CS4311 Design and Analysis of Algorithms

Homework 5

Due: 11:10 am, June 12, 2008 (before class)

1. Let $G = (V, E)$ be a *simple* graph[†] which is weighted, undirected, and connected. Suppose G contains a unique edge having the smallest weight. Let e_1 be this edge.

   (a) (30%) Prove that any minimum spanning tree of G must contain the edge e_1.

   (b) (30%) Suppose further that G contains a unique edge e_2 having the second smallest weight. Prove that any minimum spanning tree of G must also contain e_2.

   (c) (10%) What if G may contain multiple edges? That is, for vertices u and v, we may have more than one copies of the edge (u, v), each copy may have a different weight. Are the statements in (a) and (b) still correct?

   Hint: For (a) and (b), prove by contradiction. (Consider on contrary that the MST does not contain e_1 (or e_2). Can we replace some edge in the MST by this edge to reduce the total weight?)

2. Let $G = (V, E)$ be a simple graph which is weighted, undirected, and connected. Suppose G contains a unique edge having the largest weight. Let $e_{\max}$ be this edge.

   (30%) Suppose removing $e_{\max}$ in G does not disconnect G. Prove that any minimum spanning tree of G must not contain the edge $e_{\max}$.

   Hint: Prove by contradiction. (Consider on contrary that the MST contains $e_{\max}$. Can we replace this edge by some other edge to reduce the total weight?)

3. (Bonus: 10%)[§]

   In Lecture Notes 26, page 35, we describe Borůvka's algorithm for computing the minimum spanning tree, where (i) in the first step, we need to compute the cheapest edge adjacent to each vertex, and (ii) in the second step, we need to contract these cheapest edges to produce G^*.

   (a) (5%) Show how the first step can be done in $O(E)$ time. (That means, we must avoid sorting the edges!)

   (b) (5%) For the second step, we can do by first considering the subgraph formed by the cheapest edges, then giving a unique label to each connected components, and finally relabeling all the edges using the labels.[‡] Show that the above can be done in $O(E)$ time.

   Hint: Consider applying DFS to find connected components.

[†] A *simple* graph is a graph that does not contain self-loops, and does not contain multi-edges.

[§] Q3 is a bonus question. Total mark is calculated by: (Q1+Q2) × (100%+ Q3).

[‡] Precisely, an edge (u, v) will be relabeled as (x, y), where x is the label of the connected component containing u, and y is the label of the connected component containing v.