CS4311 Design and Analysis of Algorithms

Homework 5 (Suggested Solution)

1. (a) **Ans.** Let T be the edges of an MST of G, and suppose on the contrary that T does not contain the edge e_1. Consider adding e_1 to T. Then, we must obtain a simple cycle containing e_1. If we now remove an arbitrary edge e other than e_1 from this cycle, the resulting edges $T \cup \{e_1\} - \{e\}$ must still connect the graph. Moreover, it will become a spanning tree of G, whose total weight is strictly less than the total weight of edges in T (since $w(e) > w(e_1)$). Thus, a contradiction occurs, and the the proof completes.

   (b) **Ans.** (The proof is very similar to (a).) Let T be the edges of an MST of G, and suppose on the contrary that T does not contain the edge e_2. Consider adding e_2 to T. Then, we must obtain a cycle containing e_2. In addition, the cycle must contain *at least three* edges, because the graph is simple. *Thus, there must be an edge e whose weight is more than e_2.* If we now remove this edge e other than e_2, the resulting edges $T \cup \{e_2\} - \{e\}$ must still connect the graph. Moreover, it will become a spanning tree of G, whose total weight is strictly less than the total weight of edges in T (since $w(e) > w(e_2)$). Thus, a contradiction occurs, and the the proof completes.

   (c) **Ans.** If G may contain multiple edges, any MST must still contain e_1, by the same argument as in (a). However, an MST may not contain e_2 in case e_1 and e_2 have exactly the same endpoints.

2. **Ans.** Suppose on the contrary that some MST of $G = (V, E)$ contains $e_{\max}$. Let T be the edges of one such MST. By removing $e_{\max}$ from T, the MST will be partitioned into two connected components, say C and C'.

   On the other hand, from the given condition, we know that removing $e_{\max}$ in G does not disconnect G. This implies that must be some edge e, neither in T nor equal to $e_{\max}$, joining C and C'. Thus, the edges $T - \{e_{\max}\} \cup \{e\}$ will form a spanning tree of G whose total weight is less than the total weight of edges in T (since $w(e_{\max}) > w(e)$). Thus, a contradiction occurs, and the the proof completes.

3. (a) **Ans.** Each vertex v scans its adjacency list to find the cheapest adjacent edge e_v. The total time is thus $O(|E|)$.

   (b) **Ans.** When all the cheapest edge e_v's are found, we form a subgraph that includes all e_v's, and perform a DFS on this subgraph. This takes $O(|V|)$ time. After that, for each connected component C in the subgraph, we relabel the vertices by a new label, say c. This takes $O(|V|)$ time.

   The desired graph G^* can be obtained easily from G if the endpoints of each edge in G is now relabeled according to the new labels. This takes $O(|E|)$ time. The total time is thus $O(|V| + |E|)$.