CS 473: Undergraduate Algorithms, Spring 2009
HBS 6.55

1. Suppose you are given a directed graph $G = (V, E)$ with non-negative edge lengths; $\ell(e)$ is the length of $e \in E$. You are interested in the shortest path distance between two given locations/nodes s and t. It has been noticed that the existing shortest path distance between s and t in G is not satisfactory and there is a proposal to add exactly one edge to the graph to improve the situation. The candidate edges from which one has to be chosen is given by $E' = \{e_1, e2, \ldots, e_k\}$ and you can assume that $E \cup E' = \emptyset$. The length of the e_i is $\alpha_i \geq 0$. Your goal is figure out which of these k edges will result in the most reduction in the shortest path distance from s to t. Describe an algorithm for this problem that runs in time $O((m+n)\log n + k)$ where $m = |E|$ and $n = |V|$. Note that one can easily solve this problem in $O(k(m+n)\log n)$ by running Dijkstra's algorithm k times, one for each G_i where G_i is the graph obtained by adding e_i to G.

2. Let G be an undirected graph with non-negative edge weights. Let s and t be two vertices such that the shortest path between s and t in G contains all the vertices in the graph. For each edge e, let $G\backslash e$ be the graph obtained from G by deleting the edge e. Design an $O(E \log V)$ algorithm that finds the shortest path distance between s and t in $G\backslash e$ for all e. [*Note that you need to output E distances, one for each graph $G\backslash e$*]

3. Given a Directed Acyclic Graph (DAG) and two vertices s and t you want to determine if there is an s to t path that includes at least k vertices.