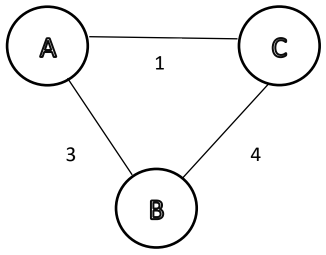
CSCI 4041, Fall 2018, Written Assignment 9

1. In order to create a minimum spanning tree using the new graph G', we can use Prim's algorithm, and only take into account the edges from T (the old minimum spanning tree) and the new edges. This algorithm would run in O(VlgV), because it is not looping over all of the edges E in the graph like normal Prim's, but rather a subset of V edges.

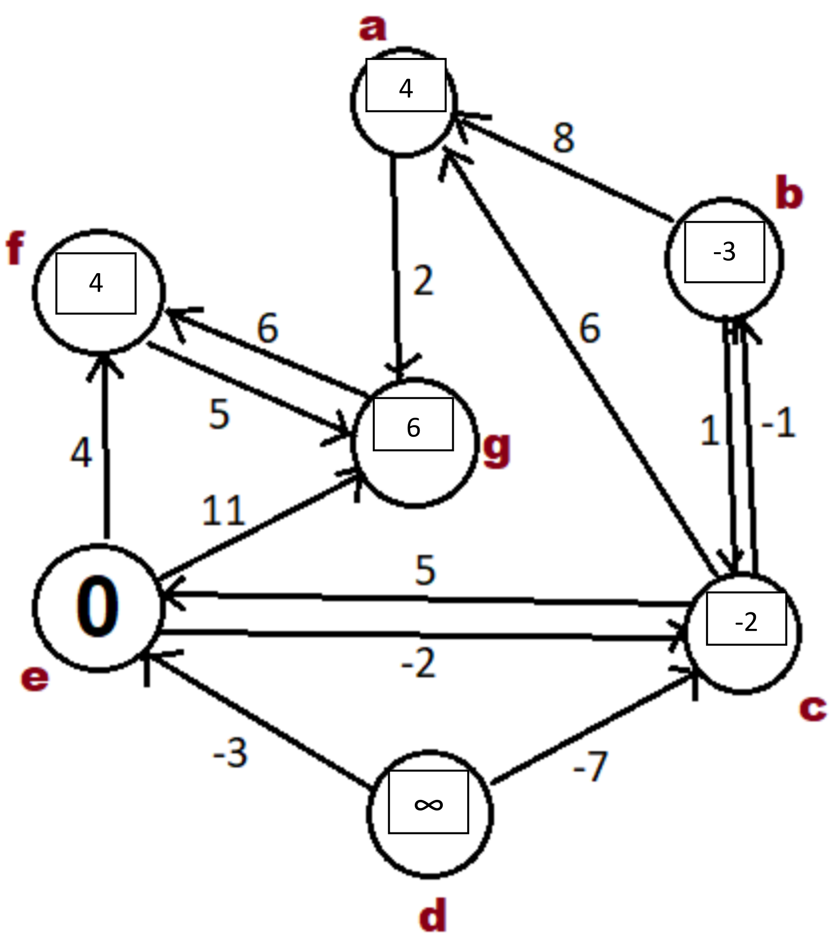
2. <u>MAYBE_MST_A(G, w):</u>
   This graph does return a MST. We can build a max priority queue to sort the edges in decreasing order, and make T that queue. We can then check if T is still a connected graph when removing an edge (u, v) from T by implementing a variation Breadth-First-Search that starts on the node u and finishes when/if it finds the node v. If it does, we simply remove the edge from T by running Heap-Extract-Max. The sort runs in O(ElgE) time, and the connection check runs in O(E(V+E)) time since the search occurs E times. Thus the algorithm runs in O(ElgE + E(V +E)) time.

   <u>MAYBE_MST_B(G, w):</u>
   This graph would return a tree, but not a minimum spanning tree. Because edges are taken arbitrarily, if for example the order that edges were taken in were in happened to be in decreasing order, then the final tree would contain edges with larger weights. This algorithm misses the chance to select smaller weights.
   Ex. [(A, B), (B, C)] could be arbitrarily chosen to create a tree, but it does not create a MST



   <u>MAYBE_MST_C(G, w):</u>
   This graph would return a MST. We can initially make T an empty max priority queue. To insert an edge (u, v) to T, we call Max-Heap-Insert. To check for a cycle, we can use a modified Depth-First-Search to check for any back edges. If one was found, we can call Heap-Extract-Max on T to remove the largest weighted edge. The insertion and removal of an edge runs in O(ElgE) time, while the cycle check runs in O(EV) time. This entire algorithm runs in O(ElgE + EV) time.

3.



Node d is not in a valid shortest path starting at e since there are no paths from e that can reach it. Nodes b and c create a zero weight cycle so they can exist in the shortest path, however it should be noted that there exists an equally short path without the cycle.