Assignment 13

Module: Graphs

Honor code: *Work on this assignment alone or with one partner. Between different teams, collaboration is at level 1 [verbal collaboration only]. There are lots of resources online, such as animations, visualizations, practice problems, videos, and solutions— which you are encouraged to explore to deepen your understanding. However, you must be careful not to search for the specific problems in the assignment with the intent of getting hints for the solution. Searching for the assignment problems on the internet violates academic honesty for this class.*

---

1. **Most reliable path:** 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.

   We expect: Pseudocode, justification, analysis.

2. **Max-bandwidth path:** 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.

   We expect: Pseudocode, justification, analysis.

3. **Computing all-pair shortest paths with dynamic programming:** You are given a directed graph $G = (V, E)$ with positive or negative edge weights but no negative cycles. Denote the number of vertices $|V| = n$. The goal is to find the length of the shortest paths from v_i to v_j, for all vertices $1 \leq i, j \leq n$.

   One way to do this is to run an SSSP algorithm n times, once with each vertex v_i as source. An improved algorithm was proposed by Floyd and Warshall, and is known as the Floyd-Warshall algorithm. In this problem you will reconstruct it.

   The idea is to use dynamic programming, with the following choice of subproblem:

   shpath(i, j, k): returns the length of the shortest possible path (if one exists) from v_i to v_j among all paths that use only vertices from the set $\{v_1, v_2, ..., v_k\}$ as intermediate vertices along the way.

   Given this subproblem, our goal is to compute the shortest path from every v_i to every v_j allowing *any* vertex along the way, i.e. shpath(i, j, n).

Recursive definition: Clearly if we don't allow any intermediate vertices, then shpath$(i, j, 0)$ will be the weight of the edge (v_i, v_j) if this edges exists, and ∞ otherwise. For $k \geq 1$: shpath(i, j, k) could be either:

- a path that does not go through vertex v_k (and therefore uses only vertices in the set $\{v_1, v_2, ...v_{k-1}\}$)
- a path that goes through vertex v_k: $v_i \rightsquigarrow v_k \rightsquigarrow v_j$. Since a shortest path cannot contain a vertex more than once, it follows that the paths $v_i \rightsquigarrow v_k$ and $v_k \rightsquigarrow v_j$ only go through vertices $\{v_1, v_2, ...v_{k-1}\}$.

(a) Optimal substructure: What can you say about the subpath $v_i \rightsquigarrow v_k$ and $v_k \rightsquigarrow v_j$?

(b) Recursive definition: Express shpath(i, j, k) recursively in terms of $k - 1$ and don't forget the basecase.

(c) Denote by $d[1..n][1..n]$ a 2-dimensional array such that $d[i][j]$ represents the length of the shortest path from v_i to v_j. Using the recursive definition above, give pseudocode for an iterative algorithm to compute $d[i][j]$ for all $1 \leq i, j \leq n$. What is the runnig time?

We expect: Pseudocode, analysis.