CS577
Homework #6, Problem #2
Sreeram Danda

#2a) Pseudocode

```
graph maximumEdgeDelete( edges[], G ){
      edges[] cycle = Use DFS to obtain first seen cycle;
      while(cycle != null){
            int maxWeight = cycle[0].weight;
            edge maxEdge = cycle[0];
            for(i = 0; i < cycle.length; i++){
                  if(cycle[i].weight > maxWeight){
                        maxWeight = cycle[i].weight;
                        maxEdge = cycle[i];
                  }
            }
            G.remove(maxEdge);
            edges[].remove(maxEdge);
            cycle = Use DFS to obtain next cycle;
      }
      return G;
}
```

Time Complexity: Using DFS to find a cycle can take up to O(V+E). In the worst case scenario, the graph can be fully connected which means there are V(V-1)/2 edges. A minimum spanning tree contains V-1 edges. The algorithm removes one edge at a time so to obtain V - 1 edges DFS is run V(V-1)/2 - (V-1) times. Simplified it is run (V-2)(V-1)/2 times. Therefore, the time complexity for the algorithm in the worst-case is $O(V^2(V+E))$.

#2b) Lemma

Case #1: $e \notin E_T$

Since *e* is the maximum weight edge in the cycle. Removing *e* would result in an acyclic graph that is still a minimum spanning tree. By having the greatest weight in the cycle, removing any other edge e' would result in a spanning tree that has a greater total weight than E_T. The minimum spanning tree of a cyclic graph can only be obtained by removing the edge with the greatest weight, in this case that is *e*.

Case #2: $e \in E_T$

If $e \in E_T$ and *e* is also the maximum weight edge on the cycle *C*, then it must mean that every other edge on the cycle C is strictly less than or equal to e. Therefore, if *e* is swapped with *e'* from the cycle C this should result in a spanning tree that has a total weight less than the one that included *e*. However, this is a contradiction because if the minimum spanning tree does not contain *e* but rather contains *e'*. The spanning tree that contains *e* is not a minimum spanning tree.

In both cases the minimum spanning tree is obtained by removing edge *e* which is the edge with the greatest weight in the cycle *C*. Thus $E_T = A - \{e\}$.

#2c) Correctness

The algorithm begins by using DFS to find any cycles. Upon finding the first cycle the algorithm enters a loop that finds the maximum weighted edge in the cycle. That edge is then removed and the algorithm looks for another cycle. This is correct due to the fact that a minimum spanning tree does not contain any cycles. Additionally, removing the maximum weight edge in the cycle is the only way to obtain the minimum spanning tree as proved above in #2b. If no cycles are found the algorithm returns the graph which should be a minimum spanning tree. If the given graph has size $|E| = n + O(1)$, then there is one more edge than there are vertices. Prim's algorithm would be able to return a minimum spanning tree in $O(E \log(V))$ time whereas there is a chance that this algorithm takes $O(V^3)$.