1 Outline

The following topics were addressed during the first lecture.

- Overview of the Course/Sublinear Algorithms
- Diameter of a Point Set
- Number of Connected Components in a Graph

We refer the reader to the slides on the course homepage for the first item.

2 Diameter of a Point Set

Our first example of a sublinear algorithm (due to Piotr Indyk) will be computing a 2-approximation to the diameter of a point set in sublinear time. This algorithm has the unique property of being the only deterministic algorithm in this class.

Input - We are given $m$ points described by a distance matrix $D$ such that $D_{i,j}$ is the distance from $i$ to $j$. Furthermore we are guaranteed that the distances satisfy

- (Symmetry) $D_{i,j} = D_{j,i}$ for all $i,j \in [m]$
- (Triangle Inequality) $D_{i,j} \leq D_{i,k} + D_{k,j}$ for all $i,j,k \in [m]$

Note here that the input size is $n = \Theta(m^2)$ as we are given all pairs of distances.

Output - Let the diameter $D = \max_{i,j} D_{i,j}$. Our output is a pair $(k,\ell)$ such that $D_{k,\ell} \geq D/2$ (e.g. a 2-approximation to the diameter).

Algorithm - Choose an arbitrary index $k$. Output $(k,\ell)$ such that $D_{k,\ell}$ is maximized. (The pseudocode for the algorithm is given below.)

Algorithm 1: Diameter-Estimator

1. Pick $k$ arbitrarily from $\{1, \cdots, m\}$;
2. $\ell = \arg\max_j D_{k,j}$;
3. Return $k, \ell, D_{k,\ell}$

Running Time - Note that we read only $O(m) = O(\sqrt{n})$ entries of the distance matrix $D$.

Correctness - Let $D = D_{i,j}$. Now note that

$$D_{i,j} \leq D_{i,k} + D_{k,j}$$  \[\text{Triangle Inequality}\]
$$D_{i,j} \leq D_{k,i} + D_{k,j}$$  \[\text{Symmetry}\]
$$D_{i,j} \leq D_{k,t} + D_{k,\ell}$$  \[\text{Definition of } \ell\]
$$D_{i,j} = D_{k,\ell}.$$  

The desired result follows immediately.

Lower Bound - We now sketch an argument that any $(2-\delta)$ approximation to the diameter requires reading the entire matrix $D$. (This answers a question raised by one of the students in class.)

- Define the distance matrix $M$ to have $M_{i,i} = 0$ and $M_{i,j} = 1$ otherwise.
• Define the distance matrix \( N_{i,j} \) to be identical to \( M \) except \( N_{i,j} = N_{j,i} = (2 - \delta) \).

• It can easily be checked that \( M, N \) satisfy the triangle inequality and symmetry. Furthermore, even if one is given the promise that the distance matrix \( D \) is one of the \( \left( \frac{m^2}{2} \right) + 1 \) examples given it takes take \( \Theta(m^2) \) time to tell if any of the entries is larger than 1 giving the desired lower bound as \( N \) has diameter \( 2 - \delta \) while \( M \) has diameter 1.

3 Number of Connected Components in a Graph

Our second example of a (randomized) sublinear time algorithm that will be an \( \varepsilon n \)-approximation the the diameter of an input graph \( G \) in time \( \text{poly}(1/\varepsilon) \).

**Input** - We are given \( G = (V,E) \) in an adjacency list representation. As is standard, we will let \( n = |V| \) and \( m = |E| \).

**Output** - Let \( C \) denote the number of connected components. We will output \( \hat{C} \) such \( |C - \hat{C}| \leq \varepsilon n \) with probability 3/4.

The first key insight we will need is an alternate characterization of the number of connected components of a graph \( G \).

**Lemma 1** Fix a graph \( G = (V,E) \). For a vertex \( v \in V \), let \( n_v \) denote the number of vertices in the connected component of \( v \) and let \( C \) be the total number of connected components. Then we have that

\[
C = \sum_{v \in V} \frac{1}{n_v}.
\]

**Proof** By splitting \( G \) into connected components, it suffices to prove the claim for a graph \( G \) which is connected. However, in this case, note that \( n_v = |V| \) and therefore

\[
\sum_{v \in V} \frac{1}{n_v} = |V| \left( \frac{1}{|V|} \right) = 1
\]

as desired.

One naive attempt given this characterization is to simply sample small number of vertices \( v \) at random from the graph \( G \), compute \( n_v \) for each sampled vertex, and output a the average of \( 1/n_v \) over the vertices sampled. However, there is a large issue in that computing \( n_v \) already is already takes linear time! The second insight therefore is to realize that if \( n_v \) is large, \( 1/n_v \) is small and therefore we do not need to compute \( n_v \) as precisely.

**Lemma 2** Let

\[
\hat{n_v} = \min(n_v, 2/\varepsilon).
\]

We have that

\[
\left| \sum_{v \in V} \frac{1}{n_v} - \sum_{v \in V} \frac{1}{\hat{n_v}} \right| \leq \frac{\varepsilon n}{2}
\]

and that for a given vertex \( v \), \( \hat{n_v} \) can be computed in \( O(1/\varepsilon^2) \) time.

**Proof** We first prove that

\[
\left| \frac{1}{n_v} - \frac{1}{\hat{n_v}} \right| \leq \frac{\varepsilon}{2}.
\]

the first claim then follows by noting that by triangle inequality

\[
\left| \sum_{v \in V} \frac{1}{n_v} - \sum_{v \in V} \frac{1}{\hat{n_v}} \right| \leq \sum_{v \in V} \left| \frac{1}{n_v} - \frac{1}{\hat{n_v}} \right| \leq n \cdot \frac{\varepsilon}{2}.
\]
To prove that
\[ \left| \frac{1}{\n_v} - \frac{1}{\hat{n}_v} \right| \leq \frac{\varepsilon}{2} \]
we split into cases based on the size of \( n_v \).

- If \( n_v \leq \frac{2}{\varepsilon} \), we are done immediately as \( n_v = \hat{n}_v \).
- If \( n_v \geq \frac{2}{\varepsilon} \), note that \( \frac{1}{n_v} \leq \frac{1}{\hat{n}_v} \) and \( \hat{n}_v = \frac{2}{\varepsilon} \) and therefore
  \[ \left| \frac{1}{n_v} - \frac{1}{\hat{n}_v} \right| = \frac{1}{n_v} - \frac{1}{\hat{n}_v} \leq \frac{1}{\hat{n}_v} = \frac{2}{\varepsilon}. \]

Now in order to compute the \( \hat{n}_v \) in \( \Theta(1/\varepsilon^2) \) time we simply run BFS starting at the vertex \( v \) and output the number of vertices in the corresponding component, short-cutting if we ever have processed more than \( 2/\varepsilon \) vertices. Note that if the connected component of \( v \) is less than \( 2/\varepsilon \) vertices we will read the entire component in \( O(1/\varepsilon^2) \)-time and thus we are able to compute \( n_v \) and thus \( \hat{n}_v \) exactly. Otherwise we have \( n_v \geq \frac{2}{\varepsilon} \) and the BFS will short-circuit after reading \( 2/\varepsilon \) vertices and we will compute (correctly) that \( \hat{n}_v = \frac{2}{\varepsilon} \). For the running time in this case note that we only process \( 2/\varepsilon \)-vertices and for each vertex we only process at most \( 2/\varepsilon \) vertices in total (as otherwise we can short-circuit).

Given the above we are now in position to state our algorithm.

**Algorithm** - Choose \( s = \Theta(1/\varepsilon^2) \) vertices \( v_1, \ldots, v_s \) uniformly at random from the vertices of \( G \). Compute \( \hat{n}_{v_i} \) for \( i \in [s] \) and return
\[ \hat{C} := \frac{n}{s} \left( \sum_{i \in [s]} \frac{1}{\hat{n}_{v_i}} \right). \]
(The psuedocode for the algorithm is given below.)

### Algorithm 2: Connected Components-Estimator

1. \( \text{sum} \leftarrow 0; \)
2. \( \text{for} \ 1 \leq i \leq s \ \text{do} \)
   3. \( \quad \)Sample \( v_i \) uniformly from \( V; \)
   4. \( \quad \)\( \text{sum} \leftarrow \text{sum} + 1/\hat{n}_{v_i}; \)
5. \( \hat{C} \leftarrow \frac{n}{s}(\text{sum}) \ \text{return} \hat{C} \)

**Running Time** - The running time is dominated by computing \( \hat{n}_{v_i} \) for sampled vertices \( n_{v_i} \). There are \( \Theta(1/\varepsilon^2) \) vertices and each run takes \( \Theta(1/\varepsilon^2) \)-times giving a total running time of \( \Theta(1/\varepsilon^4) \).

**Correctness** - In order to prove correctness it essentially suffices by Lemma 2 to prove that
\[ \frac{1}{s} \sum_{i \in [s]} \frac{1}{\hat{n}_v} \approx \frac{1}{n} \sum_{v \in V} \frac{1}{\hat{n}_v}; \]
the key tool here will be Chernoff bounds.

**Theorem 3 (Chernoff Bounds)** Fix \( \delta \in [0, 1] \). Let \( X_i \) be iid random variables in \([0, 1]\) with \( p = \mathbb{E}[X_i] \). Let \( X = \sum_{i=1}^n X_i \) and \( \mu = \mathbb{E}[X] = rp \). Then
\[ \mathbb{P}[|X - \mu| \geq \delta \mu] = \mathbb{P}[|X - rp| \geq \delta rp] \leq \exp(-\Theta(\delta^2 rp)). \]
**Theorem 4** Let $C$ be the number of connected components of $G$. The output of Algorithm 2, $\hat{C}$, satisfies that
\[
P\left[|C - \hat{C}| \geq \varepsilon n\right] \leq \frac{1}{4}.
\]

**Proof** By the first part of Lemma 2 and triangle inequality it suffices to prove that
\[
P\left[\left|\sum_{v \in V} \frac{1}{n_v} - \hat{C}\right| \geq \frac{\varepsilon n}{2}\right] \leq \frac{1}{4}.
\]
Note by definition that
\[
\hat{C} = \frac{n}{s} \left( \sum_{i \in [s]} \frac{1}{n_{v_i}} \right)
\]
and therefore the desired claim is equivalent to
\[
P\left[\left|\sum_{v \in V} \frac{1}{n_v} - \frac{n}{s} \left( \sum_{i \in [s]} \frac{1}{n_{v_i}} \right)\right| \geq \frac{\varepsilon n}{2}\right] \leq \frac{1}{4}.
\]
This is equivalent to the expression
\[
P\left[\left|\left( \sum_{i \in [s]} \frac{1}{n_{v_i}} \right) - \frac{s}{n} \sum_{v \in V} \frac{1}{n_v}\right| \geq \frac{\varepsilon s}{2}\right] = P\left[\left|\left( \sum_{i \in [s]} \frac{1}{n_{v_i}} \right) - \mathbb{E}\left[\sum_{i \in [s]} \frac{1}{n_{v_i}}\right]\right| \geq \frac{\varepsilon s}{2}\right] \leq \frac{1}{4}.
\]
Note that we have simply applied linearity of expectation at this stage. This is precisely the setup for Chernoff-bounds and now it is simply a matter of picking parameters appropriately.

First note that expected summand is at least $\varepsilon/2$ as we always have $1/n_v \geq \varepsilon/2$ and thus $p \geq \varepsilon/2$. Now choosing $\delta = \frac{\varepsilon}{4p} \leq 1$ we find that
\[
P\left[\left|\left( \sum_{i \in [s]} \frac{1}{n_{v_i}} \right) - \mathbb{E}\left[\sum_{i \in [s]} \frac{1}{n_{v_i}}\right]\right| \geq \frac{\varepsilon s}{2}\right] \leq \exp(-\Theta(\delta^2 sp)) = \exp(-\Theta(\varepsilon^2 s/(4p))) \leq \exp(-\Theta(\varepsilon^2 s)).
\]
Note in the final step we have used that $p \leq 1$ which follows as $1/n_v \in [0, 1]$. Thus taking $s$ a sufficiently large multiple of $\Theta(1/\varepsilon^2)$ the result follows. $\blacksquare$