Algorithms Lab 10

Homework (Collaboration[1] level:1)

1. Consider a directed graph G, and assume that instead of shortest paths we want to compute *longest paths*. Longest paths are defined in the natural way, i.e. the longest path from u to v is the path of maximum weight among all possible paths from u to v. Note that if the graph contains a positive cycle, then longest paths are not well defined (for the same reason that shortest paths are not well defined when the graph has a negative cycle). So what we mean is the *longest simple path*, (a path is called *simple* if it contains no vertex more than once).

   Show that the the *longest simple path* problem does not have optimal substructure by coming up with a small graph that provides a counterexample.

   Note: Finding longest (simple) paths is a classical *hard* problem, and it is known to be NPC (NP-complete).

2. Prove that the following claim is false by showing a counterexample:

   Claim: Let $G = (V, E)$ be a directed graph with negative-weight edges, but no negative-weight cycles. Let $w, w < 0$, be the smallest weight in G. Then one can compute SSSP in the following way: transform G into a graph with all positive weights by adding $-w$ to all edges, run Dijkstra, and subtract from each shortest path the corresponding number of edges times $-w$. Thus, SSSP can be solved by Dijkstra's algorithm even on graph with negative weights.

3. (CLRS 24.3-6) We are given a directed graph $G = (V, E)$ on which each edge (u, v) has an associated value $r(u, v)$, which is a real number in the range $[0, 1]$ that represents the reliability of a communication channel from vertex u to vertex v. We interpret $r(u, v)$ as the probability tht the channel from u to v will not fail, and we assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices.

[1]Collaboration level 1: verbal collaboration without solution sharing. You are allowed and encouraged to discuss ideas with other class members, but the communication should be verbal and additionally it can include diagrams on board. Noone is allowed to take notes during the discussion (being able to recreate the solution later from memory is proof that you actually understood it). Communication cannot include sharing pseudocode for the problem. Check complete guidelines at: https://turing.bowdoin.edu/dept/collab.php

4. (GT C-7.7) Suppose you are given a diagram of a telephone network, which is a graph G whose vertices represent switching centers, and whose edges represent communication links between the two centers. The edges are marked by their bandwidth. The bandwidth of a path is the *minimum* bandwidth along the path. Give an algorithm that, given two switching centers a and b, will output a maximum bandwidth path between a and b.

5. **All-Pair-Shortest-Paths via matrix multiplication:** In the APSP problem the goal is to compute the shortest path between all pairs of vertices $u, v \in V$. Note that the output is of size $\Theta(|V|^2)$ which means any algorithm for APSP runs in $\Omega(|V|^2)$.

   (a) We can solve the problem simply by running Dijkstra's algorithm $|V|$ times. What is the running time of this approach? What does the running time become for sparse graphs ($E = \theta(V)$) and for dense graphs ($E = \theta(V^2)$)?

   We can obtain another APSP algorithm by working on adjacency matrix of the graph, which we denote by A: for weighted graphs, a_{ij} is equal to the weight w_{ij} of the edge (v_i, v_j); w_{ij} is assumed to be ∞ is the edge does not exist.
   Let A, B be two matrices, and let $C = A \cdot B$. Remember that

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

   We redefine the $\sum$ and $\cdot$ operators in matrix multiplication to mean *minimum* and $+$, respectively. That is,

$$c_{ij} = min_{k=1..n}\{a_{ik} + b_{kj}\}$$

   (b) What does $A \cdot A$ represent in terms of paths in graph G? What about $\min\{A, A \cdot A\}$?

   (c) Sketch an algorithm for computing APSP using this approach and estimate its running time.

   (d) Improve your algorithm by being smart about how you compute powers.
   Hint: aim to compute a^n in $O(\lg n)$ rather than in $O(n)$ time.

   (e) Describe how this corresponds to dynamic programming.
   Hint: consider the following subproblem: $d_k(u, v)$ is the shortest path from u to v that consists of at most k edges.