These lecture notes have not undergone rigorous peer-review. Please email quanquan.liu@yale.edu if you see any errors.

1 Introduction

Having divided into examples of uses of the median trick in streaming algorithms, now we'll see various applications of our techniques in the sublinear model in the next few lectures. We'll start with the seminar result of Chazelle, Rubinfeld, and Trevisan '05 [CRT05] for estimating the number of connected components in a graph in sublinear time. Estimating the number of connected components in a graph is a fundamental problem in graph theory and network analysis. In the sublinear model, we aim to estimate this quantity without examining the entire graph, which is particularly useful when dealing with large-scale networks.

2 Sublinear Time Algorithms for Graphs

Sublinear algorithms are algorithms that run in time substantially less than the input size. They are particularly useful for large-scale problems where it's computationally expensive or even impossible to examine the entire input.

2.1 Query Model for Graph Problems

When designing sublinear time algorithms, specifying the exact data model, or rather the query model, is very important as the algorithm cannot even read the entire input once. A query model then specifies what type of queries can be made to the input. In the context of graph problems, we typically work with one of the following models: adjacency list model, adjacency matrix model, or the general query model. See the PowerPoint presentation included with this lecture for details about these models. In this lecture, we'll specifically focus on the adjacency list query model.

3 Estimating the Number of Connected Components

We first formally give the result we will prove in this lecture.

Theorem 1. *In the adjacency list query model, given a graph $G = (V, E)$, approximation parameter $\varepsilon \in (0, 1)$ (typically constant), and failure probability parameter $\delta \in (0, 1)$, output an approximation of the number of connected components $\tilde{C}$ such that*

$$\mathsf{Pr}\left(|\tilde{C} - C| \leq \varepsilon n\right) \geq 1 - \delta$$

where C is the actual number of connected components in the graph.

A Few Notes This is an *additive* approximation with respect to n, not a multiplicative approximation with respect to C (when C is small). This is because event distinguishing between two connected components (i.e. whether the entire graph is connected) better than a 2-approximation requires $\Omega(n^2)$ time.

3.1 Algorithm

First, let n_u denote the number of nodes in the connected component containing u.

Lemma 3.1. *Observe the following:*

- *If A is a connected component, then*

$$\sum_{u \in A} \frac{1}{n_u} = \frac{|A|}{|A|} = 1.$$

- *Then, knowing 1,*

$$\sum_{u \in V} \frac{1}{n_u} = \sum_{i=1}^{C} \sum_{u \in D_i} \frac{1}{n_u} = \sum_{i=1}^{C} 1 = C.$$

Using Lemma 3.1, we can first attempt the following algorithm: just calculate n_u for all $u \in V$. However, this obviously takes too much time as BFS/DFS takes too much time when n_u is large. (Recall we want a $o(n)$ sublinear algorithm). However, we make the following very important intuitive observation using the above equations: a large n_u *does not contribute to the summation much*. Hence, we can afford to *lower bound their size*. Now, let's do just that.

Strategy: We sample some vertices S (IID with replacement) and approximate the n_u values of the vertices we sampled. Let's denote $\hat{n}_u$ as the estimated value of the size of the component containing u. Then, for each sampled sampled $u \in S$, we set

$$\hat{n}_u = \min\{n_u, \frac{2}{\varepsilon}\}.$$

We first show a few characteristics of why it is fine to set $\hat{n}_u$ in this way. First, let $\hat{C} = \sum_{u \in V} \frac{1}{\hat{n}_u}$.

Lemma 3.2. $|\hat{C} - C| \leq \frac{\varepsilon n}{2}$.

Proof. For each $u \in V$, $0 \leq \frac{1}{\hat{n}_u} - \frac{1}{n_u} \leq \frac{\varepsilon}{2}$. This inequality holds since either:

- $n_u > \frac{1}{\varepsilon}$, we set $\hat{n}_u = \frac{2}{\varepsilon}$, and $\frac{1}{\hat{n}_u} - \frac{1}{n_u} \leq \frac{\varepsilon}{2} - \frac{1}{n} < \frac{\varepsilon}{2}$, or
- $\hat{n}_u = n_u$.

Thus, $0 \leq \sum_{u \in V} \frac{1}{\hat{n}_u} - \frac{1}{n_u} = \hat{C} - C \leq \sum_{u \in V} \frac{\varepsilon}{2} = \frac{n\varepsilon}{2}$. □

This provides the estimate we want; however, we cannot afford to estimate all $\hat{n}_u$ in sublinear time. Hence, as mentioned earlier, we must do some sampling.

Runtime of Each Sample The runtime of each sample is $O\left(\frac{1}{\varepsilon}\right)$ time per vertex using BFS since we perform BFS until we either see $\frac{2}{\varepsilon}$ unique vertices or we have seen all of the vertices in the component. This means that the maximum number of edges we traverse before we stop the BFS is $\frac{4}{\varepsilon^2}$.

Before we introduce the sampling based algorithm below, we make one additional observation. Notice that $\frac{\hat{C}}{n}$ is the *average* of the $\frac{1}{\hat{n}_u}$ values over all nodes $u \in V$. However, estimating the *average* is enough to get $\hat{C}$ since n is fixed and known.

Now, we give the algorithm in Algorithm 1.

```
Data: A graph $G = (V, E)$, a parameter $\varepsilon > 0$
Result: An estimate $\tilde{C}$ of the number of connected components of $G$
$S \leftarrow \frac{8}{\varepsilon^2}$.
for $i \leftarrow 1$ to $S$ do
    $u_i \leftarrow$ a random node from $V$
    $\hat{n}_{u_i} \leftarrow$ the number of nodes visited by BFS from $u_i$, limited to $\frac{2}{\varepsilon}$.
end
$\tilde{C} \leftarrow n \cdot \left(\frac{1}{S}\sum_{i=1}^{S}\frac{1}{\hat{n}_{u_i}}\right)$.
return $\tilde{C}$.
```
Algorithm 1: Pseudocode for estimating the number of connected components

Runtime For each node, BFS takes $O\left(\frac{1}{\varepsilon^2}\right)$ time and we perform $O\left(\frac{1}{\varepsilon^2}\right)$ queries. Hence, we take $O\left(\frac{1}{\varepsilon^4}\right)$ time. When, ε is constant, this sampling scheme is completely independent of the input!

Now, we show the concentration bounds.

Lemma 3.3. $\mathbb{E}[\tilde{C}] = \hat{C}$.

Proof.

$$\begin{aligned}
\mathbb{E}\left[\tilde{C}\right] &= \mathbb{E}[\frac{n}{S}\sum_{i=1}^{S}\frac{1}{\hat{n}_{u_i}}] \\
&= \frac{n\varepsilon^2}{8}\cdot\sum_{i=1}^{S}\mathbb{E}\left[\frac{1}{\hat{n}_{u_i}}\right] \\
&= \frac{n\varepsilon^2}{8}\cdot\frac{8}{\varepsilon^2}\cdot\sum_{u\in V}\mathsf{Pr}(\text{sample vertex } u)\cdot\frac{1}{\hat{n}_u} \\
&= \frac{n\varepsilon^2}{8}\cdot\frac{8}{\varepsilon^2}\cdot\frac{1}{n}\cdot\sum_{u\in V}\frac{1}{\hat{n}_u} \\
&= \hat{C}
\end{aligned}$$

The last line follows by definition of $\hat{C}$. □

Now, we can use the following form of the Chernoff bound to show our concentration bounds. Note that *unlike previous lectures*, we can directly use the additive Chernoff bound instead of going through Chebyshev's because our random variables have values in $[0, 1]$.

Theorem 2 (Additive Chernoff Bound). *Let $Y_1, Y_2, \dots, Y_k$ be k independent random variables with values in $[0, 1]$ and $Y = \sum_{i=1}^{k} Y_i$. Then, for any $b \geq 1$,*

$$\mathsf{Pr}(|Y - \mathbb{E}[Y]| > b) \leq 2\cdot e^{-2b^2/k}.$$

Lemma 3.4. $\mathsf{Pr}\left(|\tilde{C} - \hat{C}| \leq \frac{\varepsilon n}{2}\right) \geq \frac{7}{8}$.

Proof. Let $Y_i = \frac{1}{\hat{n}_{u_i}}$, then $Y = \sum_{i=1}^{S}\frac{1}{\hat{n}_{u_i}}$. Hence, $\tilde{C} = \frac{n\cdot Y}{S}$. Since $\mathbb{E}[\tilde{C}] = \hat{C}$, then we know that $\mathbb{E}[\tilde{C}] = \hat{C} = \frac{n}{S}\cdot\mathbb{E}[Y]$ and solving for $\mathbb{E}[Y]$ gives $\mathbb{E}[Y] = \frac{S\cdot\hat{C}}{n}$. Substituting this into the additive Chernoff bound, we obtain,

$$
\begin{aligned}
|Y - \mathbb{E}[Y]| &= |\frac{S \cdot \tilde{C}}{n} - \frac{S \cdot \hat{C}}{n}| \\
\mathsf{Pr}[|\tilde{C} - \hat{C}| > \frac{\varepsilon n}{2}] &= \mathsf{Pr}[|Y - \mathbb{E}[Y]| > \frac{\varepsilon S}{2}] \\
\mathsf{Pr}[|Y - \mathbb{E}[Y]| > \frac{\varepsilon S}{2}] &\leq 2e^{-2(\varepsilon S/2)^2/S} = 2e^{-\varepsilon^2 S/2} = 2e^{-4} < \frac{1}{8}.
\end{aligned}
$$

□

We can then amplify the probability of success to $1 - \delta$ either by using the median trick discussed in previous lectures or by taking $O(\log(1/\delta))$ additional samples and using the above analysis with the additive Chernoff bound.

Content used in these notes include material from the following sources [Ass20], [Ras20], and [Rub07].

References

[Ass20] Sepehr Assadi. CS 514: Advanced Algorithms II – Sublinear Algorithms. https://people.cs.rutgers.edu/ sa1497/courses/cs514-s20/lec2.pdf, 2020.

[CRT05] Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on computing*, 34(6):1370–1379, 2005.

[Ras20] Sofya Raskhodnikova. CS 591 Sublinear Algorithms. https://cs-people.bu.edu/sofya/sublinear-course/lecture-notes/lecture3.pdf, 2020.

[Rub07] Ronitt Rubinfeld. 6.896 Sublinear Time Algorithms. https://people.csail.mit.edu/ronitt/COURSE/S07/l11.pdf, 2007.