CS 7880: Algorithms for Big Data (Fall'22) **Northeastern University**

Lecture 3

September 16, 2022

Instructor: Soheil Behnezhad *Scribe: Thien Nguyen*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

1 Sublinear Graph Connectivity

There are two common query models for graphs: the *adjacency list* and *adjacency matrix*:

- **Adjacency List:** The query input is a vertex v and an integer i. The output is the ID of the ith neighbor of v, or FAIL if $\deg(v) < i$.
- **Adjacency Matrix:** The query input is a pair of vertices u, v, and the output is 1 if the edge $\{u, v\}$ exists and 0 otherwise.

Let us first focus on the adjacency list model. We seek to solve the problem of *graph connectivity*.

Problem 1 (Graph Connectivity)**.**

Input: An n-vertex graph $G = (V, E)$ in adjacency list format.

Goal: Let C be the number of connected components of G. We seek an estimate $\tilde{C}$ such that

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

We consider both the case where the graph is simple (i.e. when there's no parallel edge) and the case where the graph might contain parallel edges.

Note that this is *not* a multiplicative approximation but is an additive approximation of ϵn which can be quite far from C. We will see later why this is the best we can do.

Naively, BFS and DFS take $\Theta(m+n)$ times due to having to go through all the edges and vertices. Instead, we look to estimate some other value that makes up C. For any vertex v, let S_v be the size of the connected component that v is in. Then

$$\sum_{v \in V} \frac{1}{S_v} = C.$$

Since S_v can be quite large, we use a smaller proxy $S'_v = \min(S_v, 2/\epsilon)$. We can show that this quantity can estimate C well. Letting $C' = \sum_{v \in V} \frac{1}{S'_v}$, :

Claim 1. $|C' - C| \leq \frac{\epsilon n}{2}$.

Proof. For any $v \in V$, we have $0 \leq \frac{1}{S'_v} - \frac{1}{S_v} \leq \frac{\epsilon}{2}$ due to $S'_v \leq S_v$ and $\frac{1}{S'_v} - \frac{1}{S_v} \leq \frac{1}{S'_v} \leq \epsilon/2$. Hence, $\left|\sum_v \frac{1}{S'_v} - \frac{1}{S_v}\right| \leq \frac{\epsilon n}{2}$. □

This fact allows us to estimate C' instead of C. We present Algorithm 1 that attempts so. In short, we are computing S'_v for enough random vertices v to ensure that we are concentrating around the true value and then scaling the sum by an appropriate constant to ensure that our estimate is not biased. We show its correctness and analyze its running time next.

Algorithm 1 Approx. Connectivity

1. Let $K := \frac{3}{\epsilon^2} \ln 2/\delta$ (set from our analysis).
2. For $i = 1$ to K:
   (a) Sample a vertex v_i uniformly at random from V.
   (b) For vetex v_i, start exploring the connected component of v_i with BFS or DFS and truncate if $2/\epsilon$ vertices are seen. Let S'_{v_i} be the number of vertices seen in the connected components of v_i.
3. **Return**: $\tilde{C} := \frac{n}{K} \sum_{i=1}^{K} \frac{1}{S'_{v_i}}$.

1.1 Analysis of Algorithm 1

First, we show that $\tilde{C}$ approximates C' in expectation. Then we show that it concentrates around its expectation (of which we expect since it's a sum of bounded independent r.v.'s).

Claim 2. $\mathbb{E}\left[\tilde{C}\right] = C'$.

Proof. Let $X_i = \frac{1}{S'_{v_i}}$. We have $\mathbb{E}\left[\tilde{C}\right] = \mathbb{E}\left[\frac{n}{K}\sum_{i=1}^{K} X_i\right] = \frac{n}{K}\sum_{i=1}^{K}\mathbb{E}\left[X_i\right]$. Then

$$\begin{aligned}\mathbb{E}\left[X_i\right] &= \sum_{v\in V} \underbrace{\Pr\left[v_i = v\right]}_{\frac{1}{n}} \cdot \frac{1}{S'_v} \\ &= \frac{1}{n}\sum_{v\in V}\frac{1}{S'_v} = \frac{C'}{n}.\end{aligned}$$

Then, $\mathbb{E}\left[\tilde{C}\right] = \frac{n}{K}\sum_{i=1}^{K}\mathbb{E}\left[X_i\right] = \frac{n}{K}\sum_{i=1}^{K}\frac{C'}{n} = C'$. □

We can now establish the concentration result:

Theorem 1. *Let $\tilde{C}$ be returned by Algorithm 1, then*

$$\Pr\left[\left|\tilde{C} - C'\right| \le \frac{\epsilon n}{2}\right] \ge 1 - \delta.$$

This implies that with probability $\ge 1-\delta$, $\left|\tilde{C} - C\right| \le \epsilon n$.

Proof. Let $X_i = \frac{1}{S'_{v_i}}$ and $X = \sum_{i=1}^{K} X_i$. Since $\tilde{C} = \frac{n}{K}X$, we have $\left|\tilde{C} - C'\right| = \left|\tilde{C} - \mathbb{E}\left[\tilde{C}\right]\right| = \frac{n}{K}\left|X - \mathbb{E}\left[X\right]\right|$. So $\left|\tilde{C} - C'\right| \ge \frac{\epsilon n}{2} \iff |X - \mathbb{E}X| \ge \frac{\epsilon K}{2}$.

We will use the additive Chernoff bound, which recall shows $\Pr\left[|X - \mathbb{E}X| \ge t\right] \le 2\exp\left(-\frac{t^2}{3\mathbb{E}X}\right)$ for any $0 \le t \le \mathbb{E}X$. Note that $\mathbb{E}X \le K$ due to $X_i \le 1$. We have:

$$\begin{aligned}\Pr\left[\left|\tilde{C} - C'\right| \ge \frac{\epsilon n}{2}\right] &= \Pr\left[|X - \mathbb{E}X| \ge \frac{\epsilon K}{2}\right] \\ &\le 2\exp\left(-\frac{1}{3\mathbb{E}X}\cdot\frac{\epsilon^2 K^2}{4}\right) \\ &\le 2\exp\left(-\frac{\epsilon^2 K}{12}\right) && (\mathbb{E}X \le K) \\ &= \delta.\end{aligned}$$

The last equality comes from our choice of K: $2\exp\left(-\frac{\epsilon^2 K}{12}\right) = \delta$ implies $K = \frac{3}{\epsilon^2}\ln 2/\delta$. Taking the complement of the probability gives us the result. □

Running time. For each step of the for loop, we perform a BFS to obtain at most $\frac{1}{\epsilon}$ vertices (truncated BFS). At worst, it can visit every edge of its $\frac{1}{\epsilon}$ vertices. Let Δ be the maximum degree of the graph. Then the running time would be $O(K) \cdot O\left(\frac{1}{\epsilon}\Delta\right) = O\left(\frac{\log(1/\delta)}{\epsilon^3}\Delta\right)$.

For a simple graph, the worst case is a complete graph with $O\left(\frac{1}{\epsilon^2}\right)$ edges. Hence, the running time is $O\left(\frac{\log\frac{1}{\delta}}{\epsilon^4}\right)$ for a simple graph.

Remark (Multiplicative Approximation)**.** Our current approximation of $C - \epsilon n \leq C' \leq C + \epsilon n$ seems not too satisfying. If n is large and C is small, then ϵ would have to be set very small to achieve a reasonable approximation. Worse still, for parallel graphs, n and C might be small but if the maximum degree is large, the run time can be much larger than both C and n.

Hence, one might wonder if it's possible to obtain a multiplicative approximation $(1-\epsilon)C \leq C' \leq (1+\epsilon)C$ rather than weaker $C - \epsilon n \leq C' \leq C + \epsilon n$. Sadly, the answer is no. For some intuition, consider a graph G consisting of two fully connected subgraphs K_1, K_2 of size $n/2$ each. Consider two cases where K_1 and K_2 are disconnected and one where they are connected by only 1 edge. Then to distinguish between those two cases require the ability to find the vertex that connects the two graphs together. A formal proof will be presented later.

2 Sublinear Minimum Spanning Tree

Let us now study the Minimum Spanning Tree (MST) problem.

Problem 2 (Minimum spanning tree)**.**

Input: A connected edge-weighted graph $G = (V, E, w)$ with $w : V \to \{1, \ldots, W\}$ in adjacency list format.

Goal: Let T be the weight of the MST of G, which is a spanning tree with the smallest weight. We wish to compute an estimate $\tilde{T}$ such that

$$\Pr\left[\tilde{T} \in (1 \pm \epsilon)T\right] \geq 1 - \delta.$$

Note that this is a multiplicative approximation unlike approximated connected components.

Remark (Kruskal algorithm)**.** Recall the greedy Kruskal algorithm that finds an MST in $O(m \log n)$ time, where we sort all the edges and one-by-one add the lowest weighted edge that joins 2 connected components together until there's no more edge left to add. We now look for an approximation algorithm that runs in much faster time.

To gain some intuition for this problem, we first examine the simple case when $w_e \in \{1, 2\}$. Let $G^{(i)}$ be the subgraph of G including the edges of weights $\{1, \ldots, i\}$. Let $C^{(i)}$ be the number of connected components in graph $G^{(i)}$. Observe from Kruskal's algorithm that the optimal solution is to first find a spanning forest in $G^{(1)}$, which will have $n - C^{(1)}$ edges. Then we should connect these forests into a tree using $(C^{(1)} - 1)$ edges of weight two (recall that we assume G is a connected graph). Hence, the overall value of MST is

$$(n - C^{(1)}) + (C^{(1)} - 1) \cdot 2 = n - 2 + C^{(1)}.$$

The following claim generalizes this for any $W \geq 2$.

Algorithm 2 Sublinear MST

1. **Input**: Graph G with maximum edge weight W and number of vertices n.
2. For $i = 1, \ldots, W$
   (a) Construct $G^{(i)}$ by removing edges of weights strictly greater than i from G.
   (b) Run connectivity (i.e. Algorithm 1) on $G^{(i)}$ to obtain $\tilde{C}^{(i)}$.
3. **Return** $n - W - \sum_{i=1}^{W-1} \tilde{C}^{(i)}$.

Theorem 2. *It holds that* $T = n - W + \sum_{i=1}^{w-1} C^{(i)}$.

Proof. Let $\alpha_j = \#$ of edges of weight j in the MST. This definition gives $T = \sum_{j=1}^{W} \alpha_j \cdot j$. Note that

$$\sum_{j=1}^{W} \alpha_j \cdot j = \sum_{i=1}^{W} \sum_{j=i}^{W} \alpha_j,$$

since for each α_j in the inner sum, we count it j times: once for each $i \leq j$ of the outer sum. As another way to see this, consider the grid where each row represents weight i:

$$\begin{array}{cccc} \alpha_W & \alpha_W & \alpha_W & \alpha_W \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_3 & \alpha_3 & \alpha_3 & \\ \alpha_2 & \alpha_2 & & \\ \alpha_1 & & & \end{array}$$

Then the first sum is just summing the rows first and the second sum is summing the columns first.

Now, let $\beta_j := \sum_{j=i+1}^{W} \alpha_j = \#$ of edges of weight $> j$ in the MST. Recall the definition of $G^{(j)}$ where we remove all the edges of weight $> j$ from G. Then, since the number of connected components in $G^{(j)}$ is $C^{(j)}$, there must be $C^{(j)} - 1$ edges of weights $> j$ connecting the $C^{(j)}$'s components together. Similarly to how Kruskal's algorithm can add edges greedily to build the tree, the MST of G must also contain $C^{(j)} - 1$ of edges of weights $> j$. That means $\beta_j = C^{(j)} - 1$.

Then, combining everything, we have:

$$T = \sum_{i=1}^{W} \sum_{j=i}^{W} \alpha_j = \sum_{i=1}^{W} \beta_{i-1} = \sum_{i=1}^{W} C^{(i-1)} - 1 = \sum_{i=0}^{W-1} C^{(i)} - W = n - W - \sum_{i=1}^{W-1} C^{(i)}.$$

This establishes our claim. □

This theorem means that we can solve or approximate MST by computing or approximating $C^{(i)}$ for each $i = 1, \ldots, W-1$. We can do that via the connectivity algorithm from Theorem 1. We present this algorithm formally in Algorithm 2.

Correctness. We must first inspect the failure probability of W connectivity runs:

$$\begin{aligned} \Pr\left[\text{all } W \text{ calls to the connectivity algorithm succeed}\right] &\geq 1 - \sum_{i=1}^{W} \Pr\left[i^{th} \text{ call fails}\right] \\ &\geq 1 - \delta W. \end{aligned}$$

We can then set $\delta = \delta'/W$ to get a $1 - \delta'$ bound. To handle the estimation error, we note that the error overall is $W \cdot \epsilon n$ if we run connectivity W times. Hence, we can set $\epsilon = \epsilon'/W$.

Time Complexity. Since each connectivity run takes $O\left(\frac{\log \frac{1}{\delta}}{\epsilon^4}\right)$ time, plugging in the choices of δ and ϵ above for each W connectivity run, we get an overall time complexity of

$$O\left(W^5 \frac{\log \frac{\delta'}{W}}{\epsilon'^4}\right).$$

> **Remark.** We can further reduce the dependency on W by bucketing weights together, but that's a topic for another time.

Combining the above, we recover the following result:

Theorem 3. *[CRT05] In any n-vertex edge-weighted simple graph G of average degree d, there is an $O\left(W^5 \frac{\log \frac{\delta}{W}}{\epsilon^4}\right)$ time algorithm to produce an estimate $\widetilde{mst}(G)$ such that*

$$\Pr\left[\widetilde{mst}(G) \in (1 \pm \epsilon) \cdot mst(G)\right] \geq 1 - \delta.$$

Exercise: Prove that finding the *edges* of (any) appropriate MST requires $\Omega\left(n^2\right)$ time.

References

[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. 5