6.5240 Sublinear Time Algorithms September 11, 2024

Lecture 3

Lecturer: Ronitt A. Rubinfeld *Scribe: Fares Pasuni*

The content of this class is follows:

- We present algorithms for estimating the number of connected components and the weight of the minimum spanning tree (MST) in a graph, aiming for efficient computation with controlled approximation errors.
- For estimating connected components, we introduce a sampling-based algorithm that approximates the number within an additive error of ϵn, utilizing partial breadth-first searches to limit runtime to $\mathcal{O}\left(\frac{\Delta}{\epsilon^4}\right)$, where Δ is the maximum degree.
- For estimating the MST weight, we leverage properties of Kruskal's algorithm by estimating connected components at various weight thresholds. This approach achieves a multiplicative ϵ-approximation of the MST weight with a runtime of $\mathcal{O}\left(\frac{\Delta w^5}{\epsilon^4}\right)$, where w is the maximum edge weight.

1 Estimating Number of Connected Components

1.1 Problem Description

We are given a graph $G = (V, E)$ represented as an adjacency list. Let $n = |V|$ be the number of vertices, and $m = |E|$ be the number of edges. The maximum degree of any vertex in the graph is denoted as Δ. Our objective is to estimate the number of connected components in G with a given accuracy parameter ϵ, where the estimate is close to the actual number of connected components with a small additive error. The algorithm returns an estimate $\tilde{C}$ of the number of connected components C in the graph such that:

$$C - \epsilon n \leq \tilde{C} \leq C + \epsilon n.$$

Here, $\tilde{C}$ is the estimated number of components, and ϵn is the bound on the error between the estimate and the true number of components.

Query model: degree queries (number of edges connected to a vertex) and neighbor queries in constant time.

1.2 Algorithm Overview

Step 1: Sampling and Computing the Sizes of Components The algorithm begins by randomly sampling a set of vertices $U \subseteq V$. For each sampled vertex $u \in U$, the algorithm calculates the size of the connected component it belongs to, denoted as n_u. The size n_u can be computed by performing a breadth-first search (BFS) or depth-first search (DFS) starting from u.

The critical observation is that for any connected component $A \subseteq V$:

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

Thus, the total number of connected components in the graph can be estimated by summing the reciprocal of component sizes for all vertices:

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

Step 2: Approximation of Component Sizes For efficiency, the algorithm introduces an approximation to handle large components. Specifically, if the size of a connected component n_u exceeds $\frac{2}{\epsilon}$, the component is approximated coarsely. For components smaller than $\frac{2}{\epsilon}$, we compute the exact size. The approximation is given by:

$$\hat{n}_u = \min(n_u, \frac{2}{\epsilon}).$$

This approximation works as follows: while performing BFS (Breadth-First Search) starting from vertex u, the algorithm continues to explore vertices in the component of u. However, once the BFS reaches either $\frac{2}{\epsilon}$ unique vertices or the entire connected component, whichever happens first, the search stops. If the BFS completes before reaching $\frac{2}{\epsilon}$ vertices, the exact size of the component is used. Otherwise, the size is approximated as $\frac{2}{\epsilon}$. This ensures that large components are handled more efficiently without fully exploring them, thereby saving computational time.

The algorithm will estimate the following quantity $\hat{C}$:

$$\hat{C} = \sum_{u \in V} \frac{1}{\hat{n}_u}.$$

The algorithm ensures the approximation error is controlled because the reciprocal of large numbers are very close to each other. Formally, we have the following lemma:

Lemma 1 *For all vertices $u \in V$, the following inequality holds:*

$$\left| \frac{1}{\hat{n}_u} - \frac{1}{n_u} \right| \leq \frac{\epsilon}{2}.$$

This lemma provides a bound on the error between the estimated size of the connected component containing vertex u, $\hat{n}_u$, and its true size n_u, in terms of their reciprocals. We now analyze two cases based on the definition of $\hat{n}_u$.

Case 1: When $\hat{n}_u = n_u$ This case occurs when the size of the connected component containing vertex u is smaller than or equal to $\frac{2}{\epsilon}$. That is, $n_u \leq \frac{2}{\epsilon}$.

In this case, the estimate $\hat{n}_u$ is exactly equal to the true size n_u, meaning:

$$\hat{n}_u = n_u.$$

Therefore, the difference between the reciprocals is:

$$\left| \frac{1}{\hat{n}_u} - \frac{1}{n_u} \right| = \left| \frac{1}{n_u} - \frac{1}{n_u} \right| = 0.$$

Since the error is exactly zero in this case, the bound $\frac{\epsilon}{2}$ is trivially satisfied.

Case 2: When $\hat{n}_u = \frac{2}{\epsilon}$ This case occurs when the size of the connected component containing vertex u is larger than $\frac{2}{\epsilon}$, meaning $n_u > \frac{2}{\epsilon}$. In this case, the algorithm approximates the size of the component by setting:

$$\hat{n}_u = \frac{2}{\epsilon}.$$

The difference between the reciprocals is:

$$\left| \frac{1}{\hat{n}_u} - \frac{1}{n_u} \right| = \left| \frac{\epsilon}{2} - \frac{1}{n_u} \right|.$$

Since $n_u > \frac{2}{\epsilon}$, we know that $\frac{1}{n_u} < \frac{\epsilon}{2}$. Thus, the error is bounded by:

$$\left|\frac{\epsilon}{2} - \frac{1}{n_u}\right| \leq \frac{\epsilon}{2}.$$

Therefore, in this case, the bound $\frac{\epsilon}{2}$ is also satisfied.

From this lemma, we derive that the total error between the true number of components C and the estimate $\hat{C}$ is bounded by:

$$|C - \hat{C}| \leq \sum_{u \in V} |\frac{\epsilon}{2} - \frac{1}{n_u}| \leq n \cdot \frac{\epsilon}{2}.$$

1.3 Algorithm for Estimating $\hat{C}$

Let $b > 0$ be a large enough constant. We will specify it in the analysis.

Algorithm 1: EstimateNumberOfConnectedComponents

```
Input: Graph $G = (V, E)$, accuracy parameter $\epsilon$
Output: Estimated number of connected components $\hat{C}$
1 $r \leftarrow \frac{b}{\epsilon^3}$ // Set sample size based on accuracy parameter ;
2 $U \leftarrow \{u_1, u_2, \ldots, u_r\}$ // Randomly sample $r$ vertices from $V$ ;
3 $S \leftarrow 0$ ;
4 for each vertex $u_i \in U$ do
5     $\hat{n}_{u_i} \leftarrow \text{BFS}(u_i)$ // Perform BFS to compute component size of $u_i$ stopping when reach
        enough vertices;
6     $S \leftarrow S + \frac{1}{\hat{n}_{u_i}}$;
7 $\tilde{C} \leftarrow \frac{n}{r} \times S$ // Scale the estimate by the total number of vertices ;
8 return $\tilde{C}$;
```

Note that the final estimate $\tilde{C}$ is computed by scaling the sum of the reciprocals by $\frac{n}{r}$, where n is the total number of vertices and r is the size of the sample.

Runtime Analysis: Each BFS has a time complexity of $\mathcal{O}(\Delta/\epsilon)$. Since the BFS is performed for each of the $r = \frac{1}{\epsilon^3}$ sampled vertices, the overall runtime of the algorithm is:

$$\mathcal{O}(r \cdot \Delta/\epsilon) = \mathcal{O}\left(\frac{\Delta}{\epsilon^4}\right).$$

Theorem 2 *The probability that the estimate of the number of connected components $\tilde{C}$ deviates from the quantity $\hat{C}$ by more than $\epsilon \cdot \frac{n}{2}$ is bounded as follows:*

$$\Pr\left[\left|\tilde{C} - \hat{C}\right| \leq \frac{\epsilon n}{2}\right] \geq \frac{3}{4}.$$

Proof Let $X_i = \frac{1}{\hat{n}_{u_i}}$, where $\hat{n}_{u_i}$ is the estimated size of the connected component for the sampled vertex u_i. Define the sum of these values over the sample set $U = \{u_1, u_2, \ldots, u_r\}$ as:

$$S = \sum_{i=1}^{r} X_i.$$

We compute the average of these values:

$$\frac{S}{r} = \frac{1}{r}\sum_{i=1}^{r} \frac{1}{\hat{n}_{u_i}} = \frac{\tilde{C}}{n}.$$

The expected value of each X_i is:

$$p = \mathbb{E}[X_i] = \frac{1}{n}\sum_{u \in V} \frac{1}{\hat{n}_u} = \frac{\hat{C}}{n}.$$

Define $\delta = \frac{\epsilon}{2}$. Recall that:

$$r = b/(\epsilon)^3.$$

$$0 \leq \frac{1}{n_u} \leq \frac{1}{\hat{n}_u} \leq \frac{\epsilon}{2}.$$

We know that:

$$n \geq \sum_{u \in V} \frac{1}{\hat{n}_u} \geq \frac{\epsilon n}{2}.$$

Thus:

$$\frac{\epsilon}{2} \leq \frac{\hat{C}}{n} \leq 1.$$

We want:

$$\begin{aligned} Pr[|\tilde{C} - \hat{C}| \geq \frac{\epsilon \cdot n}{2}] &\leq Pr[|\tilde{C} - \hat{C}| \geq \frac{\epsilon}{2} \cdot \hat{C}] \\ &= Pr[|\frac{\tilde{C}}{n} - \frac{\hat{C}}{n}| \geq \frac{\epsilon}{2} \cdot \frac{\hat{C}}{n}] \\ &\leq e^{-\Omega(r \cdot p \cdot \delta^2)} = e^{-\Omega(\frac{b}{\epsilon^3} \cdot \frac{\hat{C}}{n} \cdot \frac{\epsilon^2}{4})} \leq e^{-\Omega(b)}, \end{aligned}$$

where we use the Chernoff bound, and the fact that $\frac{\hat{C}}{n} \geq \frac{\epsilon}{2}$ to bound the last exponent. This is the reason why we set $\frac{1}{\hat{n}_u}$ to be $\frac{\epsilon}{2}$ instead of 0.

We can now pick b so that $e^{-\Omega(b)} \leq \frac{1}{4}$

This implies:

$$\Pr[|\tilde{C} - \hat{C}| \leq \frac{\epsilon n}{2}] \geq 3/4.$$

■

Furthermore, based on our last section we have:

$$|C - \hat{C}| \leq \frac{\epsilon n}{2}.$$

Therefore, based on the triangular inequality:

$$|\tilde{C} - C| \leq \epsilon n.$$

2 Estimation of Minimum Spanning Tree (MST)

2.1 Problem Description

We are given a graph $G = (V, E)$ with maximum degree Δ, where each edge $e = (u, v) \in E$ has an integer weight $w(u, v) \in \{1, \ldots, w\} \cup \{\infty\}$ where ∞ means an edge does not exist. The graph is connected, and our objective is to estimate the weight of the minimum spanning tree (MST), denoted by M, within a factor of ϵ. The algorithm estimates the weight $\hat{M}$ of the minimum spanning tree such that:

$$(1 - \epsilon)M \leq \hat{M} \leq (1 + \epsilon)M.$$

where M is the true weight of the MST, and $\hat{M}$ is the estimate returned by the algorithm.

Query model: degree queries (number of edges connected to a vertex) and neighbor queries in constant time.

2.2 Approximation Algorithm Overview

Step 1: Estimation of Connected Components for Each Weight The algorithm starts by estimating the number of connected components $C^{(i)}$ in the subgraph induced by edges with weights $\leq i$, for $i = 1, \ldots, w$. Each $C^{(i)}$ is computed with an additive error of $\epsilon' = \frac{\epsilon}{2w}n$.

Step 2: Estimating the MST Weight The weight of the MST is approximated using the number of connected components in each subgraph:

$$\hat{M} = n - w + \sum_{i=1}^{w-1} \hat{C}^{(i)}.$$

Here, $\hat{C}^{(i)}$ is the estimated number of components with edge weights $\leq i$.

2.3 Why the Solution Works (Intuition from Kruskal's Algorithm)

Kruskal's algorithm constructs the MST by adding edges in increasing order of weight, always adding the smallest edge that connects two previously unconnected components. The idea behind our algorithm is to approximate the same process by progressively considering edges of increasing weight.

For each weight i, The algorithm constructs a subgraph G_i that contains all edges with weight $\leq i$. The number of connected components $\hat{C}^{(i)}$ in each subgraph is estimated, which gives an approximation of how many additional edges need to be added to span all components. Critically, The algorithm handles weights in a manner similar to how Kruskal's algorithm would process them in sorted order. It first considers all edges of weight 1, then weight 2, and so on. This ensures that the MST weight is approximated correctly by the careful accounting for the contribution of edges.

2.4 Detailed Explanation: Two ways of computing the weight of the MST

The weight of the MST can be expressed as the sum of contributions from edges of different weights. Define α_i as the number of edges with weight i in the MST. Then, the total weight of the MST M is:

$$M = \sum_{i=1}^{w} i \cdot \alpha_i.$$

This sum can be expanded as:

$$M = \sum_{i=1}^{w} \alpha_i + \sum_{i=2}^{w} \alpha_i + \sum_{i=3}^{w} \alpha_i + \cdots + \sum_{i=w}^{w} \alpha_i.$$

Alternatively, in terms of connected components, the MST weight can be written as:

$$M = (C^{(0)} - 1) + (C^{(1)} - 1) + \cdots + (C^{(w-1)} - 1).$$

This equation follows because the corresponding summandi, i.e. $\sum_{i=j}^{w} \alpha_i$ and $C^{(j-1)}$ are equal for every $j = 1, \dots, w$. The reason why the two are equal is because the number of edges with weight $\geq i$ in the MST that are needed to make the graph connected is exactly the number of components in G_{i-1} and the expression $C^{(i)}$ represents the number of connected components when considering edges with weights $\leq i$. The total weight of the MST is:

$$M = n - w + \sum_{i=1}^{w-1} C^{(i)}.$$

Here, n is the total number of vertices, and w is the maximum edge weight.

2.5 Pseudocode for MST Estimation

Algorithm 2: EstimateMSTWeight

```
Input: Graph $G = (V, E)$, maximum edge weight $w$, accuracy parameter $\epsilon$
Output: Estimated weight of the MST $\hat{M}$
1 $\hat{M} \leftarrow n - w$ // Initialize the estimate of the MST weight;
2 for $i \leftarrow 1$ to $w - 1$ do
3     $G_i \leftarrow$ Subgraph of $G$ with edge weights $\leq i$;
4     $\hat{C}^{(i)} \leftarrow \textsc{EstimateConnectedComponents}(G_i, \frac{\epsilon}{2w})$;
5     $\hat{M} \leftarrow \hat{M} + \hat{C}^{(i)}$;
6 return $\hat{M}$;
```

2.6 Runtime and Approximation Guarantee

Runtime Analysis: The runtime of the algorithm is dominated by the calls to estimate the connected components. The complexity is:

$$\mathcal{O}\left(\frac{\Delta w^5}{\epsilon^4}\right).$$

where Δ is the maximum degree of the graph.

Approximation Guarantee: We call the approximate connected components algorithm with a failure probability of $\delta \leq \frac{1}{4w}$. The probability that all calls return an ϵ'-approximate result is:

$$\Pr[\text{all calls give } \epsilon' \text{ approximation}] \geq 1 - \frac{w}{4w} = \frac{3}{4}.$$

If this condition holds, the error between the true MST weight M and the estimated weight $\hat{M}$ is:

$$|M - \hat{M}| = |\sum_{i=1}^{w-1} C^{(i)} - \sum_{i=1}^{w-1} \hat{C}^{(i)}| \leq w \cdot \epsilon' = w \cdot \frac{\epsilon}{2w} \cdot n = \frac{\epsilon n}{2}.$$

Since $M \geq n - 1 \geq \frac{n}{2}$, this implies:

$$|M - \hat{M}| \leq \epsilon M.$$

This ensures that the approximation error is within a multiplicative factor of ϵ relative to the true MST weight.