# CS 522, Data Structures and Algorithms II
# Second Problem Set

April 29, 2020

**Due on 11:00am May 6th, 2020. Collaboration is not allowed**

For full credit, you need to prove the correctness of your algorithms as well as a high level argument about their running time. If you model a question in the homework as any optimization problem we studied in the class you need to show the correctness of your transformation:

- Show that the solution you derive in the derived instance of the optimization problem is also a valid solution to the original problem
- Show that any solution to the original problem is also a solution to your transformed instance.
- Prove the relation of the value of the optimization of the two problems. Note they don't have to be the same but simply be related in a way that we can derive one from the other.

## Problem 1 (30 points)

An ordered stack is a data structure that stores a sequence of items and supports the following operations.

- $Push'(x)$ removes all items smaller than x from the beginning of the sequence and then adds $x$ to the beginning of the sequence.
- *Pop* deletes and returns the first item in the sequence

Prove that if we start with an empty data structure,the amortized cost of each $Push'$ and $Pop$ operation is $O(1)$.

## Problem 2 (35 points)

You're in charge of food rationing, or as the locals call it, *the dining halls*. Every person is assigned to a dining hall. These assignments are based on where the people live - the municipal government has decreed that no person should have to travel more than a quarter mile from their house to get to a dining hall. After making this promise, the government realized that implementing this plan might be a bit tricky; as you may know, dining halls aren't great places when they're overcongested. Your job is to figure out if there's a way to feed everyone without overtaxing any one dining hall.

Formally, you're given a list of all $n$ people and a list of the $m$ dining halls. Each person $i$ is within a quarter mile of some set $S_i$ of dining halls, and must be assigned to some dining hall in $S_i$. Each dining hall $j$ has a capacity $C_j$, which is an upper limit on the number of people it can feed. Your job is to assign people to dining halls in a way that feeds all people and doesn't overtax any dining hall. Give an algorithm that computes such an assignment, or outputs that no such assignment exists. Your algorithm should run in time polynomial in $n$ and $m$. Analyze your algorithm's runtime and prove that it is correct.

You may use any algorithm we studied during lecture for max flow as a black box. Mention which you are using in your runtime analysis.

## Problem 3 (35 points)

One intuitive story for the max flow problem is that the input graph $G = (V, E)$ is a rail network, the capacities are shipping capacities along rail lines, and the algorithm designer interested in maximizing the volume of shipments from some source city $s$ to the sink city $t$.

You're interested in ruining this particular algorithm designer's plans. Specifically, we're given a rail network $G = (V, E)$ with *unit capacities*, a source $s$, and a sink $t$. We have enough explosives to destroy $d$ rails. Our goal is to reduce the volume of shipping between $s$ and $t$ (computed via max flow) by as much as possible. The question is: which ones should we destroy?

Give an efficient algorithm which takes $G$, $s$, $t$, and $d$ as inputs, and returns a set of rails $S \subseteq E$ which, when removed, reduces the total shipments from $s$ to $t$ by the maximum amount. Prove correctness and runtime for your algorithm.