This assignment is **due on August 30** and should be submitted on Gradescope. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

Important: the assignment contains 7 problems, "easy" (Problem 1, out of 10 marks), "medium" (Problems 2, 3, and 4, each out of 25 marks), and "hard" (Problems 5, 6, 7, each out of 40 marks). Problem 7 is a programming exercise, and one of its questions depends on (a subquestion of) Problems 2 and 3.

You do not have to solve every problem, but instead are meant to "choose your own adventure." You *must* solve (1) Problem 1, (2) at least two out of the 3 medium $\star$ problems, and (3) at least one of the 3 hard $\star\star$ problems. Your mark will be calculated as

$$\text{mark}(\text{Problem 1}) + \text{marks}(\text{best 2 out of medium}) + \text{mark}(\text{best 1 out of hard})$$

which gives a total mark out of $10 + 2 \times 25 + 40 = 100$. So be strategic: do as much as you can or want, but don't spend hours stuck on a problem you don't need to solve.

As a first step go to the last page and read the section: "Advice on how to do the assignment."

Throughout the assignment, we are given access to an undirected, connected graph $G = (V, E)$ with a (known) number $n = |V|$ of vertices and an *unknown* number $m = |E|$ of edges (note that the vertex set V itself is not known either, only its size n is). We can query the graph through the following:[1]

- Vertex query: get a uniformly random vertex $v \in V$
- Degree query: given a vertex $v \in V$, get its degree $d_v = \deg v$
- Neighbour query: given a vertex $v \in V$, get one of its neighbours uniformly at random.

Define the *average degree* $\bar{d}(G)$ of the graph by

$$\bar{d}(G) = \frac{1}{n} \sum_{v \in V} d_v \,.$$

Our goal is to obtain an algorithm which, for some choice of $\alpha \geq 1$, returns an estimate $\hat{d}$ of $\bar{d}(G)$ such that, with probability at least $9/10$,

$$\frac{1}{\alpha} \cdot \bar{d}(G) \leq \hat{d} \leq \alpha \cdot \bar{d}(G)$$

We call such a value $\hat{d}$ an *α-estimate* of the average degree. We want our algorithm to makes as few queries (of any kind) to G as possible.

[1] The randomness involved in each of the queries is independent of that used in previous queries. In particular, the sampling is done *with* replacement.

Prelude

Problem 1. (10 points)

a) Show that the task is equivalent to providing an α-estimate $\hat{m}$ of the number of edges m.

b) Give an algorithm which provides an exact ($\alpha = 1$) estimate and makes $O(n \log n)$ queries, and (briefly) justify its correctness.

Part I: A natural sampling approach

In the first part, we will consider the following natural algorithm to estimate the average degree: "sample k vertices uniformly at random, query their degrees, take the average." In more detail:

Algorithm 1 Empirical Average via Vertex Sampling

Require: Query access to unknown graph G, number of vertices n, parameter $k \geq 1$

1: Make k vertex queries to get uniformly random vertices $v_1, \ldots, v_k$
2: Make k degree queries to get their degrees $d_{v_1}, \ldots, d_{v_k}$
3: **return** the empirical average $\hat{d} = \frac{1}{k} \sum_{i=1}^{k} d_{v_i}$

Our overall goal is to show that, if we set $k = \Theta(\sqrt{n})$, Algorithm 1 provides a 2-estimate of the average degree. Before doing so, in the next two problems we will see why we are aiming for $\alpha = 2$ (why not smaller?) and $k = \Theta(\sqrt{n})$ samples (why not fewer?).

Problem 2. (25 points $*$) Consider the graph $G^\star$ on n vertices, called a *star graph*: one vertex is connected to all other $n - 1$ vertices, and there is no other edge. We

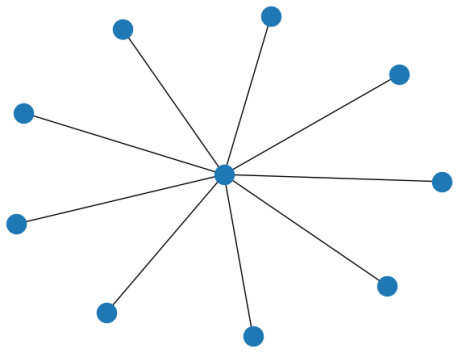

Figure 1: The star graph on $n = 10$ vertices.

will analyse the behaviour of Algorithm 1 when the input graph G is the n-vertex star graph.

a) State the average degree $\bar{d}(G^\star)$ of the star graph on n vertices.

b) Suppose that all k vertices sampled by Algorithm 1 are leaves. What is the output of the algorithm?

c) What is the probability that all k vertices sampled by Algorithm 1 are leaves?

d) Conclude that Algorithm 1 cannot provide a 1.99-estimate of the average degree unless it takes $\Omega(n)$ samples.

Problem 3. (25 points $*$) Fix any integer $a \geq 1$, and consider the following graph $G^\odot$ on $n \geq a^2$ vertices, which we'll call the *Saturn graph*: the graph is the disjoint union of two graphs K and C, where (1) K is a clique on $n_1 = a\lfloor\sqrt{n}\rfloor$ vertices, and (2) C is a cycle on $n_2 = n - n_1$ vertices. We will analyse the behaviour of Algorithm 1

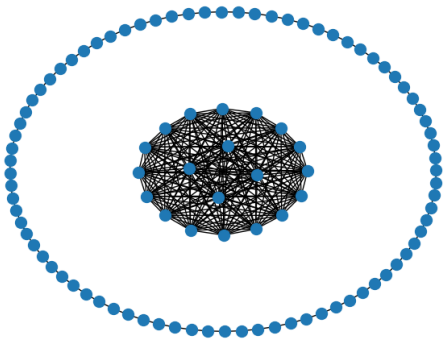

Figure 2: The Saturn graph on $n = 100$ vertices for $a = 2$, which is the disjoint union of a clique on $n_1 = 2 \cdot \sqrt{100} = 20$ vertices, shown in the middle, and a cycle on $n_2 = 80$ vertices, forming the "ring."

when the input graph G is the n-vertex Saturn graph.

a) State the average degree $\bar{d}(G^\odot)$ of the Saturn graph on n vertices as a function of n_1, n_2, n, then as a function of n only. What does this become when n is a square?

For simplicity, in what follows we assume that n is a square (to ignore $\lfloor\cdot\rfloor$'s).

b) Suppose that all k vertices sampled by Algorithm 1 are from the "ring" (i.e., from C). What is the output of the algorithm?

c) What is the probability that all k vertices sampled by Algorithm 1 are from the ring?

d) Conclude that Algorithm 1 cannot provide an $\frac{a^2}{2}$-estimate of the average degree unless it takes $\Omega(\sqrt{n})$ samples.

Now that we have shown *why* we cannot hope for better than an approximation factor $\alpha = 2$ (at least if we want $k = o(n)$) and $k = O(\sqrt{n})$, we will show that Algorithm 1 does, indeed, achieve these.

Problem 4. (25 points $*$) In this exercise, you are asked to compute the expectation and bound the variance of the output $\hat{d}$ of Algorithm 1.

a) Show that $\mathbb{E}[\hat{d}] = \bar{d}(G)$.

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

c) Deduce that, to get a 2-estimate with probability at least $9/10$, it would be sufficient to have

$$k \geq 10n \cdot \frac{\sum_{v \in V} d_v^2}{\left(\sum_{v \in V} d_v\right)^2} \tag{1}$$

d) Show that, for every connected graph G,

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

Conclude that taking $k \geq 5n$ in Algorithm 1 is enough.

e) Unfortunately, this is a number of samples *linear in* n, while we are hoping for much smaller (we want $O(\sqrt{n})$). Sadly, the analysis above cannot do better than this: show that, for every $n \geq 2$, there exists a connected graph $G = (V, E)$ 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

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

(Hint: we have already seen this graph.)

f) Conclude that the sufficient condition shown in c) is not enough for our goals.

The last problem shows how to analyse our algorithm better, going "beyond" a naive expectation/variance argument.

Problem 5. (40 points $\star\star$) Let $\beta \in (0, 1)$ be a constant. Given a graph $G = (V, E)$ with $n \geq 2$ vertices and m edges, we say a vertex $v \in V$ is *light* if $d_v < \sqrt{m/\beta}$, and *heavy* otherwise. Similarly, an edge $(u, v) \in E$ is *heavy* if both u, v are heavy vertices, *light* is both u, v are light, and *medium* otherwise.

a) Show that there are at most $2\sqrt{\beta m}$ heavy vertices.

b) Deduce that there are at most βm heavy edges.

c) Letting m_L be the number of light edges and m_M the number of medium edges, show that

$$\sum_{v \text{ light}} d_v = 2m_L + m_M$$

d) Deduce from the above that $\sum_{v \text{ light}} d_v \geq (1-\beta)m$, and that

$$\frac{1-\beta}{2} \cdot \bar{d}(G) \leq \frac{1}{n} \sum_{v \text{ light}} d_v \leq \bar{d}(G)\,.$$

This suggests that it would be sufficient, to get a 2-estimate, to only consider the light vertices. Of course, one issue is that given a vertex, v, we cannot tell whether it is a light vertex or not (since this would require checking if $d_v < \sqrt{m/\beta}$, *and we don't know* m – by the first problem, this is equivalent to what we are trying to estimate! But we can still use this *for the analysis of our algorithm*.

Consider the following thought experiment: in Line 7 of Algorithm 1, instead of

$$\hat{d} = \frac{1}{k} \sum_{i=1}^{k} d_{v_i}$$

we define

$$\hat{d}_L = \frac{1}{k} \sum_{i=1}^{k} d_{v_i} \mathbf{1}_{v_i \text{ is light}}$$

Again, we cannot actually run this new algorithm, but we *can* analyse it:

e) Show that $\frac{1-\beta}{2} \cdot \bar{d}(G) \leq \mathbb{E}[\hat{d}_L] \leq \bar{d}(G)$.

f) Show that $\mathrm{Var}[\hat{d}_L] \leq \frac{\sqrt{m/\beta}}{k} \cdot \mathbb{E}[\hat{d}_L]$.

g) Argue that $\hat{d} \geq \hat{d}_L$ (always), and that $\Pr[\hat{d} < t] \leq \Pr[\hat{d}_L < t]$ for all t.

h) Deduce that

$$\Pr\left[\hat{d} < \frac{(1-\beta)^2}{2}\bar{d}(G)\right] \leq \frac{\sqrt{m/\beta}}{k\beta^2\mathbb{E}[\hat{d}_L]} \leq \frac{1}{\beta^{5/2}(1-\beta)} \cdot \frac{n}{k\sqrt{m}} \leq \frac{\sqrt{2}}{\beta^{5/2}(1-\beta)} \cdot \frac{\sqrt{n}}{k}$$

(Prove all 3 inequalities.)

i) Show that $\Pr\left[\hat{d} > \frac{2}{(1-\beta)^2}\bar{d}(G)\right] \leq \frac{1-\beta}{2}$.

j) Conclude by showing that, for every $\varepsilon \in (0,1)$, there exists $C = C(\varepsilon)$ and a constant $c > 0$ such that Algorithm 1, run with parameter

$$k \geq C\sqrt{n}$$

returns a $(2+\varepsilon)$-estimate of $\bar{d}(G)$ with probability at least $1/2 + c \cdot \varepsilon$.

k) Briefly explain how to amplify the probability of success to $9/10$.

Part II: An elegant algorithm, for a more civilised age

We will now state and analyse a different algorithm, more recent, and (arguably) quite "elegant." To do so, we first need to introduce one piece of notation: given two vertices u, v of a graph G, we write

$$u \prec v$$

if $d_u < d_v$ or ($d_u = d_v$ and $u \leq v$); where for the second condition we assume that the vertices of a graph are lexicographically orders, for instance by giving them a unique number between 1 and n. Let us now state the algorithm:

Algorithm 2 A simple algorithm

Require: Query access to unknown graph G, number of vertices n, parameter $k \geq 1$

1: Make k vertex queries to get uniformly random vertices $v_1, \ldots, v_k$
2: Make k degree queries to get their degrees $d_{v_1}, \ldots, d_{v_k}$
3: **for all** $1 \leq i \leq k$ **do**
4: Make one neighbour query to get a uniformly random neighbour u_i of v_i
5: Make one degree queries to get its degree d_{u_i}
6: Set

$$X_i = \begin{cases} 2d_{v_i} & \text{if } v_i \prec u_i \\ 0 & \text{otherwise} \end{cases}$$

7: **return** $\widetilde{d} = \frac{1}{k}\sum_{i=1}^{k} X_i$

Problem 6. (40 points $\star\star$) For the sake of the analysis, define the *out-degree* of a vertex $v \in V$ as

$$d_v^+ = |\{u : (u, v) \in E \text{ and } v \prec u\}|$$

(equivalently, this is the out-degree of v in the directed graph $\vec{G}$ obtained by orienting the edges of G according to the total order $\prec$ defined above).

a) Show that $\sum_{v \in V} d_v^+ = m$.

b) Deduce that $\mathbb{E}[\widetilde{d}] = \bar{d}(G)$.

c) Show that, for every $v \in V$,

$$(d_v^+)^2 \leq \sum_{\substack{u:(u,v)\in E \\ v \prec u}} d_u$$

(Hint: $d_v^+ \leq d_v \leq d_u$ for every u such that $v \prec u$.)

d) Deduce that $\max_{v \in V} d_v^+ \leq \sqrt{2m}$.

e) Deduce that $\mathrm{Var}[\widetilde{d}] \leq \frac{1}{k}\sqrt{2m} \cdot \bar{d}(G)$.

f) Show that, for every $\varepsilon > 0$,

$$\Pr[|\widetilde{d} - \bar{d}(G)| \geq \varepsilon \bar{d}(G)] \leq \frac{n}{k\varepsilon^2\sqrt{2m}}$$

g) Conclude that there exists a constant $C > 0$ such that, for every $\varepsilon > 0$, Algorithm 2, when run with parameter

$$k \geq C\sqrt{n}/\varepsilon^2$$

outputs an $(1+\varepsilon)$-estimate with probability at least $9/10$.

Part III: Implementation

You are provided the following API in Python: the first 3 functions return graphs, and allow you to generate various test graphs on n vertices.

```
# Generates a "star graph" on n vertices
def starGraph(n):
# Generates a "Saturn graph"  on n vertices with a=2
def saturnGraph(n):
# Generates a random graph on n vertices
def randomGraph(n):
```

The next 4 functions provide the graph query access discussed in this assignment.

```
# Returns a uniformly random vertex from an input graph G
def randomNode(G):
# Same, but k random vertices (independently chosen) at once
def randomNodes(G, k):
# Returns a uniformly random neighbour of vertex v from G
def randNeighbour(G, v):
# Returns the degree of a given vertex v from G
def degree(G, v):
```

Problem 7. (40 points) You are asked to provide the code, as well as the plots asked.

a) Implement Algorithm 1 using the above API. (Your code needs to pass at least 75% of the "Algorithm 1" test cases)

b) Implement Algorithm 2 using the above API. (Your code needs to pass at least 75% of the "Algorithm 2" test cases)

c) For both the star and the Saturn graph on $n = 10000$ vertices, and for each of the two algorithms: run the algorithm on the graph for each value of $1 \leq k \leq 500$, repeating it $t = 1000$ times for each k. Compute the empirical probability $p(k)$ that the output of the algorithm is a 2-estimate of the true value (which, for each graph, you have computed in Problems 2 and 3). Provide the 4 corresponding plots, correctly labelled (empirical probability of success as a function of k, for each of the two graphs and two algorithms).

d) Discuss the plots and the results: which algorithm seems better? Is there anything counterintuitive or unexpected in some or all of these plots? If so, mention it.

e) Run the same setup as in b), but only on Algorithm 1 and the star graph for $n = 100$ (only one, much smaller graph). Interpret the resulting plot the best you can.

f) Run both algorithms on the "random graph" (third API function) on n vertices, for $2 \leq n \leq 10000$ and $k = 10\lfloor\sqrt{n}\rfloor$, computing for each value of n the average, median, and standard deviation of $t = 20$ runs of each algorithm. Present the corresponding results in two plots:

   1. One plot with the average, and the average $\pm$ one standard deviation, as a function of n (both algorithms on the same plot)
   2. One plot with the median and the average, as a function of n (both algorithms on the same plot)

   Discuss the results: is there a difference between taking the average and the median? If so, which seems better? Which of the two algorithms seems better?

Advice on how to do the assignment

- Assignments should be typed and submitted as pdf (no pdf containing text as images, no handwriting). If you want to use LaTeX(encouraged, but not mandatory), a template is provided.
- Start by typing your student ID at the top of the first page of your submission. Do **not** type your name.
- Submit only your answers to the questions. Do **not** copy the questions.
- Indicate at the top which problems you are answering (cf. assignment rules at the beginning: you don't need to answer *all* problems to get full marks)
- Conciseness is appreciated. Brief accurate answers are better than long, vague ones: there is no marks for "throwing in the kitchen sink."
- Some of the questions are very easy (with the help of the slides or book). You can use the material presented in the lectures, tutorials, or lecture notes without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst-case running times, etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources and explain it in your own words. Only citing a source doesn't show your understanding and will thus get you very few (if any) marks. Copying from any source without reference is considered plagiarism.