

Lecture 2

Lecturer: Ronitt Rubinfeld

Scribe: Valentio Iverson

Topics covered.

1. Estimating number of connected components
2. Estimating the weight of a minimum spanning tree

1 Estimating the Number of Connected Components

Problem. We are given an undirected graph $G = (V, E)$ in the adjacency-list representation, with $n = |V|$, maximum degree Δ , and an accuracy parameter $\varepsilon \in (0, 1)$. We have degree-query access, where a query on v returns $\deg(v)$, and neighbor-query access, where a query on (v, j) returns the j th neighbor of v . If C denotes the number of connected components of G , the goal is to output an estimate $\tilde{C}$ such that $|\tilde{C} - C| \leq \varepsilon n$ with probability at least $\frac{3}{4}$.

1.1 Why do we ask for additive error?

Before deciding what approximation guarantee is reasonable, consider the two graphs in Figure 1. The graph $C_6 \sqcup C_6$ has two connected components, while C_{12} has one. But if we start from a random vertex and inspect only a small neighborhood, these two inputs look the same: locally, we simply see a path of degree-two vertices, without a way to distinguish between these two unless you look sufficiently many vertices. For larger versions of the same example, an algorithm has to keep exploring until it learns how the path eventually closes.

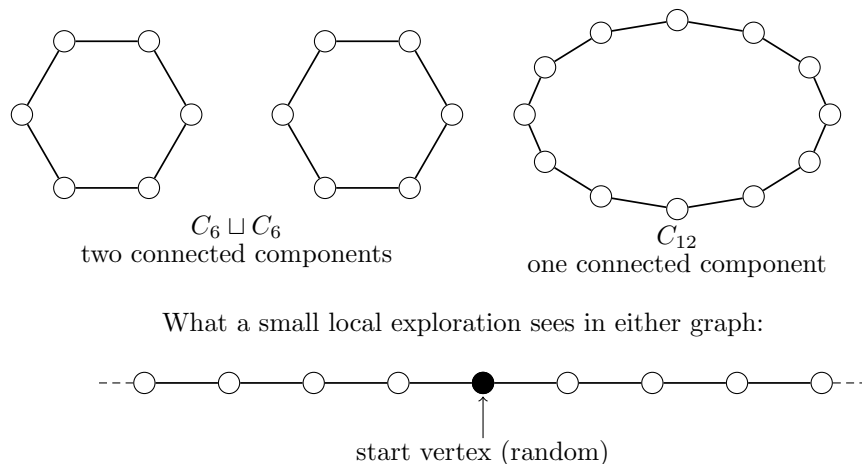


Figure 1: One component and two components can have the same local view: both of these locally give a path of degree-two vertices.

A multiplicative approximation must distinguish one connected component from two, and using these *local* ideas, one can show that this in fact requires $\Omega(n)$ queries.

This same indistinguishability idea also rules out a constant additive-error approximation in sublinear time. The example above is meant to illustrate the general idea of the lower bound, by arguing that two different construction essentially looks identical locally. We will learn formal techniques for transforming this kind of intuition into an actual lower bound later in the course. This motivates asking instead for additive error εn .

1.2 A different characterization

The main insight here is to get a different characterization of the number of connected components.

For each vertex $u \in V$, let n_u be the number of vertices in the connected component containing u . If $A \subseteq V$ is a connected component, then $|A| = n_u$ for every $u \in A$, and hence

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

Therefore, we have

$$C = \sum_{u \in V} \frac{1}{n_u} = n \cdot \underbrace{\frac{1}{n} \sum_{u \in V} \frac{1}{n_u}}_{\text{average value of } 1/n_u}. \quad (1)$$

It's worth noting here that it's not super clear why this would be better: since we have a sum of n terms, and computing each n_u naively may take $O(n)$ time. The key is to estimate both of these: we will use Chernoff bound to estimate this sum of n terms and we show that we can take constant time to estimate $1/n_u$.

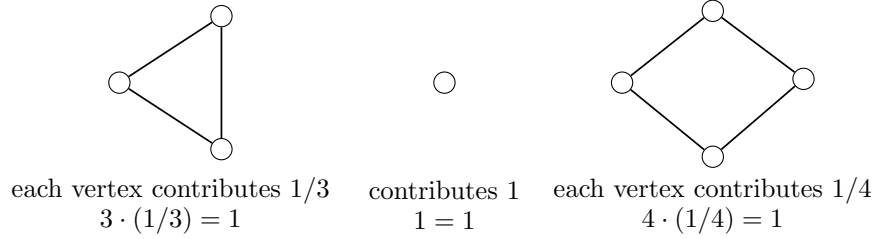


Figure 2: Examples for n_u and $1/n_u$ in various graphs: each connected component contributes exactly one to the sum in (1). In this example the three components therefore contribute $1 + 1 + 1 = 3$.

The identity (1) therefore suggests a two-step plan:

1. Given a sampled vertex u , estimate its contribution $1/n_u$ quickly.
2. Estimate the average of these contributions by random sampling.

1.3 Estimating $1/n_u$ in constant time

If the component containing u is small, simply running a BFS can find n_u exactly. However, if the component is large, finding its exact size may be costly. However, note that in this case $1/n_u$ gets really small. The key here is to develop some threshold for which we stop the search once the component size surpass the threshold, as the contribution to the value of $1/n_u$ will be negligible.

Indeed, to do this, we will define

$$\hat{n}_u := \min \left\{ n_u, \frac{2}{\varepsilon} \right\} \text{ and } \hat{C} := \sum_{u \in V} \frac{1}{\hat{n}_u}.$$

By definition, we have $1/\hat{n}_u = \max\{1/n_u, \varepsilon/2\}$.

Lemma 1. *For every $u \in V$, $|1/\hat{n}_u - 1/n_u| \leq \varepsilon/2$.*

Proof. If $n_u \leq 2/\varepsilon$, then $\hat{n}_u = n_u$ and the difference is zero. Otherwise $n_u > 2/\varepsilon$ and $\hat{n}_u = 2/\varepsilon$, and so

$$0 < \frac{1}{n_u} < \frac{\varepsilon}{2} = \frac{1}{\hat{n}_u}.$$

Hence, the difference is at most $\varepsilon/2$. ■

Corollary 2. $|C - \hat{C}| \leq \varepsilon n/2$.

Proof. Summing the pointwise errors from Lemma 1 and using triangle inequality gives us

$$|C - \widehat{C}| = \left| \sum_{u \in V} \frac{1}{n_u} - \sum_{u \in V} \frac{1}{\widehat{n}_u} \right| = \left| \sum_{u \in V} \left(\frac{1}{n_u} - \frac{1}{\widehat{n}_u} \right) \right| \leq \sum_{u \in V} \left| \frac{1}{n_u} - \frac{1}{\widehat{n}_u} \right| \leq \frac{\varepsilon n}{2},$$

as desired. ■

The reason this approximation is algorithmically useful is that as soon as we know $n_u > 2/\varepsilon$, we already know $\widehat{n}_u = 2/\varepsilon$ and may stop searching. This means that we can compute $\widehat{n}_u$ in $O(1/\varepsilon)$ queries.

Algorithm 1 Computing $\widehat{n}_u$

```

Start a breadth-first search from  $u$ .
Continue until either the whole component of  $u$  has been explored or  $2/\varepsilon$  distinct vertices have been
discovered.
if the whole component was explored and contains  $s$  vertices then
    return  $s$ .
else
    return  $2/\varepsilon$ .
end if

```

Let us analyze the runtime of this algorithm. A naive analysis shows that we can do this in $O(\Delta/\varepsilon)$ queries: indeed, we visit at most $2/\varepsilon$ vertices, and may inspect up to Δ neighbors of each. We can improve this to $O(1/\varepsilon^2)$. Note that before scanning the adjacency list of a discovered vertex, query $\deg(v)$. If $\deg(v) > 2/\varepsilon$, then the component containing v , and hence u , already has more than $2/\varepsilon$ vertices, so we can stop immediately. Otherwise v has at most $2/\varepsilon$ neighbors to scan. Since at most $2/\varepsilon$ such adjacency lists are scanned, the whole procedure uses $O(1/\varepsilon^2)$ queries.

1.4 Estimating the sum by sampling

We now estimate $\widehat{C}$. Let b be a sufficiently large absolute constant and set $r = b/\varepsilon^3$. We sample r vertices independently and uniformly from V , compute $\widehat{n}_u$ for each sampled vertex, and average their reciprocal contributions.

Algorithm 2 Estimating $\widehat{C}$

```

Set  $r \leftarrow b/\varepsilon^3$ , where  $b$  is a sufficiently large constant.
Choose  $u_1, \dots, u_r$  independently and uniformly from  $V$ .
for  $i = 1, 2, \dots, r$  do
    Compute  $\widehat{n}_{u_i}$  using Algorithm 1.
end for
return  $\widetilde{C} \leftarrow n \cdot \frac{1}{r} \sum_{i=1}^r \frac{1}{\widehat{n}_{u_i}}$ .

```

The quantity after n in the definition of $\widetilde{C}$ is simply the sample average of the vertex contributions $1/\widehat{n}_u$. Since one computation of $\widehat{n}_u$ costs $O(1/\varepsilon^2)$ queries and we take $O(1/\varepsilon^3)$ samples, the total running time is thus $O(1/\varepsilon^5)$.

It remains to show that this sample average is accurate. We will use the following form of the Chernoff bound.

Lemma 3 (Chernoff bound). *Let $X_1, \dots, X_r$ be independent and identically distributed random variables taking values in $[0, 1]$. Let $S := \sum_{i=1}^r X_i$ and let $p := \mathbb{E}[X_i] = \mathbb{E}[S]/r$. Then, for every $\delta \in (0, 1)$,*

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

For our estimator, let $X_i := 1/\widehat{n}_{u_i}$ and $S := \sum_{i=1}^r X_i$. Since u_i is uniformly distributed over V ,

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

Similarly, by the definition of the output of Algorithm 2,

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

At this point, it becomes apparent why we chose $\varepsilon/2$, rather than 0, as the proxy contribution of a large component to $1/n_u$. By the definition of $\widehat{n}_u$, every sampled contribution satisfies $\varepsilon/2 \leq X_i \leq 1$. Consequently, $\varepsilon/2 \leq p \leq 1$. Informally, after truncation, the largest possible sampled value is only a factor $2/\varepsilon$ larger than the smallest possible sampled value. More importantly for the analysis, the mean p can no longer become arbitrarily small.

This is exactly what we need for a multiplicative Chernoff argument. The exponent in Lemma 3 is proportional to $rp\delta^2$. If we had replaced the contribution of a large component by 0, then p could be arbitrarily close to 0, and the factor rp in the exponent could be too small to obtain a bound using only a number of samples depending on ε . By truncating the contribution from below at $\varepsilon/2$, we guarantee $p \geq \varepsilon/2$.

We can now prove the sampling guarantee.

Theorem 4. *For a sufficiently large absolute constant b , we have*

$$\mathbb{P}\left[|\widetilde{C} - \widehat{C}| \leq \frac{\varepsilon n}{2}\right] \geq \frac{3}{4}.$$

Proof. Recall that $\widetilde{C} = nS/r$ and $\widehat{C} = np$. We want to show that the sampling error is at most $\varepsilon n/2$ with probability at least $3/4$.

Since $1/\widehat{n}_u \leq 1$ for every u , we have $\widehat{C} \leq n$. Therefore, $\varepsilon/2 \cdot \widehat{C} \leq (\varepsilon n)/2$.

It follows that

$$\begin{aligned} \mathbb{P}\left[|\widetilde{C} - \widehat{C}| \geq \frac{\varepsilon n}{2}\right] &\leq \mathbb{P}\left[|\widetilde{C} - \widehat{C}| \geq \frac{\varepsilon}{2} \widehat{C}\right] \\ &= \mathbb{P}\left[\left|n\frac{S}{r} - np\right| \geq \frac{\varepsilon}{2} np\right] \\ &= \mathbb{P}\left[\left|\frac{S}{r} - p\right| \geq \frac{\varepsilon}{2} p\right]. \end{aligned}$$

We now apply Lemma 3 with $\delta = \varepsilon/2$. Since $r = b/\varepsilon^3$ and, by our choice of truncation, $p \geq \varepsilon/2$, we obtain

$$\begin{aligned} \mathbb{P}\left[|\widetilde{C} - \widehat{C}| \geq \frac{\varepsilon n}{2}\right] &\leq \exp\left(-\Omega\left(rp\left(\frac{\varepsilon}{2}\right)^2\right)\right) \\ &\leq \exp\left(-\Omega\left(\frac{b}{\varepsilon^3} \cdot \frac{\varepsilon}{2} \cdot \frac{\varepsilon^2}{4}\right)\right) \\ &= e^{-\Omega(b)}. \end{aligned}$$

Thus, by choosing the absolute constant b sufficiently large, we can make this probability at most $1/4$. Hence

$$\mathbb{P}\left[|\widetilde{C} - \widehat{C}| \leq \frac{\varepsilon n}{2}\right] \geq \frac{3}{4}. \quad \blacksquare$$

We can now combine the two sources of error. By Corollary 2, the truncation error always satisfies

$$|C - \widehat{C}| \leq \frac{\varepsilon n}{2},$$

and by Theorem 4, with probability at least $3/4$ the sampling error satisfies

$$|\widehat{C} - \widetilde{C}| \leq \frac{\varepsilon n}{2}.$$

Therefore, on this event, the triangle inequality gives

$$|C - \widetilde{C}| \leq |C - \widehat{C}| + |\widehat{C} - \widetilde{C}| \leq \varepsilon n.$$

Thus Algorithm 2 gives an additive εn approximation to the number of connected components using $O(1/\varepsilon^5)$ queries, with success probability at least $3/4$. As usual, the success probability can be amplified from $3/4$ to $1 - \delta$ using the method from Homework 0.

2 Estimating the Weight of a Minimum Spanning Tree

Problem. We are given a connected, undirected graph $G = (V, E)$ in the adjacency-list representation, with $n = |V|$, maximum degree Δ , and an accuracy parameter $\varepsilon \in (0, 1)$. Each edge $(u, v) \in E$ has a weight $w_{uv} \in \{1, 2, \dots, w\}$. We have degree-query access, where a query on v returns $\deg(v)$, and neighbor-query access, where a query on (v, j) returns the j th neighbor of v along with the edge weight. For a spanning tree T , let $\text{wt}(T) := \sum_{e \in T} w_e$. If we have

$$M := \min\{\text{wt}(T) : T \text{ is a spanning tree of } G\}$$

denotes the minimum weight of a spanning tree, the goal is to output an estimate $\widetilde{M}$ such that $(1 - \varepsilon)M \leq \widetilde{M} \leq (1 + \varepsilon)M$ with probability at least $\frac{3}{4}$.

Every spanning tree has exactly $n - 1$ edges. Under our normalization $1 \leq w_e \leq w$, this gives $n - 1 \leq M \leq w(n - 1)$. The runtime will depend on the range of edge weights. More generally, after scaling the minimum positive weight to one, w should be thought of as the ratio between the largest and smallest relevant edge weights. The assumption of integral weights is only for convenient notation.

The strategy is the same kind of move that worked for connected components: rather than estimate the MST weight directly, find a different characterization that can be expressed using quantities we already know how to estimate.

2.1 A different characterization of MST

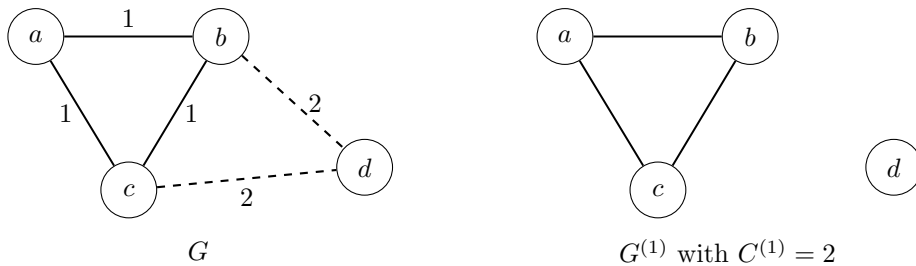
For $i \in \{1, \dots, w\}$, define

$$E^{(i)} := \{(u, v) \mid w_{u,v} \in \{1, \dots, i\}\} = \text{set of edges of weight at most } i.$$

We define $G^{(i)} := (V, E^{(i)})$, and let $C^{(i)}$ be the number of connected components of $G^{(i)}$. Thus, $G^{(i)}$ contains exactly the edges whose weight is at most i .

We first build intuition in some easy examples:

Warm-up: $w = 1$. Every edge has weight one, so every spanning tree has weight exactly $n - 1$. Hence $M = n - 1$.



$G^{(1)}$ has two components, so one weight-two edge is still needed.

$$\#\{\text{weight-two edges in an MST}\} = C^{(1)} - 1 = 1.$$

Figure 3: The $w = 2$ example. The threshold graph $G^{(1)}$ forgets the heavier edges. Its component count tells us exactly how many weight-two edges are necessary to finish a spanning tree.

Warm-up: $w = 2$. We will use Kruskal's idea for constructing minimum spanning tree: first use as many weight-one edges as possible, which connect vertices inside the components of $G^{(1)}$, but they cannot connect two different components of $G^{(1)}$. We need to use the weight two edges to connect the remaining components. Since there are $C^{(1)}$ such components, then exactly $C^{(1)} - 1$ additional edges are needed

to connect them into one tree, and those additional edges have weight two. Therefore, the total MST weight is

$$\begin{aligned} M &= 1 \cdot \#\{\text{weight-one edges}\} + 2 \cdot \#\{\text{weight-two edges}\} \\ &= (n-1) + \#\{\text{weight-two edges}\} \\ &= (n-1) + (C^{(1)} - 1) = n - 2 + c^{(2)} \end{aligned}$$

This is the motivating picture for all larger w , which is Kruskal's idea for constructing minimum spanning tree: first use as many weight-one edges as possible; then use weight-two edges to merge as many remaining components as possible; then use weight-three edges, and so on.

Lemma 5. *For every $i \in \{1, \dots, w\}$, every minimum spanning tree contains exactly $n - C^{(i)}$ edges of weight at most i . In particular, any MST have the same number of edges of weight i , for every $i \in \{1, \dots, w\}$.*

Proof. Inside a component of $G^{(i)}$ on s vertices, a forest contains at most $s - 1$ edges. Summing over the $C^{(i)}$ components, any spanning tree contains at most $n - C^{(i)}$ edges from $E^{(i)}$.

Suppose an MST T contained fewer. Then within some component of $G^{(i)}$, the forest $T \cap E^{(i)}$ would have at least two connected pieces. Since that component is connected in $G^{(i)}$, there is an edge $e \in E^{(i)}$ joining two such pieces. Adding e to T creates a unique cycle. Because the endpoints of e were not already connected using edges from $E^{(i)}$, this cycle contains some edge f of weight strictly larger than i . Replacing f by e gives a spanning tree of strictly smaller total weight, contradicting minimality. ■

Lemma 6. *We have*

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

Proof. Let α_i be the number of weight- i edges in an MST, which we can define since shows that this quantity is well-defined even if the MST itself is not unique. In fact, the above shows that $\alpha_i = C^{(i-1)} - C^{(i)}$ for $i \in \{2, \dots, w\}$ and $\alpha_1 = n - C^{(1)}$.

Equivalently, the number of MST edges with weight larger than i is

$$\sum_{j=i+1}^w \alpha_j = (n-1) - (n - C^{(i)}) = C^{(i)} - 1. \quad (2)$$

Now expand the MST weight by counting a weight- j edge once at each level below j :

$$\begin{aligned} M &= \sum_{j=1}^w j\alpha_j = \sum_{j=1}^w \alpha_j + \sum_{j=2}^w \alpha_j + \dots + \sum_{j=w}^w \alpha_j \\ &= \sum_{i=0}^{w-1} (C^{(i)} - 1) = n - w + \sum_{i=1}^{w-1} C^{(i)}. \end{aligned}$$

■

Therefore, estimating the MST weight reduces to estimating the number of connected components in the threshold graphs $G^{(1)}, \dots, G^{(w-1)}$, which is what we have done in the previous section.

2.2 The approximation algorithm

We can now get our approximation algorithm as follow:

Algorithm 3 Estimating the MST weight

for $i = 1, 2, \dots, w-1$ **do**

 Let $G^{(i)}$ contain the edges of G whose weight is at most i .

 Estimate $C^{(i)}$ to additive error $\varepsilon/(2w) \cdot n$ with failure probability at most $1/(4w)$; call the estimate $\tilde{C}^{(i)}$.

end for

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

Set $\varepsilon' = \varepsilon/(2w)$ for convenience. The bounded-degree connected-components implementation costs

$$\tilde{O}\left(\frac{\Delta}{(\varepsilon')^4}\right) = \tilde{O}\left(\frac{\Delta w^4}{\varepsilon^4}\right)$$

per call. Note that we can't save factor of Δ since we're calling the approximate connected component on subgraphs on G . Doing this for all $i = 1, 2, \dots, w-1$ therefore give a runtime of

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

In particular, this running time has no dependence on n . One can improve the running time to

$$O\left(\frac{\Delta w}{\varepsilon^2} \log\left(\frac{\Delta w}{\varepsilon}\right)\right)$$

and show that we can get a lower bound of

$$\Omega\left(\frac{\Delta w}{\varepsilon^2}\right).$$

Finally, to guarantee that the approximation above works, we see that we run the connected-components estimator at each threshold with failure probability $\delta \leq 1/(4w)$. By a union bound, all $w-1$ estimates are simultaneously correct with probability at least $1 - (w-1)/(4w) > 3/4$. On this event,

$$|M - \widetilde{M}| \leq \sum_{i=1}^{w-1} |C^{(i)} - \widetilde{C}^{(i)}| \leq (w-1) \frac{\varepsilon n}{2w} \leq \frac{\varepsilon n}{2} \leq \varepsilon M,$$

where the last inequality uses $M \geq n-1 \geq n/2$ for $n \geq 2$. Hence $(1-\varepsilon)M \leq \widetilde{M} \leq (1+\varepsilon)M$ with probability at least $3/4$.

Some further questions can be asked: which edges are actually on an MST and what should we do when the weight range w is large?