# Assignment 1

## Algorithms Complexity (CIS 522-01)

*Javier Arechalde*

February 2, 2018

# Problem 1. Matching Residents to Hospitals

## Problem description

We have $m$ hospitals, each one of them has certain number of available positions to hire residents.

In a given year $n$ medical students were graduating, each one of them interested in joining one of the hospitals.

Each one of the hospitals had a ranking of the students in order of preference, and each student had a ranking of hospitals in order of preference.

In this problem we will assume that $n > m$.

We are looking to assign each student to at most one hospital, in a way that all available positions in hospitals are filled. As $n > m$ some students may have none hospitals assigned at the end of the algorithm run.

For this problem there will be two types of unstability:

- First type

  There are students $s$ and $s'$, and hospital $h$.

  - $s$ is assigned to $h$.
  - $s'$ is assigned to no hospital.
  - $h$ prefers $s'$ to $s$.

- Second type

  There are students $s$ and $s'$ and hospitals $h$ and $h'$.

  - $s$ is assigned to $h$
  - $s'$ is assigned to $h'$
  - $h$ prefers $s'$ to $s$
  - $s'$ prefers $h$ to $h'$

## Overview of the problem

So the difference between this problem and stable matching problem is that hospitals want more than one student usually, and there is more students that positions available in the hospitals.

In this case I think it makes more sense to start from the students side, checking if there is any positions available in the first hospital they want to get a position in, and if there is one get it, otherwise, check if the hospital prefers this student to the ones that they have assigned already. If the hospital prefer this student to one of the students that they have already assigned, kick the least preferred student, and get this new student in. This algorithm should run while the number of students "free" is different from the number of positions available, and all the students havent been checked yet, because even though, all the positions are filled, we may be running in the first type of unstability, as $h$ prefers $s'$ to $s$ and $s'$.

Now we will proceed to describe our algorithm structure.

## Algorithm implementation

- $H, S$ are the set of Hospitals and Students respectively.
- $m, n$ are the total number of hospitals and students respectively.
- $h_{pref}$ is the hospital student preference list for each $s \in S$.
- $s_{pref}$ is the student hospital preference list for each $h \in H$.
- $n_{avail}(h_i)$ is the total number of possitions available in each hospital.
- $n_{assign}(h_i)$ is the total number of students assigned to a hospital.
- $N_{pos}$ is the total number of positions available $\sum n_{avail}(h_i)$.
- $S_{assign}$ is the total number of students that have a hospital assigned.
- $S_{check}$ is the total number of students that we tried to assign a hospital to.

**Algorithm 1** My implementation

**while** $S_{assign} < N_{pos}$ & $S_{check} < n$ & student $s$ hasnt proposed to all $h_i \in s_{pref}$ **do**
    Try to assign to that student a hospital in his preference list
    **if** $n_{assign}(h_i) < n_{avail}(h_i)$ **then**
        Student $s$ is assigned to hospital $h_i$
    **else if** $n_{assign}(h_i) \geq n_{avail}(h_i)$ **then**
        Check $h_{pref}(h_i)$
        **if** student $s$ is higher in the list than any student $s_i \in S$ **then**
            Least prefered student $s'$ in $h$ is now free, and will try to be reassigned
            Student $s$ is now assigned to hospital $h_i$
        **else if** Student $s$ is not higher in $h_{pref}$ than any $s_i \in h_{pref}(h_i)$ **then**
            Student $s$ remains free
        **end if**
    **end if**
**end while**
Return the set of hospitals and assigned students to each hospital.

## Proof that the assignment is stable:

In order to prove that our assignment is stable, we need to prove that none of the two types of instability can happen.

**First type of instability:**

- $s$ is assigned to $h$.

- $s'$ is assigned to no hospital.

- $h$ prefers $s'$ to $s$.

In the execution of our algorithm that resulted in the set of students and hospitals $S$, $s'$ last proposal was to $h$ by definition. In our algorithm implementation, if a student $s$ proposes to a hospital, and the hospital, even though it has all its positions filled, prefers this new student that the ones that are already assigned to it, the hospital will dump the least prefered student, and assign that position to the new student. This contradicts the fact that $h$ prefers $s'$ to $s$, because in that case, $s$ would have been dumped in the execution of the algorithm, and its position would have been assigned to the new student $s'$.

**Second type of instability:**

- $s$ is assigned to $h$

- $s'$ is assigned to $h'$

- $h$ prefers $s'$ to $s$

- $s'$ prefers $h$ to $h'$

In the execution of our algorithm that produced the set of students and hospitals $S$, $s$'s last proposal was to $h$ by definition. The question is now: Did $s'$ propose to $h$ at some earlier point in the execution of the algorithm? If this happened, and $s'$ was higher in the preference list than $s$, $s'$ would have been reassigned to $h$, so if this didn't happen that means that $h$ prefers $s$ rather than $s'$, which contradicts the fact than $h$ prefers $s'$ to $s$. Then it is proven, that this type of instability can't happen in our algorithm implementation.

## Problem 2. Implementation of Propose-and-Reject Algorithms

See the source code attached to this assignment.