CS41 lab 3

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. However, it will be common to have some overlap between lab exercises and homework sets.

The goal of this lab session is to gain more experience with asymptotic analysis, and to start thinking about graphs. Do not expect to complete all parts of all problems by the end of the lab. Consider it a successful lab session if you can mostly complete two problems.

For these problems, your example functions should have domain and range the positive integers $\mathbb{N}$. Keep it simple.

Gale-Shapley Running Time

1. In class, I claimed (without proving it) that the Gale-Shapley Algorithm runs in $O(n^2)$ time. For this problem, we will look at the Gale-Shapley Algorithm and see how we can implement parts of it to achieve the desired running time. This will be a good opportunity to review some of the data structures from CS 35.

   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 residency position (i.e., d has not yet proposed to h)
   4       d asks h for a job
   5       if h is free
   6            (h, d) become (tentatively) matched
   7       else if h prefers d to their current match d':
   8                 h and d' become unmatched (d' becomes free)
   9                 (h, d) become matched
   10 return the set S of matches
   ```

   (a) First, focus on the tasks that need to be performed in *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 will maintain a list of free doctors. We start with all doctors as free. We want to delete a doctor from this list when it gets matched, and add a doctor to the list when it gets unmatched. 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: $O(1)$.

Now, do something similar for each of the other tasks that need to be performed.

(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 determine the final running time, you will need to multiply the total time taken in each iteration by n^2, and add the total preprocessing time (time taken before the first iteration of the while loop). So, 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.

(**Hint:** It may be helpful to start by finding *any* data structure to use to implement the operations, even if the total time taken is more than $O(n^2)$, and then think of ways to refine it. It should not be too hard to implement all steps so that the total running time is $O(n^3)$. You may also see if you can achieve a running time of $O(n^2 \log n)$.)

Graphs

This week, we will start discussing Graph algorithms. Here, we will see some review and practice with graphs and proofs involving graphs.

1. **Proofs with Graphs.** An undirected graph $G = (V, E)$ is a set of vertices V along with edges E, where each edge $e \in E$ is a pair of vertices $\{u, v\}$ with $u, v \in V$. A *triangle* in a graph G is a set of exactly three vertices such that there is an edge in G for every pair of vertices in the triangle. An *independent set* in a graph G is a set of vertices for which *no* pair of vertices in the set is an edge in G.

   Prove or disprove: Every graph on exactly five vertices contains either a triangle or an independent set of size three.

   Extra (Challenge): What about graphs with five vertices?

2. **Connectivity.** A graph $G = (V, E)$ is *connected* if there is a path between any two vertices. Design an algorithm to detect whether an undirected graph is *connected*. Your algorithm should return YES if the graph is connected; otherwise, return NO. Your algorithm should run in $O(m + n)$ time on a graph with n vertices and m edges. (Hint: BFS and DFS both run in $O(m + n)$ time, if you choose a reasonable data structure to store the graph.)

More with Asymptotics

1. **Asymptotics of Polynomials.** Let $d \in \mathbb{N}$ be a positive integer constant, and $f(n) = \sum_{i=0}^{k} a_i n^i = a_d n^d + a_{d-1} n^{d-1} + \cdots + a_1 n + a_0$. Prove that f is $\Theta(n^d)$.

2. **Properties.** 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 $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.

   (b) If f is not $O(g)$, then g is $O(f)$.

   (c) If f is $O(g)$ and g is $O(f)$, then f is $\Theta(g)$.

   (d) If f is $\Omega(g)$ and g is $\Omega(h)$, then f is $\Omega(h)$

3. **General asymptotic analysis.** For these problems, your example functions should have domain and range the positive integers $\mathbb{N}$.

   Let k be a fixed constant and suppose that $f_1, \ldots, f_k$ and h are functions such that $f_i = O(h)$ for all i.

   (a) Let $g_1(n) := f_1(n) + \cdots + f_k(n)$. Is $g_1 = O(h)$? Prove or give a counterexample. If you give a counterexample, give an alternate statement that best captures the relationship between the two functions.

   (b) Let $g_2(n) := f_1(n) \cdot \cdots \cdot f_k(n)$. Is $g_2 = O(h)$? Prove or give a counterexample. If you give a counterexample, give an alternate statement that best captures the relationship between the two functions.