CS41 lab 2

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design, and to gain experience collaborating with others on algorithm design and analysis. However, it will be common to have some overlap between lab exercises and homeworks.
The goal of this lab is to gain more experience with asymptotic analysis and algorithm design. There are more problems than you have time to complete during the lab. Consider it a successful lab session if you can mostly complete two problems.

1. **Better Runtime for Gale-Shapley Algorithm.** In class, we saw that the Gale-Shapley Algorithm runs in $O(n^3)$ time. However, with a bit more care on selecting data structures, this runtime can be improved to $O(n^2)$.

   ```
   Gale-Shapley()
   1  Initialize all hospitals, doctors := free.
   2  while there is a free doctor $d$:
   3      $h$ := highest ranked hospital in $d$'s preference list that $d$ hasn't yet asked for a job
   4      $d$ asks $h$ for a job
   5      if $h$ is free
   6          $(h, d)$ become matched
   7      else if $h$ prefers $d$ to their current match $d'$:
   8              $(h, d)$ become matched
   9              $d'$ becomes free again.
   10 return the set $S$ of matches
   ```

   (a) First, focus on each iteration of the While loop. We've given you two below.

   - Identify a free doctor d
   - Identify the highest ranked hospital h in d's preference list that d hasn't yet asked for a job

   What other tasks must be performed?

   (b) For each task, identify a data structure that can be used to manage this data/task, and analyze how much time it will take to initialize the data and how much time the task will take during the while loop.

   As an example, we've given you a solution for the first task below.

   - For the first task (identify a free doctor d), we start with all doctors as free. We want to delete a doctor from this list when it gets matched, and add a doctor when its match is broken. However, of all the free doctors that haven't yet asked every hospital, we're free to choose which is next.
     Solution: Use a LinkedList FreeDoctors. Initialize this list to contain all doctors. Call First() to get the next free doctor. Add doctors to the end of the list.
     Initialization time: $O(n)$. Loop Iteration time: $O(1)$.

(c) Now, pull everything together to analyze the overall runtime. How much time is spent before the while loop? How much time is spent during each iteration of the while loop?

We've seen in class that there are $O(n^2)$ loop iterations. In order to get an overall $O(n^2)$ runtime, your preprocessing (i.e., work before the while loop) step must take $O(n^2)$ time and your work inside each loop iteration must run in $O(1)$ time.

2. **Asymptotic analysis.** Arrange the following functions in ascending order of growth rate. That is, if g follows f in your list, then it should be the case that $f = O(g)$.

- $f_1(n) = \frac{\sqrt{n}}{6}$
- $f_2(n) = 12n\log(n)$
- $f_3(n) = 5\log(n)^4$
- $f_4(n) = \pi \cdot 2^n$
- $f_5(n) = 7n^3$
- $f_6(n) = 16n^2 + 22n$

No proofs are necessary.

3. **Complete analysis.** Let $f(n) = 12n^{4/5}$ and $g(n) = n^{3/5}(\log n)^6$. Prove that $g(n) = O(f(n))$. You may use techniques and facts from class and the textbook; your proof should be formal and complete.

4. **Asymptotic analysis properties, part 1.** Assume you have functions f, g, and h. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.

(a) If f is $O(h)$ and g is $O(h)$, then $f + g$ is $O(h)$.
(b) If f is $O(h)$ and g is $O(h)$, then $f \cdot g$ is $O(h)$.
(c) If f is $O(g)$, then g is $\Omega(f)$.

5. **Asymptotic analysis properties, part 2.** Assume you have functions f and g such that $f(n)$ is $O(g(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.

(a) $\log_2(f(n))$ is $O(\log_2(g(n)))$.
(b) $2^{f(n)}$ is $O(2^{g(n)})$.
(c) $(f(n))^2$ is $O((g(n))^2)$.
(d) If $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.