6.890 HOMEWORK 4 **Due: November 10, 2021**
Submit by email to `6890mit@gmail.com` Handed out: October 28, 2021

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

The maximum number of points in the problem set is 26.

Problem 1: Radius approximation. [10pt = 5 check+s, one per bullet]

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 unweighted.

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, $d(s,c) \le R/2$, then $R \le \min_{s\in S} \epsilon(s) \le 3R/2$, and hence one can return an estimate R' of the radius so that $R \le R' \le 3R/2$.
- 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 $d(w,c) \le R/2$, then $R \le \epsilon(w) \le 3R/2$.
- 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 $\epsilon(x) \le 3R/2$, and hence $R \le \min_{x\in T_w} \epsilon(x) \le 3R/2$.
- Give pseudocode for the radius approximation algorithm.

Problem 2: Emulators. [8pts = 4 check+s: 2 for construction, 1 for sparsity bound, 1 for error bound]

Give an algorithm that given an undirected, unweighted graph $G = (V,E)$ on n nodes, creates a graph H (called an emulator) on the same vertex set V as G and with $O(n^{4/3}\log^2 n)$ edges with edge weights in $\{1,\ldots,n-1\}$ so that for all $u,v \in V$, $d(u,v) \le d_H(u,v) \le d(u,v)+4$.

Here $d(u,v)$ and $d_H(u,v)$ are the distances between u and v in G and H, respectively.

Problem 3: Distance Oracle Creation. [8pt= 4 check+s, one per subpart]

Consider the construction for $2k-1$-approximate distance oracles from class. There we constructed sets $A_{k-1} \subseteq A_{k-2} \subseteq \ldots \subseteq A_1 \subseteq A_0 = V$ so that each A_i was random of size $\tilde{O}(|A_{i-1}|/n^{1/k})$. Then we constructed the distance oracle by finding for each vertex v,

- for $j \in \{0,\ldots,k-1\}$, the closest vertex $p_j(v)$ in A_j to v,
- for $i < k-1$, $B_i(v) = \{x \in A_i \mid d(x,v) < d(p_{i+1}(v),v)\}$, and $B_{k-1}(v) = A_{k-1}$,
- setting $B(v) = A_{k-1} \cup \bigcup_{i=0}^{k-2} B_i(v) = \bigcup_i B_i(v)$

In order to compute $B(v)$, we assumed that we knew $d(v,x)$ for every $v \in V$ and $x \in B(v)$.
In this problem we will show how to compute these distances in $\tilde{O}(mn^{1/k})$ time, given the sets $A_0, \ldots, A_{k-1}$.

(a) Design an $O(m + n \log n)$ time algorithm that given $i \in \{0, \ldots, k-1\}$, computes for every $v \in V$, the quantity $\min_{x \in A_i} d(x,v)$ and the closest node $p_i(v) \in A_i$ to v.

(b) Consider any $w \in A_i \setminus A_{i+1}$. Let $C(w) = \{v \in V \mid d(v,w) < d(v, p_{i+1}(v))\}$. Show that for every $v \in V$ and $w \in A_i \setminus A_{i+1}$ (for some i), $w \in B(v)$ if and only if $v \in C(w)$.

In the following problems, let $C(w) = V$ for any $w \in A_{k-1}$.

(c) Let $C(w)$ be as above. Show that $\sum_{w \in V} \sum_{x \in C(w)} deg(x) = \sum_{x \in V} deg(x) \left(\sum_i |B_i(x)|\right)$ and conclude that with high probability, $\sum_{w \in V} \sum_{x \in C(w)} deg(x) \leq \tilde{O}(kmn^{1/k})$.

(d) Show how to modify Dijkstra's algorithm so that for any $w \in V$, one can compute $C(w)$ and the distances from w to every $x \in C(w)$ in $\tilde{O}(\sum_{x \in C(w)} deg(x))$ time. This together with part (c) shows that the preprocessing time of the distance oracles from class can be made $\tilde{O}(mn^{1/k})$ (with high probability).