Lec 2 Sublinear Time Algorithms

Today:

Sublinear time algorithms for:

(1) Estimating MST weight

(2) Estimating Average degree

Recall from last time:

- Can estimate # connected components of degree $\leq d$ graph to within $\underbrace{\pm \varepsilon n}_{\text{fall in this range with prob } \geq 3/4 \text{ "success probability"}}$ in time $O(d/\varepsilon^4)$

- HW0 $\Rightarrow$ thus can give estimate to $\underbrace{\text{within } \pm \varepsilon n}_{\text{falls in this range with prob } \geq 1-\beta}$ in time $O(d/\text{poly}(\varepsilon) \cdot \log \frac{1}{\beta})$

"failure probability"

Approximating Min Spanning Tree (MST)

Input (1) $G=(V,E)$ adj list representation $n=|V|$

max degree d

each edge has weight $w_{uv} \in \{1..w\} \cup \{\infty\}$

ie. $w_{uv} \notin E$

G connected

(2) ε

Output

let $M = \min_{T \text{ spans } G} \{ w(T) \}$

tree   tree touches every node $\sum_{(i,j)\in T} w_{ij}$

output $\hat{M}$ s.t. $(1-\varepsilon)M \le \hat{M} \le (1+\varepsilon)M$ $\Leftarrow$ multiplicative approximation

Assumption on wts $\Rightarrow$ $n-1 \le w(T) \le w(n-1)$

A different characterization of MST:

def $G^{(i)} = (V, E^{(i)})$ where $E^{(i)} = \{(u,v) \mid w_{uv} \in \{1..i\}\}$

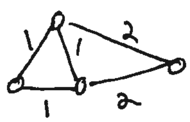
$c^{(i)}$ = # conn comp of $G^{(i)}$

Some examples before characterization:

1) $w=1$ only size 1 weights + connected by assumption

here $M=n-1$

2) $w=2$ weights $\in \{1,2\}$



$G^{(1)}$

$C^{(1)}=2$

idea of Kruskal:

use as many wt 1 edges as you can

only need wt 2 edges to connect the components

$\Rightarrow$ need $C^{(1)}-1$ wt 2 edges in MST

(recall that total # of edges is $n-1$)

Total wt of MST:

$$M = \underbrace{(n-1)}_{\substack{\uparrow \\ \text{1 for each edge}}} + \underbrace{(C^{(1)}-1)}_{\substack{\uparrow \\ \text{additional 1 for wt 2 edges}}} = n-2+C^{(1)}$$

<u>Claim</u> $M = n - w + \sum_{i=1}^{w-1} C^{(i)}$

<u>Pf</u>.

let α_i = # edges of wt i in any MST of G

Kruskal's tells us that <u>all</u> MST's have <u>same</u> value of α_i

why?

$$\sum_{i \geq l} \alpha_i = \# \text{ conn comp of } G^{(l)} - 1$$
$$= C^{(l)} - 1$$

where $C^{(0)} = n$ (no edges in $G^{(0)}$)

$$M = \sum_{i=1}^{w} i \cdot \alpha_i$$
$$= \sum_{i=1}^{w} \alpha_i + \sum_{i=2}^{w} \alpha_i + \sum_{i=3}^{w} \alpha_i + \ldots + \underbrace{\sum_{i=w}^{w} \alpha_i}_{= \alpha_w}$$
$$= (n-1) + (C^{(1)} - 1) + (C^{(2)} - 1) + \ldots + (C^{(w-1)} - 1)$$
$$= n - w + \sum_{i=1}^{w-1} C^{(i)}$$

$\blacksquare$

Approximation Algorithm:

For $i = 1$ to $w-1$

$\hat{c}^{(i)}$ = approx # c.c. of $G^{(i)}$ to within $\overbrace{\frac{\varepsilon}{2w}}^{\varepsilon'} \cdot n$ (additive error)

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

adds poly$(\frac{w}{\varepsilon})$ factor to runtime

Runtime:

$\tilde{O}(d/(\varepsilon')^4) = \tilde{O}(d w^4/\varepsilon^4)$ for each call to approx # c.c.

how do you recompute $G^{(i)}$? ignore edges of wt $> i$

Total $\tilde{O}(d w^5/\varepsilon^4)$

(Can improve to $O(\frac{dw}{\varepsilon^2} \log \frac{dw}{\varepsilon})$

+ need $\Omega(dw/\varepsilon^2)$)

Approximation guarantee:

"failure" if approx error of approx #cc is too big ($> \varepsilon'$)

how? HW0!

Call approx #cc with "failure" probability $\leq \frac{1}{4 \cdot w}$

Pr[all calls to approx #cc give output that is ε' additive approx] $\geq 1 - \frac{w}{4w}$ $= 3/4$

union bound

If ↑ happens: $|M - \hat{M}| \leq w \cdot \frac{\varepsilon n}{2w} = \frac{\varepsilon n}{2}$ ← small additive error

this is where we use lower bnd on edge wts → since $M \geq n-1 \geq n/2$, $|M - \hat{M}| \leq \varepsilon M$ ← small multiplicative error

$\boxed{\text{Conclusion:}}$ runtime depends only on $d, w, 1/\varepsilon$
gives additive/multiplicative error

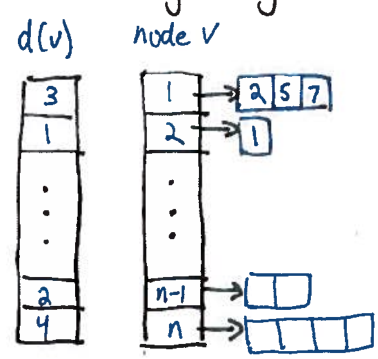
Approximating Average Degree

def Average degree $\bar{d} = \dfrac{\sum_{u \in V} d(u)}{n}$

Assume: G simple (no parallel edges, self-loops)

$\Omega(n)$ edges (not "ultra-sparse")

representation: adjacency list + degrees



- degree queries: on v return $d(v)$
- neighbor queries: for (v,j) return j^{th} nbr of v

Naive sampling:

Pick ?? sample nodes $v_1 \cdots v_s$

output $\frac{1}{s} \sum_i d(v_i)$ (ave degree of sample)

using straightforward Chernoff/Hoeffding $\Rightarrow$ $\Omega(n)$ samples needed

Degree sequences are special?

$(n-1, 0, 0, 0, \ldots, 0)$ not possible

$(n-1, 1, 1, 1, \ldots, 1)$ is possible

Some lower bounds:

"ultrasparse case":

need linear time to get any multiplicative approx

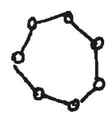
graph with 0 edges vs. graph with 1 edge

ave deg = 0 ave deg = $\frac{1}{n}$

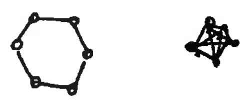
need $\Omega(n)$ queries to distinguish

ave deg ≥ 2:

n-cycle $\bar{d} = 2$



$n - cn^{1/2}$ cycle + $cn^{1/2}$-clique $\bar{d} \approx 2 + c^2$



need $\Omega(n^{1/2})$ queries to find clique node

Algorithm idea:

group nodes of similar degrees

estimate average w/in each group

+ each group has bounded variance

− doesn't work for estimating ave of arbitrary numbers, why should it work here?

Bucketing:

set parameters $\beta = \varepsilon/c$

$t = O(\log n/\varepsilon)$ # buckets

$$B_i = \{v \mid (1+\beta)^{i-1} < d(v) \le (1+\beta)^i\}$$

for $i \in \{0..(t-1)\}$ ⇐ can add extra bucket for degree 0 nodes or can assume (for now) that there are none (lets do the latter)

Note:

total degree of nodes in B_i

$$(1+\beta)^{i-1}|B_i| \le d_{B_i} \le (1+\beta)^i |B_i|$$

total degree of graph

$$\sum_i (1+\beta)^{i-1}|B_i| \le d_{total} \le \sum_i (1+\beta)^i |B_i|$$

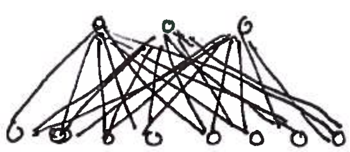
First idea for algorithm:

- Take sample S of nodes
- $S_i \leftarrow S \cap B_i$ (samples that fall in i^{th} bucket use degree queries to determine this)
- estimate average degree contribution from B_i using S_i

  i.e. $\rho_i \leftarrow \frac{|S_i|}{|S|}$ $\leftarrow$ note: $\forall i$ $E[\rho_i] = E\left[\frac{|S_i|}{|S|}\right] = \frac{|B_i|}{n}$
- Output $\sum_i \rho_i (1+\beta)^{i-1}$ $\nwarrow$ undercounting

Problem: $\underbrace{i \text{ s.t. } |S_i| \text{ is small}}_{\text{likely come from } i \text{ s.t. } |B_i| \text{ small}}$ $\leftarrow$ for these, our estimate of $|S_i|$ could be terrible

example of problem:



$\leftarrow$ 3 nodes, deg $n-3$

$\leftarrow$ $n-3$ nodes, deg 3

$a \leftarrow i$ s.t. $(1+\beta)^{i-1} \leq 3 \leq (1+\beta)^i$ $\quad |B_a| = n-3$ contributes $(n-3)\cdot 3$ edges

$b \leftarrow i$ s.t. $(1+\beta)^{i-1} \leq n-3 \leq (1+\beta)^i$ $\quad |B_b| = 3$ contributes $3\cdot(n-3)$ edges $\leftarrow$ Never sampled but contributes ½ edges!!

$\forall c \neq a,b \quad |B_c| = 0$

Still, maybe good enough for 2-approximation?

Next idea: use "0" for small buckets

Algorithm:

- sample S $\leftarrow$ how big?
- $S_i \leftarrow S \cap B_i$
- For all i

$\leftarrow$ so $|S| > \overset{\text{\# buckets}}{t} \sqrt{\frac{n}{\varepsilon}} = \Omega\left(\frac{\log n}{\varepsilon} \cdot \sqrt{\frac{n}{\varepsilon}}\right)$

let $|S| = \Theta(\sqrt{n}$ polylog n $\times$ poly $1/\varepsilon)$

$\Rightarrow |S_i| \geq \Omega($polylog $n \times$ poly $1/\varepsilon)$

if $|S_i| \geq \sqrt{\frac{\varepsilon}{n}} \cdot \frac{|S|}{c \cdot t}$

use $\rho_i \leftarrow \frac{|S_i|}{|S|}$ $\Leftarrow$ call i "big"

else $\rho_i \leftarrow 0$ $\Leftarrow$ call i "small"

- output $\sum_i \rho_i (1+\beta)^{i-1}$

Analysis:

1) Output not too large

idealistic (but unrealistic) case $\Rightarrow$ Suppose $\forall i$ $\rho_i = \frac{|B_i|}{n}$, then $\sum_i \rho_i (1+\beta)^{i-1} = \sum_i \frac{|B_i|}{n} \underbrace{(1+\beta)^{i-1}}_{\leq \text{ deg of node in } B_i}$

$\leq \bar{d}$

realistic case Suppose $\forall_i$ $\rho_i \leq \frac{|B_i|}{n} \underbrace{(1+\gamma)}_{\text{bound on sampling error}}$ when $|S_i|$ is big (note that trivial when $|S_i|$ not big since $\rho_i \leftarrow 0$)

$$\Rightarrow \sum_i \rho_i (1+\beta)^{i-1} \leq \bar{d}\,(1+\gamma)$$

✓

ave deg 6

2) Can output be too small?

if $\forall i \quad \rho_i = \frac{|B_i|}{n}$ then $\sum_i \rho_i (1+\beta)^{i-1} = \sum_i \frac{|B_i|}{n} (1+\beta)^{i-1}$

$$\geq (1-\beta) \sum_i \frac{|B_i|}{n} \underbrace{(1+\beta)^i}_{\geq \text{ deg of node in } B_i}$$

$$\geq (1-\beta)\,\bar{d}$$

By sampling, for big i, $\rho_i \geq \frac{|B_i|}{n}(1-\gamma)$

For small i ????

How much undercounting?

divide edges into 3 types:

type is determined by run of algorithm

1) big-big - both endpts in big buckets — counted twice
2) big-small - one endpt in big bucket — counted once
   " " " small "
3) small-small - both endpts in small buckets — never counted

[See example]

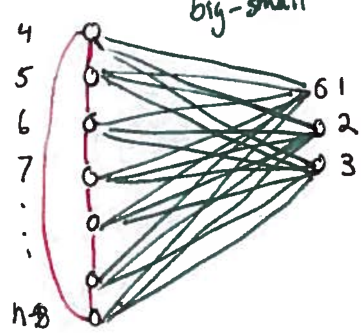
note: big-big + big-small get counted (off by factor of two)
but small-small can be a real problem

ave deg 7

Example

big-big

big-small



ave deg 5

bucket a

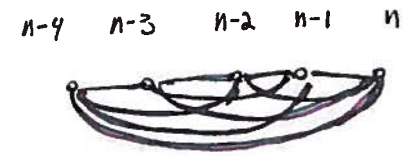
big

ave deg n-5

bucket b

Small

Small-small



5-clique

ave deg 4

bucket c

Small

Total degree

LHS of Biparhte ⇓ RHS of biparhte ⇓ 5-clique ⇓

$5(n-8) + (n-8)(3) + 4\cdot 5 = 8(n-8)+20$

ave deg $\approx 8n$ Algorithm will output $\approx 5n$

Samples

6, 104, 22, 157, 74, 41, ...

bucket a

ϕ

bucket b

ϕ

bucket c

↑

most nodes here

$\Rightarrow$ (whp) bucket a is big, in fact,

whp $\rho_a \leftarrow 1$

output ≈ 5

very few nodes in these buckets

so unlikely to see any samples

$\Rightarrow$ (whp) b & c are small

$\rho_b \leftarrow 0$ $\rho_c \leftarrow 0$

big-small edges: $3(n-8)$

Fraction of big-big + big-small : $\frac{3(n-8)}{5(n-8)} = 3/5$

$E[a_j] = \frac{3}{5}$

Output $1\cdot(1+\frac{3}{5})(1+\underbrace{\rho)^a}_{\approx 5} \approx 8$