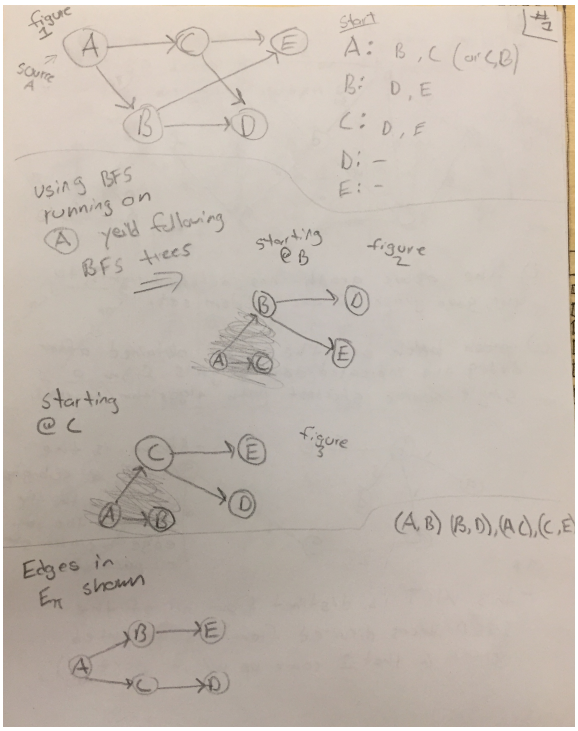
1. (10 pts) Hermione needs your help with her wizardly homework. She's trying to come up with an example of a directed graph $G = (V, E)$, a source vertex $s \in V$ and a set of tree edges $E_\pi \subseteq E$ such that for each vertex $v \in V$, the unique path in the graph (V, E_π) from s to v is a shortest path in G, yet the set of edges E_π cannot be produced by running a breadth-first search on G, no matter how the vertices are ordered in each adjacency list. Include an explanation of why your example satisfies the requirements.

   The explanation is referring to the figure below with included graphs and short explanations next to those graphs. Figure 1 is the example graph. To show an example that E_π cannot be a breadth-first-tree we need to assume the adjacency list of A contains C before B. BFS will add the edges (A,C) and (A,B) to the breadth-first-tree. Since C is added before B, BFS then adds the edges (C,E) and (C,D)...the adjacency list order of C for E and D doesn't matter. Symmetrically looking at the tree, if the adjacency list of A adds B before C, the BFS adds edges (A,B) and (A,C) to the breadth-first-tree. B is added before C so the BFS adds edges (B,D) and (B,E) but the order of D and E do not matter in the adjacency list of B.



2. (15 pts) Prof. Dumbledore needs your help to compute the in- and out-degrees of all vertices in a directed multigraph G. However, he is not sure how to represent the graph so that the calculation is most efficient. For each of the three possible representations, express your answers in asymptotic notation (the only notation Dumbledore understands), in terms of V and E, and justify your claim.

   (a) An *edge list* representation. Assume vertices have arbitrary labels.

   The edge list representation of the in- and out- degrees of a directed multigraph G, time complexity is $\Theta(E)$. This is because the edge list is an array of edges from each each vertex. If we want to find a particular edge in the graph G, we must traverse through the edge list which would be done in $\Theta(E)$ time complexity.

   (b) An *adjacency list* representation. Assume the vector's length is known.

   To compute the out-degrees of all the vertices in a directed multigraph G, you must traverse the adjacency list for each vertex and keep count of the length of its list. The sum of all lengths of all the adjacency lists is E. The time to compute the out-degree of every vertex is $\Theta(V + E)$. For the in-degree of vertices in a directed multigraph G, you need to keep track of the count of in-degrees with an array (A[1...n]) where n is the number of vertices. The values in A will be the in-degrees of every vertex. This will be done in $\Theta(V + E)$ but additional storage/space will be needed. Reference pseudocode is shown below.

$\forall j, 1 \leq j \leq n, A[j] \leftarrow 0;$
For each vertex $i \in V(G)$
    For each edge $(i, j) \in Adj(i)$
        $A[j] \leftarrow A[j] + 1.$

(c) An *adjacency matrix* representation. Assume the size of the matrix is known.

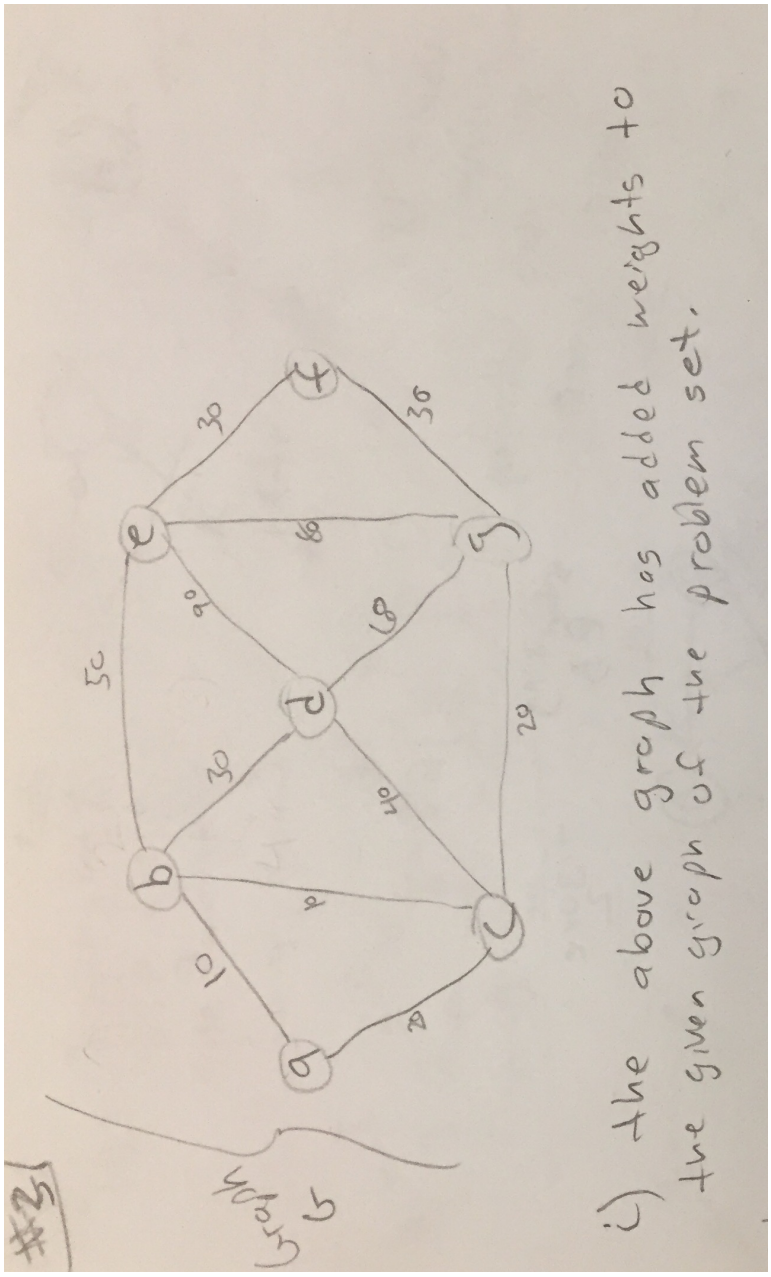
Assuming the size of the matrix A is known, the adjacency matrix is V x V and

furthermore the adjacency matrix would V^2 entries. In a directed multigrpah G, computing the out-degree of a vertex u is equal to searching the row corresponding to u in the matrix A and adding up all the 1's. where computing the out-degree of every vertex is equivalent to searching all the entries of the matrix A and getting a total. Therefore the time complexity if $\Theta(V^2)$. Also computing the in-degrees of a vertex u is equal to searching the column corresponding to u in the matrix A and totalling all the 1's thus the time complexity required is also $\Theta(V^2)$.

3. (25 pts) Professor Snape gives you the following unweighted graph and asks you to construct a weight function w on the edges, using positive integer weights only, such that the following conditions are true regarding minimum spanning trees and single-source shortest path trees:

   - The MST is distinct from any of the seven SSSP trees.
   - The order in which Jarník/Prim's algorithm adds the safe edges is different from the order in which Kruskal's algorithm adds them.

   Justify your solution by (i) giving the edges weights, (ii) showing the corresponding MST and all the SSSP trees, and (iii) giving the order in which edges are added by each of the three algorithms.

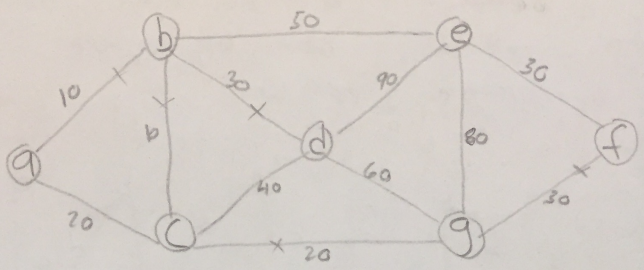

#2
Graph G
i) the above graph has added weights to the given graph of the problem set.
f
e
g
d
b
c
a
30
30
50
20
18
30
20
40
20
10
8

ii) shown below are the MST obtained after adding the indicated edge weights from a single-source shortest path algorithm/traversal.

- shown is, the MST, a sub-graph of G, with the sum of the min. edge weight given in part i) =

- this MST is distinct from all of the SSSP trees divined from the weighted graph G that I came up w/ in part i).

3iii)



Order of adding for Prim's Algorithm: starting @ vertex (a) ending vertex (f)

{(a, b), (b, c), (c, g), ([redacted]), (b, d), (b, e), (g, f)}

Order of adding for Kruskal's Algorithm: (a,b)(b,c)(c,g)(b,d) (g,f)(f,e)

1) sort edges in non-degreasing order of their weight.

2) pick smallest edge, if forms cycle do not include.

3) repeat step 2 until all edges are present.