

Your Name

September 23, 2026

6.5240 Problem Set 1

Collaborators: *Firstname Lastname*

Due: September 24, 2026

Homework guidelines: You may work with other students, as long as (1) they have not yet solved the problem, (2) you write down the names of all other students with which you discussed the problem, and (3) you write up the solution on your own. No points will be deducted, no matter how many people you talk to, as long as you are honest. If you already knew the answer to one of the problems (call these “famous” problems), then let us know that in your solution writeup – it will not affect your score, but will help us in the future. It’s ok to look up famous sums and inequalities that help you to solve the problem, but don’t look up an entire solution.

1. **Estimating average degrees via empirical sampling:** In this exercise, you will compute the expectation and bound the variance of the following algorithm for estimating the average degree $\bar{d}(G)$ of a graph G , and you will show that these quantities alone are not enough to show a sublinear bound on the query complexity.

Empirical Average via Vertex Sampling

Require: Query access to unknown graph G , number of vertices n , parameter k

- 1 Take k uniformly random vertices $v_1, \dots, v_k$
- 2 Make k degree queries to get their degrees $d_1, \dots, d_k$
- 3 Return the empirical average $\hat{d} := \frac{1}{k} \sum_{i=1}^k d_i$

a) Show that $\text{Var}[\hat{d}] \leq \frac{1}{nk} \sum_{v \in V} d_v^2$.

b) Deduce that, to get a 2-approximation with probability $\geq 9/10$, it would suffice to have

$$k \geq 40n \cdot \frac{\sum_{v \in V} d_v^2}{(\sum_{v \in V} d_v)^2}$$

c) Show that, for every connected graph G ,

$$\frac{\sum_{v \in V} d_v^2}{(\sum_{v \in V} d_v)^2} \leq \frac{1}{2}$$

Conclude that taking $k \geq 20n$ is enough.

- d) Unfortunately, this is a number of samples that is *linear in n* , while we are hoping for much smaller ($O(\sqrt{n})$). One way to achieve this would be to show that:

$$\frac{\sum_{v \in V} d_v^2}{(\sum_{v \in V} d_v)^2} \leq O(1/\sqrt{n}).$$

Unfortunately, this is not true in general: show that for every $n \geq 2$, there exists a connected graph G on n vertices such that, for this graph, $\sum_{v \in V} d_v^2 = \Omega(n^2)$ and $\sum_{v \in V} d_v = O(n)$, and so

$$\frac{\sum_{v \in V} d_v^2}{(\sum_{v \in V} d_v)^2} = \Omega(1)$$

2. **Almost Uniform Edge Sampling:** In this problem, we will build an algorithm that samples an edge e in a graph $G(V, E)$ of n vertices and m edges, such that every edge is sampled with probability within the interval $[(1 - 10\epsilon)/m, (1 + 10\epsilon)/m]$ for some small constant ϵ . The algorithm will use $\tilde{O}(n/\sqrt{\epsilon m})$ number of queries. For simplicity, we assume that the exact value of m is given to the algorithm. We use adjacency lists to represent the graph, and we allow the following three types of queries:

- Vertex query: Return a uniform random vertex $v \in V$;
- Degree query: Give a vertex $v \in V$, return $\deg(v)$;
- Neighborhood query: Give a vertex $v \in V$ and $i \in [n]$, return the i -th neighbor of v or return *fail* if $\deg(v) < i$.

A vertex is *light* if its degree is at most $\sqrt{m/\epsilon}$ and it is *heavy* otherwise. An edge is *heavy* if both endpoints are heavy and is *light* otherwise.

- a) Provide a procedure with $O(1)$ queries that returns a light edge in E or *fail* such that each light edge is sampled with probability $1/(n\sqrt{m/\epsilon})$.
- b) Show that for every heavy vertex, the proportion of light vertices in its neighborhood is at least $(1 - 2\epsilon)$.
- c) Provide a procedure with $O(1)$ queries that returns a heavy vertex in V or *fail* such that each heavy vertex v is sampled with probability within the interval

$$\left[(1 - 2\epsilon) \frac{\deg(v)}{n\sqrt{m/\epsilon}}, \frac{\deg(v)}{n\sqrt{m/\epsilon}} \right].$$

Conclude with a procedure with $O(1)$ queries that returns a heavy edge in E or *fail* such that each heavy edge is sampled with probability within the interval

$$\left[(1 - 2\epsilon) \frac{1}{n\sqrt{m/\epsilon}}, \frac{1}{n\sqrt{m/\epsilon}} \right].$$

- d) Design an algorithm that outputs an edge $e \in E$ or *fail* within $\tilde{O}(n/\sqrt{\epsilon m})$ queries, such that each edge is returned with probability within $[(1 - 10\epsilon)/m, (1 + 10\epsilon)/m]$.

3. **Estimating the number of triangles:** In this problem we will design and analyze an algorithm for estimating the number of triangles in a graph $G(V, E)$. We allow the following four types of queries in the instance:

- Degree query: Give a vertex $v \in V$, return $\deg(v)$;
- Pair query: Give a pair of vertices u, v , return whether $(u, v) \in E$;
- Neighborhood query: Give a vertex $v \in V$ and $i \in [n]$, return the i -th neighbor of v or return *fail* if $\deg(v) < i$;
- *Uniform edge sample*: Return a uniform edge (u, v) drawn from E .

We assume that the number of edges m is known, as well as a lower bound $\hat{T}$ for the number of triangles in G . Our final running time will be $\text{poly}(1/\epsilon, 1/\hat{T}) \cdot o(m^3)$, thus a better lower bound $\hat{T}$ on the number of triangles will yield an improved running time.

- a) Consider sampling an edge uniformly, then choosing one of its endpoints uniformly. For a vertex v , what is the probability that v is the chosen vertex, in terms of $\deg(v)$?
- b) Let heavy vertices be defined as those with degree at least $\sqrt{2m}$. Give a procedure that uses $O(1)$ queries and outputs a vertex u (or $\perp$) as follows: if u is heavy, then it is returned with probability $1/\sqrt{2m}$. If u is light then it is returned with 0 probability.
- c) Recall from lecture that every vertex has at most $\sqrt{2m}$ *outgoing* neighbors (neighbors with higher degree, with ties broken by vertex ID, *and we assume that vertex ID is known to the algorithm*). Design a procedure for sampling outgoing neighbors such that each outgoing neighbor is sampled with probability $1/\sqrt{2m}$. As in (b), the procedure may output $\perp$. (Hint: consider separately nodes of degree $> \sqrt{2m}$ and nodes of degree $\leq \sqrt{2m}$).
- d) Prove that we can assign each triangle to one of its edges such that each edge has at most $\sqrt{2m}$ assigned triangles. (Hint: consider ordering the vertices according to degree).
- e) Design an algorithm that samples each triangle with probability $\frac{1}{m\sqrt{2m}}$. As in (b) and (c), the algorithm may output $\perp$.
- f) Design an algorithm that outputs an estimate t such that, with probability at least $2/3$, $(1 - \epsilon)T(G) \leq t \leq (1 + \epsilon)T(G)$.