Shortest Path - FAQ

How do I calculate the Euclidean distance between two points again?

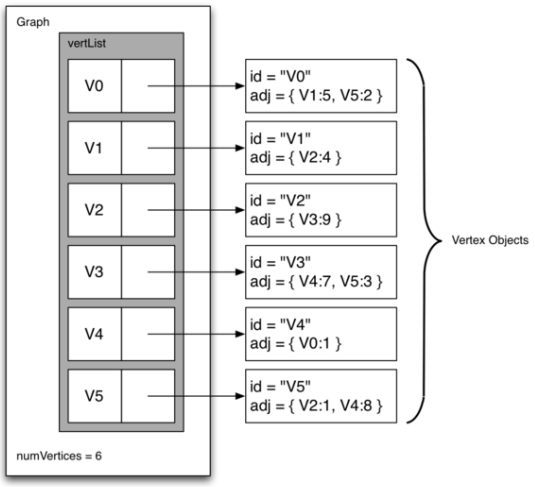
In the Euclidean plane, if $\mathbf{p} = (p_1, p_2)$ and $\mathbf{q} = (q_1, q_2)$ then the distance is given by

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

The abstractions in this lab make no sense. How else could I represent the graph?

The suggested representation is an "adjacency list", basically a list of vertices where each maintains its own list of edges. There are many ways this could be done, here are a couple more:

- Make a Node class that wraps a Vertex and a `next` reference (essentially making your own linked list). The adjacency list would then be an array of Nodes.
- Use a map instead of a list. For example, there could be a master map of the vertices in the graph. Each would maintain its own list of edges (as values linked to the keys).
    - You could also have each vertex maintain its *own* map, e.g. where the keys are the connected nodes and the values are the distances (weights) to that node. This wouldn't be a bad choice for this project. Example, borrowed from bradfieldcs.com:



I've read all the notes, but I still don't understand Dijkstra's algorithm and/or graphs.

This stuff isn't easy. Thankfully, there are a *lot* of resources on the web covering shortest path problems (and specifically Dijkstra's algorithm). Find an explanation / visualization that works for you. Here are a few more to try if you're not satisfied with the information I provided:

1. https://www.hackerrank.com/topics/shortest-paths-in-graphs
2. http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf
3. https://www.cs.usfca.edu/~galles/visualization/Dijkstra.html
4. https://bradfieldcs.com/algos/graphs/dijkstras-algorithm/
    a. The code examples in this link are in Python, but the article is excellent
5. http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html#model

Why do I need Dijkstra's algorithm if a BFS finds a shortest path already?

This only works with un-weighted graphs; when there is a cost associated with an edge, a BFS *won't* (necessarily) find a shortest path.