A looped tree $G = (V, E)$ is an edge-weighted directed graph built from some (directed) binary tree $T$ on $V$ rooted at some node $r \in V$, by adding an edge from every leaf in $T$ back to $r$ (e.g., if $T$ was a long directed path, the looped tree $G$ would be a cycle). Assume that the vertices are labeled from 1, ..., $n$, with the root $r$ having label 1, and the edges are given in the form of an adjacency list, along with the corresponding edge weights. Moreover, assume that all the edge weights are non-negative.

Your goal in this problem be to develop a faster-than-Dijkstra single-source shortest-path algorithm computing all shortest distances $d[v]$ from a given input node $u \in V$ to all other nodes $v$ of $G$. You will do it by solving the following sub-problems:

(a) (1 point) Show that the number of edges in $G$ is $O(n)$.

Solution: INSERT YOUR SOLUTION HERE

(b) (2 points) Compute the running time of Dijkstra’s algorithm to compute the shortest distance $d[v]$ from a given vertex $u$ to all $v \in V$.

Solution: INSERT YOUR SOLUTION HERE

(c) (3 points) Modify the BFS algorithm from the lecture appropriately to give an $O(n)$ algorithm that computes for all $v \in V$ the shortest distance $c[v]$ from the root $r$ to $v$.

Solution: INSERT YOUR SOLUTION HERE

(d) (5 points) Give an $O(n)$ algorithm that computes the shortest distance $\alpha$ from $u$ to the root $r$, for the given source vertex $u$. (Hint: Think of recursion, but make sure you terminate, and fast!)

Solution: INSERT YOUR SOLUTION HERE

(e) (3 points) Let $T = (V, E')$ be the original rooted tree you started from (before adding the edges from the leaves of $T$ to $r$). Give an $O(n)$ algorithm to compute the shortest distance $b[v]$ from $u$ to $v$ in $T$ (not $G$), for all $v \in V$.

Solution: INSERT YOUR SOLUTION HERE

(f) (4 points) In $G$, express (with proof) the shortest distance $d[v]$ from $u$ to $v$ in terms of $b[v], c[v]$, and $\alpha$. Use this expression to obtain an algorithm to compute $d[v]$ for all $v \in V$ with running time asymptotically faster than Dijkstra’s algorithm.

Solution: INSERT YOUR SOLUTION HERE
You are given a directed graph $G = (V, E)$ with non-negative edge weights $w(u, v)$ representing the time it takes to go from a node $u$ to $v$ when $u$ and $v$ are directly connected. (We assume all $w(u, v) < \infty$ for simplicity.) Some subset $F$ of vertices in $G$ are pharmacies, while some other subset $D$ are doctor offices. You live at a node $s$ and have a medical emergency. You now need to go to some doctor office $d \in D$ to get a prescription for the drug, then to some pharmacy $f \in F$, and finally back home. Recall that a proper output must be the full path described above. However some paths are better then others. Solve the following variants of this problem, where each variant describes how to compare different valid paths when choosing the optimal path you are looking for. Each variant could be solved by one “clever” run of the Dijkstra’s algorithm, on an appropriate modification of our graph $G$. You have to explain your choice and state the resulting running time as the function of $n$ (remember, we assume $m = \Theta(n^2)$ here, as $w(u, v) < \infty$).

(a) (3 points) Assume only the distance from your home $s$ to the doctor’s office matters (e.g., you need to get doctor’s emergency help asap, and the time it then takes to the pharmacy and back home are not important).

**Solution:** INSERT YOUR SOLUTION HERE

(b) (5 points) Assume only the distance between the doctor office $d$ and the pharmacy $f$ matters. E.g., the doctor would perform some painful procedure, but does not have a tranquilizer to numb the subsequent pain. Thus, you must choose an office $d \in D$ and pharmacy $f \in F$ with the smallest distance between them, but do not care how far $d$ is from $s$ or $s$ is from $f$. (E.g., if there is a pharmacy in some doctor’s office in Antarctica, you do not mind going there, as the answer you care will be 0, even if you live in NYC.)

**(Hint:** Add an artificial source node $s' \rightarrow G$ and compute the distance from $s'$ to “fill the blank” in your new graph.)

**Solution:** INSERT YOUR SOLUTION HERE

(c) (5 points) Assume only the distance from the pharmacy $f$ back to home $s$ matters. E.g., the medicine must be administered by the pharmacist and has some quick side effect, so you must get to bed asap after taking the medicine in the pharmacy.

**(Hint:** Two solutions are possible: one does something to the edges of $G$, and the other again adds some artificial source similar to part (b).)

**Solution:** INSERT YOUR SOLUTION HERE
(d) (7 points) Finally, assume that the overall trip time from $s$ to $d$ to $f$ to $s$ matters. Show how to combine the ideas in parts (a)-(c) to solve this variant.

(Hint: Make several “copies” of $G$ and connect them “appropriately” by 0-weight edges. The copies will ensure that every valid path must pass through a doctor office followed by a pharmacy.)

**Solution:** INSERT YOUR SOLUTION HERE
Assume we define the length of a path to be the maximum weight among all the edges of the path rather than the sum of all the edge weights. Argue that the Floyd-Warshall algorithm can be modified to handle this situation. That is, for every pair of nodes, find the value of the “shortest” path connecting these nodes, where the meaning of “shortest” is appropriately modified according to the above. Make sure to write the pseudo-code of your new algorithm and briefly argue why it is correct.

**Solution:** INSERT YOUR SOLUTION HERE