Math552/Comp552: Combinatorial Optimization *Due: Friday Sept 27th*

Assignment 1

1. **Edge Cover.** An *edge cover* in a graph $G = (V, E)$ is a set of edges $F \subseteq E$ such that every vertex is incident to at least one edge in F. Give a polynomial time algorithm to find a minimum cardinality edge cover.

2. **Balanced Matching Problem.** Suppose each edge e in a graph G has a weight w_e. Let $\mathcal{M}$ be the collection of perfect matchings in G.

   (a) Suppose you want to find a perfect matching with the property that the maximum edge weight in M is minimised; that is, solve $\min_{M \in \mathcal{M}} \max_{e \in M} w_e$. How can you solve this by running a maximum cardinality matching algorithm $O(\log n)$ times?

   (b) Now suppose you want to you find a perfect matching with the property that the difference between the maximum and minimum edge weight in M is minimised; that is, solve $\min_{M \in \mathcal{M}} \max_{e,e' \in M} |w_e - w_{e'}|$. How can you solve this by running a maximum cardinality matching algorithm $O(m)$ times?

3. **Matchings.** A edge e is termed *unmatchable* if it is not contained in any perfect matching in G. How quickly can you identify all the unmatchable edges in a graph?

4. **Shortest Paths.** Suppose we are told that the distances of each vertex from $s = v_1$ satisfy $d(v_1) \le d(v_2) \le d(v_3) \le \cdots \le d(v_n)$.

   (a) Use this knowledge to design an $O(m)$ algorithm to find the shortest paths if the graph has only positive cost edges?

   (b) Does you algorithm for a) work if the graph has negative cost edges?

5. **Maximum Profit Cut.** Take a directed graph G with a profit π_a on each arc. The profit $\pi(S)$ of an $s - t$ cut S is the sum of the profits on arcs leaving S minus the sum of the profits of arcs entering S, that is

$$\pi(S) = \sum_{a \in \delta^+(S)} \pi_a - \sum_{a \in \delta^-(S)} \pi_a$$

   Give a polynomial time algorithm to find a maximum profit cut?

6. **Maximum Flows.** Consider the following algorithm for finding maximum flow. Starting with a suitably chosen Δ, repeatedly augment along paths with capacity at least Δ in the residual graph. If no such paths exist then set $\Delta := \frac{1}{2}\Delta$ and repeat. How long does this algorithm take to find a maximum flow?