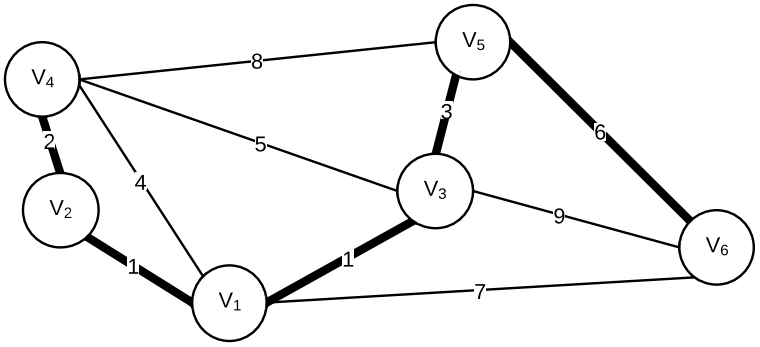
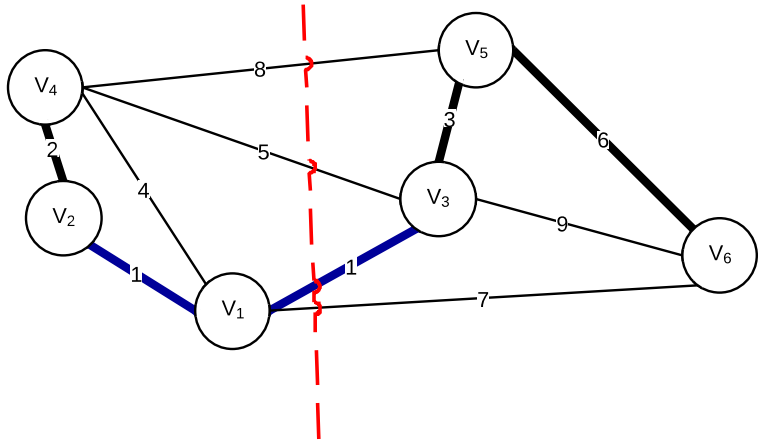
(a)

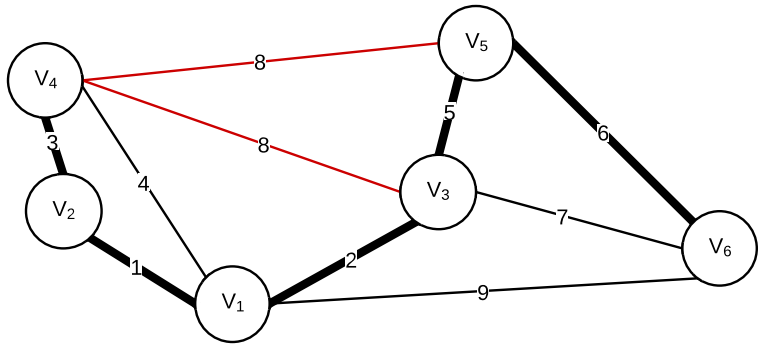


Note that edges v_1, v_2 and v_1, v_3 are of the same weight (x=1). If we use Kruskal's algorithm, we can order all edges by their weights from smallest to largest. An edge is included as long as it does not make a cycle. If there are two minimum edge weights such that w(e) = w(e'), then we know that they will both be selected for our MST because the first two edges cannot make a cycle. Therefore, since all other edges are distinct, you will end up with one minimum spanning tree. That is, whatever order of selecting the two minimum edges, the tree would be the same. QED.

(b) i) If we partition our example graph from 3(a) through edge V_4 - V_5 down to V_1 - V_3 , we see that, although the edge V_1 - V_3 is the minimum weight edge with one endpoint in each subset, it is not unique (it has the same weight as V_1 - V_2), yet the MST for our graph is still unique:

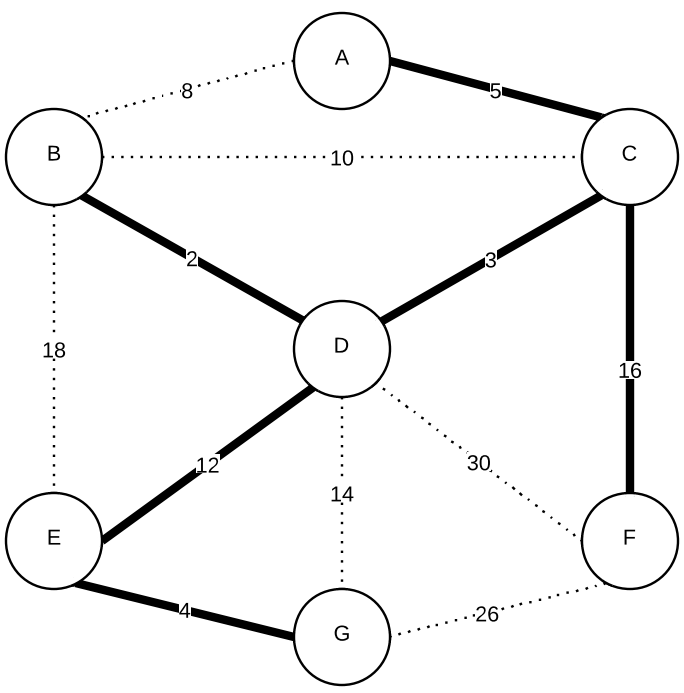


ii) We can modify our example graph as follows:

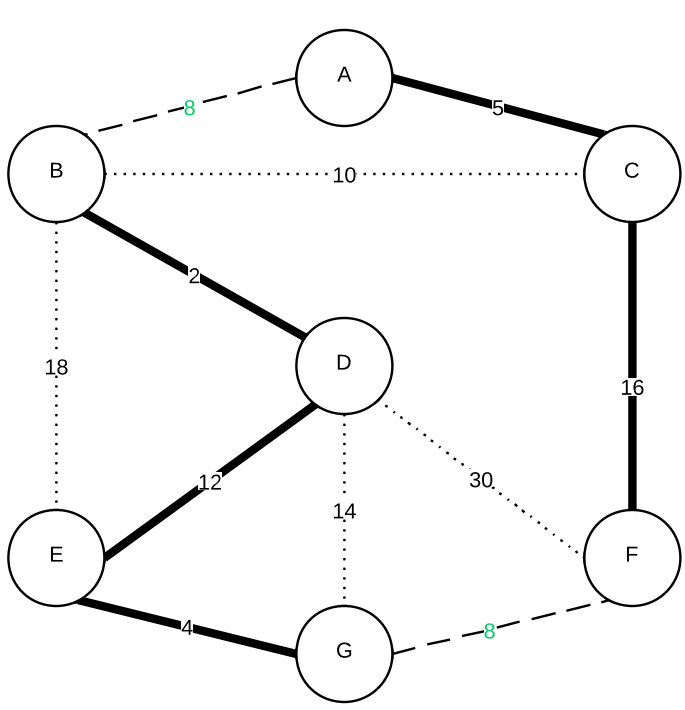


We see that this graph still has only one unique MST, yet the maximum weight edges V_4 - V_5 and V_3 - V_4 are NOT unique. Thus, it is possible that a graph with non-unique minimum weight edges (as seen in (b) i)) and/or non-unique maximum weight edges (as seen in (b) ii)) has a unique MST, and Gollum is incorrect.

(c) For this claim, we can prove our claim using contradiction. Also, we will use the MST example graph from Prof Clauset's Lecture 23-25 notes:

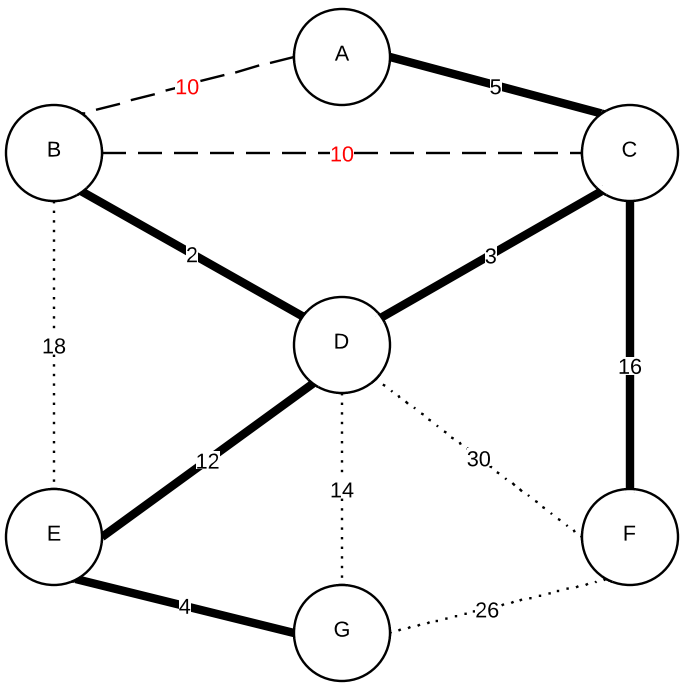


Let's look at part i). Suppose that we remove edge (C, D) and that edges (A, B) and (G, F) have the same weight, call it w = 8:



Since edges (A, B) and (F, G) are BOTH minimum weight edges, either edge could be used as the connector with one endpoint in each of our subtrees. The simple fact that we have a choice and that, in choosing one edge over the other, we have the same total weighted tree w(G) either way, means that graph G DOES NOT have a unique MST, and that in order to have one, graph G must have unique edges for any cut we decide to make.

For part ii) of our claim, suppose we have a graph with NON-unique maximum edges with which to make a cycle:



Since edges (A, B) and (B, C) are maximums, both with the same weight w = 10, the cycle induced by adding either of these edges to G's MST would equate to the same weighted value of the MST. Therefore, the MST would NOT be unique. Any edges spanned by a cut in graph G must be unique for the MST to be unique.

(d) This algorithm is quite simple since our goal is only to find out if a given graph has a unique MST. We must create vertex sets using two vertices at a time. Then we need to compare each edge to see if it is a maximum weight and if so, if it's unique. Also, if it's not a max edge, we need to check to see if it's a minimum weight edge. If so, we need to check if it's unique. If, in either condition, we find two minimum or maximum edges that have the same weight, we know that our graph cannot possibly have a unique MST and we're done. Below is the pseudo-code:

```
boolMST_Unique(G)
{
   bool unique = True;
   sort EdgeSet E by weight
   for ( each vertex v in VertexSet V)
   {
      for ( each edge connected to v_n)
      {
         compare all safe edges with current edge
         if (v_is_minimum && w(v) == all w(v_n) loop)
         {
            unique = False;
            break;
         }
         else if (v_is_maximum && w(v) == all w(v_n) loop)
         {
            unique = False;
            break;
         }
         e_n++;
      }
      v_n++;
   }
   return unique;
}
```

Since this is a minor change from Kruskal's algorithm and navigates through all edges in the exact same way, the algorithm performs effectively in O(E log V) time. This is because we have only added a boolean value and comparison, which takes O(1) time.