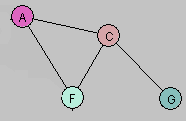
**Problem 1**

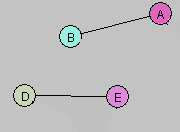
a. Let U = {A, B}. Draw G[U].

C:\Users\huynh\Desktop\Problem 1 a.png

b. Let W = {A, C, G, F}. Draw G[W].



c. Let Y = {A, B, D, E}. Draw G[Y].

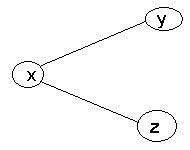


d. Consider the following subgraph H of G:

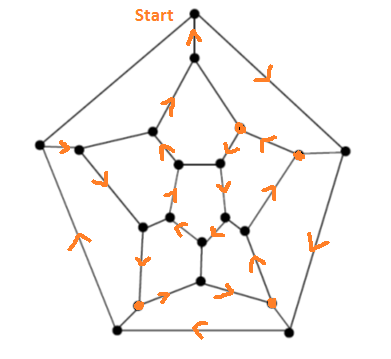
**Problem 2**

* Suppose (u, v) € T, if we remove (u, v) from T, then T will be disconnected. It is resulting in a cut (S, V - S). The edge (u, v) is a light cut crossing the cut (S, V - S)
* Suppose (x, y) € T’ that crosses (S, V- S), so it is a light edge crossing this cut, too.
* Because the light edge crossing (S, V - S) is unique, the edges (u, v) and (x, y) are the same edge. Thus, (u, v) € T’
* Since we choose (u, v) arbitrarily, every edge in T is also in T’

Here is a counter-example



The graph is its own minimum spanning tree, and so the minimum spanning tree is unique. Consider the cut ({x}, {y, z}). Both of the edges (x, y) and (x, z) are light edges crossing the cut, and they are both light edges.

**Problem 3**

**Problem 4**

Finding the maximum spanning tree is the same problem as finding the minimum spanning tree in a graph which had costs negated (relative to the originals)

One method for computing the maximum weight spanning tree of a network G – due to Kruskal – can be summarized as follows.

1. Sort the edges of G into decreasing order by weight. Let T be the set of edges comprising the maximum weight spanning tree. Set T = ∅.

2. Add the first edge to T.

3. Add the next edge to T if and only if it does not form a cycle in T. If there are no remaining edges exit and report G to be disconnected.

4. If T has n−1 edges (where n is the number of vertices in G) stop and output T. Otherwise go to step 3.

Note: A maximum spanning tree is a spanning tree of a weighted graph having maximum weight. It can be computed by negating the weights for each edge and applying Kruskal's algorithm (Pemmaraju and Skiena, 2003, p. 336).