24.1 Introduction

Shortest Paths

Recalls

So far, we have methods to do the following

- find a path from a given vertex, `s`, to every reachable vertex in the graph.
- find a **shortest** path from a given vertex, `s`, to every reachable vertex in the graph. (...or do we?)

Before we answer the mysterious question posed below, let's further recall the two types of searches we could use to do the above two things: BFS or DFS.

Are both going to always be correct? Yes. Does one give better results? BFS finds you the **shortest** paths whereas DFS does not. Is one more efficient than the other, runtime-wise? No. Is one more efficient than the other, space-wise?

- DFS is worse for spindly graphs. Imagine a graph with 10000 nodes all spindly. We'll end up making 10000 recursive calls, which is bad for space.
- BFS is worse for "bushy" graphs, because our queue gets used a lot.

Answering the mysterious question

Did we develop an algorithm to find the **shortest** path from a given vertex to every other reachable vertex? Well, kind of. We developed an algorithm that works well on graphs with no edge labels. Here's what we did: we developed an algorithm that finds us the shortest (**where shortest means the fewest number of edges**) paths from a given source vertex.

But that's not always the correct definition of shortest. Sometimes, our graph edges might have 'weights', and A-B is considered farther than A-C if the A-B edge has weight, say, 5 and the A-C edge only has weight, say, 3.

That is why we need a different algorithm for finding the shortest path when we have the edge weights in the graph.

Previous
24. Shortest Paths

Next
24.2 Dijkstra's Algorithm

Last updated 1 year ago