COL351: Analysis and Design of Algorithms
Instructor: Ragesh Jaiswal

1. You are given a directed graph $G = (V, E)$ in which each node $u \in V$ has an associated *price*, denoted by $price(u)$, which is a positive integer. The *cost* of a node u, denoted by $cost(u)$, is defined to be the price of the cheapest node reachable from u (including u itself). Design an algorithm that computes $cost(u)$ for all $u \in V$. Discuss correctness and running time.

2. You are a chemical thief who raids a chemical shop that has n liquid chemicals that are mutually immiscible. You have a large vessel of total volume V that you want to use this to steal the chemicals. You know that the total volume of the i^{th} chemical available in the shop is $v(i)$ and the total volume of this $v(i)$ volume of chemical is $w(i)$. You have to decide how much volume of each of the chemical to steal so as to maximize your profit. Design an algorithm that is given inputs $V, v(1), w(1), v(2), w(2), ..., v(n), w(n)$ and outputs $(s_1, ..., s_n)$, the volume of items to steal. Note that since you can only steal a total of V volume of liquids, $\sum_i s_i \leq V$.

   (a) Consider the greedy algorithm that considers chemicals in decreasing order of value $w(i)$'s and steals as much of an item as possible before considering the next item. Does this algorithm always give optimal solution?

   (b) Consider the greedy algorithm that considers chemicals in increasing order of volume $v(i)$'s and steals as much of an item as possible before considering the next item. Does this algorithm always give optimal solution?

   (c) Consider the greedy algorithm that considers chemicals in decreasing order of value per unit volume. That is, the ratio $\frac{w(i)}{v(i)}$ and steals as much of an item as possible before considering the next item.

   We will show that this greedy algorithm always gives an optimal solution. For simplicity of argument, assume that $\frac{w(1)}{v(1)} \geq \frac{w(2)}{v(2)} \geq ... \geq \frac{w(n)}{v(n)}$. Let $(o_1, ..., o_n)$ be any optimal solution and let $(g_1, ..., g_n)$ be the output of our greedy algorithm. We will show that for all i:

   $$g_1 \cdot \frac{w(1)}{v(1)} + ... + g_i \cdot \frac{w(i)}{v(i)} \geq o_1 \cdot \frac{w(1)}{v(1)} + ... + o_i \cdot \frac{w(i)}{v(i)} \qquad (3.0.1)$$

   This would prove that the greedy algorithm produces an optimal solution. To show this, we first need to prove the following claim.

   <u>Claim 1</u>: For all i, $g_1 + g_2 + ... + g_i \geq o_1 + o_2 + ... + o_i$.

   Let j be the item such that after stealing item j (as per the greedy algorithm), the vessel becomes full. Note that this means $g_{j+1} = 0, g_{j+2} = 0, ..., g_n = 0$. Let us do a case analysis based on the value of g_j. Consider the case when $g_j \leq o_j$. The case when $g_j > o_j$ is symmetric and is left as one of the exercises. The next claim shows inequality (3.0.1) when $i < j$.

Claim 2: For all $i < j$, we have $g_1 \geq o_1, g_2 \geq o_2, ..., g_i \geq o_i$ and so $g_1 \cdot \frac{w(1)}{v(1)} + ... + g_i \cdot \frac{w(i)}{v(i)} \geq o_1 \cdot \frac{w(1)}{v(1)} + ... + o_i \cdot \frac{w(i)}{v(i)}$.

Now, if $i \geq j$, then we can write:

$$\begin{aligned}
(g_1 - o_1) + (g_2 - o_2) + ... + (g_{j-1} - o_{j-1}) &\geq (o_j - g_j) + (o_{j+1} - g_{j+1}) + ... + (o_n - g_n) \\
&\quad \text{(From Claim 1)} \\
\Rightarrow (g_1 - o_1) \cdot \frac{w(j)}{v(j)} + ... + (g_{j-1} - o_{j-1}) \cdot \frac{w(j)}{v(j)} &\geq (o_j - g_j) \cdot \frac{w(j)}{v(j)} + ... + (o_n - g_n) \cdot \frac{w(j)}{v(j)} \\
&\quad \text{(since all terms in the first inequality are } \geq 0) \\
\Rightarrow (g_1 - o_1) \cdot \frac{w(1)}{v(1)} + ... + (g_{j-1} - o_{j-1}) \cdot \frac{w(j-1)}{v(j-1)} &\geq (o_j - g_j) \cdot \frac{w(j)}{v(j)} + ... + (o_n - g_n) \cdot \frac{w(n)}{v(n)} \\
&\quad \text{(since } \frac{w(1)}{v(1)} \geq ... \geq \frac{w(n)}{v(n)}) \\
\Rightarrow g_1 \cdot \frac{w(1)}{v(1)} + ... + g_i \cdot \frac{w(i)}{v(i)} &\geq o_1 \cdot \frac{w(1)}{v(1)} + ... + o_i \cdot \frac{w(i)}{v(i)}
\end{aligned}$$

For part (c) you have to prove Claim 1 and Claim 2 and show that analysis for the case when $g_j > o_j$.

3. Given a graph $G = (V, E)$, a subgraph induced by vertex set $S \subseteq V$ is a graph $G_S = (S, E')$, where $(u, v) \in E'$ iff $((u, v) \in E$ and $u \in S$ and $v \in S)$.

   Prove or disprove: For any strongly connected, weighted, undirected graph $G = (V, E)$ with distinct edge weights and any non-empty proper subset S of V such that G_S and G_{V-S} are strongly connected, $M(G) = M(G_S) + M(G_{V-S}) + w(e)$, where $M(.)$ denotes the weight of the minimum spanning tree, and $w(e)$ is the weight of the minimum weight edge $e = (u, v)$ such that $u \in S$ and $v \in V - S$.

4. We know that when edge weights are not distinct, then there may be multiple MSTs of the graph. How do we argue that the Prim's and Kruskal's algorithm outputs one of the MSTs of the given graph? We will do this using an exchange argument. Needless to say that in both Prim's and Kruskal's algorithm, at every step, instead of considering "the" minimum weight edge we will consider "a" minimum weight edge (that is, break ties arbitrarily). We will first argue about the optimality of the Prim's algorithm.

   Let $e_1, ..., e_{n-1}$ denote the sequence of edges picked by the Prim's algorithm and let T be any MST of G. Let e_i be the first edge in the sequence $e_1, ..., e_{n-1}$ that is not present in T. Let $(S, V - S)$ denote the cut (that includes the starting vertex) just before picking e_i during the execution of the Prim's algorithm. Suppose we add the edge e_i in T. This creates a cycle. Consider the edges in this cycle that go across the cut $(S, V - S)$. All these edges have weights equal to the weight of e_i (otherwise there is a spanning tree of smaller weight as per class discussion). Consider any one such edge e_j and let

$T' = T - \{e_j\} \cup \{e_i\}$. Note that T' is a spanning tree with the same cost as T. Now, if T' contains all edges $e_1, ..., e_{n-1}$, then we are done. Otherwise, we repeat the argument to construct a spanning tree of same cost as T but that includes all edges $e_1, ..., e_{n-1}$.

5. Do a similar argument as above but for Kruskal's algorithm.