Sublinear Subgraph Counting

C. Seshadhri (Seshadhri Comandur)

Dept. of Computer Science & Engg
UC Santa Cruz

Amazon Scholar
AWS

Thanks to my teachers

Manindra Agrawal

Bernard Chazelle

Mike Saks

Tamara Kolda

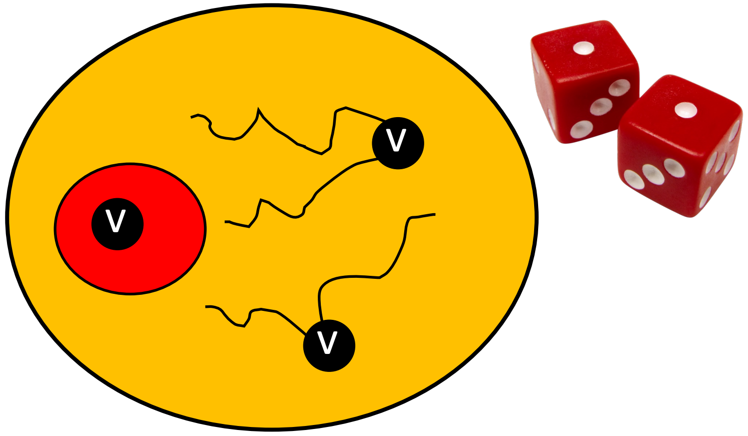
Ali Pinar

Thanks to my collaborators

Talya Eden

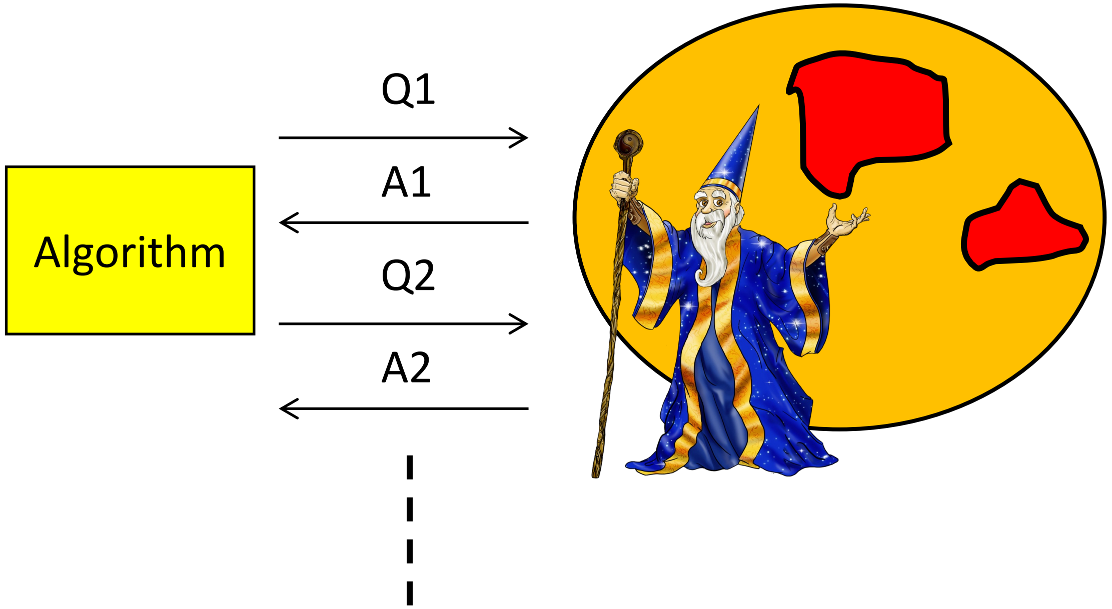
Dana Ron

Sublinear graph algorithms

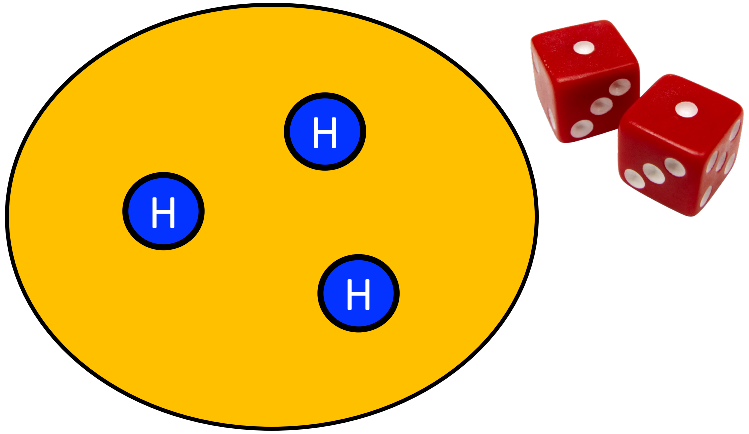


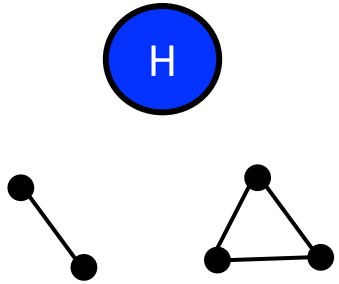
- How much of a graph needs to be seen for an (approximate) algorithmic task?

- How to sample a large graph?



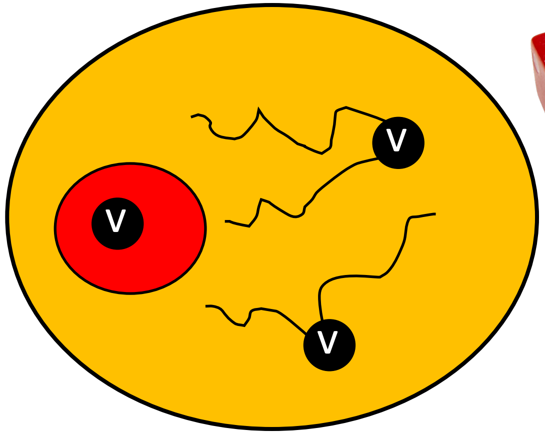
Sublinear subgraph counting





- Approximate H-count in G
- G is simple, undirected
  - Think about G sparse, results hold in general
  - Not property testing!
- G stored as adjacency list

What’s the model?



n known initially

V is not known
Nothing else known

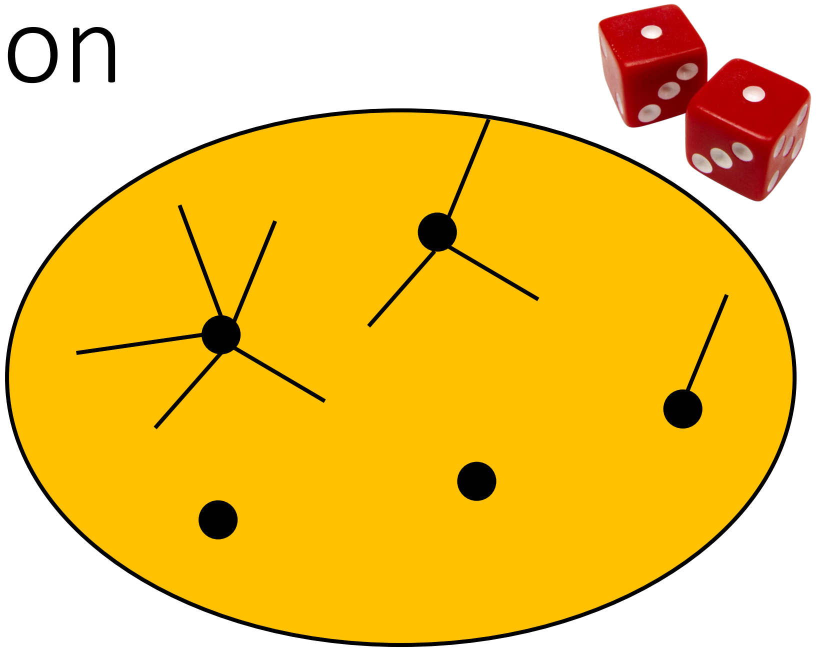
Algorithm can crawl/BFS from some random starting vertices

- [Goldreich-Ron 02] “Standard sparse graph model”
- Vertex query: Get a uniform random vertex
- Degree query: For vertex v, get degree d_v
- Neighbor query: For vertex v, get a uar neighbor u
- Edge query: Given vertices u, v, check if edge (u,v) present
- (Get uar edge)

Tool #1: Heavy vertices/edges

Shedding weight

A simple question

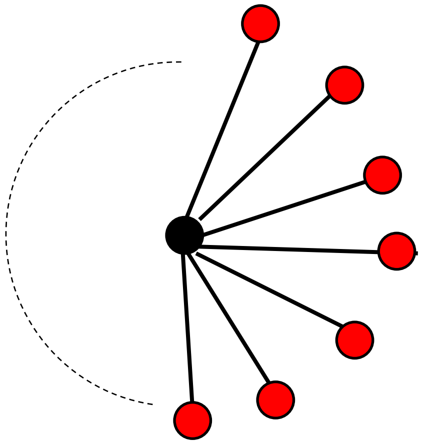


m = #edges
n = #vertices

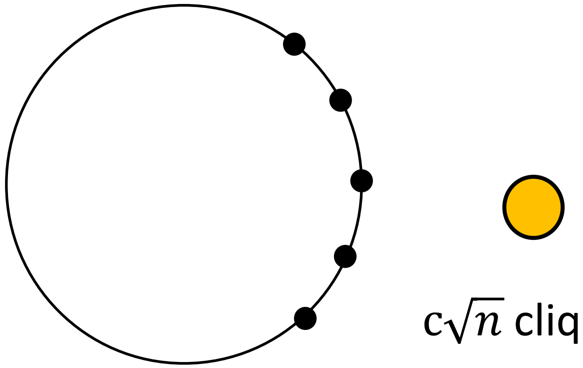
Avg degree = $\sum_v d_v$/n
$\bar{d}$ = 2m/n

- Estimate the average degree of a graph
- Consider "obvious procedure": sample uniform random vertices, take the average
- [Feige 02] O($\sqrt{n}$) samples give a 2-approximation
  - Average is in [m/n, 2m/n]
  - For n = 10^8, only 10,000 samples!

Why 2? And why $\sqrt{n}$?



Star graph



c$\sqrt{n}$ clique

$n - c\sqrt{n}$ cycle

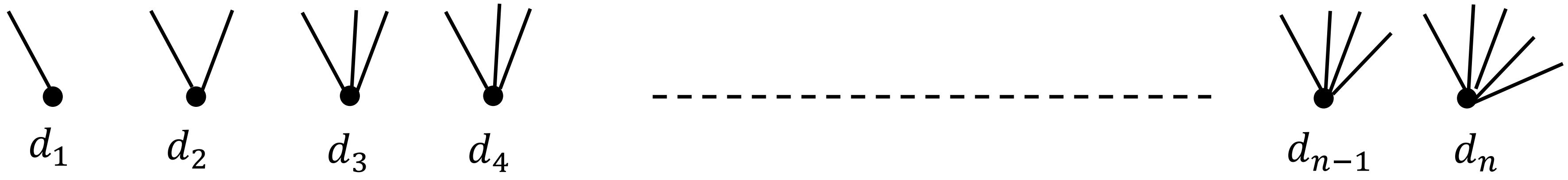
Average degree ≈ 2

o(n) samples only leaves, so empirical avg = 1

Average degree ≈ c/2

<< $\sqrt{n}/c$ samples will not hit clique. Empirical avg = 2

The variance problem



$\mathbf{E}[X_1] = \overline{d}$

$\text{var}[X_1] \le \frac{\sum_v d_v^2}{n}$

Choose k iid samples, so $\mathbf{E}[X] = k \cdot \overline{d}$

$\text{var}[X] = k \cdot \text{var}[X_1]$

Chebyshev

$$\Pr\Big[|X - \mathbf{E}[X]| \ge \varepsilon \mathbf{E}[X]\Big] \le \frac{\text{var}[X]}{\varepsilon^2 \mathbf{E}[X]^2} = \frac{\text{var}[X_1]}{\varepsilon^2 k \mathbf{E}[X_1]^2}$$

$$k \approx \frac{\text{var}[X_1]}{\mathbf{E}[X_1]^2} \qquad k \approx \frac{\sum_v d_v^2}{n\overline{d}^2}$$

We can have numerator n^2
but denominator Θ(n)

We can have avg deg = O(1)
but avg sq deg = Ω(n)

The variance problem

$$k \approx \frac{\sum_v d_v^2}{n\bar{d}^2}$$

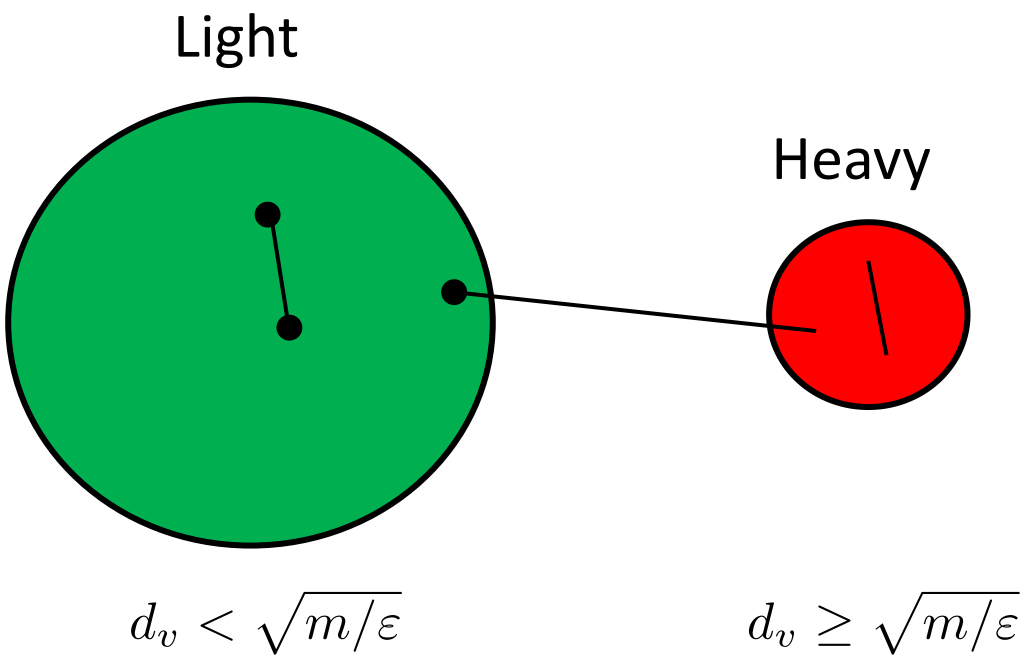
- Need to reduce variance
- Can we simply drop “large” outcomes?
  - Word of the day: Winsorize

But these are degrees!

Sum of degrees of light vertices ≥

2 x (Light–light edges) + (Light-heavy edges) ≥

$$(1-\varepsilon)m$$

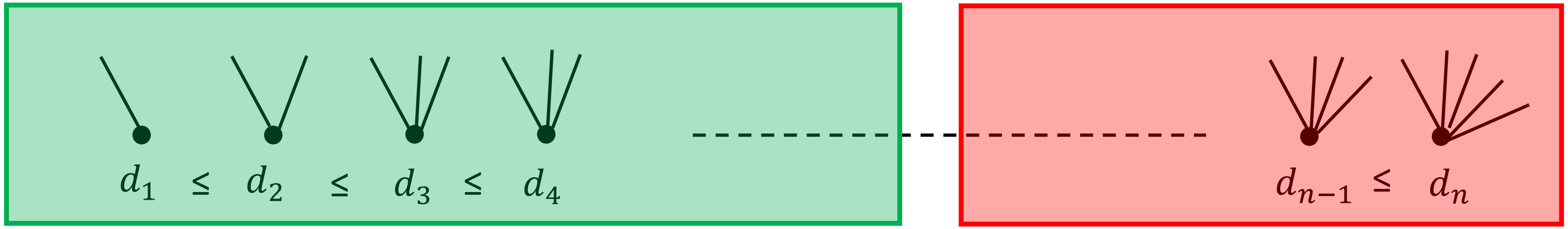


At most $2\sqrt{\varepsilon m}$ heavy vertices

At most εm heavy-heavy edges

- Avg degree of light vertices is (1/2)-approx of avg degree
- But light degrees cannot be too large
  - So avg light degree has lower variance

The variance problem



$$Y_1 = \begin{cases} d_v & d_v < \sqrt{m/\varepsilon} \\ 0 & \text{else} \end{cases}$$

$$k \approx \frac{\text{var}[Y_1]}{\mathbf{E}[Y_1]^2} \le \frac{\max(Y_1)}{\mathbf{E}[Y_1]} \le \frac{\sqrt{m}}{\bar{d}} \le \frac{n}{\sqrt{m}}$$

$$\text{var}[Y_1] \le \mathbf{E}[Y_1^2] \le \max(Y_1)\mathbf{E}[Y_1]$$

- $\sqrt{n}$ samples suffice to estimate average light degree
  - That gives (1/2)-approximation to true average degree
- Clean expression that deals with all (sparse to dense) cases

- But wait…how do we even sample Y?

A tale of two tails

X = n*(avg of k uar degrees)

E[X] = 2m

Y = n*(avg of k uar Winsorized degrees)

E[Y] = sum of light degrees ≥ (1-ε)m

X ≥ Y

$$\Pr[X < (1-\varepsilon)m] \leq \Pr[Y < (1-\varepsilon)m]$$

Chebyshev on Y, like previous slide

$$\Pr[X \geq (1+\varepsilon)(2m)] \leq 1-\varepsilon$$

Markov on X ([Feige 02] is tighter)

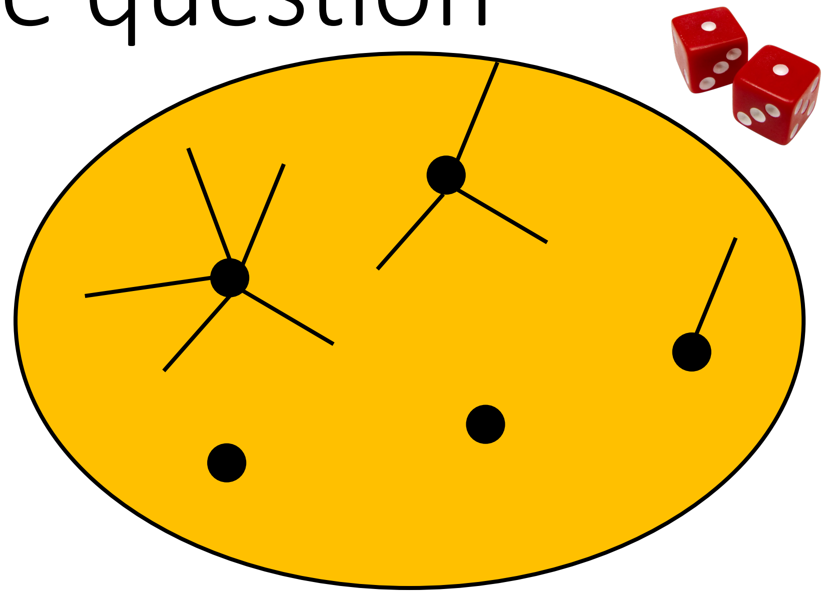
Take min of O(1/ε) estimates for full proof

$\sqrt{n}$ samples give (2+ε)-approx to average degree

Tool #2: Graph orientations

Get some direction

Back to a simple question



m = #edges
n = #vertices

Avg degree = $\sum_v d_v/n$

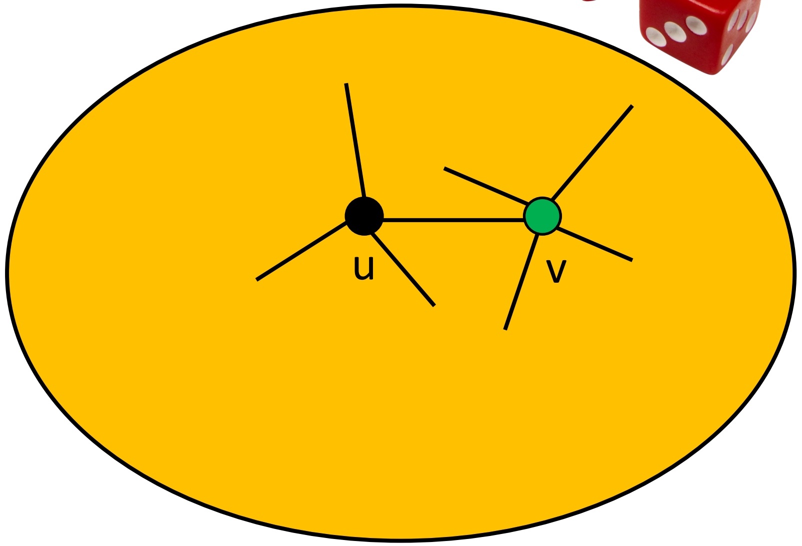
$\overline{d}$ = 2m/n

- Estimate the average degree of a graph
  - Beat the obvious procedure of sampling random degrees
  - Can we exploit graph structure?
- [Goldreich-Ron 08] (1+ε)-approximation in O($\sqrt{n}$) time

Know thy neighbor

[Eden-Ron-S 17]

Y_1

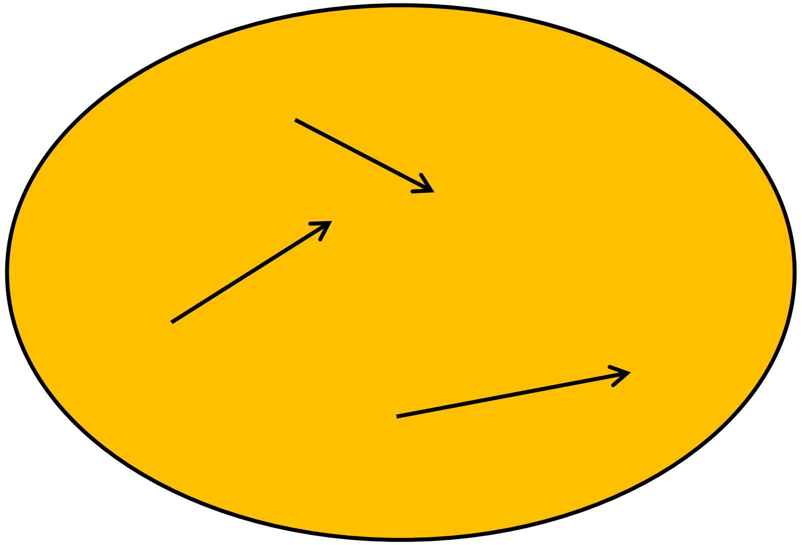


1. Pick uar vertex u
2. Pick uar neighbor v
3. If $d_u < d_v$, output $2d_u$
4. If $d_u > d_v$, output 0

(If equal, break ties consistently.)

d_v : Degree

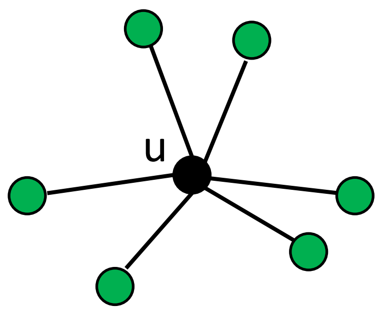
d_v^+ : Outdegree



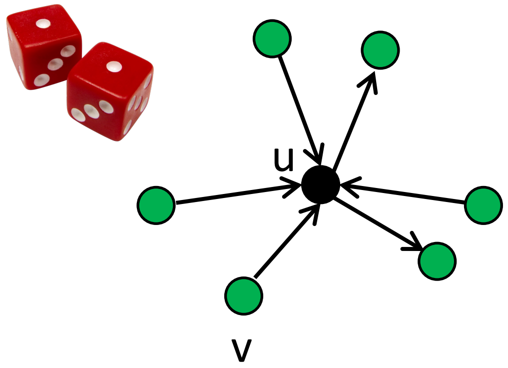
Orient G into a DAG as follows

$u \prec v$: if $d_u < d_v$ or
$d_u = d_v$ and $id(u) < id(v)$

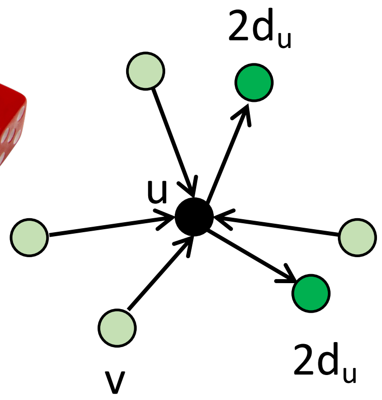
What do you expect?



What do you expect?



What do you expect?



Sum of outdegrees is m

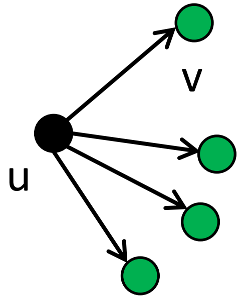
$$\mathbf{E}[Y_1] = \frac{1}{n}\sum_u \frac{d_u^+}{d_u} \cdot 2d_u = \frac{1}{n}\sum_u 2d_u^+ = \frac{2m}{n} = \bar{d}$$

- We have an unbiased estimator for average degree

$$k \approx \frac{\mathrm{var}[Y_1]}{\mathbf{E}[Y_1]^2} \le \frac{\max(Y_1)}{\mathbf{E}[Y_1]}$$

What's the max?

$d_v \geq d_u \geq d_u^+$



$$2m \geq \sum_{\text{green } v} d_v \geq (d_u^+)^2$$

$$\max_u d_u^+ \leq \sqrt{2m}$$

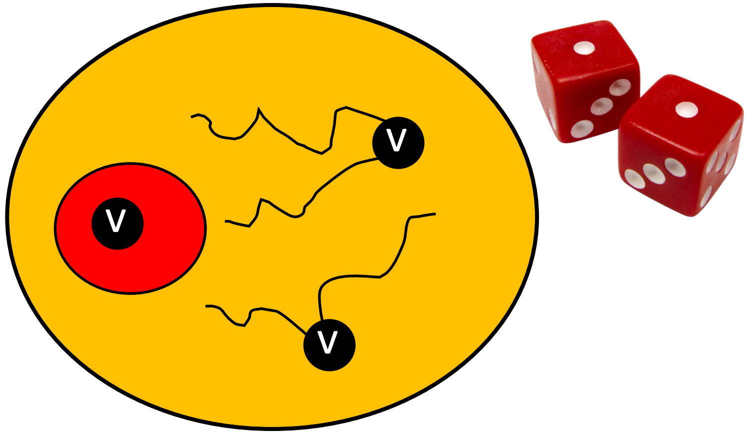
$$k \approx \frac{\text{var}[Y_1]}{\mathbf{E}[Y_1]^2} \leq \frac{\max(Y_1)}{\mathbf{E}[Y_1]} \leq \frac{\sqrt{2m}}{\bar{d}} \leq \frac{n}{\sqrt{m}}$$

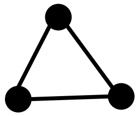
- So $O(\sqrt{n}\,)$ queries suffice to get (1+ε)-approx of average degree

Tool #3: Chiba-Nishizeki

A really really useful fact

Triangle counting





- Approximate triangle in G
- About as classic as it gets

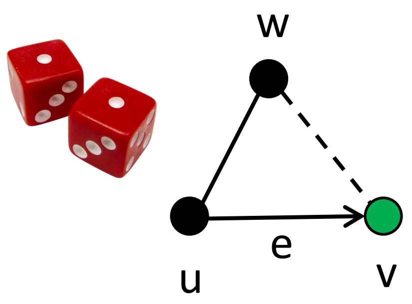
- [Eden-Levi-Ron-S 15] (1+ε)-estimate to t in time:

Ignoring log and ε

$$O^*\left(\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t}\right)$$

Optimal!

A simple estimator

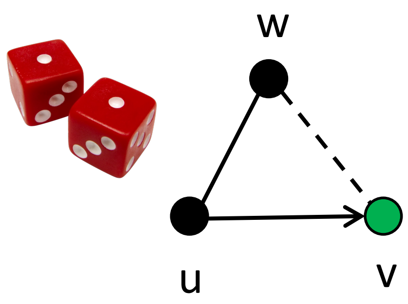


1. Pick uar (u,v)
2. Pick uar neighbor w from lower degree endpoint
3. Check if (u,v,w) is a triangle

Success prob $= \frac{1}{m} \sum_{e=(u,v)\in E} \frac{t_e}{\min(d_u, d_v)}$

- Assume access to uar edges
  - [Assadi-Kapralov-Khanna 18]
- We want to estimate average t_e , # triangles containing e
  - t = $3m(\sum_e t_e/m)$

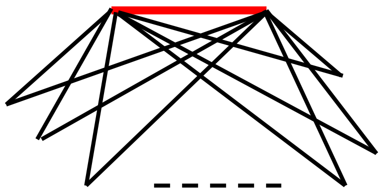
An unbiased estimator



1. Pick uar (u,v)
2. Pick uar neighbor w from lower degree endpoint
3. Check if (u,v,w) is a triangle, output $Y_1 = \min(d_u, d_v)$, else 0

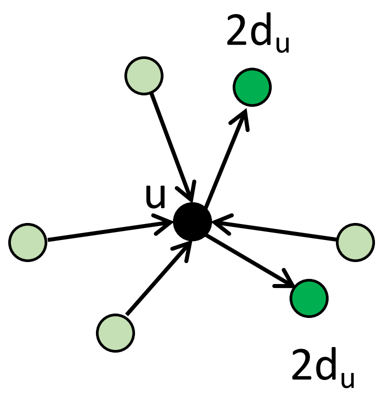
Expectation $= \frac{1}{m} \sum_{e=(u,v) \in E} \frac{t_e}{\min(d_u, d_v)} \cdot \min(d_u, d_v) = \frac{1}{m} \sum_{e \in E} t_e$

$$k \approx \frac{\mathrm{var}[Y_1]}{\mathbf{E}[Y_1]^2} \leq \frac{\max(Y_1)}{\mathbf{E}[Y_1]} = \frac{\max_{(u,v)\in E} \min(d_u, d_v)}{\mathbf{E}[Y_1]}$$



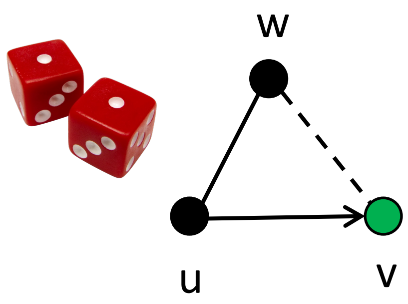
Chiba-Nishizeki to the rescue

$$\sum_{(u,v)\in E} \min(d_u, d_v) \le m \cdot \sqrt{2m}$$



- [Chiba-Nishizeki 85] In the context of clique counting and arboricity
- So average $\min(d_u, d_v)$ is at most $\sqrt{m}$

An unbiased estimator



1. Pick uar (u,v)
2. Pick uar neighbor w from lower degree endpoint
3. Check if (u,v,w) is a triangle, output $Y_1 = \min(d_u, d_v)$, else 0

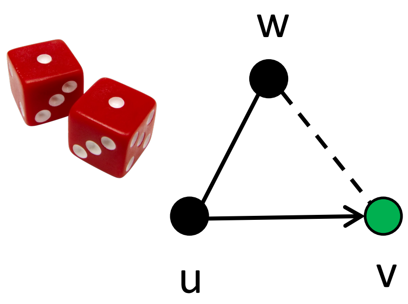
If $\min(d_u, d_v) \leq \sqrt{m}$

$$k \approx \frac{\mathrm{var}[Y_1]}{\mathbf{E}[Y_1]^2} \leq \frac{\max(Y_1)}{\mathbf{E}[Y_1]} \leq \frac{\sqrt{m}}{\mathbf{E}[Y_1]}$$

$$= \frac{m^{3/2}}{t}$$

$$\mathbf{E}[Y_1] = \frac{\sum_e t_e}{m}$$

Reducing variance



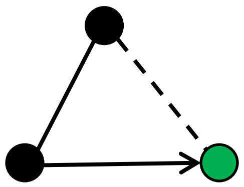
1. Pick uar (u,v)
2. Repeat (1 + min(d_u,d_v)/$\sqrt{m}$)) times
   a) Pick uar neighbor w from lower degree endpoint
   b) Check if (u,v,w) is a triangle, set Z_i = min(d_u, d_v), else 0
3. Output Y_1 = average Z_i

Variance of average of iid variables = Average of variance

$$\mathrm{var}[Y_1] = \frac{\mathrm{var}[Z_1]}{\min(d_u, d_v)/\sqrt{m}} \leq \frac{\max(Z_1)\mathbf{E}[Z_1]}{\min(d_u, d_v)/\sqrt{m}} = \sqrt{m}\mathbf{E}[Y_1]$$

$$k \approx \frac{\mathrm{var}[Y_1]}{\mathbf{E}[Y_1]^2} \leq \frac{\sqrt{m}}{\mathbf{E}[Y_1]}$$

The punchline



1. Pick uar (u,v)
2. Repeat (1 + min(d_u,d_v)/$\sqrt{m}$)) times
   a) Pick uar neighbor w from lower degree endpoint
   b) Check if (u,v,w) is a triangle, set Z_i = min(d_u, d_v), else 0
3. Output Y_1 = average Z_i

[Chiba-Nishizeki 85]!

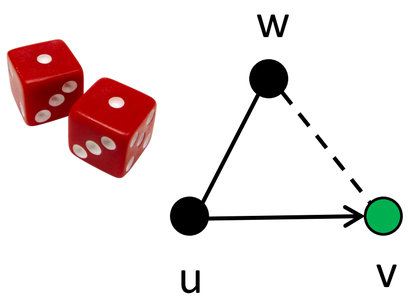
$$k \approx \frac{\mathrm{var}[Y_1]}{\mathbf{E}[Y_1]^2} \leq \frac{\sqrt{m}}{\mathbf{E}[Y_1]} = \frac{m^{3/2}}{t}$$

$$\leq \sqrt{2} m^{3/2}$$

Runtime per sample = $\frac{1}{m} \sum_{(u,v)\in E} \left(1 + \frac{\min(d_u, d_v)}{\sqrt{m}}\right) = 1 + \frac{\sum_{(u,v)\in E} \min(d_u, d_v)}{m^{3/2}}$

$$\leq 3$$

To finish...



1. Pick uar (u,v)
2. Repeat ($1 + \min(d_u,d_v)/\sqrt{m}$) times
   a) Pick uar neighbor w from lower degree endpoint
   b) Check if (u,v,w) is a triangle, set $Z_i = \min(d_u, d_v)$, else 0
3. Output Y_1 = average Z_i

$$k \approx \frac{\mathrm{var}[Y_1]}{\mathbf{E}[Y_1]^2} = \frac{m^{3/2}}{t}$$

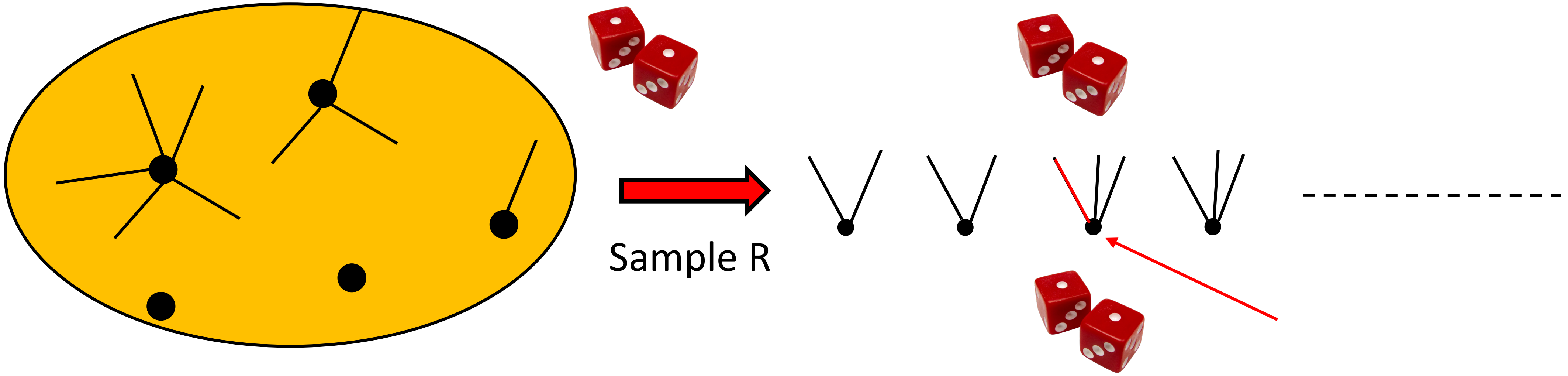
Runtime per sample < 3

- $m^{3/2}/t$ algorithm for estimating triangle count
  - Assuming uar edges
- Optimal!

Tool #4: Simulating edge samples

Fake it till you make it

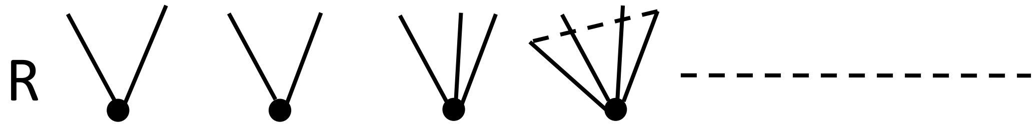
Fake uar edge samples



- Query all degrees in R
- Set up data structure that:
    1. Samples u in R proportional to d_u/d_R
    2. Output uar edge incident to v (uar nbr of u)

- This gives uar edge incident to R, in O(1) time
- Can we use these as generic "uar" edges?

What do we need?

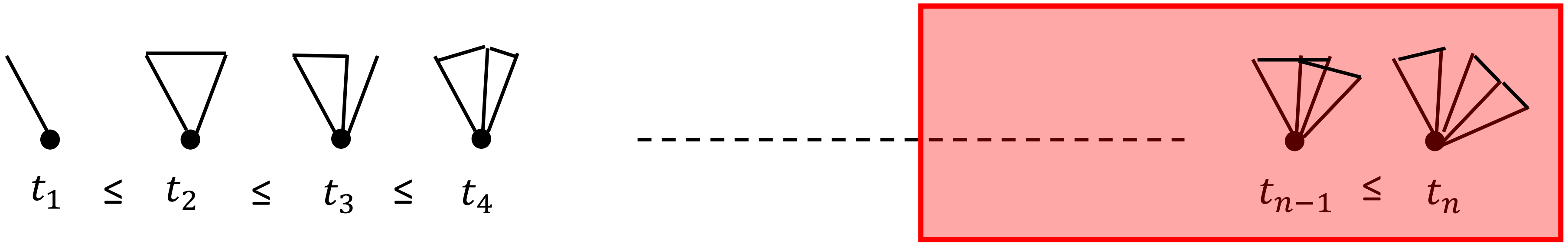


1. Pick uar (u,v)
2. Repeat $(1 + \min(d_u,d_v)/\sqrt{m})$ times
    a) Pick uar neighbor w from lower degree endpoint
    b) Check if (u,v,w) is a triangle, set $Z_i = \min(d_u, d_v)$, else 0
3. Output Y_1 = average Z_i

$$k \approx \frac{\text{var}[Y_1]}{\mathbf{E}[Y_1]^2} = \frac{\sum_{(u,v)\in E_R} \min(d_u, d_v)}{t_R}$$

- When is t_R good estimate for total triangle count?
    - Denominator (t_R) should not too small
- Numerator is easy to deal with (Markov)

Tool #1: Heavy Vertices

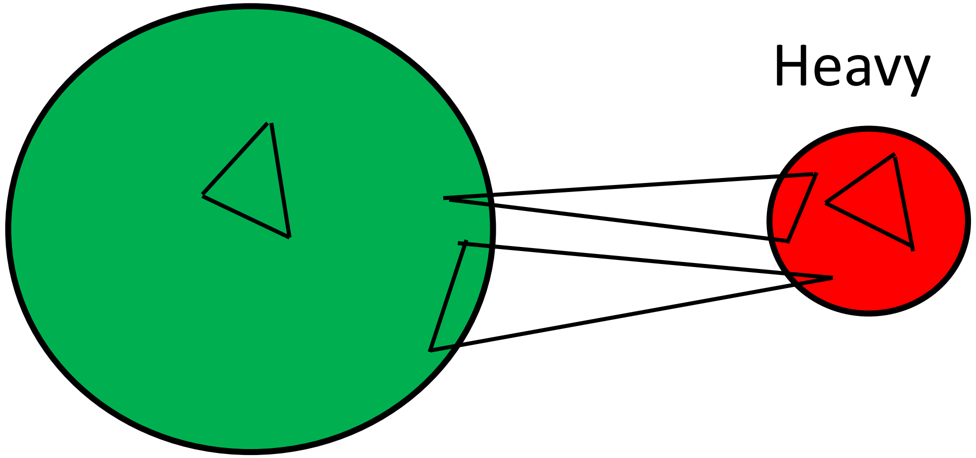


- Can we simply drop "large" outcomes?
  - Word of the day: Winsorize

But these are degrees!

$Y_1 = t_v$

Light



Heavy

At most $(\varepsilon t)^{1/3}$ heavy vertices

At most εt fully heavy triangles

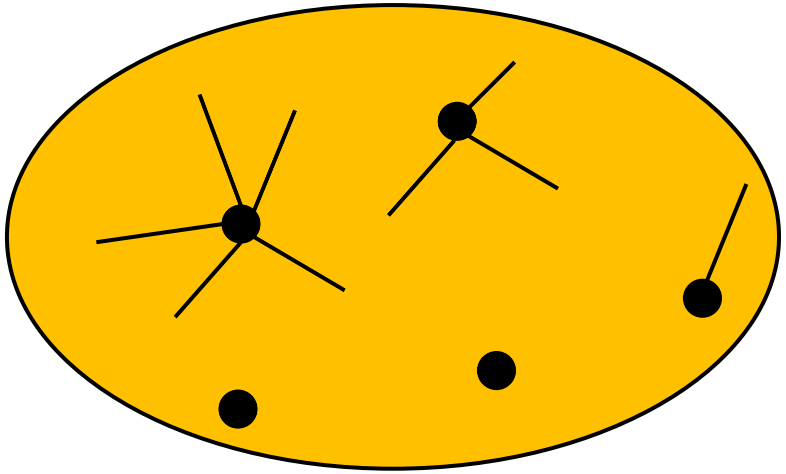
$t_v < t^{2/3}/\varepsilon^{1/3}$ $\quad$ $t_v \geq t^{2/3}/\varepsilon^{1/3}$

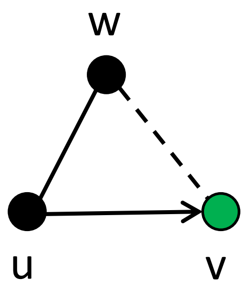
- At least (1-ε)t triangles incident to light vertices
- Average t_v of light vertices gives (1/3)-approx to average t_v

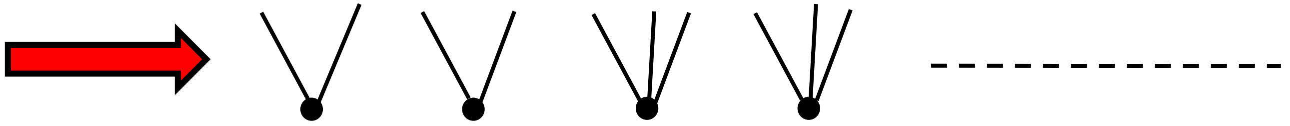
$$k \approx \frac{\mathrm{var}[Y_1]}{\mathbf{E}[Y_1]^2} \leq \frac{\max(Y_1)}{\mathbf{E}[Y_1]} \approx \frac{t^{2/3}}{t/n} = \frac{n}{t^{1/3}}$$

In total…

1. Pick uar (u,v)
2. Repeat $(1 + \min(d_u,d_v)/\sqrt{m})$ times
   a) Pick uar neighbor w from lower degree endpoint
   b) Check if (u,v,w) is a triangle, set $Z_i = \min(d_u, d_v)$, else 0
3. Output Y_1 = average Z_i





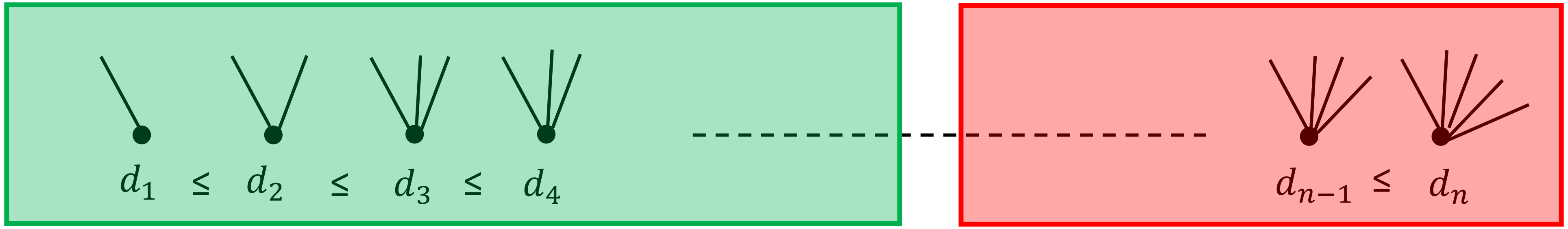


Sample R

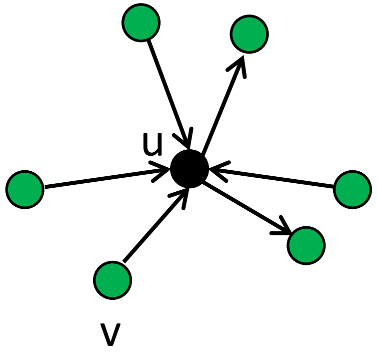
$$\frac{n}{t^{1/3}} + \frac{m^{3/2}}{t}$$

- Direct analysis gives 3-approx for t
  - Optimal complexity for constant factor approx
- Getting (1+ε)-approx needs little more work
  - Same tools, just need to determine whether vertex is heavy/light

Tool #1: Heavy Vertices



Tool #2: Graph orientations

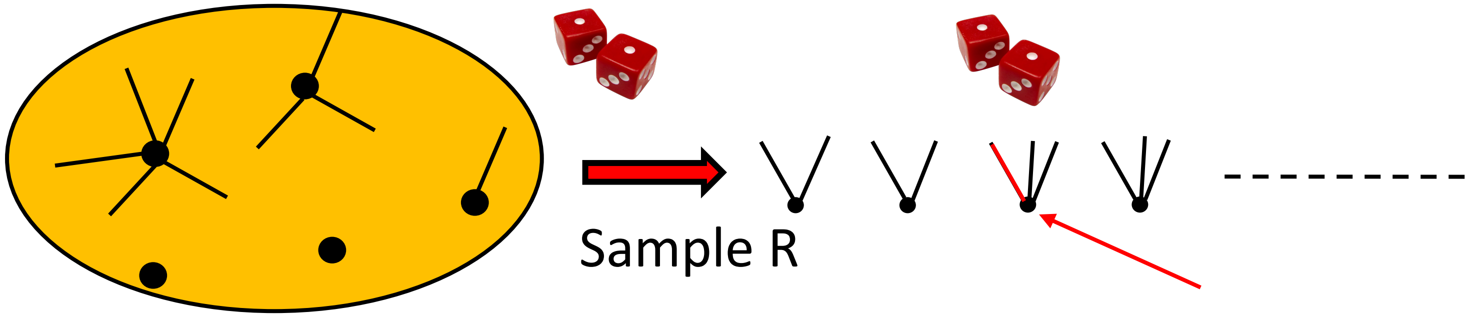


d_u vs d_v

Tool #3: Chiba-Nishizeki

$$\sum_{(u,v)\in E} \min(d_u, d_v) \le m \cdot \sqrt{2m}$$

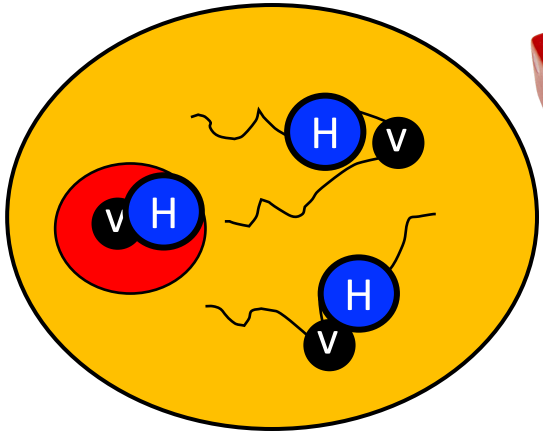
Tool #4: Simulating edge samples

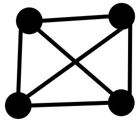


Some survey-ish slides

If you're in the audience, I hope I cited you

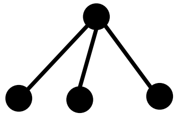
Sublinear subgraph counting





- [Eden-Levi-Ron-S 15, Eden-Ron-S 20] Clique counting, standard model

$$\frac{n}{C^{1/3}} + \frac{m^{k/2}}{C}$$



- [Gonen-Ron-Shavitt 15, Eden-Ron-S 17] k-Star counting, standard model

$$\frac{n}{C^{1/(k+1)}} + \frac{m}{C^{1/k}} \leq n^{1-1/(k+1)}$$

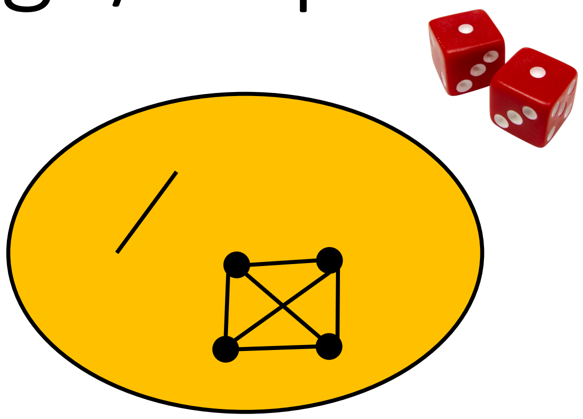
The arboricity connection

- The degeneracy/arboricity α is: max min (or avg) degree of a subgraph
  - The $\sqrt{m}$ is really α !

- [Eden-Ron-S 18, Eden-Ron-S 20] One can get α in all the complexities

- For any minor-free family graphs:
  - Clique estimation in O(n/C)
  - k-Star estimation in $n^{1-1/k}$ (instead of $n^{1-1/(k+1)}$)

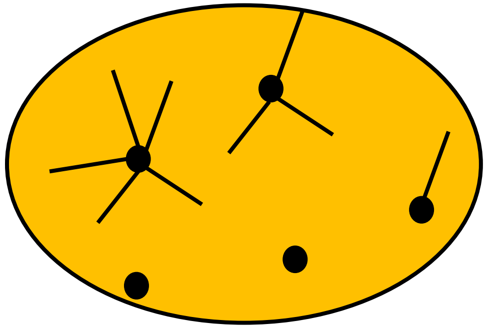
Sampling uar edge/clique



- [Eden-Ron-Rosenbaum 18, Eden-Rosenbaum 20, Eden-Mossel-Rubinfeld 21, Tetek-Thorup 22, Eden-Narayanan-Tetek 23] Sampling uar edges

- [Fichtenberger-Gao-Peng 20, Eden-Ron-Rosenbaum 22] Sampling cliques
  - [FGP20] does arbitrary subgraphs but needs uar edges

Model with uar edges



- Access to uniform random edges

- [Aliakbarpour-Biswas-Goulekis-Peebles-Rubinfeld-Yodpinyanee 18] k-star counting

$$\frac{n}{C^{1/(k+1)}} + \frac{m}{C^{1/k}}$$

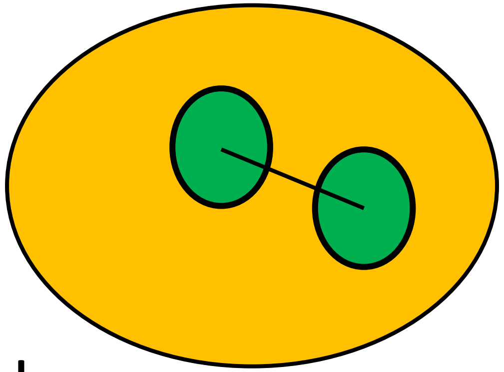
- [Assadi-Khanna-Kapralov 19, Fichtenberger-Gao-Peng 20] Any H-subgraph!

$$\frac{m^{e(H)}}{C}$$

$$\frac{n}{C^{1/3}} + \frac{m^{k/2}}{C}$$

- [Chierichetti-Dasgupta-Kumar-Lattanzi-Sarlos 16, Tetek-Thorup 22] Full neighbor list in one query

Independent set queries



- A different model, but again, you don't see the whole graph

- [Beame-HarPeled-Ramamoorthy-Rashtchian-Sinha 18] Edge estimation
- [Addanki-McGregor-Musco 22]

- [Bhattacharya-Bishnu-Ghosh-Mishra 21] Triangles with tripartite queries

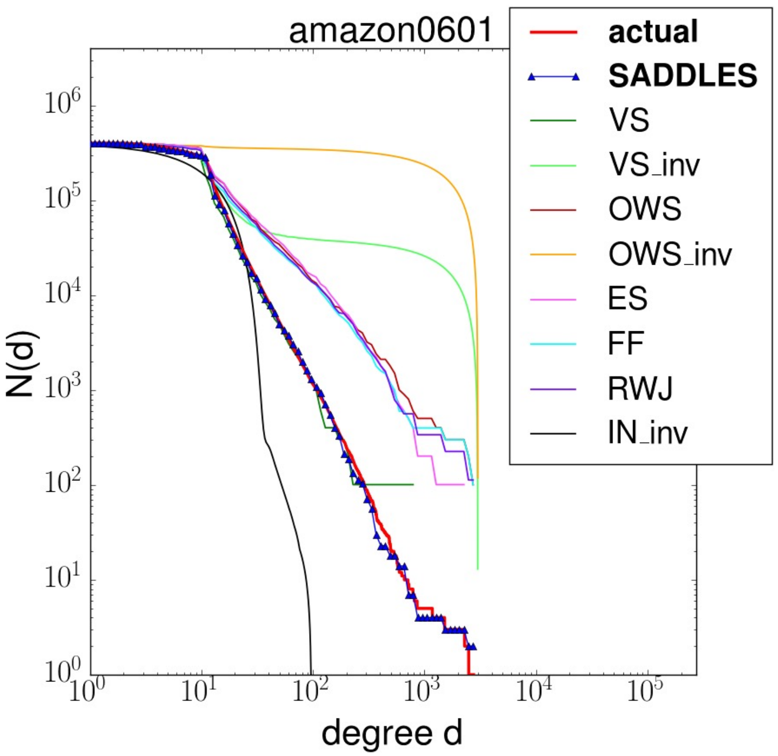
But is it practical?

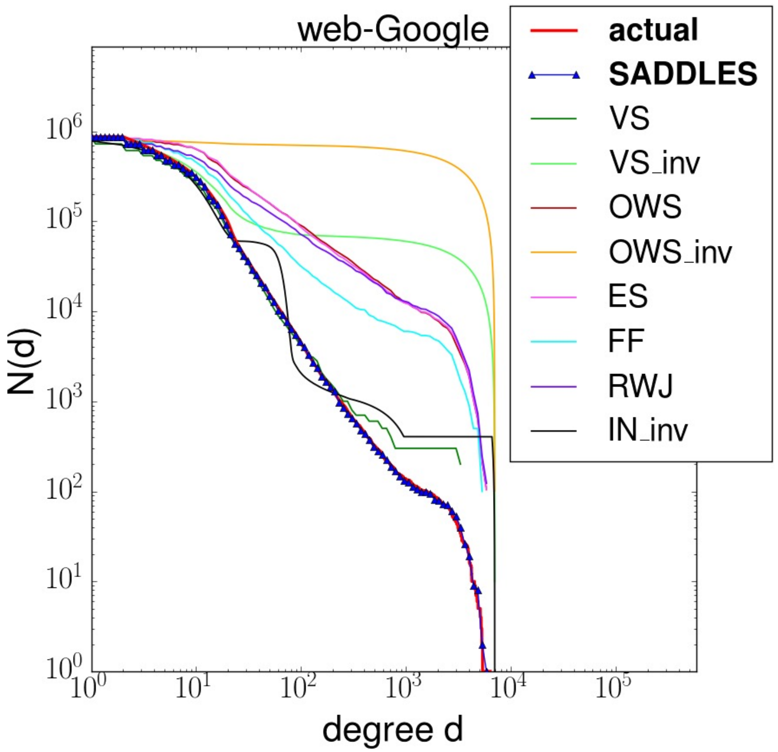
Constants matter, only when they don’t

What do you mean?

- Algorithms can be implemented “off the shelf”?
  - Small constant factors
- Ideas can be used for faster algorithms?
  - Coding sublinear sampling algorithms
- Write papers in other conferences?
  - Design algorithms that non-TCS people care about
- Solve algorithmic problems others care about?
  - Either write SciPy package everyone uses, or make some money

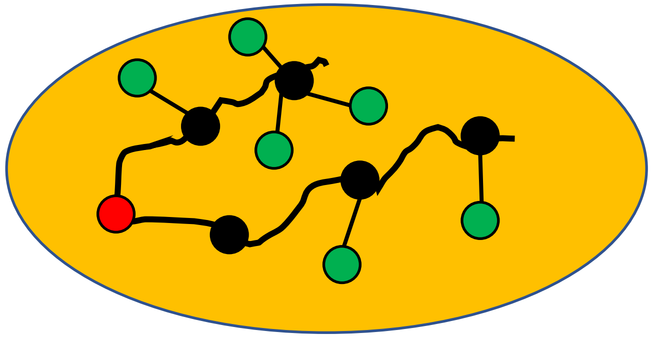
Estimating the degree distribution

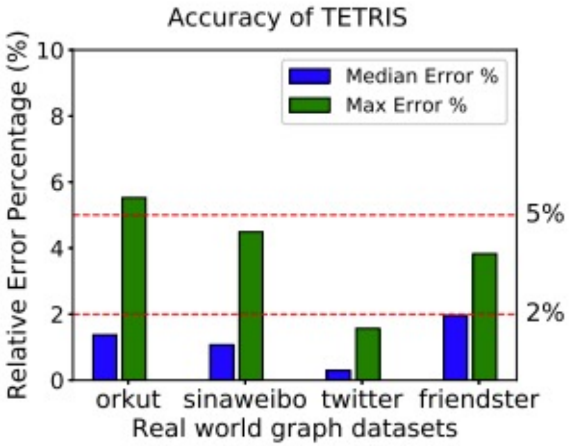




- [Eden-Jain-Pinar-Ron-S 18] Total of 0.01n degree queries in all cases

Sublinear triangle counting (for real)





Accuracy over 100 independent runs
3% of edges seen, graphs have 300M – 30B edges

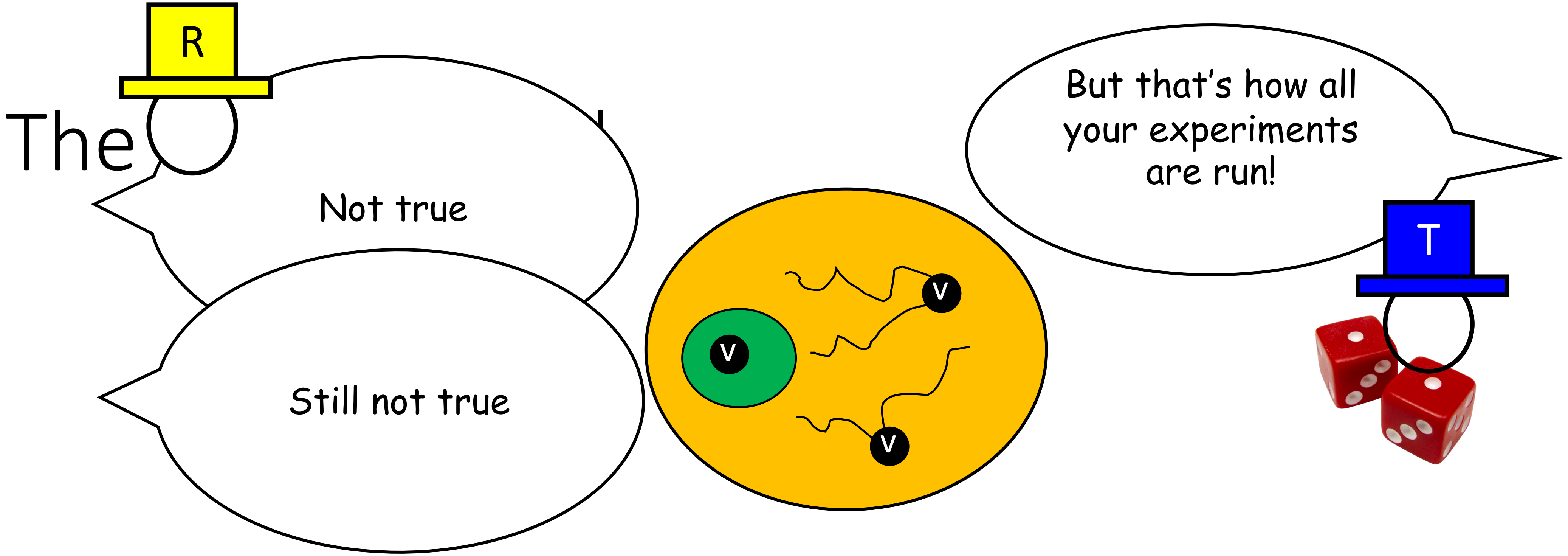
- [Bera S 20] Sublinear triangle counting
- In the real-world, one cannot sample uniform random vertices
  - Need to use random walks from "seed vertices"
  - Assume mixing time bounds
- Need to couple random walk with Tools #1 - #4
- [Bera-Choudhari-Haddadan-Ahmadian 24] General clique counting

Estimating m (or n)

- [Dasgupta-Kumar-Sarlos 14, Chierichetti-Dasgupta-Kumar-Lattanzi-Sarlos 16, BenEliezer-Eden-Oren-Fotakis 22]

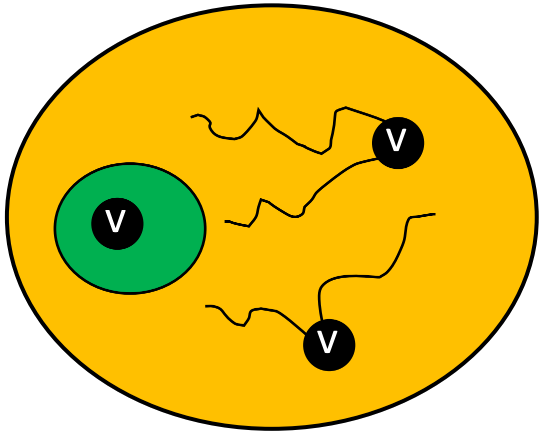
- What is the right model?

The



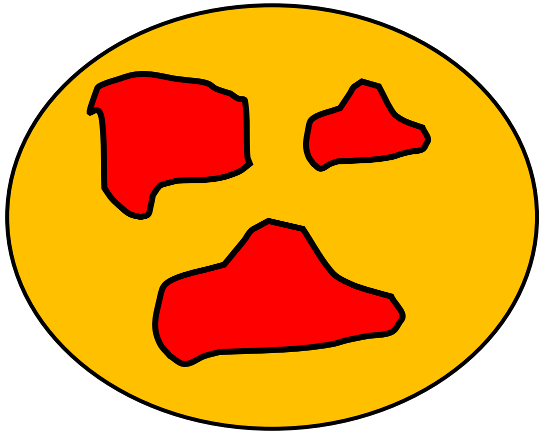
- [Goldreich-Ron 02] G is bounded degree, stored as adjacency list
  - n vertices, d degree bound
- ~~You can select (random) vertices/seeds~~
- You can crawl from these seeds
  - BFS, Random walks
- You can look up edges

The query models…?



- You start with one/few random vertices
- You can crawl from these seeds
  - BFS, Random walks
- You can look up edges
- Mixing time of graph is small

Words of wisdom

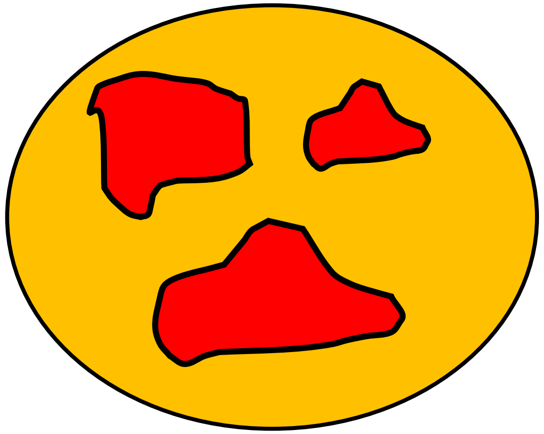


Tina Eliassi-Rad

"All of us (applied researchers) are really running sublinear graphs algorithms, because our data collection is incomplete.

Our data is a random snapshot of the ground truth"

A deep question

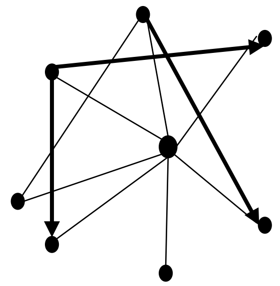


Tina Eliassi-Rad

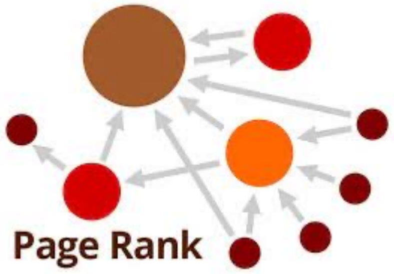
"If I run my favorite graph algorithm on the sample, what does that say about the whole?

How should I collect my graph data?"

Concrete sublinear questions

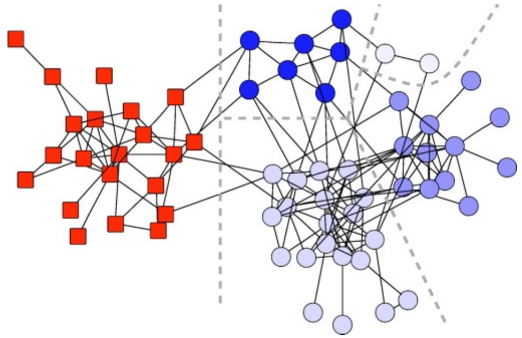


- Triangle statistics and clustering coefficients



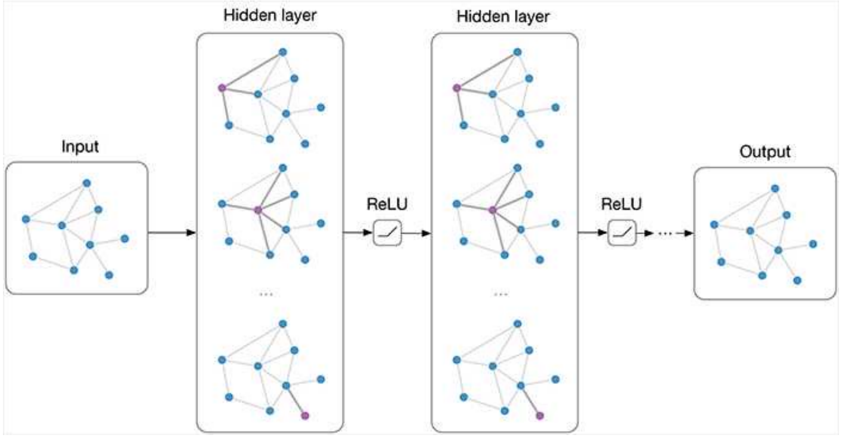
- Distribution of PageRank values

- Cluster/community structure of the graph

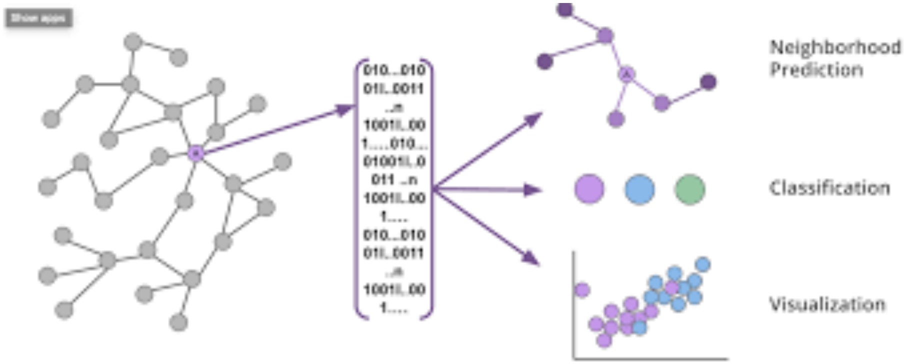


Less concrete sublinear questions

- Output of Graph Neural Net



- Output of downstream ML task



Time for coffee?

Dana, Talya, and I are working on a survey