

Lecture 1

Lecturer: Ronitt Rubinfeld

Scribe: Michael Ngo

Overview. In Section 1, we construct a 2-multiplicative approximation algorithm for estimating the diameter of a point set. In Section 2, we study estimation of the average degree of graphs: for graphs of n vertices, we provide a $\Omega(\sqrt{n})$ lower bound in run time in a standard setting (Section 2.2), and then construct and prove that an algorithm achieves it (Section 2.3).

1 Estimating Diameter of a Point Set

The problem of estimating the diameter of a point set is to estimate the max distance across all pairs of points. We do not necessarily imagine these points being in Euclidean space, but that there is some distance metric defined on them. Informally, a distance matrix D has entries such that D_{ij} is the distance between points i and j . We note that the distance matrix satisfies two important properties: (1) distances are symmetric ($D_{ij} = D_{ji}$), and (2) distances satisfy the triangle inequality ($D_{ij} + D_{jk} \geq D_{ik}$). More formally:

Definition 1 Suppose that there are $p \in \mathbb{Z}^{\geq 0}$ points. Then $D \in \mathbb{R}_{\geq 0}^{p \times p}$ is a distance matrix if it satisfies the following:

1. (Symmetry) For all $i, j \in [p]$ $D_{ij} = D_{ji}$;
2. (Triangle Inequality) For all $i, j, k \in [p]$, $D_{ij} + D_{jk} \geq D_{ik}$;
3. (Identity) For all $i \in [p]$, $D_{ii} = 0$.

For the exact version of this problem, the input is D , and the goal is to output the points (i, j) that maximize D_{ij} . The matrix D is $p \times p$ dimensional, so the total input size is $n = p^2$. Additionally, the algorithm is given random access memory to D . That means that the algorithm is allowed to look-up any entry of D in constant time.

The exact version is “like trying to find a needle in the haystack.” Instead we will provide an $\sqrt{n}$ -time algorithm that solves the 2-multiplicative approximation diameter problem. That is, it outputs points (k, ℓ) such that $D_{k\ell}$ is at least half of the max distance, or $D_{k\ell} \geq \frac{1}{2} \max_{i,j} D_{ij}$. This algorithm is deterministic and is simple enough to just state as is:

Algorithm 1 Two-Approximately Estimating Diameter

Pick a point k arbitrarily.
 Pick the point ℓ that maximizes $D_{k\ell}$.
 Output (k, ℓ) .

It’s that simple! Runtime analysis follows immediately and correctness follows via the triangle inequality.

Lemma 2 Algorithm 1 runs in time $O(\sqrt{n})$.

Proof In order to choose ℓ that maximizes $D_{k\ell}$, the algorithm maximizes over $p - 1$ choices of ℓ . This takes $O(p) = O(\sqrt{n})$ time. All other steps take constant time. ■

Lemma 3 If Algorithm 1 outputs (k, ℓ) , then $D_{k\ell} \geq \frac{1}{2} \max_{i,j} D_{ij}$.

Proof Let (i, j) be the points that maximize D_{ij} . In order to compare $D_{k\ell}$ and D_{ij} , we employ the triangle inequality on (i, k, j) . Then:

$$D_{ij} \leq D_{ki} + D_{kj}$$

Recall that ℓ was chosen to maximize $D_{k\ell}$, so $D_{ki} \leq D_{k\ell}$ and $D_{kj} \leq D_{k\ell}$. Therefore:

$$D_{ij} \leq D_{k\ell} + D_{k\ell}$$

And consequently, $D_{k\ell} \geq \frac{1}{2}D_{ij}$. ■

From these two lemmas, we have shown:

Theorem 4 *Algorithm 1 runs in $O(\sqrt{n})$ -time and 2-approximates the diameter of a point set.*

1.1 Can Deterministic Algorithms Do Any Better?

No, deterministic sublinear time algorithms cannot achieve anything better than a 2-approximation. This is shown via a counterexample where even if the algorithm reads $n - 1$ distances, it could still output half of the max distance. In other words, the one distance it doesn't read contains essential information about the diameter of the entire graph.

Theorem 5 *No sublinear time deterministic algorithm can achieve better than a 2-approximation for diameter estimation.*

Proof Choose any $p \in \mathbb{Z}^{\geq 0}$, and let $n = p^2$. Consider two distance matrices D and D' which are the same except for one entry:

- $D_{ij} = 1$ for all $i \neq j$ except that $D_{12} = 2$;
- $D'_{ij} = 1$ for all $i \neq j$ except that $D'_{12} = \frac{1}{2}$.

The reader can verify these are, in fact, distance matrices. Take any sublinear time algorithm. It can read at most $n - 1$ of the distances. Without loss of generality, assume that on either inputs, D or D' , it does not read the distance between $(1, 2)$. If the algorithm outputs $(1, 2)$, then it is only a 2-approximation on D' . Else if the algorithm outputs any other pair (i, j) , then it is only a 2-approximation on D . Thus, the algorithm is at best a 2-approximation algorithm. ■

2 Estimating Average Degree of a Graph

The goal of this problem is to estimate the average degree of a graph. Suppose we have a graph $G = (V, E)$ where there are n vertices, and m edges. Then its average degree is:

$$\bar{d} := \frac{1}{n} \sum_{u \in V} \deg(u) = \frac{2m}{n}.$$

The goal is to output $\tilde{d}$ that is within an additive factor of $\varepsilon \bar{d}$ of $\bar{d}$ for some approximation factor $\varepsilon \in (0, 1)$. More formally, we define the approximate average degree problem as follows:

Definition 6 *Given a graph $G = (V, E)$ where $|V| = n$ and $|E| = m$, an approximation parameter $\varepsilon \in (0, 1)$, and a confidence parameter $\delta \in (0, 1)$, the goal of the approximate average degree problem is to output $\tilde{d}$ such that*

$$\Pr \left[\left| \tilde{d} - \bar{d} \right| \leq \varepsilon \bar{d} \right] \geq 1 - \delta$$

Moreover we are interested in sublinear time algorithms that achieve this with the following input access. The algorithm has access to an adjacency list. In particular, it can query on the i -th node and receive random memory access to an array of all the neighbors of i . Additionally the algorithm has random access memory to an array of the degree of each node. This time, the input size is considered to be n , the number of vertices.

2.1 A Naive Algorithm

Let's go over a naive algorithm and see why it doesn't work. We construct an unbiased estimator for $\bar{d}$ by randomly sampling vertices and averaging their degree. More formally:

Algorithm 2 Random Sampling

```

 $s \leftarrow ?$  (an integer we may compute later)
for  $i \leftarrow 1, 2, 3, \dots, s$  do
    Pick a point  $v_i$  uniformly at random from  $V$ .
    Let  $x_i \leftarrow \deg(v_i)$ .
end for
Output  $\tilde{d} \leftarrow \frac{1}{s} \sum_{i=1}^s x_i$ .
```

Unfortunately, in order to solve this approximate average degree problem, the algorithm will need s that is linear in n , and thus run in linear time.

Lemma 7 *Algorithm 2 runs in time linear in n .*

Sketch of Proof In order to compute the runtime, we will choose the smallest s such that our estimate $\tilde{d}$ is $\varepsilon\bar{d}$ -accuracy with δ -confidence.¹ Take the example where the graph is a star. The first node is connected to all of the other $n - 1$ nodes, and there are no other edges. Then the max range over all pairs $\deg(v_i)$ is $O(n)$ and the average degree is $O(1)$. Therefore, Chernoff bounds (via Hoeffding's Inequality) yield:

$$\begin{aligned}
 s &\geq \frac{n}{\varepsilon\bar{d}} \sqrt{2 \log(2/\delta)} \\
 &\geq \Omega\left(\frac{n}{\varepsilon} \sqrt{\log(1/\delta)}\right)
 \end{aligned}$$

which is linear in n . Thus Algorithm 2 runs in linear time. ■

Remark Curiously, if the max range over $\deg(v_i)$ is bounded by a constant multiplicative factor—say $\deg(v_i) \in [\Delta, 10\Delta]$ for some constant Δ — then Chernoff bounds yield the following sample complexity:

$$s \geq \frac{50}{\varepsilon^2} \log(2/\delta).$$

This is independent of n , so in this bounded degree setting, the algorithm will in sublinear time, and in fact in near-constant time $O(\log(1/\delta)/\varepsilon^2)$. See the full proof in the handwritten notes.

¹Technically, we provide an upperbound on the smallest s , so this proof provides an upper bound on run time. But since Chernoff is almost asymptotically tight this provides very good evidence on the true minimum runtime possible.

2.2 General Lower Bounds on Runtime

We will show algorithms that solve approximate average degree require at least linear time. To show lower bounds on runtime, we will reduce to distinguishing between two families of graphs of differing average degree. If all sublinear time algorithms fail to distinguish the two families, then that shows that all sublinear time algorithms cannot solve approximate average degree.

Theorem 8 (Ultrasparse Lower Bound) *The approximate average degree problem requires at least $\Omega(n)$ time.*

Sketch of Proof Consider the empty graph versus the graph with exactly one edge, selected uniformly at random. They differ in average degree: 0 versus $1/n$. In order for any algorithm to distinguish between them, it must access all nodes of the graph. Since sublinear time algorithms cannot read all edges, they cannot solve approximate average degree—at least for sufficiently small $\delta > 0$. ■

This ultrasparse “needle-in-a-haystack” example is perhaps unfair for any reasonable algorithm: both graphs have average degree that tend to 0. The next lower bound is constructed from families whose average degrees are at least 2.

Theorem 9 (Average Degree Two Lower Bound) *Under the promise that $\bar{d} \geq 2$, the approximate average degree problem requires at least $\Omega(\sqrt{n})$ time.*

Sketch of Proof Fix a constant $c > 0$. Consider the following two families of graphs:

1. The graph where all n are connected into a single cycle.
2. The graph where $n - c\sqrt{n}$ of the nodes are connected into a single cycle, and the rest of the $c\sqrt{n}$ nodes form a clique.

Observe that the average degree of the first graph is 2 whereas the average degree of the second graph is roughly $2 + c^2$. Once again, we can argue how sublinear time algorithms cannot distinguish these families with access to only a subset of the nodes. An algorithm that randomly samples nodes until it finds the clique requires $\Omega(\sqrt{n})$ samples via a coupon-collecting argument. Other clever tricks like traversing through the neighbors doesn’t help to find the clique faster. Therefore, all algorithms that solve approximate average degree in this setting require $\Omega(\sqrt{n})$ run time. ■

2.3 Sublinear Time Algorithm in The Promise Setting

While we cannot achieve sublinear time algorithms in general (See Theorem 8), if we promise that $\bar{d} \geq 2$ and that the graph is simple, we will be able to construct a sublinear time algorithm that matches the lower bound in Theorem 9.

Redistributing Edges. The key problem from the naive algorithm (Algorithm 2) was that the max range of $\deg(v_i)$ was large, say $O(n)$. That range forces algorithms to “find the needle in the haystack.” We will define a new notion of degree for which the maximum degree is $O(\sqrt{m})$, thus reducing the max range to $O(\sqrt{m})$. Simply, we order the edges to point to high degree nodes and only count the outdegree, or more formally:

Definition 10 *Suppose we are given a set of n nodes, each with a unique integer ID denoted $ID(u)$ for node u . We define the total order $\preceq$ over the nodes such that for all nodes (u, v) , $u \preceq v$ if either*

- $\deg(u) < \deg(v)$; or
- $\deg(u) = \deg(v)$ and $ID(u) < ID(v)$.

Then we define the outdegree $\deg^+$ as for all nodes u , $\deg^+(u)$ is the number of neighbors v of u such that $u \propto v$.

Observe that summing $\deg^+$ recovers m , the number of edges, since all edges are counted exactly once. Therefore, we can note:

$$\sum_{u \in V} \deg^+(u) = \frac{n}{2} \cdot \bar{d}.$$

Thus, if we use the naive sampling algorithm, but querying $\deg^+$ instead of $\deg$ (See Algorithm 3), we will still have an unbiased estimator for $\bar{d}$.

Algorithm 3 Random Sampling With Degree Plus

Assumption: query access to $\deg^+$.
 $s \leftarrow ?$ (an integer we may compute later)
for $i \leftarrow 1, 2, 3, \dots, s$ **do**
 Pick a point v_i uniformly at random from V .
 Let $x_i \leftarrow \deg^+(v_i)$.
end for
Output $\tilde{d} \leftarrow 2 \cdot \frac{1}{s} \sum_{i=1}^s x_i$.

Now we show that $\deg^+$ is bounded above by $O(\sqrt{m})$. While m can be as large as n^2 , this means the edges are distributed pretty well. Think about there being about n edges. Then the largest outdegree is $O(\sqrt{n})$.

Lemma 11 For all u in V , $\deg^+(u) \leq \sqrt{2m}$.

Proof Let H be the $\sqrt{2m}$ nodes with the highest rank according to $\propto$. Then for all v in H , their outdegree can only be accounted for by higher rank nodes. There are $\sqrt{2m}$ of these “heavy” nodes, so $\deg^+(v) \leq \sqrt{2m}$.

Suppose for contradiction that for some $u \in V \setminus H$ that $\deg^+(u)$ is more than $\sqrt{2m}$. Then $\deg(u)$ is more than $\sqrt{2m}$. Then because all of H is ordered above u , every node in H must have degree of more than $\sqrt{2m}$. But that implies, the edges in H will account for more than $\frac{\sqrt{2m} \cdot \sqrt{2m}}{2} = m$ edges, and m is already supposed to be the total number of edges. This is a contradiction, so $\deg^+(u) \leq \sqrt{2m}$. ■

In general, computing $\deg^+$ exactly can take linear time. To get around this, we estimate $\deg^+$ using a random sampler.

Algorithm 4 A sublinear time algorithm

$s \leftarrow \frac{16}{\epsilon^2} \sqrt{n}$
for $i \leftarrow 1, 2, 3, \dots, s$ **do**
 Pick a node v_i uniformly at random from V .
 Pick a node u_i uniformly from the neighbors $N(v_i)$.
 If $v_i \propto u_i$, then $x_i \leftarrow \deg(v_i)$.
 Else $x_i \leftarrow 0$.
end for
Output $\tilde{d} \leftarrow 2 \cdot \frac{1}{s} \sum_{i=1}^s x_i$.

Over the choice of u_i , the expectation of x_i is $\deg^+(v_i)$. Then, it follows from our prior analysis that $\tilde{d}$ is an unbiased estimator for $\bar{d}$. We formally show that this is the case:

Claim 12 For all i , $\mathbb{E}[x_i] = \bar{d}/2$.

Proof We begin by unrolling the expectation using the law of total probability.

$$\begin{aligned}\mathbb{E}[x_i] &= \sum_{v \in V} \Pr[v_i = v] \cdot \mathbb{E}[x_i | v_i = v] \\ &= \sum_{v \in V} \Pr[v_i = v] \sum_{u \in N(v)} \Pr[u_i = u | v_i = v] \cdot \mathbb{E}[x_i | u_i = u \wedge v_i = v]\end{aligned}$$

Recall that v_i is selected uniformly from V , u_i uniformly from $N(v_i)$, and the expectation on x_i hides an indicator over the random variable over $v_i \propto u_i$.

$$\begin{aligned}\mathbb{E}[x_i] &= \sum_{v \in V} \frac{1}{n} \sum_{\substack{u \in N(v) \\ v \propto u}} \frac{1}{\deg(v)} \cdot \deg(v) \\ &= \frac{1}{n} \sum_{v \in V} \sum_{\substack{u \in N(v) \\ v \propto u}} 1 && \text{(linearity)} \\ &= \frac{1}{n} \sum_{v \in V} \deg^+(v) && \text{(definition of } \deg^+ \text{)} \\ &= \frac{m}{n} && \text{(prior observation)} \\ &= \bar{d}/2 && \text{(definition of } \bar{d} \text{)}\end{aligned}$$

■

But how many samples do we need to achieve a good enough approximation? First, we will show that the variance of x_i is bounded with the help of $\deg^+$ concentration lemma.

Claim 13 For all i , $\text{Var}[x_i] \leq \sqrt{2m} \cdot \bar{d}$.

Proof

$$\begin{aligned}\text{Var}[x_i] &= \mathbb{E}[x_i^2] - \mathbb{E}[x_i]^2 \\ &\leq \mathbb{E}[x_i^2]\end{aligned}$$

Here, we use similar analysis on $\mathbb{E}[x_i^2]$ as we did in Claim 12 .

$$\begin{aligned}\text{Var}[x_i] &\leq \sum_{v \in V} \frac{1}{n} \sum_{\substack{u \in N(v) \\ v \propto u}} \frac{1}{\deg(v)} \cdot \deg(v)^2 \\ &= \frac{1}{n} \sum_{v \in V} \sum_{\substack{u \in N(v) \\ v \propto u}} \deg(v) && \text{(linearity)} \\ &= \frac{1}{n} \sum_{v \in V} \deg^+(v) \deg(v) && \text{(definition of } \deg^+ \text{)} \\ &\leq \sqrt{2m} \cdot \frac{1}{n} \sum_{v \in V} \deg(v) && \text{(Lemma 11)} \\ &= \sqrt{2m} \cdot \bar{d}\end{aligned}$$

■

Now for the confidence $\delta = 1/4$, we show a good approximation.

Lemma 14 $\Pr[|\tilde{d} - \bar{d}| \leq \varepsilon \bar{d}] \geq 3/4$.

Proof Recall that $\tilde{d} = 2 \cdot \frac{1}{s} \sum_{i=1}^s x_i$.

- By Claim 12, $\mathbb{E}[\tilde{d}] = \bar{d}$.
- By pairwise independence of the x_i 's and Claim 13, $\text{Var}[\tilde{d}] = \frac{4}{s} \cdot \text{Var}[x_i] \leq \frac{4\sqrt{2m}}{s} \bar{d}$.

Then by applying Chebyshev's Inequality, we have the following:

$$\begin{aligned}
\Pr[|\tilde{d} - \bar{d}| > \varepsilon \bar{d}] &= \Pr[|\tilde{d} - \mathbb{E}[\tilde{d}]| > \varepsilon \bar{d}] \\
&\leq \frac{\text{Var}[\tilde{d}]}{\varepsilon^2 \bar{d}^2} && \text{(Chebyshev's Inequality)} \\
&\leq \frac{4\sqrt{2m} \cdot \bar{d}}{\varepsilon^2 \bar{d}^2 \cdot s} \\
&= \frac{4\sqrt{2m}}{\varepsilon^2 \bar{d} \cdot s} \\
&= \frac{2\sqrt{2}}{\varepsilon^2 \sqrt{m} \cdot s} && (\bar{d} = \frac{2m}{n}) \\
&= \frac{\sqrt{n}}{4\sqrt{2m}} && (\text{choice of } s = \frac{16}{\varepsilon^2} \sqrt{n}) \\
&= \frac{1}{4} \cdot \sqrt{\frac{1}{\bar{d}}} \\
&< \frac{1}{4} && (\text{assumption that } \bar{d} \geq 2)
\end{aligned}$$

■

To complete this, one can amplify the probability of success from $3/4$ to $1 - \delta$ via Chernoff bounds. See Homework 0 for examples! Altogether we have the following.

Theorem 15 *Under the promise that $\bar{d} \geq 2$ and the graph is simple, Algorithm 4 is a $O(\sqrt{n})$ -time algorithm which solves approximate average degree with confidence $\delta = 1/4$ (and arbitrary confidence with some modification).*