# 6.889 Sub-linear Time Algorithms

Prof. Ronitt Rubinfeld

Lecture 1

Topics:

- Overview
- Diameter of point set
- #Connected Components
- MST

# Overview

- Big data - how do you access it?
  too much to view it all?
  can change by the time you make a decision?

e.g. what is "diameter" of facebook graph?
Six Degrees of Separation

- Compromise:

"exactly", "for all", "there exists" statements

$\Rightarrow$ "approximately", "there is a large group"...

- Models:

1) Random access queries - can access word of input in step
   - adj matrix vs. adj list
   - locn i
2) Samples - get sample of distribution/input in step

What are we really studying?

Randomized Algorithms?

Approximation Algorithms?

Parallel Algorithms?

Communication Complexity?

Statistics?

Learning?

YES!!

What kinds of datasets?

Graphs: e.g., diameter, # connected components, MST, bipartiteness, clusterability ...

Functions: e.g., monotonicity, convexity, linearity, juntas...

Distributions: eg. uniformity, independence, entropy, monotonicity, ...

## Course requirements:

- Scribing 25%
  must be in latex
- Psets 35%
- Project 25%
- Class participation 15%
  (includes grading)

## Projects:

- groups of 1-3
- Possible ideas:
  - solve new problem (or try ideas + explain why they fail)
  - read + survey some papers
  - implement an algorithm (or two or three)

Grading psets: {not turned in, √-, √, √+}
comments / solutions + typical errors

# A first Example: Diameter of a point set

note input size $n$ is $m^2$

**Input** $m$ points described by distance matrix $D$

s.t. $D_{ij}$ is distance from $i,j$

+ $D$ is: 1) symmetric

2) satisfies $\Delta \neq$

ie. $D_{ij} \leq D_{ik} + D_{kj}$ $\forall i,j,k$

**Output**

let $i,j$ be s.t. $D_{ij}$ is max $\Leftarrow$ $D_{ij}$ is "diameter"

Output $k,l$ s.t. $D_{kl} \geq \frac{D_{ij}}{2}$ $\Leftarrow$ 2-approximation

**Algorithm**

- Pick $k$ arbitrarily
- Pick $l$ to maximize $D_{kl}$
- Output $k,l$

runtime: $O(m) = O(\sqrt{n})$

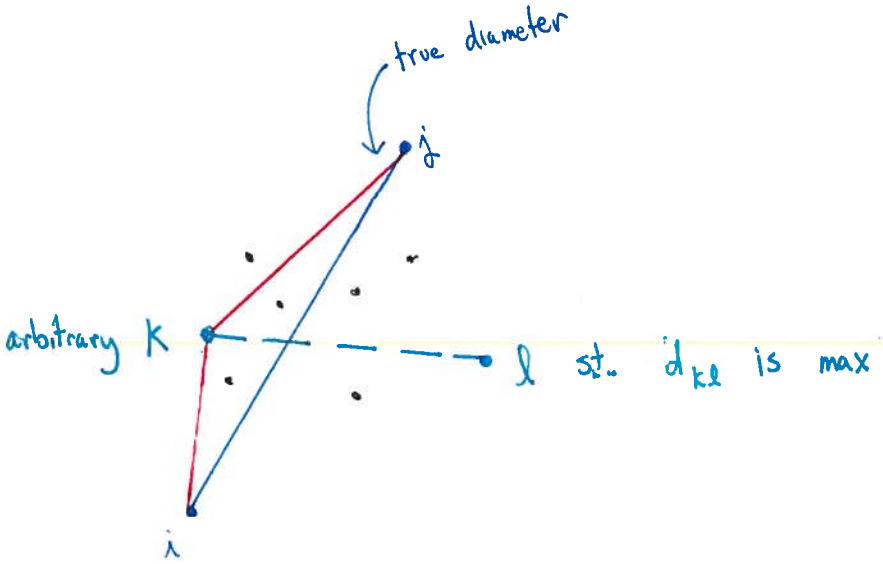
Why does it work?

$$D_{ij} \leq D_{ik} + D_{kj} \qquad \Delta\neq$$

$$= D_{ki} + D_{kj} \qquad \text{symmetry}$$

$$\leq D_{kl} + D_{kl} \qquad \text{choice of } l$$

$$\leq 2 D_{kl}$$



# A second example:

How many connected components in $G$?

← NOTE: can solve via DFS/BFS in linear time

**Input** $G = (V, E)$, $\varepsilon$

max degree $d$

adjacency list representation

$|V| = n$

$|E| = m \leq d \cdot n$

**Output** $y$ s.t. if $C = \#$ conn comp

then $C - \varepsilon \cdot n \leq y \leq C + \varepsilon n$

← "additive approx to w/in $\varepsilon n$"

## A different characterization of # conn components:

notation: $\forall v$ let $n_v \equiv \#$ nodes in $v$'s conn comp.

observation: $\forall$ connected component $A \subseteq V$

$$\sum_{u \in A} \frac{1}{n_u} = \sum_{u \in A} \frac{1}{|A|} = 1$$

---

So new characterization of # conn comp:

$$C = \sum_{u \in V} \frac{1}{n_u}$$

Why is this better?

computing $\frac{1}{n_u}$ needs $O(n)$ time? ↔ will estimate!

sum $O(n)$ terms?

Estimating $C = \sum_{u \in V} \frac{1}{n_u}$ :

Two ideas:

1) estimate $\frac{1}{n_u}$ quickly ← additive estimate

2) estimate $\sum_u \frac{1}{n_u}$ via sampling bounds ← additive estimate but will use to get multiplicative error bound

Estimating $\frac{1}{n_u}$ :

idea $n_u$ could be really big; so hard to compute exactly

but if $n_u$ really big then $\frac{1}{n_u}$ is really small so can approx $\frac{1}{n_u}$ by 0 or $\frac{\varepsilon}{2}$

def. $\hat{n}_u \equiv \min\{n_u, 2/\varepsilon\}$

$\hat{C} \equiv \sum_{u \in V} \frac{1}{\hat{n}_u}$

Lemma $\forall u \quad \left|\frac{1}{n_u} - \frac{1}{\hat{n}_u}\right| \leq \varepsilon/2$

Corr $|C - \hat{C}| \leq \frac{\varepsilon n}{2}$ ← so if can compute $\hat{C}$ faster, it is useful!

Pf of lemma

if $n_u \leq 2/\varepsilon$ then $\hat{n}_u = n_u$ so $\left|\frac{1}{n_u} - \frac{1}{\hat{n}_u}\right| = 0$

else $n_u > 2/\varepsilon$ so $\hat{n}_u = 2/\varepsilon < n_u$

$\Rightarrow \quad 0 \overset{\uparrow}{\leq} \frac{1}{n_u} < \frac{1}{\hat{n}_u} = \frac{\varepsilon}{2}$

↑ since $n_u > 0$

$\Rightarrow \quad \left|\frac{1}{\hat{n}_u} - \frac{1}{n_u}\right| < \varepsilon/2$ ∎

How long to compute $\hat{n}_u$?

**Algorithm** compute $\hat{n}_u$

Do BFS starting from $u$ until
- visit whole component of $u$
- or visit $2/\varepsilon$ distinct nodes

Output # visited nodes

**runtime**

$O(d \cdot 1/\varepsilon)$

↑ time per step of BFS

How to estimate $\sum_u \frac{1}{\hat{n}_u}$?

**Algorithm** estimate $\hat{c}$

$r \leftarrow b/\varepsilon^3$

Choose $U = \{u_1 \dots u_r\}$ random nodes

$\forall\, u \in U$
compute $\hat{n}_u$ via above algorithm

Output $\tilde{c} = \frac{n}{r} \sum_{u \in U} \frac{1}{\hat{n}_u}$

use average value estimate to estimate sum

**runtime** $O((d/\varepsilon) \cdot 1/\varepsilon^3) = O(d/\varepsilon^4)$

Why is it good?

**Thm** $\Pr\left[\,|\tilde{c}-\hat{c}| \le \varepsilon n/2\,\right] \ge 3/4$

**Pf**

Chernoff Bnd: $X_1 \cdots X_r$ iid $X_i \in [0,1]$

$$S = \sum_{i=1}^{r} X_i \qquad p = E[X_i] = E[S]/r$$

Then: $\Pr\left[\,\left|\frac{S}{r} - p\right| \ge \delta p\,\right] \le e^{-\Omega(rp\delta^2)}$

here $X_i = \frac{1}{\hat{n}_{u_i}}$

$$p = E\left[\frac{1}{\hat{n}_{u_i}}\right] = \frac{1}{n}\cdot\sum_{u\in V}\frac{1}{\hat{n}_{u_i}} = \frac{\hat{c}}{n}$$

$$\delta = \frac{\varepsilon}{2}$$

$$\frac{S}{r} = \underbrace{\frac{1}{r}\sum_{i=1}^{r}\frac{1}{\hat{n}_{u_i}}}_{\frac{\tilde{c}}{n}}$$

so should pick $r$ to be $\sim \frac{1}{p\delta^2}$ ($\Omega(\frac{1}{\varepsilon})$, $\Theta(\frac{1}{\varepsilon^2})$)

another reason why we need $\frac{1}{\hat{n}_u} \ge \frac{\varepsilon}{2}$

so $\Pr\left[\,\left|\frac{\tilde{c}}{n} - \frac{\hat{c}}{n}\right| \ge \frac{\varepsilon}{2}\cdot\frac{\hat{c}}{n}\,\right] = \Pr\left[\,|\tilde{c}-\hat{c}| \ge \frac{\varepsilon}{2}\hat{c}\,\right]$

$$\le e^{-\left(\frac{b}{\varepsilon^2}\cdot\frac{\varepsilon}{n}\cdot\frac{\varepsilon^2}{4}\right)}$$ } want this to be $\ge 2$ so that probability $\le e^{-2} \le \frac{1}{4}$

Since $\frac{\varepsilon}{2} \le \frac{1}{\hat{n}_u} \le 1$

summing over $u$: $\frac{\varepsilon n}{2} \le \hat{c} \le n$

so $\frac{\varepsilon}{2} \le \frac{\hat{c}}{n} \le 1$

$$\le e^{-\left(\frac{b}{\varepsilon}\cdot\frac{\varepsilon}{2}\cdot\frac{1}{4}\right)}$$

pick $b \ge 16$

Now we are done:

**Corr.** $\Pr[|c-\tilde{c}| \le \varepsilon n] \ge 3/4$

**Pf.** $|c-\tilde{c}| \le |c-\hat{c}| + |\hat{c}-\tilde{c}|$ by $\Delta\neq$

↑ always $\le \frac{\varepsilon n}{2}$ by corr

↖ $\le \frac{\varepsilon n}{2}$ by thm with prob $\ge 3/4$

# Approximating Min Spanning Tree (MST)

**Input** $G = (V, E)$ 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

$\varepsilon$

**Output**

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

$T$: tree; spans: tree touches every node; $w(T) = \sum_{(i,j) \in T} w_{ij}$

output $\hat{M}$ s.t. $(1-\varepsilon) M \leq \hat{M} \leq (1+\varepsilon) M$

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

## A different characterization of MST:

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

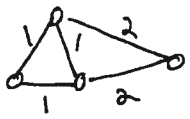
$C^{(i)}$ = # conn comp of $G^{(i)}$

Some examples before characterization:

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

here $M = n-1$

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




$G$

$G^{(1)}$

$C^{(1)} = 2$

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:

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

↑ 1 for each edge

↑ 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 $G$

↑
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 $\frac{\varepsilon'}{2w} \cdot n$ (additive error)

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

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

## Runtime:

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

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

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

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

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

## Approximation guarantee:

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

how? ↙ HW0!

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

Pr[all calls to approx #cc give output that is $\varepsilon'$ additive approx] $\geq 1 - \frac{w}{4w}$ ↙ union bound

$= 3/4$

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

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

this is where we use lower bnd on edge wts

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