CS 514: Advanced Algorithms II – Sublinear Algorithms **Rutgers: Spring 2020**

Problem set 1

Due: 11:59PM, February 25, 2020

Problem 1. Let us revisit the balls and bins experiment: We are throwing n balls into n bins by choosing a bin uniformly at random and independently for each bin. In lecture one, we proved that with high probability, namely, probability $1 - 1/n^c$, for any fixed constant $c > 0$, the maximum number of balls in any bins, i.e., the maximum load, is $O(\log n)$. We are going to improve this bound slightly in this problem and obtain asymptotically *tight* bound of $\Theta(\frac{\log n}{\log \log n})$ for the balls and bins experiment.

(a) Prove that with high probability, the maximum load in this experiment is $O(\frac{\log n}{\log \log n})$. **(10 points)**

Hint: Revisit the proof of Chernoff bound in the lecture notes and see if one can use a (slightly) stronger variant of Chernoff for this problem.

(b) Prove that with probability $1/n^d$ for some constant $d \in (0, 1)$, the first bin has load $\Omega(\frac{\log n}{\log \log n})$. **(10 points)**

Hint: Prove that the probability that the first bin receives exactly $0.1 \cdot \frac{\ln n}{\ln \ln n}$ balls is at least $\Theta(1) \cdot 1/n^d$ for some $d \in (0, 1)$. You may find the inequality $\binom{n}{k} \geq (n/k)^k$ for all n, k helpful.

Problem 2. We are going to design a sublinear time algorithm for estimating the number of triangles, namely, cliques on 3 vertices, in a given undirected graph. We will work with the general query model plus an additional random edge sample query:

- **Degree queries**: Given a vertex $v \in V$, output $\deg(v)$.
- **Neighbor queries:** Given a vertex $v \in V$ and $i \in [n]$, output the i-th neighbor of v or $\perp$ if $i > \deg(v)$.
- **Pair queries:** Given two vertices $u, v \in V$, output whether (u, v) is an edge in G or not.
- **Edge-sample queries:** Return an edge $e \in E$ uniformly at random and independently.

In the following, let n, m, and T denote the number of vertices, edges, and triangles in G, respectively. Consider the following random variable X:

(i) Sample an edge e from G uniformly at random (using an edge-sample query); let u, v be the endpoints of e such that $\text{ID}(u) < \text{ID}(v)$.

(ii) Sample a vertex w from $N(v)$ uniformly at random (using a neighbor-query); return $X = 0$ if $\text{ID}(v) > \text{ID}(w)$, otherwise go to the next line.

(iii) Check whether (u, w) is an edge in G (using a pair query); if so, let $X = m \cdot \deg(v)$ (using a degree query), otherwise let $X = 0$.

We will use this random variable X to design an algorithm for triangle counting.

(a) Prove that $\mathbb{E}[X] = T$. **(10 points)**

(b) Prove that $\text{Var}[X] \leq m \cdot n \cdot T$. **(10 points)**

(c) Let $k = \frac{10 \cdot mn}{\varepsilon^2 \cdot T}$. Consider an algorithm that computes random variable X independently for k times, denoted by $X_1, \ldots, X_k$, and return $Y = \frac{1}{k} \cdot \sum_{i=1}^{k} X_i$. Prove that, **(10 points)**

$$\Pr\left(|Y - T| \geq \varepsilon \cdot T\right) \leq \frac{1}{10}.$$

The above now implies an algorithm with runtime $O(\frac{mn}{\varepsilon^2 \cdot T})$ for estimating the number of triangles to within a $(1 \pm \varepsilon)$ multiplicative factor.

(d) **Bonus part:** There is a caveat with the algorithm above; one needs to know T to know how many times to repeat the random variable X but T is exactly the quantity we want to estimate!

Show that one can increase the runtime of the algorithm above by an $O(\log n)$ factor and removes the assumption of knowledge of T.

Hint: Assume that T is the maximum value possible, say n^3, and run the above algorithm to get an estimate; if the answer was "very" different with the assumption, repeat the process but this time make the guess equal to $n^3/2$ and then $n^3/4$ and so on. **(+20 points)**

General remark: The algorithm in this question does *not* achieve the optimal bounds for the triangle counting problem, which is known to be $O(\frac{m^{3/2}}{\varepsilon^2 \cdot T})$ (the two bounds match only when $m = \Theta(n^2)$ and otherwise the second bound is always better). However, one can use the ideas from the average degree estimation problem to improve the runtime of the above algorithm to match the optimal bounds (up to logarithmic factors).

Problem 3. Prove that any *deterministic* algorithm for estimating the number of connected components to within an $\varepsilon \cdot n$ additive factor in the adjacency list model requires $\Omega(n)$ time for some *constant* $\varepsilon \in (0, 1)$.

Hint: Use query complexity plus an adversary argument. **(20 points)**

Problem 4. We are going to design a sublinear time algorithm for estimating weight of a minimum spanning tree (MST) in *bounded degree* graphs.

Let $G(V, E)$ be a weighted connected graph with weight function $w : E \to \{1, \ldots, W\}$ on the edges and maximum degree d, specified in the adjacency list query model. The goal is to output a $(1 \pm \varepsilon)$ *multiplicative* approximation to the MST weight of G in poly$(\frac{d \cdot W}{\varepsilon})$ time, in particular, independent n, i.e., number of vertices in G.

(a) For any $i \in \{1, \ldots, W\}$, define G_i as the subgraph of G on the same set of vertices obtained using only edges e with weight $w(e) \in \{1, \ldots, i\}$ (thus $G_W = G$ and $G_0 =$ the empty graph). Let C_i denote the number of connected components of G_i. Prove that the MST weight of G is

$$\sum_{i=0}^{W} (C_i - 1) = \left(\sum_{i=1}^{W-1} C_i\right) + (n - W).$$

Hint: Think of Kruskal's algorithm! **(15 points)**

(b) Use the formula above plus the algorithm for estimating the number of connected components in lecture two on each graph G_i (in a black-box way, but by using appropriate parameters ε, δ), to design an algorithm for estimating the MST weight of G to within a $(1 \pm \varepsilon)$ approximation in poly$(\frac{d \cdot W}{\varepsilon})$ time with probability at least 3/4. **(15 points)**

Hint: Note that the algorithm for estimating number of connected components runs in $O(d/\varepsilon^3 \cdot \ln(1/\delta))$ time on graphs with maximum degree d to obtain an additive εn approximation with probability $1 - \delta$. Also use the fact that the MST weight is at least $(n - 1)$ and hence an additive $\pm\varepsilon \cdot n$ approximation to the MST weight also implies a $(1 \pm \varepsilon)$ multiplicative approximation.

Problem 5 (**Bonus Problem**)**.** We are going to consider the balls and bins experiment yet another time, but this time with a simple twist: We are throwing n balls into n bins one by one by choosing *two* bins uniformly at random for each ball independently, and placing the ball in the bin with a smaller number of assigned balls currently (breaking the ties arbitrary). Prove that with high probability, the maximum number of balls in any bins in this modified experiment is $O(\log \log n)$. **(+20 points)**

Note: Getting the exact calculations for this problem can be rather tedious. You will receive half the grade if you can provide the intuition and the high level overview of the proof correctly even if not all calculations are done completely – however, beware of the dependency issues between the random variables you define.