

## 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  $\pm \epsilon n$  in time  $O(d/\epsilon^4)$

fall in this range with prob  $\geq 3/4$  "success probability"

- HWO  $\Rightarrow$  thus can give estimate to within  $\pm \epsilon n$  in time  $O(d/\text{poly}(\epsilon) \cdot \log \frac{1}{\beta})$

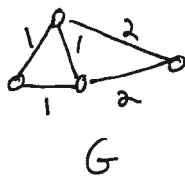
falls in this range with prob  $\geq 1 - \beta$  "failure probability"



Some examples before characterization:

1)  $w=1$  only size 1 weights + connected by assumption  
here  $M=n-1$

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



$$C^{(1)} = 2$$

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:

$$M = (n-1) + (C^{(1)} - 1) = n - 2 + C^{(1)}$$

$\uparrow$   
1 for each  
edge

$\uparrow$  additional 1  
for wt 2 edges

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

Pf.

let  $\alpha_i = \#$  edges of wt  $i$  in any MST of  $G$

$\uparrow$   
Kruskal's tells us that all MST's have same value of  $\alpha_i$   
why?

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

$$= C^{(l)} - 1$$

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

$$M = \sum_{i=1}^w i \cdot \alpha_i$$

$$= \sum_{i=1}^w \alpha_i + \sum_{i=2}^w \alpha_i + \sum_{i=3}^w \alpha_i + \dots + \underbrace{\sum_{i=w}^w \alpha_i}_{= \alpha_w}$$

$$= (n-1) + (C^{(1)} - 1) + (C^{(2)} - 1) + \dots + (C^{(w-1)} - 1)$$

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



### Approximation Algorithm :

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

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

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

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

### Runtime :

$\tilde{O}(d/(\epsilon^i)^4) = \tilde{O}(dw^4/\epsilon^4)$  for each call to approx # c.c.

Total  $\tilde{O}(dw^5/\epsilon^4)$

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

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

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

Approximation guarantee: "failure" if approx error of approx #cc is too big ( $> \epsilon^i$ )

how?  
Hwo!

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

Pr [all calls to approx #cc give output that is  $\epsilon^i$  additive approx]  $\geq 1 - \frac{w}{4w}$  ← union bound

If happens:  $|M - \hat{M}| \leq w \cdot \frac{\epsilon n}{2w} = \frac{\epsilon n}{2}$  ← small additive error = 3/4

→ since  $M \geq n-1 \geq n/2$ ,  $|M - \hat{M}| \leq \epsilon M$  ← small multiplicative error

this is where we use lower bound on edge wts

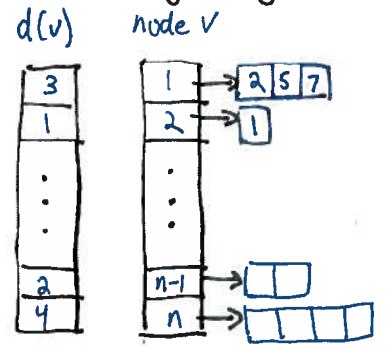
**Conclusion:** runtime depends only on  $d, w, 1/\epsilon$  gives additive/multiplicative error

# Approximating Average Degree

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

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  $j^{\text{th}}$  nbr of  $v$

## Naive sampling:

Pick ?? sample nodes  $v_1 \dots v_s$

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

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

Degree sequences are special?

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

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

Some lower bounds:

"Ultrasparse case":

need linear time to get any multiplicative approx

graph with 0 edges

ave deg = 0

vs,

graph with 1 edge

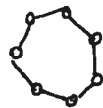
ave deg =  $\frac{1}{n}$



need  $\Omega(n)$  queries  
to distinguish

ave deg  $\geq 2$ :

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



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



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

$$\beta = \epsilon / c$$

$$t = O(\log n / \epsilon) \quad \# \text{ buckets}$$

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

for  $i \in \{0 \dots (t-1)\}$

← can add extra bucket for degree 0 nodes

or can assume (for now) that there are none (let's do the latter)

Note:

total degree of nodes in  $B_i$

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

total degree of graph

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



First idea for algorithm:

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

using  $S_i$

ie.  $p_i \leftarrow \frac{|S_i|}{|S|}$

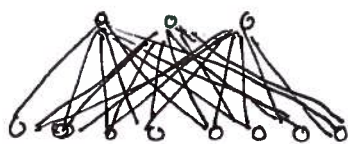
note:  $\forall i$   
 $E[p_i] = E\left[\frac{|S_i|}{|S|}\right]$   
 $= \frac{|B_i|}{n}$

Output  $\sum_i p_i (1+\beta)^{i-1}$  ← undercounting

Problem:

$i$  st.  $|S_i|$  is small for these, our estimate of  $|S_i|$  could be terrible  
 likely come from  $i$  st.  $|B_i|$  small

example of problem:



← 3 nodes, deg  $n-3$

←  $n-3$  nodes, deg 3

$a \leftarrow i$  st.  $(1+\beta)^{i-1} \leq 3 \leq (1+\beta)^i$

$b \leftarrow i$  st.  $(1+\beta)^{i-1} \leq n-3 \leq (1+\beta)^i$

$|B_a| = n-3$

$|B_b| = 3$

contributes  $(n-3) \cdot 3$  edges

contributes  $3 \cdot (n-3)$  edges

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

Still, maybe good enough for 2-approximation?

Never sampled but contributes  $\frac{1}{2}$  edges !!

Next idea: use "0" for small buckets

Algorithm:

- sample  $S$  ← how big?
- $S_i \leftarrow S \cap B_i$

• For all  $i$

if  $|S_i| \geq \sqrt{\frac{\epsilon}{n}} \cdot \frac{|B_i|}{c \cdot t}$

use  $p_i \leftarrow \frac{|S_i|}{|S|}$

call  $i$  "big"

else

$p_i \leftarrow 0$

call  $i$  "small"

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

← so  $|S| > t \sqrt{\frac{n}{\epsilon}} = \Omega\left(\frac{\log n}{\epsilon} \cdot \sqrt{\frac{n}{\epsilon}}\right)$  (# buckets)

let  $|S| = \Theta(\sqrt{n} \text{ poly log } n \times \text{poly } 1/\epsilon)$

$\Rightarrow |S_i| \geq \Omega(\text{poly log } n \times \text{poly } 1/\epsilon)$

Analysis:

1) Output not too large

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

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

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

2) Can output be too small?

$$\begin{aligned} \text{if } \forall i \quad p_i = \frac{|B_i|}{n} \quad \text{then} \quad \sum_i p_i (1+\beta)^{i-1} &= \sum_i \frac{|B_i|}{n} (1+\beta)^{i-1} \\ &\geq (1-\beta) \sum_i \frac{|B_i|}{n} (1+\beta)^i \\ &\geq (1-\beta) \bar{d} \end{aligned}$$

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

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

For small  $i$  ????

How much undercounting?

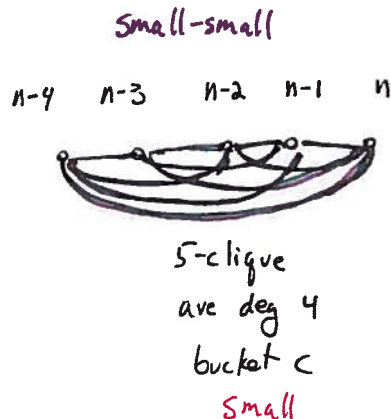
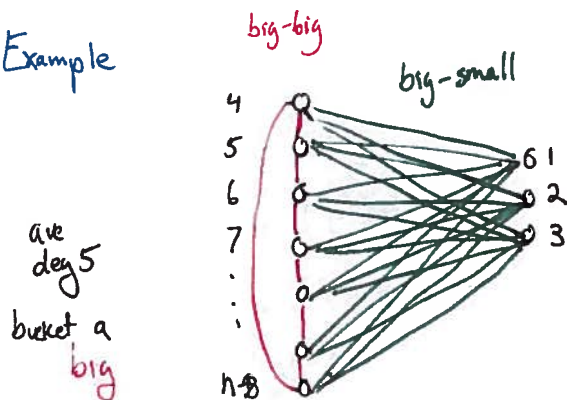
divide edges into 3 types:

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 " " " small "	counted once
		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

Example

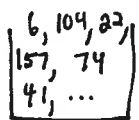


Total degree

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

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

Samples



bucket a



bucket b



bucket c

↑  
most nodes here

⇒ (whp) bucket a is big, in fact,  
whp  $P_a \leftarrow 1$

output  $\approx 5$

↙ ↘  
very few nodes in these buckets  
so unlikely to see any samples

⇒ (whp) b + c are small

$$P_b \leftarrow 0 \quad P_c \leftarrow 0$$

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

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

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

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