Sublinear Time Algorithms for MST

Minimum Spanning Trees (MST)

- In our last lecture, we saw algorithms that find an MST in $O(m \log n)$ time, which is near-linear in the input size.
- But can we do better?
- In this lecture, we will see an algorithm that runs in sublinear time, provided that we are only interested in approximating the cost of MST.

Query Access

- Adjacency List Model:

Minimum Spanning Trees (MST)

- **Theorem*:** There is an algorithm that runs in time $O(W^3 d\epsilon^{-2})$ and returns a value $\widetilde{MST}$ such that

$$E\left[\widetilde{MST}\right] \in (1 \pm \epsilon)MST.$$

- d: the maximum degree in the graph.
- W: maximum edge weight.
- ϵ: any desirable parameter in (0, 1].

- **Remark:** Often mere expectation is not enough and we desire a stronger bound of the form $\Pr\left[\widetilde{MST} \in (1 \pm \epsilon)MST\right] \geq 0.99$. This can also be achieved with a slightly more complicated analysis.

(* A simplified but slower variant of a result by [Chazelle, Rubinfeld, Trevisan '05])

Warm-up

- Suppose $W = 1$ (i.e., all edges are of weight 1). Then what do we know about MST?

Warm-up

- Suppose $W = 1$ (i.e., all edges are of weight 1). Then what do we know about MST?

$$MST = n - 1$$

- What if $W = 2$?

Warm-up

- What if $W = 2$?

Let $G^{(i)}$: the graph induced on edges of weight $\{1, 2, \dots, i\}$.

$C^{(i)}$: the number of connected components in $G^{(i)}$.

$$MST = (\#\text{weight 1 edges in MST}) + 2(\#\text{ weight 2 edges in MST})$$
$$= (\#\text{edges in MST}) + (\#\text{ weight 2 edges in MST})$$
$$= (n - 1) + (C^{(1)} - 1)$$
$$= n + C^{(1)} - 2$$

Claim: More generally, $MST = \mathrm{n} - \mathrm{W} + \sum_{i=1}^{W-1} C^{(i)}$.

MST via Connected Components

- **Claim:** $MST = \mathrm{n} - \mathrm{W} + \sum_{i=1}^{W-1} C^{(i)}$.

Let α_i: # of edges of weight i in MST. This gives $MST = \sum_{j=1}^{W} \alpha_i \cdot j$.

Observe that: $MST = \displaystyle\sum_{j=1}^{W} \alpha_j \cdot j = \sum_{i=1}^{W} \sum_{j=i}^{W} \alpha_j$

To see this, consider the grid:

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

The first sum goes over the rows, summing the columns, the second sum goes over the columns summing the rows.

MST via Connected Components

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

$$= \sum_{i=1}^{W} C^{(i-1)} - 1$$

$$= \sum_{i=0}^{W-1} C^{(i)} - 1$$ (Change of index)

$$= n - W - \sum_{i=1}^{W-1} C^{(i)}.$$ (Since $C^{(0)} = n$)

Implication:

To estimate MST cost, suffices to estimate # of connected components.

Estimating Connected Components

- **Theorem:** There is an algorithm that runs in time $O(\delta^{-2})$ and returns a value $\tilde{C}$ such that

$$\mathrm{E}[\tilde{C}] \in C \pm \delta n.$$

Here C is the number of connected components.

Remark: This is an *additive* approximation.

Estimating Connected Components

- For any vertex v, let S_v be the size of the connected component that v belongs to. Note that

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

- So, intuitively, if we compute S_v for a few random vertices we can estimate C, but the problem is that S_v can be quite large.
- Let us now define $S'_v = min\{S_v, 1/\delta\}$ and let $C' = \sum_v 1/S'_v$.

Claim: $|C' - C| \leq \epsilon n$.

Proof: Follows since $0 \leq \frac{1}{S'_v} - \frac{1}{S_v} \leq \delta$.

Estimating Connected Components

- Algorithm:
  - Sample a random vertex v.
  - Explore the connected component of v using BFS or DFS, truncating after visiting $2/\delta$ vertices. Let S'_v be the number of vertices seen in the component of v.
  - Return

$$\tilde{C} \leftarrow n \cdot \frac{1}{S'_v}.$$

Runtime: Every vertex spends $O(1/\delta)$ time to discover an unvisited neighbor, so the algorithm runs in total time at most $O(1/\delta^2)$.

Estimating Connected Components

- **Claim:** $E[\tilde{C}] = C'$.

Proof: $$\mathrm{E}[\tilde{C}] = E_v\left[n \cdot \frac{1}{S'_v}\right]$$

$$= n \sum_v \Pr[v \text{ sampled}] \frac{1}{S'_v}$$

$$= n \sum_v \frac{1}{n} \cdot \frac{1}{S'_v}$$

$$= \sum_v \frac{1}{S'_v}$$

$$= C'.$$

Putting Everything Together

We showed MST $= n - W - \sum_{i=1}^{W-1} C^{(i)}$.

Also, we showed:

Theorem 2: There is an algorithm that runs in time $O(\delta^{-2})$ and returns a value $\tilde{C}$ such that

$$\mathrm{E}[\tilde{C}] \in C \pm \delta n.$$

Provided that the algorithm has adjacency list access to the graph.

Note: Every adjacency list query to $C^{(i)}$ can be answered with $O(d)$ queries to the adjacency list of the original graph.

Note 2: We set $\delta \leftarrow \frac{\epsilon}{W}$. This way, our final estimate satisfies:

$$E[\widetilde{MST}] \in n - W - \sum_{i=1}^{W-1} (C^{(i)} \pm \delta n) = n - W - (\sum_{i=1}^{W-1} C^{(i)}) \pm \epsilon n$$

$$= MST \pm \epsilon n \in (1 \pm \epsilon) MST$$

Note 3: The final running time is $O(W \cdot d \cdot \delta^{-2}) = O(W^3 d \epsilon^{-2})$.