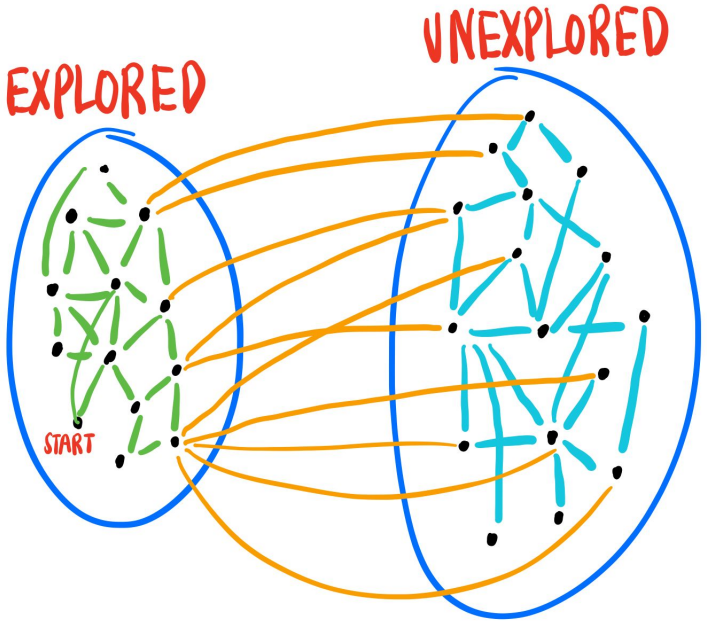
Key Idea of Dijkstra's Algorithm

"Predicted" shortest path:

Shortest path to a node if we only use green and orange edges.

green: between explored nodes

orange: from explored to unexplored



Explore the node with the shortest predicted shortest path...

Since this will be its actual shortest path!!

Description of Dijkstra's Algorithm

- Start with only the START node explored

While there are unexplored nodes...

- Find the node with the shortest predicted distance.
- Explore it... and update the predicted distances of the remaining unexplored nodes.

- We have found the shortest path from START to every node!

How fast is Dijkstra's algorithm?

Graph has n nodes and m edges...

Naively, it takes $O(n)$ time to find the next node.

Total running time: $O(n^2)$.

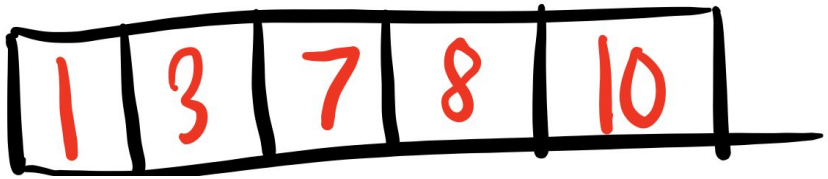
Can we do better?

Yes. We need a data structure that supports:

Priority queue

1. insert/update values of nodes
2. find + remove the node w/ minimum value

How about using a sorted array?



How fast is ...

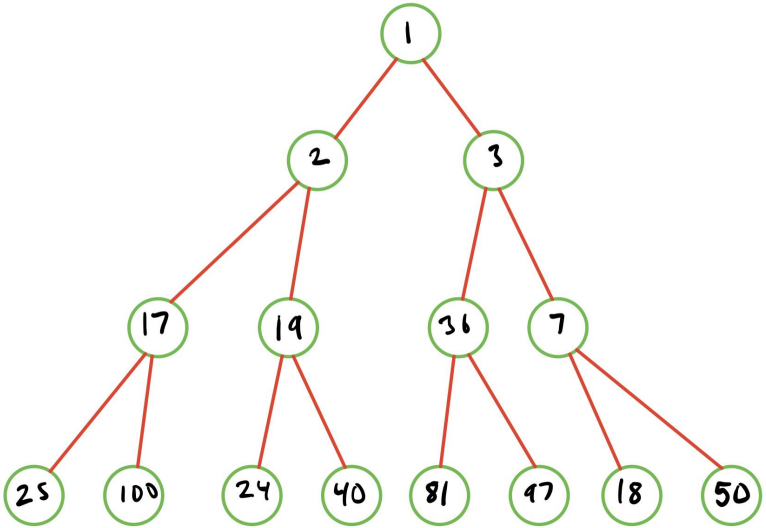
Find min?

Insertion?

The need for data structures

- Algorithms go from a fixed input to an output.
- But sometimes your data is dynamic: Can receive queries and updates – need to handle both fast!

We can do better than a sorted array.



In a binary heap...

Remove min is O(log n)

... and so is updating!

insert, delete, etc.

How fast is Dijkstra's with a binary heap?

- We have to update predicted distances at most m times.
- So Dijkstra's becomes $O(m \log n)$ with a binary heap. $\underbrace{\text{nearly linear!}}$