Sublinear Time Algorithms	October 14, 2022
Homework 3	
Lecturer: Ronitt Rubinfeld	*Due Date: October 26, 2022*

1. **Simulating Random Edges with Random Vertices**

   Let $G = (V, E)$ be an undirected graph on n nodes and m edges, given in adjacency list representation. In this model, we can query (i.e., "sample") a uniform random vertices from V, query degrees of vertices, and query the i^{th} neighbor of a given vertex for any positive integer i (when $i > \deg(v)$, this query fails).

   In this problem, we explore a way of using the above operations to sample *edges* of G, "ϵ-close to uniform" at random. Formally, let $\epsilon \in (0, 1/2)$ be a constant. Our goal is to present an algorithm which uses use a small number of queries, and with constant probability outputs an edge of the graph, such that the probability any given edge of G is output is within a $(1 \pm \epsilon)$ multiplicative factor of $1/m$.

   In what follows, for convenience we view each (undirected) edge $e = \{u, v\}$ of G as giving rise to two distinct directed edges (u, v) and (v, u).

   (a) Consider the following algorithm: (1) sample a uniform random vertex u from G, (2) sample a uniform random neighbor v of u, and (3) and output the edge (u, v). Explain why this algorithm fails to output uniform random edges of G.

   The algorithm in (a) is biased towards sampling edges (u, v) where u has low degree. To sample edges almost uniformly, we will need a way of making it more likely to sample edges (v, w) where the v has high degree.

   This motivates distinguishing edges by the degrees of their first vertices. Let $\Delta > 0$ be some degree cutoff (to be determined later). Let L be the set of vertices u with $\deg(u) \leq \Delta$ ("low-degree vertices"), and H be the set of vertices v with $\deg(v) > \Delta$ ("high-degree vertices"). Further define

   $$E_L = \{(u, v) \in E \mid u \in L\} \quad \text{and} \quad E_H = \{(v, w) \in E \mid v \in H\}$$

   to be the sets of "low-degree edges" and "high-degree edges" respectively.

   In parts (b) through (e) of this problem, you will prove the correctness of algorithms which sample from E_L and E_H separately, and then show how to combine them to get an algorithm which samples edges from E close to uniformly at random.

   (b) Present an algorithm which makes $O(1)$ queries, and with probability $|E_L|/(n\Delta)$, outputs a uniform random edge from E_L (and otherwise fails to output an edge).
   Hint: it may help to modify the algorithm from (a).

   (c) Consider the following algorithm: (1) run your procedure from part (b) to get an edge (u, v), (2) if $v \in H$, sample a random neighbor of w of v, and then (3) output the edge (v, w). If $v \notin H$ this algorithm fails to output an edge.

Prove that by setting the value of Δ appropriately, this algorithm, with some probability p satisfying

$$(1-\epsilon/2)\cdot\frac{|E_H|}{n\Delta}\le p\le\frac{|E_H|}{n\Delta},$$

samples an edge of E_H, $\epsilon/2$-close to uniform (and otherwise fails to output an edge).

(d) Consider the algorithm which with probability 1/2 runs the procedure from part (b), and with probability 1/2 runs the procedure from part (c) (in both cases, using the value of Δ determined in your soultion to part (c)).

Prove that this algorithm, with probability at least $(1-\epsilon)m/(2n\Delta)$, samples a random edge from G, ϵ-close to uniform.

(e) Finally, present an algorithm which, with probability at least 2/3, samples a random edge from G, ϵ-close to uniform, using only $\tilde{O}_\epsilon(n/\sqrt{m})$ queries.

2. **Vertex Covers & Monotonicity on DAGs**

A *vertex cover* V' of a set of edges E' is a set of nodes such that every edge of E' is adjacent to one of the nodes in V'.

For graph $G=(V,E)$, let the *transitive closure graph* $TC(G)$ be the graph $G^{(tc)}(V,E^{(tc)})$ where $(u,v)\in E^{(tc)}$ if and only if there is a directed path from u to v in G.

Let $f:V\to\{0,1\}$ be a labeling of the vertices of a known directed acyclic graph G by 0 and 1. For any pair of nodes x and y, we say that $x\le_G y$ if there is a path from x to y in G. We say that f is *monotone* if for all $x\le_G y$, $f(x)\le f(y)$. The *minimum distance of f to monotone* is the minimum number of nodes that must be relabeled in order to turn f into a monotone function.

Let E' be the set of violating edges in $TC(G)$ according to f. Show that the minimum distance of f to monotone is equal to the minimum size of a vertex cover of E'.

3. **Testing Monotonicity of Boolean Functions on Directed Graphs**

Let G be a directed graph with vertex set V. Let $f:V\to\{0,1\}$ be a function mapping nodes of G to binary values. We say f is *monotone* if for all directed edges (u,v), we have $f(u)\le f(v)$. We say that f is ϵ-close to monotone if there is a monotone function g such that g and f differ on at most $\epsilon|V|$ entries. A testing algorithm knows the graph G in advance, and for a given node u, may query $f(u)$ in one time step.

(a) Let $V=\{v_1,\ldots,v_n\}$. For each directed graph $G=(V,E)$, let $B_G=(V',E')$ be the bipartite graph where $V'=\{v_1,\ldots,v_n\}\bigcup\{v'_1,\ldots,v'_n\}$, and $(v_i,v'_j)\in E'$ iff v_j is reachable from v_i in G.

Show that a q-query testing algorithm for f over graph B_G with distance parameter $\epsilon/2$ yields a q-query testing algorithm for f over graph G with distance parameter ϵ.

(b) Let f be a function on V which is ϵ-far from monotone over graph G. Then $TC(G)$ has a matching of violated edges of size at least $(\epsilon/2)|V|$. (Recall previous problem).

(c) Show that if f is a function over bipartite graph G, there is a test for monotonicity of f with query complexity at most $O(\sqrt{|V|/\epsilon})$.

4. **Testing on Strings: Concatenations of Palindromes**

   Let $L = \{uu^r vv^r | u, v \in \{0,1\}^*, 2(|u| + |v|) = n\}$. We saw in class that given a string x, distinguishing $x \in L$ from x that is ϵ-far (meaning that $> \epsilon n$ bits of x need to be changed in order to make x a member of L) requires $\Omega(\sqrt{n})$ queries. Give an algorithm for this problem that uses $O(\sqrt{n} \log n / \text{poly}(\epsilon))$ queries to the input. The running time does not have to be sublinear.

5. **Lower Bounds for Estimating the Weight of a MST**

   Give a lower bound on computing a multiplicative estimate on the MST of a graph G in adjacency list representation: Give two distributions over graphs of degree at most d and weights in the range $\{1, \ldots, w\}$ (for $w = o(n)$) such that

   (a) graphs in one distribution have an MST weight that is at least twice the MST weight of the graphs in the in other distribution

   (b) in order to distinguish the two distributions with constant probability of success, one must make at least $\Omega(w)$ queries

   If you can get the lower bound to have some nontrivial dependence on d and ϵ, even better!

   Note: it is possible to write this lower bound without explicitly using Yao's method.