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.

1. **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 $O(g)$, then $\log_2(f(n))$ is $O(\log_2(g(n)))$.

   (d) If f is $O(g)$, then $2^{f(n)}$ is $O(2^{g(n)})$.

   (e) If f is $O(g)$, then $(f(n))^2$ is $O((g(n))^2)$.

2. **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 $g_1(n) := f_1(n) + ... + f_k(n)$. 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 $g_2(n) := f_1(n) \cdot \ldots \cdot f_k(n)$. 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.

3. **The Wedding Planner Problem.** Imagine you are a wedding planner. Among other tasks, you must help your customers with their guest lists. Couples come to you with lists of people they might invite to their wedding. They also come with demands – Alice and Bob need to be invited, but if Bob is invited, do not invite Carol (they have history). When Carol, Dave, and Eve get together, they only talk about Justin Bieber (their favorite musician), so don't invite all three. However, any two of them is ok. Call these conditions *constraints*.

   People at weddings are **demanding**. Everything needs to be exactly perfect, and they will blame you if even one constraint is not satisfied. You're not even sure if this is possible, and it when it's not, it would be nice to have a convincing argument for the wedding couple.

   Design an algorithm that takes a list of people, and a list of constraints, and outputs YES if there is an invite list that satisfies every constraint. Otherwise, output NO.

4. **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.)

5. **Strongly Connected Components.** Let $G = (V, E)$ be a directed graph. Vertices u and v are *strongly connected* if there are $u \rightsquigarrow v$ and $v \rightsquigarrow u$ paths in G. A *strongly connected component* is a set of vertices $C \subseteq V$ such that u, v are strongly connected for all $u, v \in C$ (and no other vertices are strongly connected to a vertex $u \in C$.)

   Design and analyze an algorithm to identify all strongly connected components in G. What is the runtime of your algorithm?