CSC 373 - Algorithm Design, Analysis, and Complexity **Summer 2016**

Assignment 3: Due Tuesday August 2nd, 10PM

Please follow the instructions provided on the course website to submit your assignment. You may submit the assignments in pairs. Also, if you use **any** sources (textbooks, online notes, friends) please cite them for your own safety.
You can use those data-structures and algorithms discussed in CSC263 (e.g. merge-sort, heaps, etc.) and in the lectures by stating their name. You do not need to provide any explanation or pseudo-code for their implementation. You can also use their running time without proving them: for example, if you are using the merge-sort in your algorithm you can simply state that merge-sort's running time is $\mathcal{O}(n \log n)$.

Every time you are asked to design an efficient algorithm, you should provide both a short high level explanation of how your algorithm works in plain English, and the pseudo-code of your algorithm similar to what we've seen in class. State the running time of your algorithm with a brief argument supporting your claim. You must prove that your algorithm finds an optimal solution!

1 That CS degree will finally pay off!

The cops finally cracked down the mafia in your city at a gangsta' party! Everybody was there. The police has a file on each one of them, so they know who works with who, but it's a government department, and yeah... these files are not very organized.

In order to know if there is a big cell within the group operating together or not, you were hired to determine if there are k mafiosi who know each other.

Politely prove to your police department that this problem is NP-complete :(

2 Holidays at last!

The semester is almost over, and you can't wait to plan your next big trip! Ideally, you want your itinerary to be: Start in Toronto, visit every city in your list once, come back to Toronto. Since you're a jet setter, you don't do connecting flights. Before booking any tickets, you decide to check whether you can actually make an itinerary that satisfies your constraints. Show that nope, no luck, your problem is NP-complete. Formally, given n cities (including Toronto), two cities are adjacent if and only if there is a direct flight between them. You want to see if there is a way to start a trip from Toronto, visit every city once, and come back to Toronto.

3 It's the small changes...

You're going on a long camping trip. You have n items you can potentially take with you. Each item has a weight w_i, and is useful is some way. Suppose you're given a function that gives you the value of each item.

$$\forall i \in [n], w_i \in \mathbb{Z}^{\geq 0}, v_i \in \mathbb{Z}^{\geq 0}$$

Suppose your bag has capacity $c \in \mathbb{Z}^{\geq 0}$. Ideally you want to take all the items, but you can't since your bag is small. Your goal is to select a subset of the item whose total value is maximized, without exceeding the total weight constraint of your bag.

More formally, you want a subset S of $k \leq n$ items that maximizes $\sum_{i=1}^{k} v_i$, and has total weight $\sum_{i=1}^{k} w_i \leq c$.

Consider the following greedy algorithm:

- Sort the items by $\frac{v_i}{w_i}$ in non-increasing order.
- Find the smallest k such that $\sum_{i=1}^{k} w_i > c$.
- Construct a set $S = \{1, 2, \ldots, k-1\}$.
- if $v_k > \sum_{i=1}^{k-1} v_i$, return $\{k\}$, otherwise return S.

Show that the above algorithm gives a 2-approximation.

4 Ain't no question like a graph theory question!

Let $G(V, E, w)$ be an undirected edge weighted graph, where $w(e) \geq 0, \forall e \in E$. You are asked to remove a subset of edges of minimum weight such that the remaining graph has no triangles.

Give a 3-approximation algorithm to solve this problem. Explain your algorithm, prove that it is correct, and that it achieves a 3-approximation ratio.