Instructions: Everyone needs to submit their own write-up. If you work together with other students, indicate their names on your write-up.

Problem 1: Computing R-neighborhoods in weighted graphs.

[Given the algorithm that you will design in this problem, one can also approximate the diameter of weighted graphs with the same runtime and accuracy guarantees as what we had for unweighted graphs.]

Let R be an integer between 1 and n. Modify Dijkstra's algorithm to design an $O(R^2 \log n)$ time algorithm that given any n-node graph $G = (V, E)$ with nonnegative integer edge weights $w : E \to \mathbb{Z}^+$, and a source s, finds the closest R vertices T_s to s, and the distance between s and every $v \in T_s$, under the following assumptions:

1. There is a data structure F (e.g. Binomial heap) that stores up to n pairs (e, k) (where e is an element and k is an integer value) and supports the following operations each in $O(\log n)$ time:

- insert(e, k): insert an element e into F with value k, provided e is not in F yet with any value
- decrease-key(e, k): if e is in F with value $k' \geq k$, change its value to k
- extract-min: return (e, k) where e has the minimum value k over all elements in F, deleting (e, k) from F

2. G is given in adjacency list representation, and for each $u \in V$, the neighbors of u, $N(u)$ are sorted in nondecreasing order of their edge weight.

Give pseudocode for your algorithm, prove that it is correct and that it runs in $O(R^2 \log n)$ time.

What would the running time of your algorithm be if F were a Fibonacci heap data structure? (Here all operations are $O(1)$ amortized time except for extract-min which takes $O(\log n)$ time.)

Problem 2: Radius approximation.

The radius of a graph is given by $R = \min_v \max_u d(u, v)$. In this problem we will adapt the diameter approximation algorithm given in class to obtain an $\tilde{O}(m\sqrt{n})$ time 3/2-approximation algorithm for the radius R of any given undirected graph on n nodes and m edges, whenever R is even.

The eccentricity $\epsilon(v)$ of a node v is defined as the maximum distance from v to another node, i.e. $\epsilon(v) := \max_{u \in V} d(u, v)$.

The *center* c of a graph G is the node in G of minimum eccentricity, i.e. $c := \arg\min_{v \in V} \epsilon(v)$.

Assume below that the radius of the given graph G is even. For simplicity, you can also assume that the graph is undirected.

Let S be a random sample of $O(\sqrt{n} \log n)$ nodes, let w be the node furthest from S and T_w be the closest $\sqrt{n}$ nodes to w, just as in the diameter algorithm from class. You can assume that S hits T_w, as we showed in class that it will do so with high probability.

- Show that if for some node s in the random sample S, $d(s, c) \leq R/2$, then $R \leq \min_{s \in S} \epsilon(s) \leq 3R/2$, and hence one can return an estimate R' of the radius so that $R \leq R' \leq 3R/2$.
- Show that if for all nodes $s \in S$, $d(s, c) > R/2$, then all nodes at distance $R/2$ from w are in T_w.
- Show that if $d(w, c) \leq R/2$, then $R \leq \epsilon(w) \leq 3R/2$.
- Show that if $d(w, c) > R/2$ and for all nodes $s \in S$, $d(s, c) > R/2$, then there is some node x in T_w with $\epsilon(x) \leq 3R/2$, and hence $R \leq \min_{x \in T_w} \epsilon(x) \leq 3R/2$.
- Give pseudocode for the radius approximation algorithm.

Problem 3: Approximating distances for long paths.

Design an $O(n^{2.5} \log n)$ time algorithm that, given an n-node graph with nonnegative edge weights, computes for all pairs of vertices u, v an estimate $D(u, v)$, such that with high probability, for all pairs of vertices u, v for which there is a shortest path that uses at least $\sqrt{n}$ nodes, $D(u, v)$ is the distance between u and v.

(Your algorithm does not need to know for which pairs u, v, $D(u, v)$ is the correct distance, and it doesn't have to guarantee anything about the estimate $D(u, v)$ if all shortest paths between u and v use fewer than $\sqrt{n}$ nodes.)