CS41 Lab 4

The lab and homework this week center on graph algorithms for undirected graphs. The following definitions might be helpful/relevant.

- A *path* P on a graph $G = (V, E)$ is a sequence of vertices $P = (v_1, v_2, \ldots, v_k)$ such that $(v_i, v_{i+1}) \in E$ for all $1 \leq i < k$.
- A path is *simple* if all vertices are distinct.
- The *length* of a path $P = (v_1, \ldots, v_k)$ equals $k - 1$. (Think of the path length as the number of edges needed to get from v_1 to v_k on this path).
- A *cycle* is a sequence of vertices $(v_1, \ldots, v_k)$ such that $v_1, \ldots, v_{k-1}$ are all distinct and $v_k = v_1$. A cycle is odd (even) if it contains an odd (even) number of edges.

1. **Algorithm Analysis.**

   (a) Let $f(n) := 6n^{2/3}$ and $g(n) := \frac{n}{2 \log n}$. Prove that $f(n) = O(g(n))$. You may use whatever techniques or facts from class you want, but your proof must be formal and complete.

   (b) Let $h_1(n) = n^{10}$ and $h_2(n) = 2^{(\log(n))^2}$. Choose the most accurate comparison:
   - $h_1(n) = O(h_2(n))$.
   - $h_1(n) = \Theta(h_2(n))$.
   - $h_1(n) = \Omega(h_2(n))$.

   Justify your response. (A formal proof is not required, but you should explain enough to convince the graders that your answer is correct).

2. **Graph Representation.** There are two main ways of representing a graph in a data structure.

   - **Adjacency Lists.** This data structure contains a List of vertices, and for each vertex, a List of its neighbors.
   - **Adjacency Matrix.** This data structure maintains a 2D-array of boolean values. $A[i, j]$ equals TRUE if (i, j) is an edge, and false otherwise.

   In this problem, you'll explore relative advantages of each.

   (a) List three operations you might want to perform on a graph.

   (b) For each operation, analyze the runtime assuming the graph is stored as (i) an adjacency list and (ii) an adjacency matrix.

   (c) How much space does each data structure take?

   (d) How efficient is BFS for each data structure? How efficient is DFS?

3. **Cycle Detection.** Design and analyze an efficient algorithm for finding a cycle in a graph. Your algorithm should take as input a graph $G = (V, E)$ and report a cycle (or output NO if no cycle exists). If there are multiple cycles in the graph, your algorithm should just output one.

4. **Testing Tripartiteness.** Call a graph $G = (V, E)$ *tripartite* if V can be partitioned into disjoint sets A, B, C such that for any edge $(u, v) \in E$, the vertices u, v lie in different sets. Design and analyze an algorithm to test a graph for tripartiteness.