Algorithms for Big Data (VII)

Chihao Zhang

Shanghai Jiao Tong University

Nov. 1, 2019

Review

Review

We introduced the graph stream last week.

Review

We introduced the graph stream last week.

The graph has n vertices, but the edges are given in a streaming fashion.

Review

We introduced the graph stream last week.

The graph has n vertices, but the edges are given in a streaming fashion.

Compute graph properties in $o(n^2)$ time.

Review

We introduced the graph stream last week.

The graph has n vertices, but the edges are given in a streaming fashion.

Compute graph properties in $o(n^2)$ time.

This can be done for `connectivity` and `bipartiteness`.

Shortest Path

Shortest Path

Given an undirected simple graph $G = (V, E)$.

We want to answer the query "what is the minimum distance between u and v for $u, v \in V$".

Shortest Path

Given an undirected simple graph $G = (V, E)$.

We want to answer the query "what is the minimum distance between u and v for $u, v \in V$".

Our algorithm computes a subgraph $H = (V, E_H)$ of G such that

$$\forall u, v \in V, \quad d_G(u, v) \leq d_H(u, v) \leq \alpha \cdot d_G(u, v)$$

for some constant $\alpha \geq 1$.

Algorithm Shortest Path

Init:

$E_H \leftarrow \varnothing$;

On Input (u, v):

if $d_H(u, v) \geq \alpha + 1$ **then**

  $H \leftarrow H \cup \{(u, v)\}$

end if

Output: On query (u, v)

Output $d_H(u, v)$.

Clearly, $d_H(u, v) \geq d_H(u, v)$ as H contains less edges.

Clearly, $d_H(u, v) \geq d_H(u, v)$ as H contains less edges.

Consider the shortest path from u to v in G:

$$u = x_1, x_2, \ldots, x_k = v.$$

Clearly, $d_H(u, v) \geq d_H(u, v)$ as H contains less edges.

Consider the shortest path from u to v in G:

$$u = x_1, x_2, \ldots, x_k = v.$$

Then $d_G(u, v) = \sum_{i=1}^{k-1} d(x_i, x_{i+1})$.

Clearly, $d_H(u, v) \geq d_H(u, v)$ as H contains less edges.

Consider the shortest path from u to v in G:

$$u = x_1, x_2, \ldots, x_k = v.$$

Then $d_G(u, v) = \sum_{i=1}^{k-1} d(x_i, x_{i+1})$.

If $(x_i, x_{i+1}) \in E_H$, then $d_H(x_i, x_{i+1}) = d_G(x_i, x_{i+1})$.

Clearly, $d_H(u, v) \geq d_H(u, v)$ as H contains less edges.

Consider the shortest path from u to v in G:

$$u = x_1, x_2, \ldots, x_k = v.$$

Then $d_G(u, v) = \sum_{i=1}^{k-1} d(x_i, x_{i+1})$.

If $(x_i, x_{i+1}) \in E_H$, then $d_H(x_i, x_{i+1}) = d_G(x_i, x_{i+1})$.

If $(x_i, x_{i+1}) \notin E_H$, then when we are trying to insert (x_i, x_{i+1}) into E_H, it must hold that

$$d_H(x_i, x_{i+1}) \leq \alpha.$$

Clearly, $d_H(u,v) \geq d_H(u,v)$ as H contains less edges.

Consider the shortest path from u to v in G:

$$u = x_1, x_2, \ldots, x_k = v.$$

Then $d_G(u,v) = \sum_{i=1}^{k-1} d(x_i, x_{i+1})$.

If $(x_i, x_{i+1}) \in E_H$, then $d_H(x_i, x_{i+1}) = d_G(x_i, x_{i+1})$.

If $(x_i, x_{i+1}) \notin E_H$, then when we are trying to insert (x_i, x_{i+1}) into E_H, it must hold that

$$d_H(x_i, x_{i+1}) \leq \alpha.$$

In all, we have

$$d_H(u,v) \leq \alpha \cdot d_G(u,v).$$

Space Consumption

Space Consumption

We need a bit of graph theory to analyze the space consumption.

Space Consumption

We need a bit of graph theory to analyze the space consumption.

The girth $g(G)$ of a graph G is the length of its shortest cycle.

Space Consumption

We need a bit of graph theory to analyze the space consumption.

The girth $g(G)$ of a graph G is the length of its shortest cycle.

It is clear that $g(H) \geq \alpha + 2$.

Space Consumption

We need a bit of graph theory to analyze the space consumption.

The girth $g(G)$ of a graph G is the length of its shortest cycle.

It is clear that $g(H) \geq \alpha + 2$.

Theorem

Let $G = (V, E)$ be a sufficiently large graph with $g(G) \geq k$. Let $n = |V|$ and $m = |E|$. Then

$$m \leq n + n^{1+\frac{1}{\lfloor \frac{k-1}{2} \rfloor}}.$$

The k-core of a graph G is a subgraph whose degree is at least k.

The k-core of a graph G is a subgraph whose degree is at least k.

Let $d = 2m/n$ be the average degree of G, then G contains a $d/2$-core. (Why?)

The k-core of a graph G is a subgraph whose degree is at least k.

Let $d = 2m/n$ be the average degree of G, then G contains a $d/2$-core. (Why?)

The $d/2$-core has girth at least k, so we can find a BFS tree in it with depth $\lfloor \frac{k-1}{2} \rfloor$ and width $\frac{d}{2} - 1$.

The k-core of a graph G is a subgraph whose degree is at least k.

Let $d = 2m/n$ be the average degree of G, then G contains a $d/2$-core. (Why?)

The $d/2$-core has girth at least k, so we can find a BFS tree in it with depth $\lfloor \frac{k-1}{2} \rfloor$ and width $\frac{d}{2} - 1$.

The number of the vertices satisfies

$$n \geq \left(\frac{d}{2} - 1\right)^{\lfloor \frac{k-1}{2} \rfloor} = \left(\frac{m}{n} - 1\right)^{\lfloor \frac{k-1}{2} \rfloor}.$$

The k-core of a graph G is a subgraph whose degree is at least k.

Let $d = 2m/n$ be the average degree of G, then G contains a $d/2$-core. (Why?)

The $d/2$-core has girth at least k, so we can find a BFS tree in it with depth $\lfloor \frac{k-1}{2} \rfloor$ and width $\frac{d}{2} - 1$.

The number of the vertices satisfies

$$n \geq \left(\frac{d}{2} - 1\right)^{\lfloor \frac{k-1}{2} \rfloor} = \left(\frac{m}{n} - 1\right)^{\lfloor \frac{k-1}{2} \rfloor}.$$

This bound is in fact tight, can you prove it?

Matchings

Matchings

Let $G = (V, E)$ be a graph, a matching $M \subseteq E$ consisting of edges sharing no vertex.

Matchings

Let $G = (V, E)$ be a graph, a matching $M \subseteq E$ consisting of edges sharing no vertex.

The problem of finding maximum matching is a famous polynmial-time solvable problem.

Matchings

Let $G = (V, E)$ be a graph, a matching $M \subseteq E$ consisting of edges sharing no vertex.

The problem of finding maximum matching is a famous polynmial-time solvable problem.

Now we try to approximate it in the streaming setting.

Algorithm Maximum Matching

Init:

$M \leftarrow \varnothing$;

On Input (u, v):

if $M \cup \{(u, v)\}$ is a matching **then**

  $M \leftarrow M \cup \{(u, v)\}$

end if

Output:

Output $|M|$.

Let $\widehat{M}$ denote our estimate and M^* denote the maximum matching.

Let $\widehat{M}$ denote our estimate and M^* denote the maximum matching.

Theorem

$$\frac{|M^*|}{2} \leq \left|\widehat{M}\right| \leq |M^*|.$$

Let $\widehat{M}$ denote our estimate and M^* denote the maximum matching.

Theorem

$$\frac{|M^*|}{2} \leq \left|\widehat{M}\right| \leq |M^*| .$$

M^* is a maximal matching. Each $e \in M$ intersects at most two edges in M^*.

Maximum Weighted Matching

Maximum Weighted Matching

Each edge $e \in E$ is associated with a non-negative weight $w(e) \geq 0$.

Maximum Weighted Matching

Each edge $e \in E$ is associated with a non-negative weight $w(e) \geq 0$.

Compute a matching M to maximize $\sum_{e \in M} w(e)$.

Algorithm Maximum Weighted Matching

Init: $M \leftarrow \varnothing$;

On Input (u, v):

if $M \cup \{(u, v)\}$ is a matching **then** $M \leftarrow M \cup \{(u, v)\}$

else

  $C \leftarrow \{e \in M : u \in e \vee v \in e\}$;

  if $w(u, v) > 2w(C)$ **then** $M \leftarrow (M \setminus C) \cup \{(u, v)\}$;

  end if

end if

Output: Output $|M|$.

Analysis

Analysis

We use a charging argument to analyze the algorithm.

Analysis

We use a charging argument to analyze the algorithm.

We call an edge e:

- born if we added it to M;
- die if it was removed from M;
- murdered by e' if it dies because we add e'.

Analysis

We use a charging argument to analyze the algorithm.

We call an edge e:

- born if we added it to M;
- die if it was removed from M;
- murdered by e' if it dies because we add e'.

For every $e \in M$, we define the family of victims:

$$C_0(e) = \{e\}, C_1(e) = \text{edges murdered by } e, \ldots, C_i(e) = \bigcup_{f \in C_{i-1}(e)} \text{edges murdered by } f, \ldots$$

Lemma

For every e,

$$w\left(\bigcup_{i\geq 1} C_i(e)\right) \geq w(e).$$

Lemma

For every e,

$$w\left(\bigcup_{i\geq 1} C_i(e)\right) \geq w(e).$$

Proof.

By the definition of murdering, $w(C_{i+1}) \leq w(C_i)/2$. Therefore

$$2\sum_{i\geq 1} w\left(C_i(e)\right) \leq \sum_{i\geq 0} w(C_i) = w(e) + \sum_{i\geq 1} w(C_i).$$

□

Lemma

$$w(M^*) \leq \sum_{e \in M} \left(4w(e) + 2w \left(\bigcup_{i \geq 1} C_i(e) \right) \right).$$

Lemma

$$w(M^*) \le \sum_{e \in M} \left(4w(e) + 2w\left(\bigcup_{i \ge 1} C_i(e) \right) \right).$$

We consider $e_1^*, e_2^*, \ldots$ of M^* in the order of the stream.

- if e_i^* is born, charge $w(e_i^*)$ to e_i^*;
- if e_i^* is not born, charge $w(e_i^*)$ to its conflicting edges ($w^*(e)$ is divided proportional to the weight of the conflicting edges);
- if some $e' = (u, v)$ murdered some $e = (u', v)$ and e has been charged by some $e^* = (u'', v)$, then move the charge from e to e'.

At last, we have

- for every $e \in M$, its charge is at most $4w(e)$;
- for every $e \in \bigcup_{i \geq 1} C(e')$ for some e', its charge is at most $2w(e)$.

At last, we have

- for every $e \in M$, its charge is at most $4w(e)$;
- for every $e \in \bigcup_{i \geq 1} C(e')$ for some e', its charge is at most $2w(e)$.

Therefore,

$$w(M^*) \leq \sum_{e \in M} \left(4w(e) + 2w\left(\bigcup_{i \geq 1} C_i\right)\right) \leq 6w(M).$$

At last, we have

- for every $e \in M$, its charge is at most $4w(e)$;
- for every $e \in \bigcup_{i \geq 1} C(e')$ for some e', its charge is at most $2w(e)$.

Therefore,

$$w(M^*) \leq \sum_{e \in M} \left(4w(e) + 2w \left(\bigcup_{i \geq 1} C_i \right) \right) \leq 6w(M).$$

The analysis is not pushed to the limit yet, can you improve the approximation ratio 6? (Exercise)

Counting Triangles

Counting Triangles

An important topic is to compute the number of some fixed subgraph in a graph in the streaming setting.

Counting Triangles

An important topic is to compute the number of some fixed subgraph in a graph in the streaming setting.

We study a simple algorithm for counting triangles.

Counting Triangles

An important topic is to compute the number of some fixed subgraph in a graph in the streaming setting.

We study a simple algorithm for counting triangles.

Consider an vector $\mathbf{f} = (f_T)_{T \in \binom{[n]}{3}}$, where for every $T = x, y, z$, $f_T = |\{\{x, y\}, \{x, z\}, \{y, z\}\} \cap E|$.

Counting Triangles

An important topic is to compute the number of some fixed subgraph in a graph in the streaming setting.

We study a simple algorithm for counting triangles.

Consider an vector $\mathbf{f} = (f_T)_{T \in \binom{[n]}{3}}$, where for every $T = x, y, z$, $f_T = |\{\{x, y\}, \{x, z\}, \{y, z\}\} \cap E|$.

So if for some $T = \{x, y, z\}$, $f_T = 3$, then x, y, z is a triangle in G.

Counting Triangles

An important topic is to compute the number of some fixed subgraph in a graph in the streaming setting.

We study a simple algorithm for counting triangles.

Consider an vector $\mathbf{f} = (f_T)_{T \in \binom{[n]}{3}}$, where for every $T = x, y, z$, $f_T = |\{\{x, y\}, \{x, z\}, \{y, z\}\} \cap E|$.

So if for some $T = \{x, y, z\}$, $f_T = 3$, then x, y, z is a triangle in G.

The algorithm simply outputs $F_0 - 1.5F_1 + 0.5F_2$, where $F_i = \|\mathbf{f}\|_i^i$.

We can expand $F_0 - 1.5F_1 + 0.5F_2$ as

$$\sum_{T \in \binom{[n]}{3}} 0.5f_T^2 - 1.5f_T + \mathbf{1}[f_T \neq 0].$$

We can expand $F_0 - 1.5F_1 + 0.5F_2$ as

$$\sum_{T \in \binom{[n]}{3}} 0.5 f_T^2 - 1.5 f_T + \mathbf{1}[f_T \neq 0].$$

The "polynomial" $f(x) = 0.5x^2 - 1.5x + \mathbf{1}[x \neq 0]$ satisfies

- $f(0) = f(1) = f(2) = 0$;
- $f(3) = 1$.

We can expand $F_0 - 1.5F_1 + 0.5F_2$ as

$$\sum_{T \in \binom{[n]}{3}} 0.5f_T^2 - 1.5f_T + \mathbf{1}[f_T \neq 0].$$

The “polynomial” $f(x) = 0.5x^2 - 1.5x + \mathbf{1}[x \neq 0]$ satisfies

- $f(0) = f(1) = f(2) = 0$;
- $f(3) = 1$.

The multiplicative error of the algorithm is unbounded!