University of Wrocław: Algorithms for Big Data (Fall'19) 07/01/2020

Lecture 11: Graph Algorithms

Lecturer: *Przemysław Uznański* Scribe: -

1 Model

Graph algorithms in sublinear memory/time. Need to define reasonable model, to avoid 'just run offline algorithm'.

Input model:

- Streaming of input
- Semi-streaming (only edge insertions)
- Oracle access to input (adjacency matrix + cell probe, adjacency lists)

Output model:

- Decision: is the graph bipartite?
- Optimization: weight of MST
- Sketch/summary/coreset (summary of input, e.g. mergeable sketches)
- Local query model: does this vertex belong to MIS (needs to be consistent across independent runs)

Additional assumptions:

- Promise on the input (e.g. graph is connected)
- Decision: bipartite or ε-far from bipartite (needs to flip εn^2 edges to make it bipartite)
- small memory (total)
- small memory (per vertex)

2 Cell probe MST

Model: cell probe access to graph, given by adjacency lists. Problem: given graph G with integer weights $\{1, \ldots, w\}$, find weight of MST. Assumption: max-degree is d, which is small.

2.1 Counting connected components

Simpler problem: how to count connected components (approximately) in unweighted graph? Only hope for (fast) algorithm is additive $\pm\varepsilon n$ approximation.

Observation 1. *To estimate the number of connected components in a graph H we first pick a random vertex v. If v is in a large connected component, that is an indication of a small number of connected components.*

Denote K as number of connected components. Let $\mathrm{CC}(v)$ denote conected component of v.

Lemma 2. *For v let $\alpha_v = \frac{1}{|CC(v)|}$. Then $K = \sum_v \alpha_v$.*

Proof. (Simple.) □

Algorithm 1:

1. Sample k vertices $v_1, \dots, v_k$.
2. Find $\alpha_{v_1}, \dots, \alpha_{v_k}$.
3. Output $C = \frac{n}{k} \sum_i \alpha_{v_i}$.

Issues: how large k needs to be? Computing α_{v_i} might take $\mathcal{O}(n)$ steps.

$$\mathbb{E}[C] = \frac{n}{k} \sum_i \mathbb{E}[\alpha_{v_i}] = \frac{n}{k} \cdot k \cdot \frac{K}{n} = K$$

$$\mathrm{Var}[C] \leq \frac{n^2}{k^2} \sum_i \mathbb{E}[(\alpha_{v_i})^2] \leq \frac{n^2}{k^2} \sum_i \mathbb{E}[\alpha_{v_i}] = \frac{n^2}{k^2} \cdot k \cdot \frac{K}{n} = \frac{n}{k} K \leq \frac{n^2}{k}$$

so the additive error is with e.g. 8/9 ppb at most $3\sqrt{\mathrm{Var}[C]} \leq 3n/\sqrt{k}$. So it is enough to set $k = \mathcal{O}(\frac{1}{\varepsilon^2})$.

What about other issue of DFS/BFS taking too long on large connected components? We truncate the BFS/DFS after at most $A = \frac{1}{\varepsilon}$ steps. So large CC's are reported to be of size A. This introduces additive error of $\pm\frac{1}{A}$ each α, so $\pm\varepsilon n$ to actual output of algorithm.

Total runtime: $\mathcal{O}(\frac{1}{\varepsilon^2} \cdot \frac{1}{\varepsilon} \cdot d)$.

Amplify success to whp: repeat $\mathcal{O}(\log n)$ times and take the median.

2.2 MST from connected components

Let $G_1, G_2, \dots, G_w$ be unweighted graphs such that: $e \in G_i$ iff $w(e) \leq i$. Denote by K_i the number of connected components of G_i.

Theorem 3. *Weight of MST satisfies*

$$w(MST) = (n-1) + \sum_{i=1}^{w-1} (K_i - 1)$$

Proof. (Simple.) □

If we run each MST estimator with error $\pm(\varepsilon/w)n$ (runtime $\mathcal{O}(\frac{dw^4 \log n}{\varepsilon^3})$), the total error of estimation is $\pm\varepsilon n$. Observation: since $w(\mathrm{MST}) \geq n - 1$, this is $1 \pm \mathcal{O}(\varepsilon)$ multiplicative error.

3 Graph sketching for MST

3.1 L_0 sampling

We consider a following problem:

Definition 4. *Maintain a multiset M over universe $[n]$, under insertions and deletions, and queries for random element. Random element query returns any $x \in M$ with probability $\sim \frac{1}{\|M\|_0}$, that is any unique element from M with roughly the same probability.*

Note: random element query needs to be random when queried once. Consecutive queries might be fully correlated.

We are interested in a solution that takes $\text{poly} \log n$ space.

First assume we have a guarantee that when there is a query, $\|M\|_0 = 1$ (the trick is that M might grow large and then shring). The solution is to maintain:

- $C = \sum_{x \in M} x$
- $D = \sum_{x \in M} x^2$

and to output $\frac{D}{C}$.

Generalizing: Now, to generalize to arbitrary size. If we know the $\|M\|_0 \sim k$ for some guessed value k, we can pick a hash function $h : [n] \to [k]$ and care only about x such that $h(x) = 0$. That is:

- $C_k = \sum_{x \in M} \mathbf{1}[h(x) = 0] \cdot x$
- $D_k = \sum_{x \in M} \mathbf{1}[h(x) = 0] \cdot x^2$

Decoding if actually one element hashed: return $\frac{D_k}{C_k}$. If $k \leq \|M\|_0 \leq 2k$, then there is constant probability that there is single value $h(x) = 0$ (since this happend with ppb $1/k$ for each element). To detect failure, we can change the scheme a little bit:

- pick hash function $g : [n] \to [n]$, and maintain $C'_k = \sum_{x \in M} \mathbf{1}[h(x) = 0] \cdot x \cdot g(x)$ and $D'_k = \sum_{x \in M} \mathbf{1}[h(x) = 0] \cdot x^2 \cdot g(x)$ instead of C_k and D_k
- after we decode x and r_x, number of repetitions of x in M, we verify values C'_k and D'_k

There are other ways to detect failures, e.g. add F_0 sketch to the scheme, or maintain sketches for more powers $x, x^2, \ldots, x^p$ for some small p.

General scheme:

- Maintain independently $\log n$ levels, each responsible for $k = 1, 2, \ldots, 2^{\log n}$.
- On each level, maintain $\log n$ independent repetitions. For correct level, each repetition is ok with constant probability, we need just one to work.

Total size is poly-logarithmic.

3.2 Connected components

We want to skecth G to maintain connected components of G under edge insertions and deletions. The sketch size of $\widetilde{\mathcal{O}}(n)$ – this is necessary since initially each vertex is in its own connected component.

Sketch of algorithm:

- Initialize each vertex with separate connected component (sketch).
- Proceed in rounds:
  - in each round, each connected component picks at random one incident edge
  - all components connected by edges are merged

Lemma 5. *If K is actual number of connected components, and K_i denotes number after round i, there is $K_{i+1} - K \leq \frac{K_i - K}{2}$.*

Thus, $\log n$ rounds are enough.

Edge-based sketching: we orient arbitrarily each edge, labeling its endpoints with -1 and $+1$. Then, with each vertex v we associate function $E \to \{-1, 0, 1\}$: $v(e) = 1$ or $v(e) = -1$ if v is an endpoint of e, and otherwise $v(e) = 0$.

For each connected component $X \subseteq V$ we maintain L_0 sampler for multiset set:

$$X(e) = \sum_{v \in X} v(e)$$

For any edge, it is present in the multiset X only iff its one endpoint is in X and other endpoint is in $V \setminus X$. Thus L_0 sampling over X gives us any adjacent edge.

The L_0 sampler presented in previous subsection is actually mergeable, since all functions there were linear.

3.3 MST

Recall:

$$w(\text{MST}) = (n-1) + \sum_{i=1}^{w-1} (K_i - 1)$$

so we keep connected components sketch separately for each edge weight.

However, since we can get precise values of K_i, we can do better – we can round each edge weight down to nearest power of $(1+\varepsilon)$. This introduces $1 \pm \varepsilon$ factor, but reduces number of different edge weights to $\mathcal{O}(\frac{\log w}{\varepsilon})$.

Also note that since our sketches are linear, any edge insertion and removal can be done on the go. Total space is $\mathcal{O}(\frac{\log w \cdot \text{poly} \log n}{\varepsilon})$ per vertex, and processing time per insert/removal is the same, and query time is $\mathcal{O}(n \cdot \frac{\log w \cdot \text{poly} \log n}{\varepsilon})$