Recall that your solution must be typed via Latex. Also, remember that this is a group assignment; group size must be 2. Since our class has an odd number of students, I allow exactly one group to have size 3. If you want to form a group of size 3, email me to make sure that there’s one such group.

Deliverables: Submit to Canvas two PDF files: a writeup for the homework solution, and a transcript of the discussion of your group members. Both files should contain the names of all group members.

1. (70 points) We are given a directed graph $G = (V, E)$ on which each edge $(u, v) \in E$ has an associated value $r(u, v)$, which is a real number in the range $0 \leq r(u, v) \leq 1$ that represents the reliability of a communication channel from vertex $u$ to vertex $v$. We interpret $r(u, v)$ as the probability that the channel from $u$ to $v$ will not fail, and we assume that these probabilities are independent. That is, for a path $P = v_1 \to \cdots \to v_k$, the probability that the communication won’t fail if we send data along this path is exactly

$$r(v_1, v_2)r(v_2, v_3)\cdots r(v_{k-1}, v_k).$$

Give an efficient algorithm that, for two given nodes $s$ and $t$, finds the most reliable path from $s$ to $t$. Analyze the running time of your algorithm.

[We expect an informal English description of your algorithm. You need to justify its correctness and running time.]

Hint: The key idea is to use Dijkstra’s algorithm; however, of course you can’t apply Dijkstra’s directly here. It’s tempting to modify Dijkstra’s algorithm to fit your problem, but this is bad because proving the correctness of your algorithm is not easy. A good approach is to convert your $(G, r)$ into another weighted graph, and run Dijkstra’s algorithm on it.

2. (70 points) We are given a directed graph $G = (V, E)$ on which each edge $(u, v) \in E$ has an associated capacity $c(u, v) \geq 0$ that represents the rate of data that we can send in the link $(u, v)$. The graph has $n = |V|$ nodes and $m = |E|$ edges, and assume that $m \geq n$. Let $s$ and $t$ be two nodes in the graph. For a path $s = u_0 \to u_1 \cdots \to u_k = t$, its bottleneck is

$$\min\{c(u_0, u_1), \ldots, c(u_{k-1}, u_k)\}.$$

Informally, the bottleneck of a path is the limitation of the rate of data that you can send along that path.

a) (30 points) Give an algorithm of $O(m)$ time that, given a number $B \geq 0$, finds a path from $s$ to $t$ whose bottleneck is at least $B$. If many such paths exist, you only need to find one.

[We expect an informal English description of your algorithm. You need to justify its correctness and running time.]

b) (40 points) Using part (a), design an $O(m \log(n))$ algorithm that finds the widest path from $s$ to $t$, meaning a path from $s$ to $t$ of the largest bottleneck possible. (Note: Recall that $m = O(n^2)$, so $\log_2(m) = O(\log(n))$.)

[We expect an informal English description of your algorithm. You need to justify its correctness and running time.]
3. (60 points) Let $G = (V, E, w)$ be a weighted, connected undirected graph whose weights may be negative. We want to find a connected subgraph $G' = (V, E')$ (meaning the same node set) of $G$ that minimizes the total weight. If the weights are positive then we get back to the minimum spanning tree problem. However, the situation is different if weights can be negative. For example, if all the weights are negative then the minimum spanning subgraph is $G$ itself.

Give an efficient algorithm that finds a minimum connected subgraph. Analyze the running time of your algorithm.

[We expect an informal English description of your algorithm. You need to justify its correctness.]

**Hint:** Note that any edge of negative weight must be included in a minimum connected subgraph.

4. (Bonus) (70 points) Let $G = (V, E, w)$ be a connected, weighted undirected graph. Assume that the weights are strictly positive. A set $F \subseteq E$ of edges is called a feedback-edge set if every cycle of $G$ has at least one edge in $F$. Design an efficient algorithm to find a minimum-weight feedback-edge set. Analyze the running time of your algorithm.

[We expect an informal English description of your algorithm. You need to justify its correctness.]

**Hint:** For a feedback-edge set $F$, the complement feedback graph $G_F$ is the subgraph of $G$ with the same node set $V$, but edge set $E \setminus F$. That is, $G_F$ is obtained by deleting edges of $F$ in $G$. In order to pick a feedback-edge set $F$ of minimum weight, we need to find a complement feedback graph $G_F$ of maximum weight.