Dijkstra's Algorithm

CSCI 2275

Traversal Methods

DFS: Gives…. *a* path between two vertices.

BFS: Gives the shortest *edge-wise* path between two given vertices..

We aren't always (or usually) concerned with finding the shortest edge-wise path, since this doesn't map to real life graph applications.

Weighted Graphs

A **weighted graph** is one in which the edges between nodes have ‘costs’, or ‘weights’.

The traversal methods with which we are familiar will *not* consider these weights, and therefore will fail to produce the shortest path between two given vertices.

Enter... Dijkstra's!! (DYKE-struh's)

Given a source vertex and a graph, Dijkstra's algorithm creates a **single-source shortest path tree.**

That is, it creates a path between two vertices (or, a given vertex and *all* vertices).

Important: Dijkstra's initializes all vertices other than *start* with a large (let's say, infinite) distance.

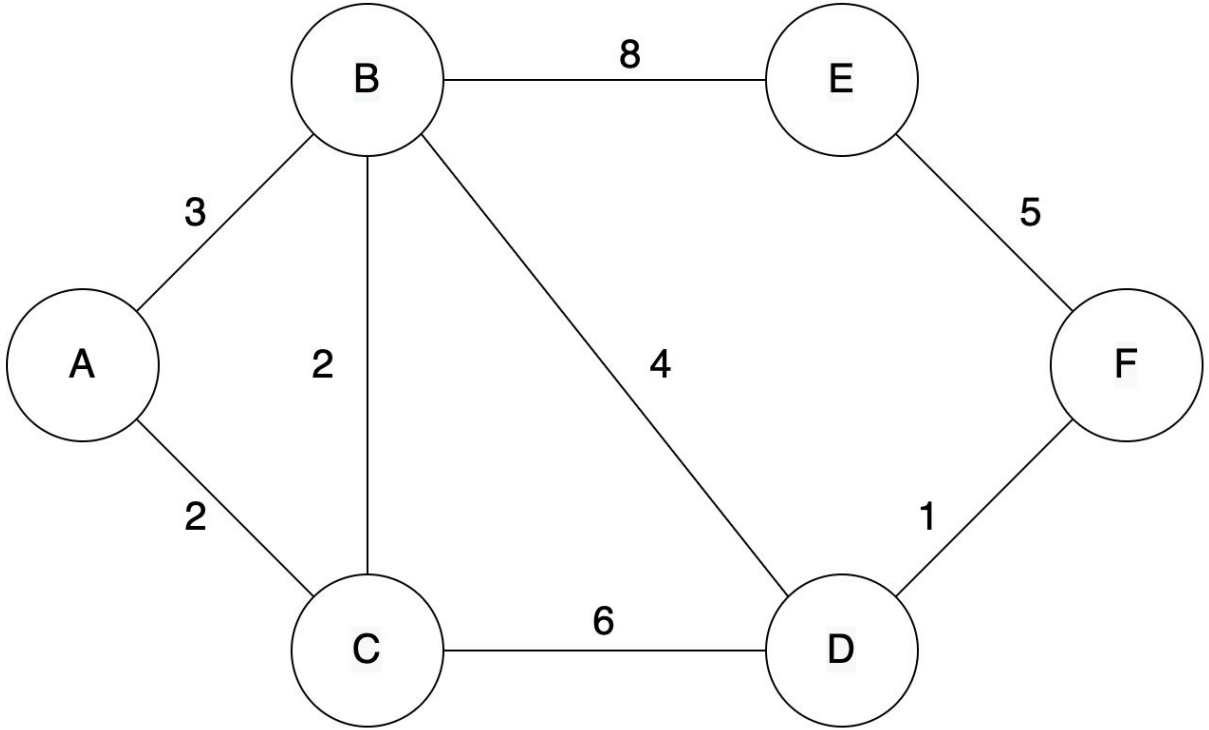
Big Picture Dijkstra's

1. Let *current* = *start*
2. Consider *current's* unvisited neighbors, update their distances from *start*.
    - What will this distance be?
3. Consider the vertex with the shortest distance, and *visit* it. Update its distance from *start*.
    - What will this distance be?

Big Picture Dijkstra's

1. Let *current* = *start*
2. Consider *current's* unvisited neighbors, update their distances from *start*.
3. Consider the vertex with the shortest distance, and *visit* it. Update its distance from *start*.
4. Repeat 2-3 until all vertices are visited/source vertex is found!!!
5. Profit. :D

Example



Your exercise

Perform Dijkstra’s on the given graph (see next slide)

Print the path back to source from all of the vertices in the graph.

Visual of Exercise Graph