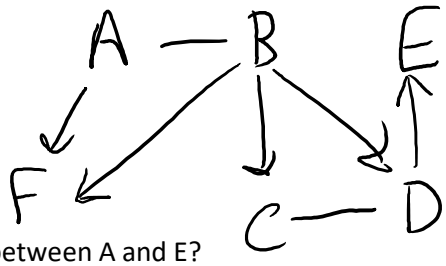
2018-04-12 Graph Tidbits

Thursday, April 12, 2018 9:38 AM

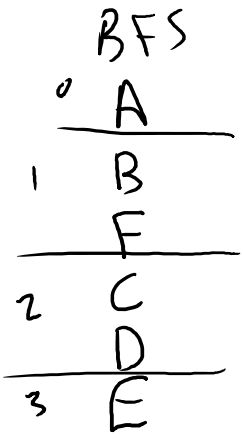
"Simple" Shortest Path

Given the graph:



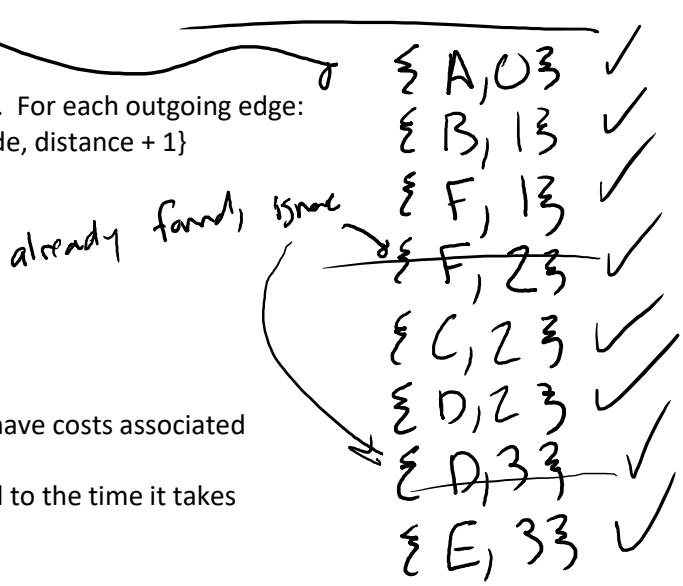
What is the shortest path between A and E?

- The solution merely requires a BFS
- Items come out of a queue based on distance
- To find distance, we must maintain a "distance" counter
- Programmatically, this requires us to push additional information into our queue.



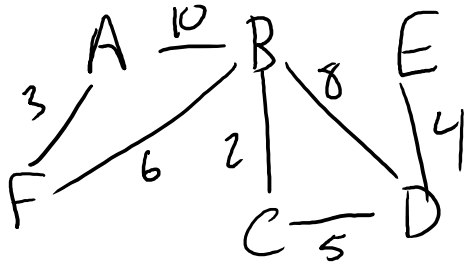
Pseudocode:

1. Push {start, 0} into queue
2. While queue is not empty:
    a. Pop top KVP {node, distance}. For each outgoing edge:
        i. If not known, push {node, distance + 1}



"Full" Shortest Path

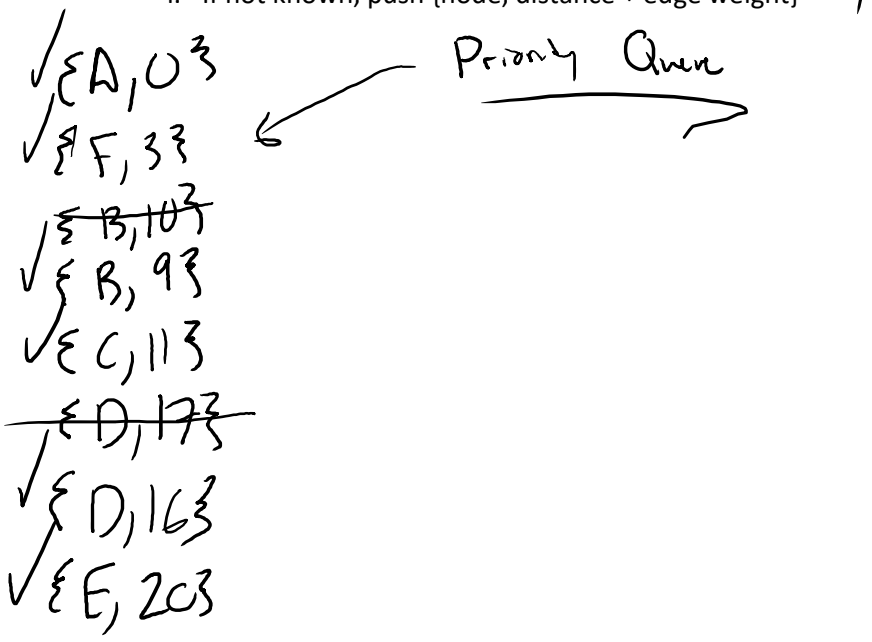
- Consider a weighted graph (edges have costs associated with them).
- Example: roads have weight related to the time it takes to travel from one point to another



Dijkstra's Algorithm

- Instead of using a normal queue, use a priority queue.
  Pseudocode:
    1. Push {start, 0} into priority queue
    2. While priority queue is not empty:
        a. Pop top KVP {node, distance}. For each outgoing edge:
            i. If not known, push {node, distance + edge weight}

a. Pop top KVP {node, distance}. For each outgoing edge:
    i. If not known, push {node, distance + edge weight}



Minimum Spanning Tree

- Given an infinite run of wire to fully connect a graph, which wire should we keep in order to use the least amount of wire
- Examples: computer network topology
    - Power wires

Prim's Algorithm (TODO: verify this)

General idea

- We need to maintain a list of visited nodes
- Given some starting location, push all outgoing edges into a priority queue
- While not all nodes have been visited:
    - Pop off the least cost edge. If that node has not been visited, push all of its edges onto the PQ. Remember that we took this edge. Mark as visited.