6.889 Sublinear Time Algorithms February 6, 2019

Lecture 1

Lecturer: Ronitt Rubinfeld *Scribe: Sandeep Silwal*

1 Outline

The following topics were covered in class:

- Overview of the course.
- Diameter of a point set.
- Estimating the number of connected components of a graph.

2 Overview of the course

The high-level goal of sublinear time algorithms is to answer questions without reading or processing the entire input. Therefore, we have to make a compromise which usually means introducing an error parameter along with a probability that your approximation holds. Hence, algorithms in this field are usually randomized. In addition to randomized algorithms, the field of sublinear time algorithms also intersects with approximation algorithms, parallel algorithms, distributed algorithms, statistics and many other fields.

The complexity of algorithms is measured in terms of the number of queries that we request from the input data, such as querying an adjacency matrix or asking for a sample from a distribution. This general framework can be applied to a variety of different types of inputs. Some examples of inputs and the properties of these inputs that we can ask questions about are shown in Table 1. We will cover most of the things in Table 1 in due time.

Type of input	**Properties that we can ask questions about**
Graphs	Diameter, # of connected components, MST, bipartiteness, clusterability, $\cdots$
Functions	Monotonicity, convexity, linearity, juntas, $\cdots$
Distributions	Uniformity, independence, entropy, monotonicity, $\cdots$

Table 1: Topics that we might cover in the course.

Let's begin by discussion the only non randomized algorithm in this course.

3 Diameter of a point set

Consider m points in some metric space whose pairwise distances are given in the matrix $\mathbf{D}$. That is, $\mathbf{D}_{ij}$ is the distance between point i and point j. Since we are in a metric space, we assume the following things about $\mathbf{D}$:

- $\mathbf{D}$ is symmetric: $\mathbf{D}_{ij} = \mathbf{D}_{ji}$ for all i, j.
- $\mathbf{D}$ satisfies the triangle inequality: $\mathbf{D}_{ij} \leq \mathbf{D}_{ik} + \mathbf{D}_{kj}$ for all i, j, k.

Our goal is to find the diameter of this point set, that is, we want to find the maximum entry in the matrix $\mathbf{D}$. Moreover, we want to find the points that achieve this diameter. We won't be able to do this exactly but we will instead give a *2-approximation* by using the fact that our points lie in a metric space. Our algorithm is the following.

Algorithm 1: Diameter-Estimator

Input : m points in a metric space, matrix $\mathbf{D}$ of all the pairwise distances.
Output: Points k, ℓ and distance $\mathbf{D}_{k,\ell}$.

1 Pick k arbitrarily from $\{1, \cdots, m\}$;
2 Pick ℓ that maximizes $\mathbf{D}_{k,\ell}$;
3 Return $k, \ell, \mathbf{D}_{k,\ell}$

Theorem 1 ***Diameter-Estimator*** *returns a* 2*-approximation to the actual diameter.*

Proof Consider the points k and ℓ outputted by **Diameter-Estimator**. Suppose that the actual diameter is achieved by points i and j. Then by the triangle inequality, we have

$$\mathbf{D}_{ij} \leq \mathbf{D}_{ik} + \mathbf{D}_{kj} = \mathbf{D}_{ki} + \mathbf{D}_{kj} \leq 2\mathbf{D}_{k,\ell},$$

as desired. ■

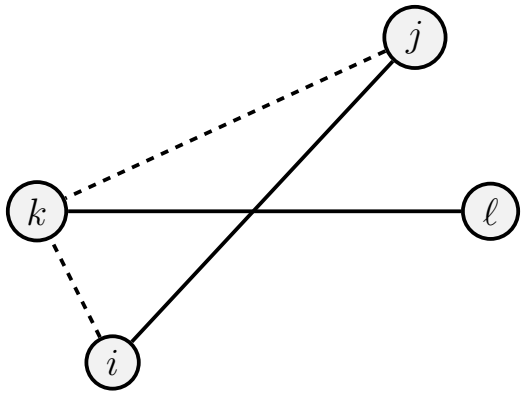

Figure 1: We use the triangle inequality on the points k, i, j in the proof of Theorem 1.

Note that the size of the input is $n = m^2$ since $\mathbf{D}$ has m^2 entries. Furthermore, the query complexity of **Diameter-Estimator** is $O(m)$ since we examine one row of $\mathbf{D}$. Therefore, the complexity of our algorithm is $O(\sqrt{n})$, which is sublinear in the input size.

4 Number of connected components

We now move onto our second topic which is to estimate the number of connected components of a graph. Given an input $G = (V, E)$ with $|V| = n$ and an error parameter $\epsilon > 0$, our goal is to output an estimate for c, the number of connected components of G. Our complexity will be measured in terms of the number of queries that we make to the graph.

We will give an algorithm that outputs an estimate $\widetilde{c}$, and we will show that we get an additive error of at most ϵn, that is, $|c - \widetilde{c}| \leq \epsilon n$ (with probability at least $\frac{3}{4}$). To motivate the algorithm, we first introduce a new notation.

Definition 2 *For $v \in V$, define n_v to be the number of nodes in the connected component of v.*

The key observation is if $A \subseteq V$ is a connected component, we have $\sum_{v \in A} \frac{1}{n_v} = 1$. Thus,

$$\sum_{v \in V} \frac{1}{n_v} = \#\text{ of connected components} = c. \tag{1}$$

Therefore, we can use Eq. (1) to calculate c. Our strategy is to first get a good approximation for n_v for a fixed v. The guiding principle behind this estimate will be that if n_v is small, the connected component that v belongs to is small and therefore we will incur small query complexity to just search over the entire connected component of v.

On the other hand, if n_v is large, we know $\frac{1}{n_v}$ is small so we will incur a small error by approximating n_v. We then approximate $\sum_{v \in V} \frac{1}{n_v}$ using our approximation for n_v. Our guiding principle here will be that we can sample vertices at random to approximate the average value of $\frac{1}{n_v}$. Towards this goal, we make the following definition.

Definition 3 $\widehat{n}_v = \min\{n_v, 2/\epsilon\}$.

Again, the intuition behind this definition is that we want to separate the cases that n_v is small versus large. We now show that this estimate incurs us a very small loss in Eq. (1).

Lemma 4 *The following estimate holds for all vertices $v \in V$:*

$$\frac{1}{\hat{n}_v} - \frac{1}{n_v} \leq \frac{\epsilon}{2}.$$

Proof If $n_v \leq \frac{2}{\epsilon}$, we have $\widehat{n}_v = n_v$ so the claim immediately follows. Otherwise, $n_v > \frac{2}{\epsilon}$ and $\widehat{n}_v = \frac{2}{\epsilon}$ and

$$0 \leq \frac{1}{n_v} \leq \frac{1}{\widehat{n}_v} = \frac{\epsilon}{2}$$

so the conclusion also follows. ■

We formalize our heuristics and present an algorithm to compute $\widehat{n}_v$.

Algorithm 2: $\widehat{n}_v$-Calculator

Input : Graph G, vertex v, ϵ
Output: $\widehat{n}_v$.

```
Initialize Breadth-first search (BFS) from v ;
while # of unique visited nodes by BFS is < 2/ε do
    Continue BFS ;
    if BFS finishes then
        Return number of visited nodes and abort
Return 2/ε
```

Note that the query complexity of this algorithm is $O(1/\epsilon^2)$. This is because we visit at most $O(1/\epsilon)$ unique vertices and for each of them, we visit at most $O(1/\epsilon)$ of their neighbors (some of the neighbors might be repeated, think of a clique of size $O(1/\epsilon)$).

We now proceed to estimate the *sum* in Eq. (1). We first define a similar quantity

$$\hat{c} = \sum_{v \in V} \frac{1}{\widehat{n}_v}. \tag{2}$$

From Lemma 4, we know that computing $\hat{c}$ gives us a desirable approximation for c, the true number of connected components. However, simply estimating $\widehat{n}_v$ for each $v \in V$ is too costly (this is sublinear time algorithms after all) so we must do something clever. Fortunately, we don't have to be too clever since computing $\hat{c}$ is the same as computing the average value of $1/\widehat{n}_v$. This last task can be easily handled by standard concentration bounds. Therefore, our task is to now find an 'good' approximation for $\hat{c}$, which will be denoted as $\tilde{c}$, and use Lemma 4 to argue that we also have a good estimate for c.

Algorithm 3: $\tilde{c}$**-Calculator**

Input : Graph G, ϵ, b
Output: $\tilde{c}$.

$r \leftarrow b/\epsilon^3$;
Sample r vertices $v_1, \cdots, v_r$ from G uniformly with replacement ;
Compute $\widehat{n}_{v_i}$ for all $1 \leq i \leq r$ using $\widehat{n}_v$**-Calculator** ;
Return $\tilde{c} = \frac{n}{r}\left(\sum_{i=1}^{r} 1/\widehat{n}_{v_i}\right)$

We present an algorithm to calculate $\tilde{c}$ in Algorithm 3. Note that from the discussion from Algorithm 2, the query complexity of Algorithm 3 is $O(1/\epsilon^5)$. We proceed to show that $\tilde{c}$ from Algorithm 3 is a 'good' approximation to $\hat{c}$.

Lemma 5 *Let $\tilde{c}$ be the output of Algorithm 3 (for some constant b to be specified later). Then*

$$\mathbb{P}[|\hat{c} - \tilde{c}| \geq \epsilon n/2] \leq \frac{1}{4}.$$

We need the following variant of Chernoff bounds to prove Lemma 5.

Theorem 6 (Chernoff) *Let $X_1, \cdots, X_r$ be i.i.d. random variables in the unit interval. Define $S = \sum_{i=1}^{r} X_i$ and $p = \mathbb{E}[X_i]$. Then,*

$$\mathbb{P}\left[\left|\frac{S}{r} - p\right| \geq \delta p\right] \leq \exp\left(-\Omega(rp\delta^2)\right).$$

We now prove Lemma 5.

Proof Consider the vertices $v_1, \cdots, v_r$ in Algorithm 3 that are sampled uniformly from G and let $\tilde{c}$ be the output of Algorithm 3. Recall Eq. (2)

$$\hat{c} = \sum_{v \in V} \frac{1}{\widehat{n}_v}.$$

We wish to upper bound $\mathbb{P}[|\hat{c} - \tilde{c}| \geq \epsilon n/2]$. Note that $\hat{c} \leq n$ so

$$\mathbb{P}[|\hat{c} - \tilde{c}| \geq \epsilon n/2] \leq \mathbb{P}[|\hat{c} - \tilde{c}| \geq \epsilon \hat{c}/2] = \mathbb{P}\left[\left|\frac{\hat{c}}{n} - \frac{\tilde{c}}{n}\right| \geq \frac{\epsilon \hat{c}}{2n}\right].$$

Now let $X_i = 1/\widehat{n}_{v_i}$. It follows that

$$\mathbb{E}[X_i] = \frac{1}{n} \sum_{i=1}^{r} \frac{1}{\widehat{n}_{v_i}} = \frac{\hat{c}}{n}.$$

Furthermore,

$$\frac{1}{r} \sum_{i=1}^{r} X_i = \frac{1}{r} \sum_{i=1}^{r} \frac{1}{\widehat{n}_{v_i}} = \frac{\tilde{c}}{n}.$$

Now let $\delta = \frac{\epsilon}{2}$. The Chernoff bound gives us

$$\mathbb{P}\left[\left|\frac{\hat{c}}{n} - \frac{\tilde{c}}{n}\right| \geq \frac{\epsilon \hat{c}}{2n}\right] \leq \exp(-\Omega(\hat{c}/\epsilon n)).$$

Since

$$\frac{\epsilon n}{2} \leq \sum_{v \in V} \frac{1}{\widehat{n}_v} = \hat{c}$$

by definition, it follows that by picking a suitably large constant for b, we have

$$\mathbb{P}[|\hat{c} - \tilde{c}| \geq \epsilon n/2] \leq \frac{1}{4},$$

as desired. ■

We now combine Lemmas 4 and 5 to prove the main theorem of this section.

Theorem 7 *Let c be the number of connected components of G and let $\tilde{c}$ be the output of Algorithm 3. Then,*

$$\mathbb{P}[|c - \tilde{c}| \leq \epsilon n] \geq \frac{3}{4}.$$

Proof By the triangle inequality,

$$|c - \tilde{c}| \leq |\hat{c} - c| + |\hat{c} - \tilde{c}|.$$

By Lemma 4,

$$|\hat{c} - c| = \sum_{v \in V} \frac{1}{\hat{n}_v} - \frac{1}{n_v} \leq \frac{\epsilon n}{2}.$$

Furthermore, by Lemma 5, $|\hat{c} - \tilde{c}| \leq \frac{\epsilon n}{2}$ with probability at least $\frac{3}{4}$. Hence with probability at least $\frac{3}{4}$, we have

$$|c - \tilde{c}| \leq \frac{\epsilon n}{2} + \frac{\epsilon n}{2} = \epsilon n,$$

as desired. ■

Theorem 7 tells us that our estimate for the number of connected components is off by an additive factor of at most ϵn. Interestingly, our overall query complexity is $O(1/\epsilon^5)$ (which can be optimized slightly) which has no dependence on n.