# COMP 251 2016, Assignment 1
# Due Wednesday September 28th 2016

[20%]

1. **Read Chapter 1. Solve Exercise 4.**

**4.** Gale and Shapley published their paper on the stable marriage problem in 1962; but a version of their algorithm had already been in use for ten years by the National Resident Matching Program, for the problem of assigning medical residents to hospitals.

Basically, the situation was the following. There were $m$ hospitals, each with a certain number of available positions for hiring residents. There were $n$ medical students graduating in a given year, each interested in joining one of the hospitals. Each hospital had a ranking of the students in order of preference, and each student had a ranking of the hospitals in order of preference. We will assume that there were more students graduating than there were slots available in the $m$ hospitals.

The interest, naturally, was in finding a way of assigning each student to at most one hospital, in such a way that all available positions in all hospitals were filled. (Since we are assuming a surplus of students, there would be some students who do not get assigned to any hospital.)

We say that an assignment of students to hospitals is *stable* if neither of the following situations arises.

- First type of instability: There are students $s$ and $s'$, and a hospital $h$, so that
  - $s$ is assigned to $h$, and
  - $s'$ is assigned to no hospital, and
  - $h$ prefers $s'$ to $s$.
- Second type of instability: There are students $s$ and $s'$, and hospitals $h$ and $h'$, so that
  - $s$ is assigned to $h$, and
  - $s'$ is assigned to $h'$, and
  - $h$ prefers $s'$ to $s$, and
  - $s'$ prefers $h$ to $h'$.

So we basically have the stable marriage problem from the section, except that (i) hospitals generally want more than one resident, and (ii) there is a surplus of medical students.

Show that there is always a stable assignment of students to hospitals, and give an efficient algorithm to find one.

```
GS Men-optimal

Initialize each person to be free.
while (some man is free and hasn't proposed to every woman)
{
    Choose such a man m
    w = 1st woman on m's list to whom m has not yet proposed
    if (w is free)
        assign m and w to be engaged
    else if (w prefers m to her fiancé m')
        assign m and w to be engaged, and m' to be free
    else
        w rejects m
```

```
GS Women-optimal

Initialize each person to be free.
while (some woman is free and hasn't proposed to every man)
{
    Choose such a woman w
    m = 1st man on w's list to whom w has not yet proposed
    if (m is free)
        assign m and w to be engaged
    else if (m prefers w to his fiancé w')
        assign m and w to be engaged, and w' to be free
    else
        m rejects w
}
```

[15%] 2. Either prove the following statement or exhibit a counter-example.

*The solutions produced by both algorithms are equal*
*if and only if*
*this is the <u>only</u> solution to the input instance.*

[15%] 3. **<u>Read Chapter 2.</u>** Prove formally that if

$$\lim_{n\to\infty} f(n)/g(n) = 0,$$

$$\text{then } f \in O(g),\ f \notin \Omega(g),$$
$$O(f) \subsetneq O(g),\ \Omega(g) \subsetneq \Omega(f).$$

[10%] 4.**Solve the following Exercises**

**Problem 3.21.** To illustrate how the asymptotic notation can be used to rank the efficiency of algorithms, use the relations "$\subset$" and "$=$" to put the orders of the following functions into a sequence, where $\varepsilon$ is an arbitrary real constant, $0 < \varepsilon < 1$.

$$n \log n \qquad n^8 \qquad n^{1+\varepsilon} \qquad (1+\varepsilon)^n \qquad n^2/\log n \qquad (n^2-n+1)^4$$

Do *not* use the symbol "$\subseteq$". Prove your answers.

**Problem 3.22.** Repeat Problem 3.21 but this time with functions

$$n! \qquad (n+1)! \qquad 2^n \qquad 2^{n+1} \qquad 2^{2n} \qquad n^n \qquad n^{\sqrt{n}} \qquad n^{\log n}.$$

[10%] Explain what is wrong with the following reasoning :

$$\sum_{i=1}^{n} i = 1+2+\cdots+n \in O(1+2+\cdots+n) = O(\max(1,2,\ldots,n)) = O(n).$$

[15%] Which of the following statements are true?
(Prove formally each of your answers.)

**i.** $n^2 \in O(n^3)$

**ii.** $n^3 \in O(n^2)$

**iii.** $2^{n+1} \in O(2^n)$

**iv.** $(n+1)! \in O(n!)$

**v.** for any function $f: \mathbb{N} \to \mathbb{R}^+$, $f(n) \in O(n) \Rightarrow [f(n)]^2 \in O(n^2)$

**vi.** for any function $f: \mathbb{N} \to \mathbb{R}^+$, $f(n) \in O(n) \Rightarrow 2^{f(n)} \in O(2^n)$

**[15%]**

6. Consider the following basic problem. You're given an array $A$ consisting of $n$ integers $A[1], A[2], \ldots, A[n]$. You'd like to output a two-dimensional $n$-by-$n$ array $B$ in which $B[i, j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$—that is, the sum $A[i] + A[i+1] + \cdots + A[j]$. (The value of array entry $B[i, j]$ is left unspecified whenever $i \geq j$, so it doesn't matter what is output for these values.)

   Here's a simple algorithm to solve this problem.

```
For $i = 1, 2, \ldots, n$
  For $j = i+1, i+2, \ldots, n$
    Add up array entries $A[i]$ through $A[j]$
    Store the result in $B[i, j]$
  Endfor
Endfor
```

   **(a)** For some function $f$ that you should choose, give a bound of the form $O(f(n))$ on the running time of this algorithm on an input of size $n$ (i.e., a bound on the number of operations performed by the algorithm).

   **(b)** For this same function $f$, show that the running time of the algorithm on an input of size $n$ is also $\Omega(f(n))$. (This shows an asymptotically tight bound of $\Theta(f(n))$ on the running time.)

   **(c)** Although the algorithm you analyzed in parts (a) and (b) is the most natural way to solve the problem—after all, it just iterates through the relevant entries of the array $B$, filling in a value for each—it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve this problem, with an asymptotically better running time. In other words, you should design an algorithm with running time $O(g(n))$, where $\lim_{n\to\infty} g(n)/f(n) = 0$.