# 15-451 Algorithms, Fall 2014

**Homework # 5** **Due: Thursday October 9, 2014**

Your homework should be submitted as a single PDF file using `autolab`. You are allowed to work in groups of 2-3, but you must not share written work, and must write your solutions yourself. **Please cite your sources, and your collaborators; mention explicitly if you did not discuss problems with others. Else we will deduct 20 points.** Solve problems #1–#3.

Problem #H is a "honors" problem. See previous writeups for rules about such problems.

(35 pts) 1. **Cell Towers**

A company is considering $n$ locations at which to build cell towers. It's going to build towers at some subset $S$ of these $n$ locations, to produce the most profit. Profit is the revenue minus cost.

The $i$th cell tower will cost $a_i$ dollars to build, so the total cost of building towers at $S$ is $\sum_{i \in S} a_i$.

The company can get revenue from these towers too. There are $m$ pairs of towers such that if both towers are built, then the company will get some revenue for that. So, as input, we are also given $m$ triples $(x_i, y_i, r_i)$, such that the company gets revenue of $r_i$ dollars if it builds on both locations $x_i$ and $y_i$. (Imagine two customers, one near tower $x_i$ and one near tower $y_i$, who talk a lot. Admitedly it's a stretch, but indulge us.)

E.g., if the costs were $7, 9, 12, 14$ and the triples were $\{(1,2,14), (1,3,11), (2,3,22), (1,4,6)\}$, then building a tower at just location 1 gets profit $-7$, building at $\{1,2\}$ gets profit $14-(7+9) = -2$, building at $\{1,2,3\}$ gets profit $(14+11+22)-(7+9+12) = 19$, etc.

Give an efficient algorithm to find the subset $S$ of these $n$ locations the company should build in order to earn the most profit. (Hint: Use maximum $s$-$t$ flow.)

(30 pts) 2. **Eliminating Negative Edges**

Let $G$ be a graph of $n$ vertices and $m$ edges with no negative edges. Dijkstra's algorithm can be applied $n$ times to solve the all-pairs shortest paths (APSP) problem in $O(nm + n^2 \log n)$ time. If there are negative edges (but no negative cycles), it seems like it requires $\Omega(n^3)$ time (Floyd-Warshall). So the Dijkstra method is a lot faster if the graph is sparse, but it does not work if there are negative edges. It turns out that there's a trick that can used to eliminate the negative edges, thus making Dijkstra's method applicable.

Let $G$ be a graph as described above with no negative cycles, but possibly containing edges of negative length. Construct the following graph $G'$ from $G$. Start with $G$ and add one extra "source" vertex $s$. Now add a zero-length directed edge from $s$ to every other vertex. This is $G'$.

(a) Prove that $G'$ has no negative cycles.

(b) Run Bellman-Ford on $G'$, computing the shortest path from source $s$ to every other vertex. Denote by $\Phi(v)$ the shortest path distance from $s$ to $v$ computed by Bellman-Ford. Denote by $\text{len}(u, v)$ the length of the edge $(u, v)$ in $G'$.

Prove that for every edge $(u, v)$ we have $\Phi(v) - \Phi(u) \le \text{len}(u, v)$.

(c) Let $G''$ be the same as $G$ except that the lengths of the edges have been replaced. Specifically, if in $G$ the length of edge $(u, v)$ is $\text{len}(u, v)$, then in $G''$ its length is defined to be $\text{len}''(u, v) = \text{len}(u, v) + \Phi(u) - \Phi(v)$. By part (b), these new lengths are non-negative.

Explain how to use the results of an APSP computation in the graph $G''$ (with non-negative edge lengths) to solve the APSP problem in $G$ (with negative edge lengths). The runtime of your process to compute APSP on $G$ should be $O(mn + n^2 \log n)$.

(35 pts) 3. **Color Me Read, Color Me Blew**

Let $G = (V, E)$ be an undirected graph with the vertex set $V$ and edge set $E$. Let $|V| = n$. A $(k, n-k)$-*partition* of $G$ is a coloring of the vertices with two colors (red/blue) such that there are $k$ red nodes and $n - k$ blue nodes.

The goal is to find a $(k, n-k)$-partition of $G$ where the number of *split* edges (edges with one endpoint red and the other blue) is minimized. In general there is no known polynomial time algorithm for this problem. However, if the graph is a tree, the problem becomes easier.

Design a polynomial-time algorithm to solve this problem when the input graph $G$ is a *binary tree*. Formally, a binary tree (for our purposes) is a tree rooted at some node $r$, and each node has at most two children. Prove the correctness of your algorithm and analyze its running time. For full credit, your algorithm should take $O(n^3)$ time.

Hint: Dynamic programming.

($20) H. **Covering a Set of Points**

You're given $n$ points in the plane on the $x$-axis, with $x$ coordinates $\{x_1 < x_2 < \ldots < x_n\}$. You have to purchase a set of custom-made umbrellas sufficient to cover all the points. Each umbrella is circular (of any radius you want), and will be placed anywhere you want on the plane. The cost of an umbrella is its area (for the fabric) plus $1 (for the handle). A point is covered if it is on the boundary of (or inside of) an umbrella's circle. The problem is to cover all the points at minimum cost.

For example, suppose the $x$-coordinates of the points are $\{0, 1, 4\}$. One solution is to put an umbrella of radius 2 centered at (2,0), which costs $13.57 (rounded). A second solution is to use three umbrellas of radius 0, which has a total cost of $3.00. A third (and optimal) solution is to use two umbrellas: one of radius .5 centered at $(.5, 0)$ and another of radius 0 centered at (4,0), for a total cost of $2.79 (rounded).

Give a linear-time algorithm that computes the minimum cost of a set of umbrellas necessary to do the job.

The first person in the class to solve this problem gets a reward of $20.