CS41 Lab 4

September 28, 2017

1. **Shortest Paths in Weighted Graphs.**

   - a directed graph $G = (V, E)$.
   - a start vertex $s \in V$.
   - each edge $e \in E$ has an *edge length* $\ell_e > 0$.

   Your goal is to output, for each $v \in V$, the length of the shortest $s \rightsquigarrow v$ path.

   If you took CS35, you saw a solution to this problem called Dijkstra's Algorithm. Dijkstra's Algorithm is a greedy algorithm which works by iteratively and greedily building a set of "visited nodes" S. The nodes are selected in increasing path length.

   DIJKSTRA(G, s, $\{\ell_e\}_{e \in E}$)
   $S = \{s\}$.
   $d[s] = 0$.
   while $S \neq V$
   pick $v \in V \setminus S$ to minimize $\min_{e=(u,v):u \in S} d[u] + \ell_e$.
   add v to S.
   $d[v] = d[u] + \ell_e$
   Return $d[\ldots]$.

   (a) Show that Dijkstra's algorithm correctly returns the minimum length of paths from s to other nodes.

   (b) What is the running time of Dijkstra's algorithm? For this answer, you may assume any data structure that you've seen from CS35 or CS41, but the running time should be as low as possible.

2. **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.

3. **Enemies on the Move.** Alice and Bob are very active students at University of Pennsylvania. They used to be best friends but now despise each other. Alice and Bob can't stand to be in the same room, or even nearby. However, they each take many classes and are active in several clubs. Is it even possible to avoid each other?

   This can be modeled as a graph problem. The input consists of:

   - a directed graph $G = (V, E)$,
   - an integer $k \leq n$,
   - start vertices s_A and $s_B \in V$, and

- end vertices t_A and $t_B \in V$.

In this problem, Alice starts at s_A and wants to travel to t_A, while Bob starts at s_B and wants to travel to t_B. At each time step, either Alice or Bob moves along a single edge. (You can assume they move separately.) At all times, Alice and Bob must be at least k edges apart.

Design and analyze a polynomial-time algorithm that determines if Alice and Bob can get where they want to go while maintaining distance.

(**Hint:** It might be helpful to use your solution to problem (2) as a subroutine.)

4. **Computer virus proliferation.** You are helping some security analysts monitor a collection of networked computers, tracking the spread of a virus. There are n computers in the system, call them C_1, C_2, $\ldots$, C_n. You are given a trace indicating the times at which pairs of computers communicated. A trace consists of m triples (C_i, C_j, k) that indicate that C_i communicated with C_j at time t_k. At that time, the virus could have spread between C_i and C_j.

   We assume that the trace holds the triples in sorted order by time. For simplicity, assume that each pair of computers communicates at most once over the time of the trace. Also, it is possible to have pairs (C_s, C_j, k) and (C_t, C_j, k): this would indicate that computer C_j opened connections to both C_s and C_t at time t_k, allowing the virus to spread any way among the three machines.

   Design and analyze an efficient algorithm that, given as input a collection of time-sorted trace data and a virus query, answers the question "if the virus was introduced to C_i at time x, could it spread to C_j at time y?" That is, is there a sequence of communications that could have lead to the virus moving from C_i to C_j?