CS41 lab 5

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 algorithm design, particularly using directed graphs.

1. **All-Pairs Shortest Paths.** Design and analyze a polynomial-time algorithm that takes a directed graph $G = (V, E)$ and for all $u, v \in V$ computes the length of the shortest $u \rightsquigarrow v$ path or determines that no such path exists. (The length of a path in an undirected graph is equal to the number of edges in the path.)

2. **Cycle Detection (K & T 3.2)** Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should **output a cycle**. Otherwise, your algorithm should output NO. Your runtime should be $O(m + n)$ for a graph with m edges and n vertices.

   Hint: Don't forget edge cases. Don't forget to return the cycle if one is detected.

3. In CS35, you likely saw the *shortest path on weighted graphs* problem. In this problem, each edge e has an edge length ℓ_e, and the length of a path $s \rightsquigarrow t$ is the sum of the edge lengths along the path. Your goal is to find, for a specific start vertex, the length of the shortest $s \rightsquigarrow v$ path for all vertices v.

   Problem: Single-Source Shortest Paths

   Inputs:

   - a graph $G = (V, E)$
   - for each edge e a positive *edge length*, ℓ_e
   - a start vertex $s \in V$

   Output: an array of distances d, where $d[v]$ is the length of the shortest $s \rightsquigarrow v$ path.

   Below is pseudocode for Dijkstra's Algorithm, which finds the shortest path in a graph G between a start vertex s and any other vertex. Dijkstra's algorithm is greedy; the greedy strategy is to pick the next not-yet-reached vertex that has shortest path from s.

   DIJKSTRA(G, s, ℓ)
   1 $S = \{s\}$.
   2 $d[s] = 0$.
   3 **while** $S \neq V$
   4 pick $v \in V \setminus S$ to minimize $\min_{e=(u,v):u \in S} d[u] + \ell_e$.
   5 add v to S.
   6 $d[v] = d[u] + \ell_e$
   7 Return $d[\ldots]$.

   (a) Show that Dijkstra's Algorithm solves the shortest path problem. (Hint: use the "stays ahead" method.)

(b) What is the asymptotic running time of Dijkstra's algorithm?
If you were to implement it (in, say, C++) what data structures would you need? Would you need any additional data structures beyond structures you've seen from CS35? If so, try to design an implementation for them.

Note: the pseudocode given above is *high-level* pseudocode. One reason why this is high-level is because it doesn't specify how to compute the edge $e = (u, v)$ such that $u \in S$ that minimized $d[u] + \ell_e$. You'll need to understand how to compute this edge efficiently in order to analyze the overall runtime.