HW 8

Due on **Wednesday, October 27, 2021 at 10am**

CS 473: Algorithms, Fall 2021 Version: **2.3**

Submission guidelines and policies as in homework 1.

22 (100 PTS.) A long path revisited.

Let G be an undirected connected graph with n vertices and m edges.

22.A. (10 PTS.) The *average degree* of G is $d_{\text{avg}}(\mathsf{G}) = \sum_{v \in \mathsf{V}(\mathsf{G})} d(v)/n = 2m/n$.
Prove that if there is a vertex v in graph, such that $d(v) < m/n$, then $d_{\text{avg}}(\mathsf{G} - v) > d_{\text{avg}}(\mathsf{G})$, where $\mathsf{G} - v$ is the graph resulting from removing v and all its adjacent edges.

22.B. (20 PTS.) Consider the algorithm that starts with an arbitrary vertex $v_1 \in \mathsf{V}(\mathsf{G})$. In the ith iteration, it picks a vertex v_{i+1} which is a neighbor of v_i, which was not visited yet. The algorithm continues in this fashion till it gets stuck, as all the neighbors of this final vertex are already visited. Show how to modify this algorithm so it computes a path with $\geq m/n+1$ vertices. What is the running time of your algorithm.

22.C. (40 PTS.) Let $\delta = \min_{v \in \mathsf{V}(\mathsf{G})} d(v)$, with $4\delta < n$, and consider a path $\pi = v_1 v_2 \ldots v_k$ in G, with at $k \leq 2\delta$ vertices (assume $k > 2$). Prove that either:

(I) The vertices v_1 or v_k have neighbors that are not in π (and thus π can be extended).

(II) The vertices v_1 and v_k are neighbors. Then the cycle C formed by $v_1 \cdots v_k$ can be modified into a simple path with $k+1$ vertices (i.e., a path made out of $k+1$ distinct vertices and no vertex is visited twice by the path).

(III) Otherwise, using the pigeonhole principle, show that there is an i, $1 < i < k$, such that $v_i v_k \in \mathsf{E}(\mathsf{G})$ and $v_{i+1} v_1 \in \mathsf{E}(\mathsf{G})$. Conclude, that one can turn π into a simple cycle, and compute from it a path with $k+1$ vertices.

Conclude that there is always a path in G with $2\delta + 1$ vertices.

22.D. (30 PTS.) Using the above, describe an algorithm, with polynomial running time, that computes a path in G with at least $2m/n + 1$ vertices. What is the running time of your algorithm?

23 (100 PTS.) Shorter question.

The input is a set $X = \{x_1, x_2, \ldots, x_n\}$ of n numbers. For a number $y \in \mathbb{R}$, the *closest number* to y is $\ell(y) = \min_{x \in X \setminus \{y\}} |x - y|$. Consider the set of distances $L = \{\ell(x_1), \ldots, \ell(x_n)\}$.

23.A. (20 PTS.) Let $\#_k = \#_k(X)$ be the kth smallest number in L. Describe how to compute $\#_k$ in $O(n \log n)$ time.

23.B. (20 PTS.) Let r be a parameter. Describe how to decide in linear time, for each number $x \in X$, whether

(i) $r < \ell(x)$,

(ii) $r > \ell(x)$, or

(iii) or $r = \ell(x)$.

23.C. (20 PTS.) Prove that if for some element $x \in X$, we have $\ell(x) > \#_k(X)$, then $\#_k(X - x) = \#_k(X)$, where $X - x = X \setminus \{x\}$.

23.D. (40 PTS.) Using the above, describe an algorithm that computes $\#_k(X)$ in $O(k \log k + n)$ expected time.

[**Hint:** Either $|X| = O(k)$, or alternatively, a random number $x \in X$, with good probability, has the property that $\ell(x) > \#_k(X)$, and at least (say) third of the numbers in X have even bigger value of ℓ. One can throw such numbers away, and continue.]

(One can get $O(n)$ expected running time here for any value of k, but it requires quite a bit more work.)

24 (100 PTS.) Shortest question.

You are given a connected undirected graph G with n vertices and m edges. Using **22** (verify that it works even if the graph is disconnected), describe a polynomial time algorithm that computes a set of $O(n \log n)$ simple paths that are edge disjoint, and cover all the edges of G. What is the running time of your algorithm?