Privatizing & Dynamizing Algorithm Design

Sricharan AR

advised by :

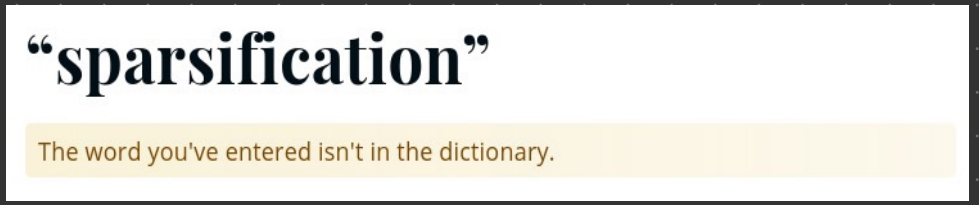
Monika Henzinger + Gramoz Goranci

Theme of the thesis

✨ SPARSIFICATION ✨

Theme of the thesis

✨ SPARSIFICATION ✨



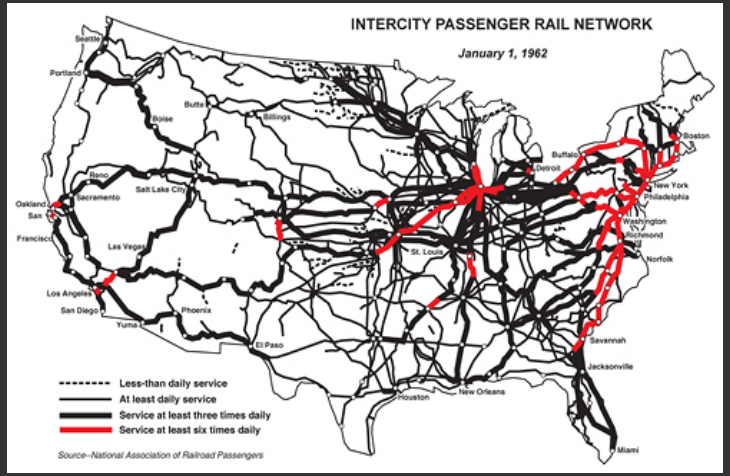
Theme of the thesis

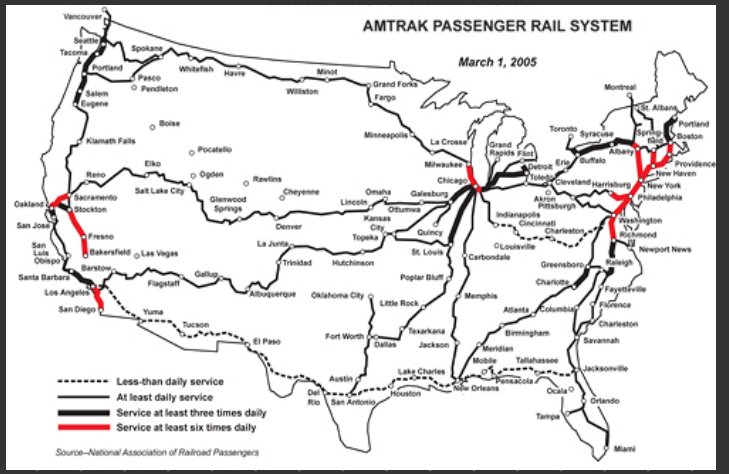
✨ SPARSIFICATION ✨

Marie Kondo

Keep only those things that speak to your heart. Then take the plunge and discard all the rest. By doing this, you can reset your life and embark on a new lifestyle.

Example : Passenger Rail Network in the USA



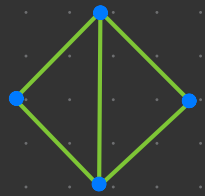


Incremental Max Flow

~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$

n vertices, m edges



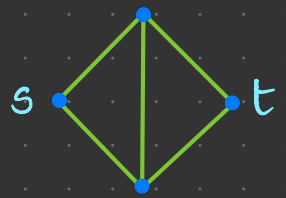
~~Incremental~~ Max Flow

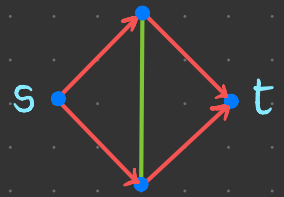
Input : undirected $G = (V, E)$

Output : max $s \to t$ flow

↙

edge disjoint $s \to t$ paths

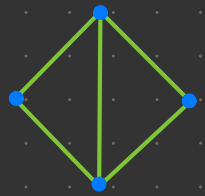


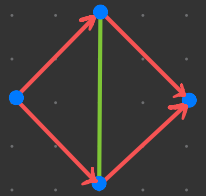


~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$

Output : max $s \to t$ flow



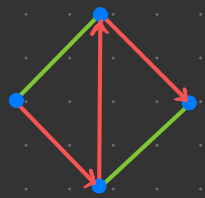


Algorithm ? Ford - Fulkerson

~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

MAIN OBJECT : Residual graph G_f



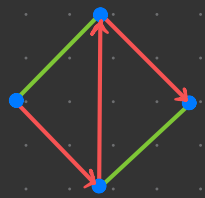
G with f as above

G_f allows "undoing" flow by reversing edge e

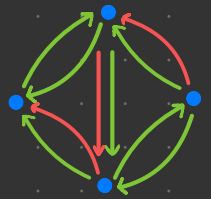
~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

MAIN OBJECT : Residual graph G_f



G with f as above



Corresponding G_f

G_f allows "undoing" flow by reversing edge e

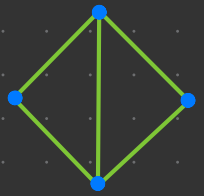
~~Incremental~~ Max Flow

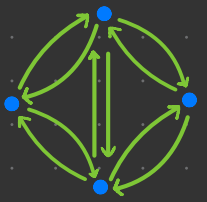
Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph :

augment f along this path

update residual graph by flipping edges on this path





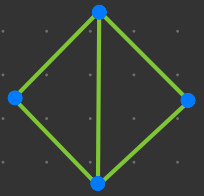
~~Incremental~~ Max Flow

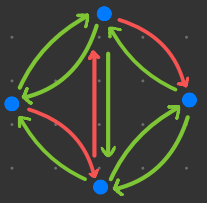
Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph :

augment f along this path

update residual graph by flipping edges on this path





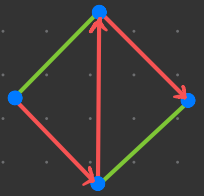
~~Incremental~~ Max Flow

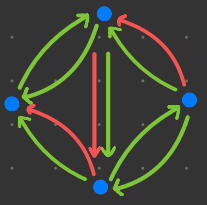
Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph :

augment f along this path

update residual graph by flipping edges on this path





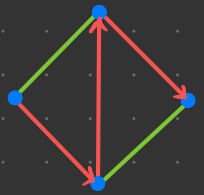
~~Incremental~~ Max Flow

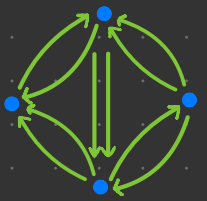
Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph :

augment f along this path

update residual graph by flipping edges on this path





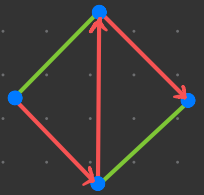
~~Incremental~~ Max Flow

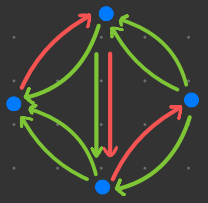
Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph :

augment f along this path

update residual graph by flipping edges on this path





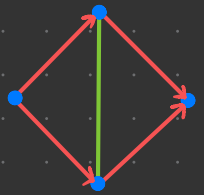
~~Incremental~~ Max Flow

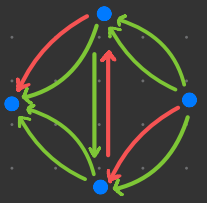
Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph :

augment f along this path

update residual graph by flipping edges on this path





~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph :

augment f along this path

update residual graph by flipping edges on this path

Running Time ?

~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph : → $O(m)$ → BFS/DFS

augment f along this path

update residual graph by flipping edges on this path

Residual graph has $\Omega(m)$ edges.

~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph : → $O(m)$ → BFS/DFS

augment f along this path → $O(n)$

update residual graph by flipping edges on this path → $O(n)$

Residual graph has $\Omega(m)$ edges.

~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph : → $O(m)$ → BFS/DFS

augment f along this path → $O(n)$

update residual graph by flipping edges on this path → $O(n)$

Residual graph has $\Omega(m)$ edges.

$\leq F^*$ such augmentations can occur $\Rightarrow$ Total time = $O(mF^*)$

~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph : → $O(m)$ → BFS/DFS

augment f along this path → $O(n)$

update residual graph by flipping edges on this path → $O(n)$

Residual graph has $\Omega(m)$ edges.

$\leq F^*$ such augmentations can occur $\Rightarrow$ Total time = $O(mF^*)$

~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph :

augment f along this path

update residual graph by flipping edges on this path

Residual graph has $\Omega(m)$ edges.

$\leq F^*$ such augmentations can occur $\Rightarrow$ Total time = $O(mF^*)$

~~Incremental~~ Max Flow

Input : undirected $G = (V, E)$ Output : max $s \to t$ flow

while $\exists$ $s \to t$ path in residual graph :

augment f along this path

update residual graph by flipping edges on this path

Sparsified Residual graph has ~~$O(m)$~~ $O(n \log n)$ edges.

$\leq F^*$ such augmentations can occur $\Rightarrow$ Total time = $\tilde{O}(nF^*)$

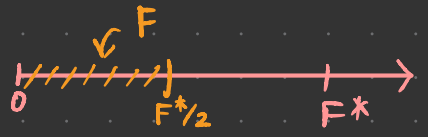
~~Incremental~~ Max Flow

Karger + Levine 2002

Sampling $n \log n$ edges sparsifies a.p. as long as $F = 0$

~~Incremental~~ Max Flow

Karger + Levine 2002

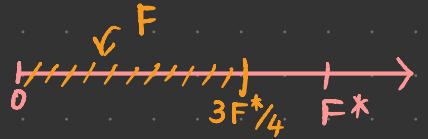


Sampling $n \log n$ edges sparsifies a.p. as long as $F = 0$

Sampling $2n \log n$ edges sparsifies a.p. as long as $F < F^*/2$

~~Incremental~~ Max Flow

Karger + Levine 2002



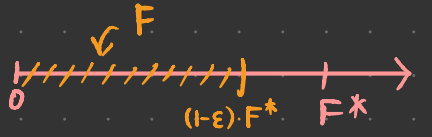
Sampling $n \log n$ edges sparsifies a.p. as long as $F = 0$

Sampling $2n \log n$ edges sparsifies a.p. as long as $F < F^*/2$

Sampling $4n \log n$ edges sparsifies a.p. as long as $F < 3F^*/4$

~~Incremental~~ Max Flow

Karger + Levine 2002



Sampling $n\log n$ edges sparsifies a.p. as long as $F = 0$

Sampling $2n\log n$ edges sparsifies a.p. as long as $F < F^*/2$

Sampling $4n\log n$ edges sparsifies a.p. as long as $F < 3F^*/4$

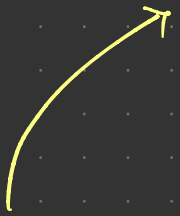
⋮

Sampling $n\log n/\varepsilon$ edges sparsifies a.p. as long as $F < (1-\varepsilon)F^*$

Incremental Max Flow

Karger + Levine 2002

NOT uniform sampling & difficult incrementally



Sampling $n \log n/\varepsilon$ edges sparsifies a.p. as long as $F < (1-\varepsilon) F^*$

Incremental Max Flow

We show:

→ that a variety of sampling schemes work for sparsification

→ One scheme that is easy to incrementally maintain

Sampling $n\log n/\varepsilon$ edges sparsifies a.p. as long as $F < (1-\varepsilon)F^*$

Incremental Max Flow

We show:

Incremental $(1-\varepsilon)$-max flow in $\tilde{O}(m + nF^*/\varepsilon)$ total time.

Incremental Max Flow

We show:

Incremental $(1-\varepsilon)$-max flow in $\tilde{O}(m + nF^*/\varepsilon)$ total time.

on edge insertion:

add edge to sampled residual graph

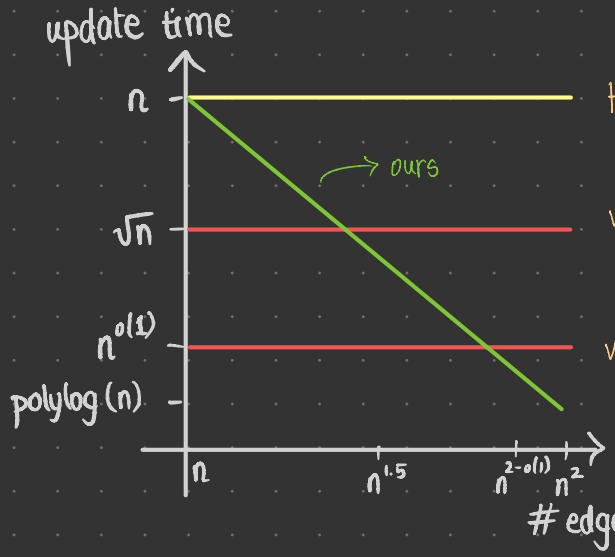
while $\exists$ $s \to t$ path in sampled residual graph:

augment f along this path

update residual graph by flipping edges on this path

resample residual graph

Incremental Max Flow – Comparison



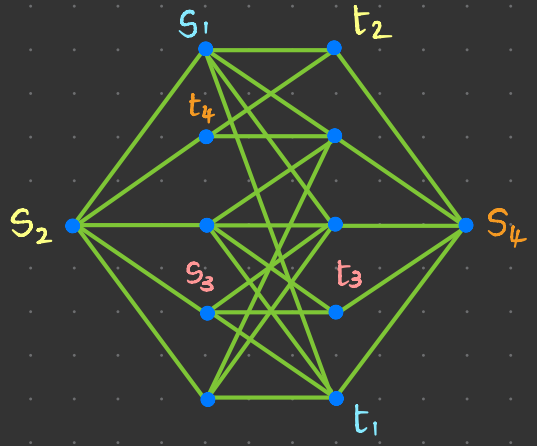
Henzinger '97 (exact)

van den Brand et al. '23, Goranci & Henzinger '23 ($\sqrt{m}$)

van den Brand et al. '24

Many flows to route

Many flows to route



$s_1 \to t_1$

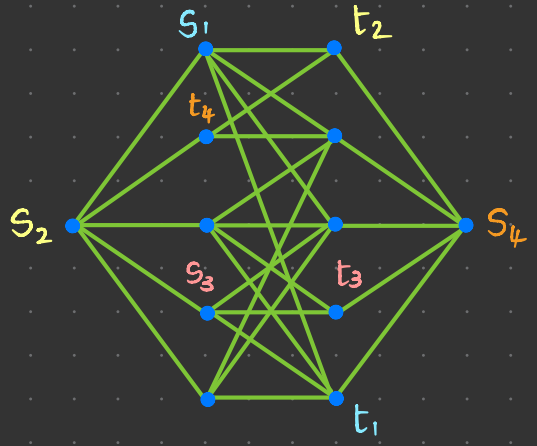
$s_2 \to t_2$

$s_3 \to t_3$

$s_4 \to t_4$

1 unit of flow

Many flows to route



$s_1 \to t_1$

$s_2 \to t_2$

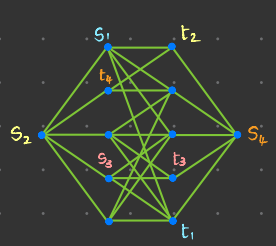
$s_3 \to t_3$

$s_4 \to t_4$

Goal: minimize maximum congestion on any edge.

↳ sum of all flows on that edge

The Lazy Solution - Oblivious Routing



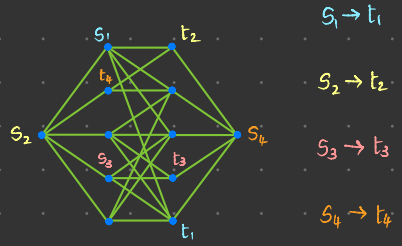
Oblivious Routing on G_L

↑

Constructed without knowing (s_i, t_i) beforehand

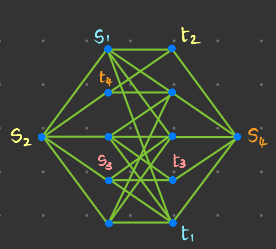
The Lazy Solution - Oblivious Routing

(source, target) → Oblivious Routing on G_t → flow



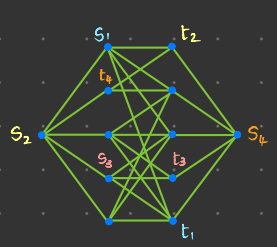
The Lazy Solution - Oblivious Routing

(s_1, t_1) → **Oblivious Routing on G_L** → f_1



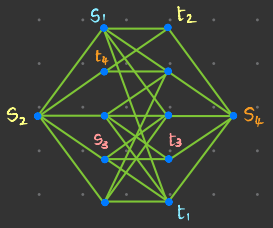
The Lazy Solution - Oblivious Routing

$(s_2, t_2) \rightarrow$ **Oblivious Routing on G_L** $\rightarrow f_2$



The Lazy Solution - Oblivious Routing

(s_2, t_2) → **Oblivious Routing on G_L** → f_2



$s_1 \to t_1$

$s_2 \to t_2$

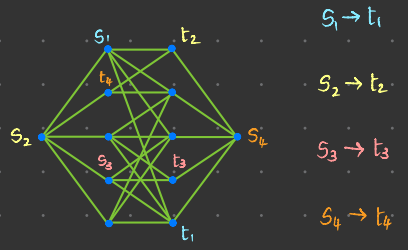
$s_3 \to t_3$

$s_4 \to t_4$

oblivious to the flow f_1

The Lazy Solution - Oblivious Routing

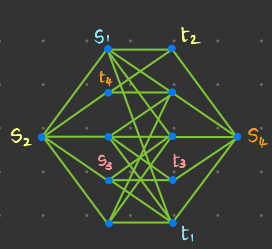
(s_3, t_3) → **Oblivious Routing on G_1** → f_3



How bad are oblivious routings?

The Lazy Solution - Oblivious Routing

$(s_3, t_3) \rightarrow$ Oblivious Routing on G_t $\rightarrow f_3$

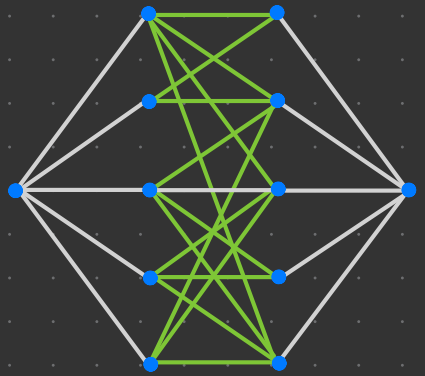


Räcke '08 showed a construction : $\forall$ demand sequences

$$\text{congestion}(\text{OBL-ROU}) \leq O(\log n) \cdot \text{OPT}$$

Many flows to route

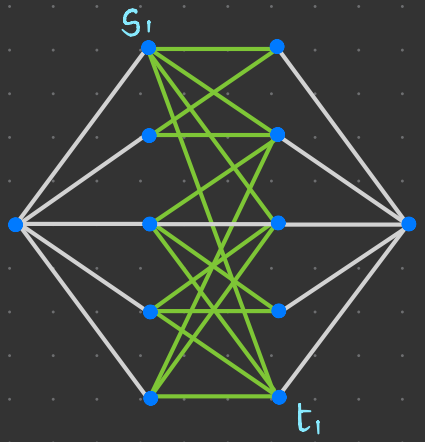
Spanning Tree Routing



Many flows to route

Spanning Tree Routing

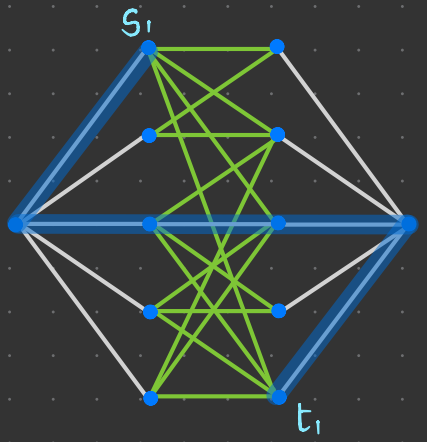
$s_1 \rightarrow t_1$



Many flows to route

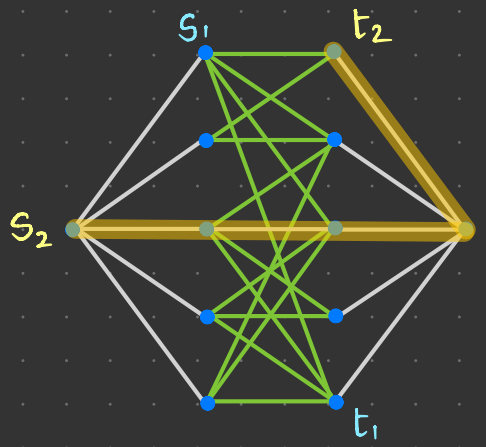
Spanning Tree Routing

$s_1 \rightarrow t_1$



Many flows to route

Spanning Tree Routing

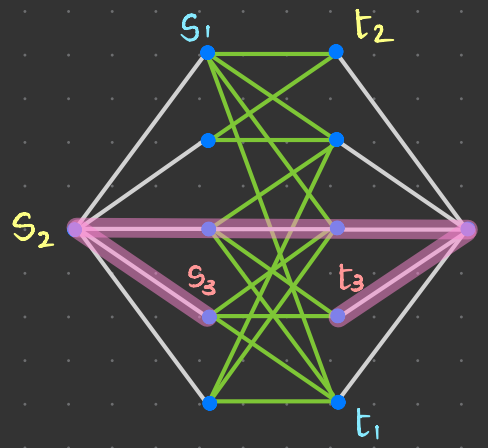


$s_1 \rightarrow t_1$

$s_2 \rightarrow t_2$

Many flows to route

Spanning Tree Routing



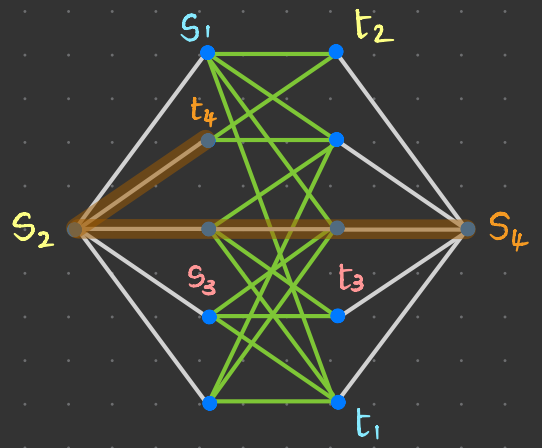
$s_1 \rightarrow t_1$

$s_2 \rightarrow t_2$

$s_3 \rightarrow t_3$

Many flows to route

Spanning Tree Routing



$s_1 \rightarrow t_1$

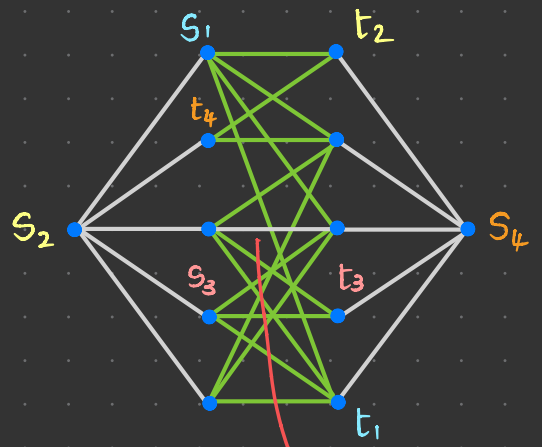
$s_2 \rightarrow t_2$

$s_3 \rightarrow t_3$

$s_4 \rightarrow t_4$

Many flows to route

Spanning Tree Routing



$s_1 \rightarrow t_1$

$s_2 \rightarrow t_2$

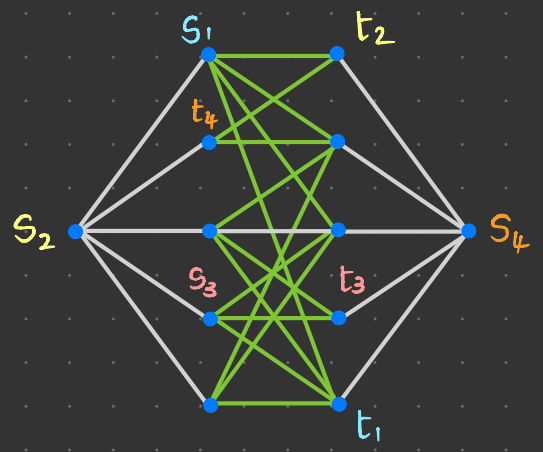
$s_3 \rightarrow t_3$

$s_4 \rightarrow t_4$

this edge has high congestion

Many flows to route

Spanning Tree Routing



$s_1 \to t_1$

$s_2 \to t_2$

$s_3 \to t_3$

$s_4 \to t_4$

Räcke's idea :

Keep finding trees that fix the bottlenecks of previous trees

Many flows to route

Andersen, Feige '09

Combining $O(m)$ spanning trees gives $\tilde{O}(\log n)$ approximations

Many flows to route

Combining $O(m)$ spanning trees gives $\tilde{O}(\log n)$ approximations

Many flows to route

Combining $O(m)$ spanning trees gives $\tilde{O}(\log n)$ approximations

"width"

low average stretch spanning trees

Abraham, Neiman '12

Many flows to route

Combining $O(m)$ spanning trees gives $\tilde{O}(\log n)$ approximations

Combining $O(\sqrt{m})$ electrical flows gives $O(\log^2 n)$ approximations

↑ our result

Many flows to route

Combining $O(m)$ spanning trees gives $\tilde{O}(\log n)$ approximations

Combining $O(\sqrt{m})$ electrical flows gives $O(\log^2 n)$ approximations

↑ electrical flow localization

Schild et al. '18

Many flows to route

Combining $O(m)$ spanning trees gives $\tilde{O}(\log n)$ approximations

Combining $O(\sqrt{m})$ electrical flows gives $O(\log^2 n)$ approximations

Many flows to route

Combining $O(m)$ spanning trees gives $\tilde{O}(\log n)$ approximations

Combining $O(\sqrt{m})$ electrical flows gives $O(\log^2 n)$ approximations

Many flows to route

Combining $O(m)$ spanning trees gives $\tilde{O}(\log n)$ approximations

Combining $O(\sqrt{m})$ electrical flows gives $O(\log^2 n)$ approximations

Can be computed in $\tilde{O}(m\sqrt{m})$ time : dimension reduction + Laplacian solvers

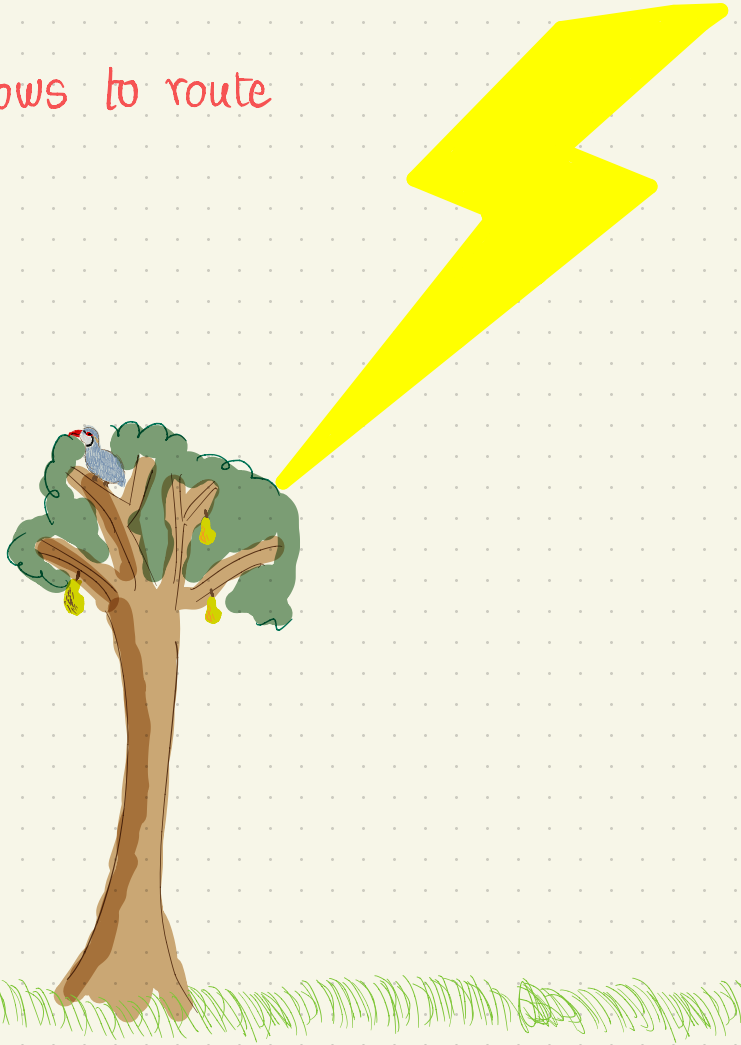
↓ Indyk '08

↓ Spielman, Teng '04

Many flows to route

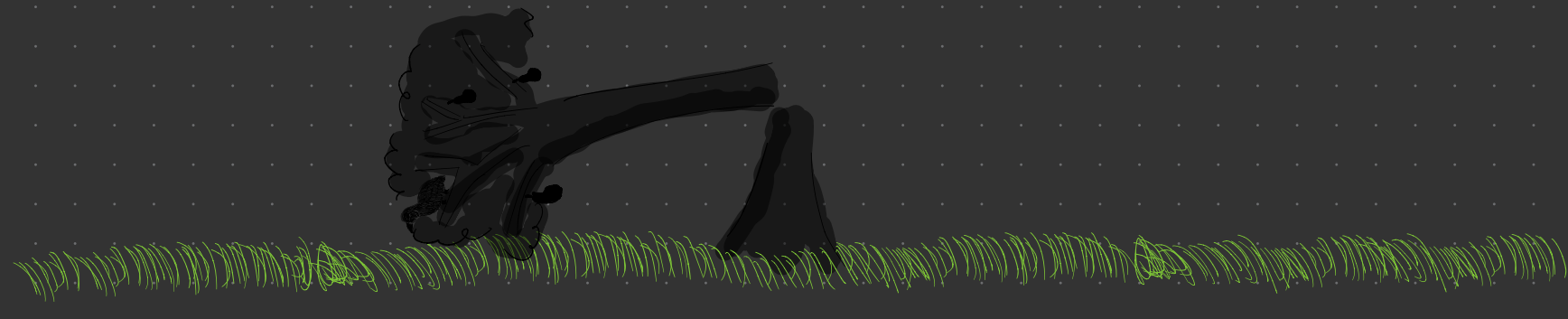

Many flows to route

Many flows to route




Many flows to route

Many flows to route



Limits of Sparsification

We saw where sparsification helps.

Is it possible that sometimes it doesn't?

Limits of Sparsification

We saw where sparsification helps.

Is it possible that sometimes it doesn't? Yes!

If problem is hard on low-degree graph,

then degree sparsification won't help.

Limits of Sparsification

We saw where sparsification helps.

Is it possible that sometimes it doesn't? Yes!

If problem is hard on ~~low-degree~~ expander graph,

then ~~degree sparsification~~ expander decomposition won't help.

Litany of results

Takeaway: There are hard dynamic problems on expanders.

Litany of results

Takeaway: There are hard dynamic problems on expanders.

Conditioned on OMv,
cannot have subpolynomial
update time

Litany of results

$s \to t$ distance

maximum matching

densest subgraph

Takeaway: There are hard dynamic problems on expanders.

Conditioned on OMv,
cannot have subpolynomial
update time

Litany of results

$s \to t$ distance

maximum matching

densest subgraph

Takeaway: There are hard dynamic problems on expanders.

Conditioned on OMv, cannot have subpolynomial update time

Constant degree graphs

constant expansion graphs

power-law graphs

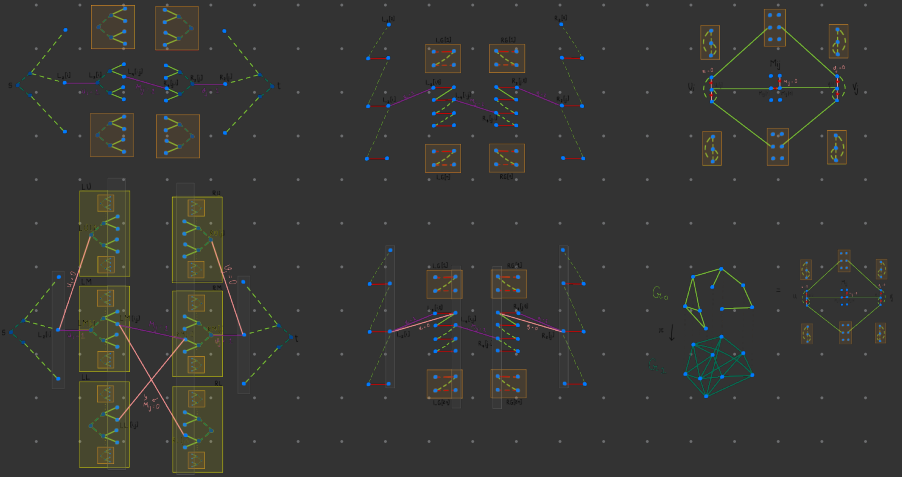
Overview of Techniques

Henzinger et al. '15 and Dahlgaard '16 show lower bounds on general graphs

Overview of Techniques

Henzinger et al. '15 and Dahlgaard '16 show lower bounds on general graphs

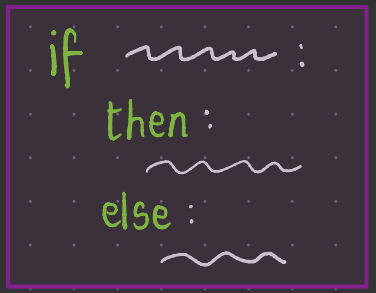
Our gadgets extend these to "sparse" graphs



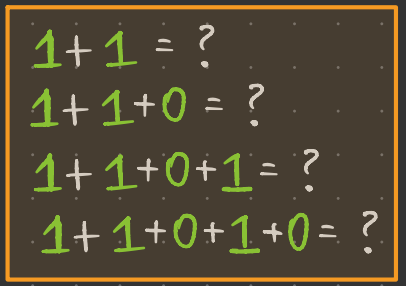
Until now: speeding up dynamic graph algorithms

Next up: private algorithms

Tools to protect privacy



private if statements



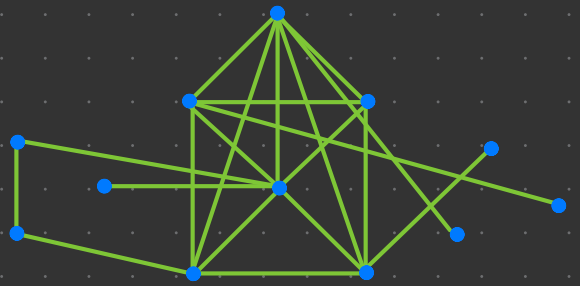
private streaming sums

Until now: speeding up dynamic graph algorithms

INTERLUDE: dense subgraphs

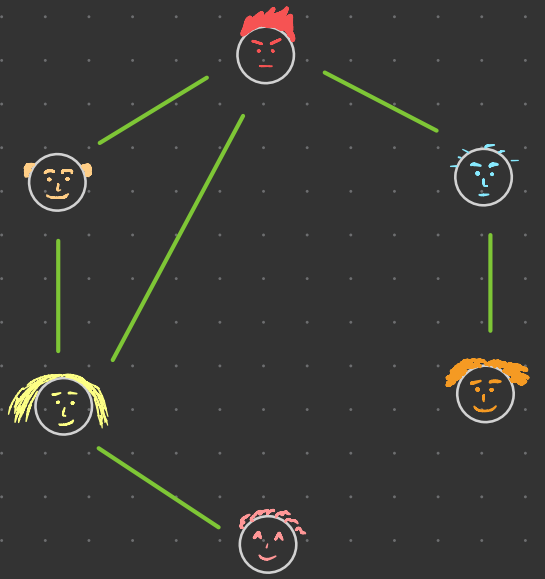
Next up: private algorithms

Densest subgraph



density = average # friendships

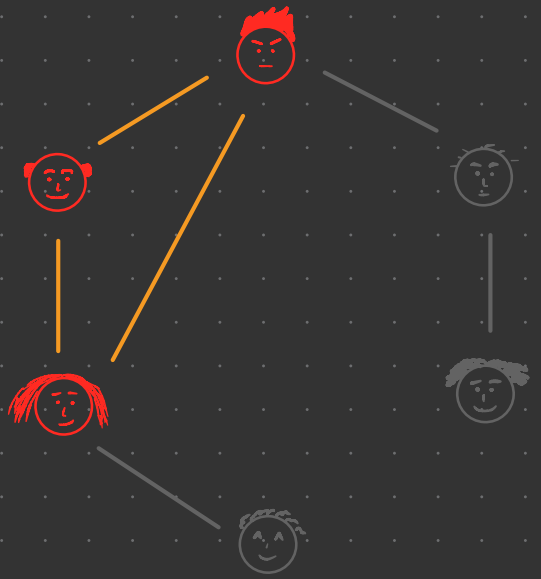
$$\text{density}(S) = \frac{\#\text{edges in } S}{\#\text{vertices in } S}$$



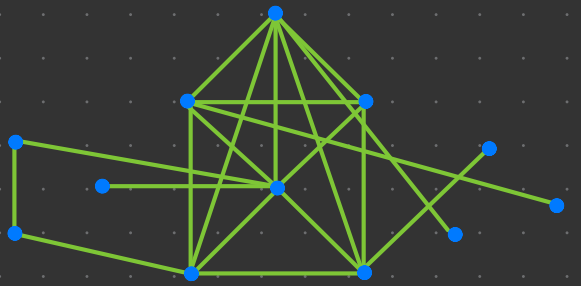
Densest subgraph

density = average # friendships

$$\text{density}(S) = \frac{\#\text{edges in } S}{\#\text{vertices in } S}$$



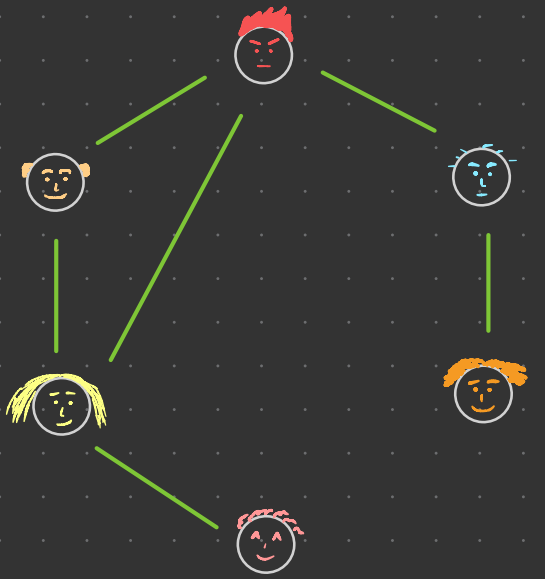
Densest subgraph



density = average # friendships

$$\text{density}(S) = \frac{\#\text{edges in } S}{\#\text{vertices in } S}$$

Goal ~ find high degree subgraph



Densest subgraph

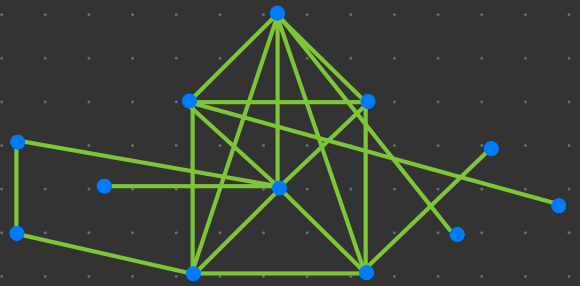
Matula, Beck '83

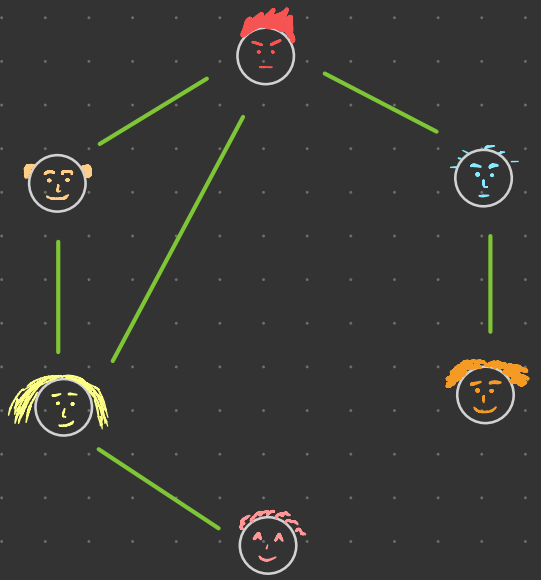
↓

while ∃ vertex :

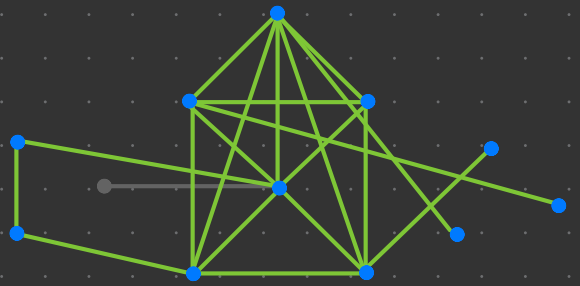
    remove lowest deg. vertex

return densest subgraph seen.





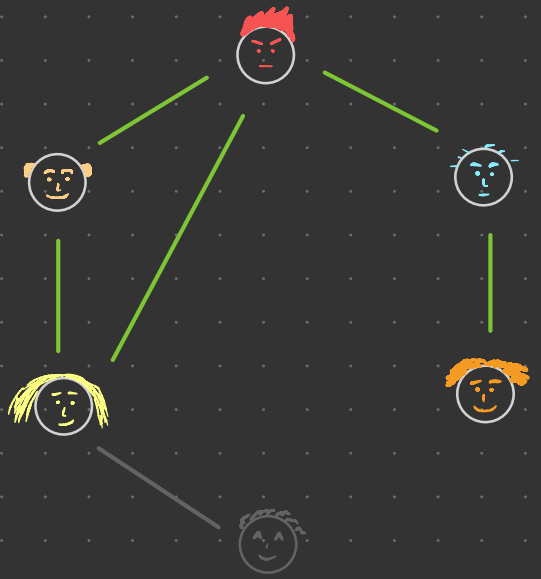
Densest subgraph



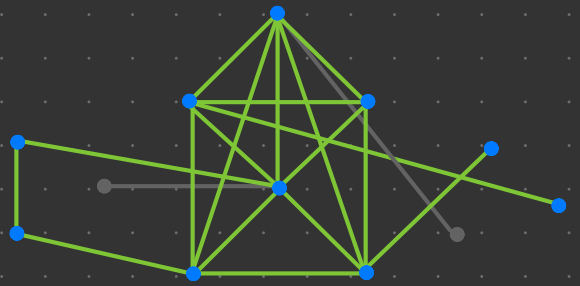
while ∃ vertex :

    remove lowest deg. vertex

return densest subgraph seen.



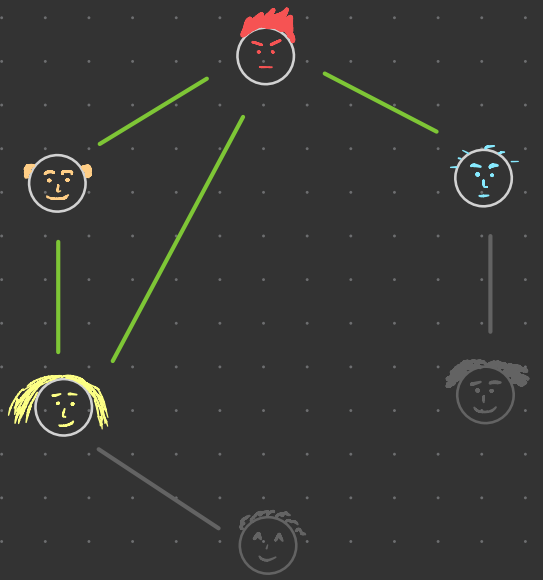
Densest subgraph



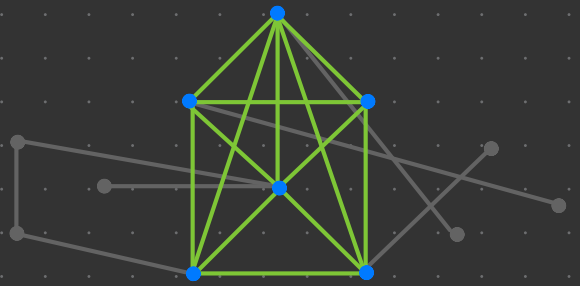
while ∃ vertex :

    remove lowest deg. vertex

return densest subgraph seen.



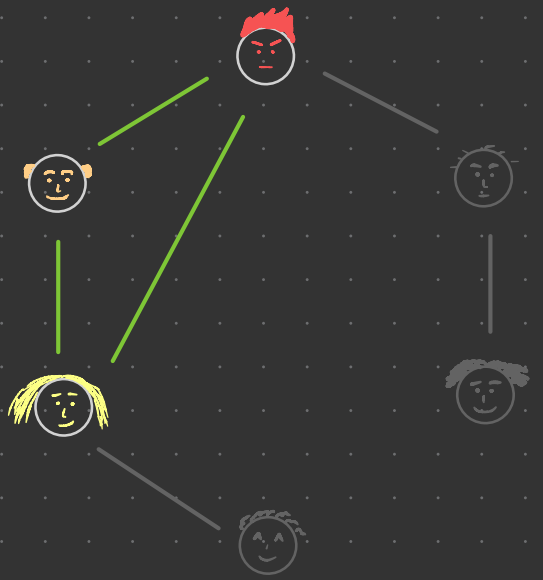
Densest subgraph



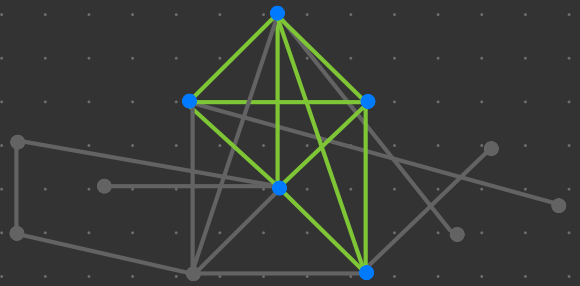
while ∃ vertex :

    remove lowest deg. vertex

return densest subgraph seen.



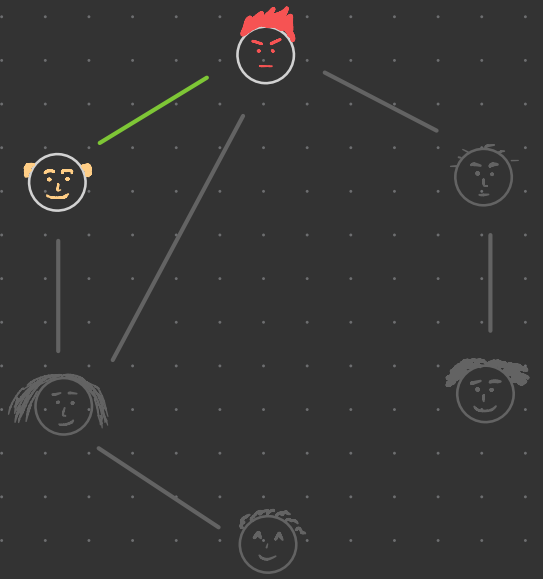
Densest subgraph



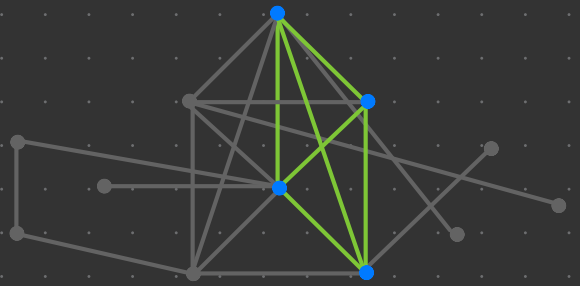
while ∃ vertex :

    remove lowest deg. vertex

return densest subgraph seen.



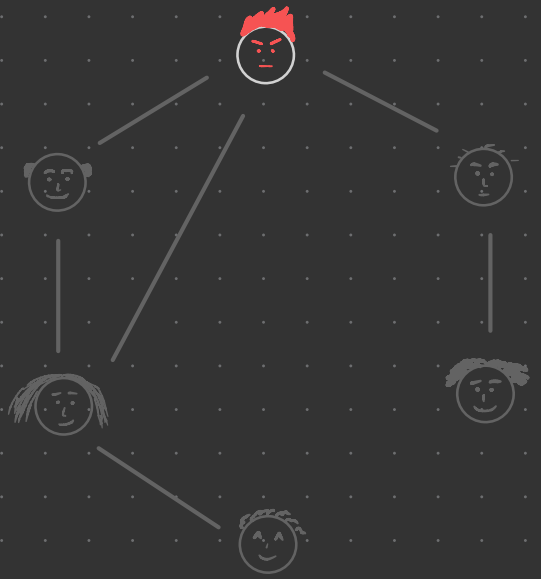
Densest subgraph



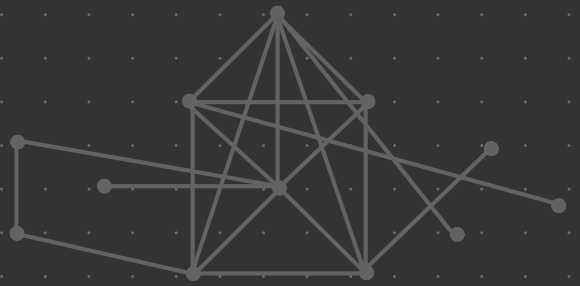
while ∃ vertex :

    remove lowest deg. vertex

return densest subgraph seen.



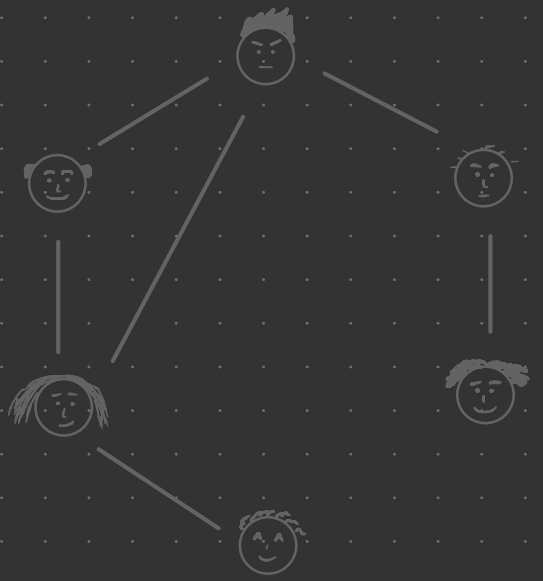
Densest subgraph



while ∃ vertex :

    remove lowest deg. vertex

return densest subgraph seen.

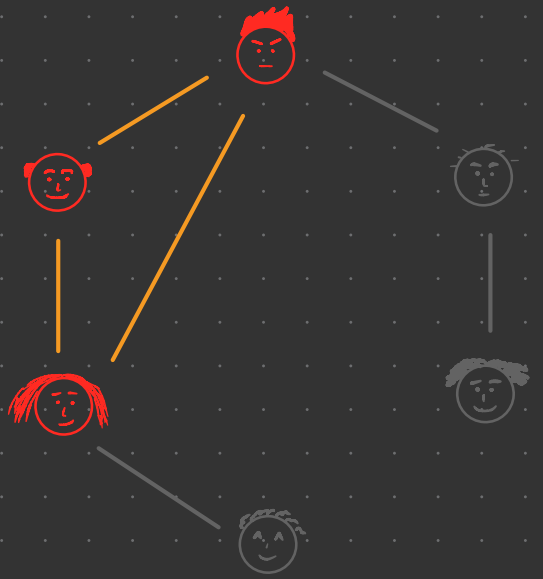


Densest subgraph

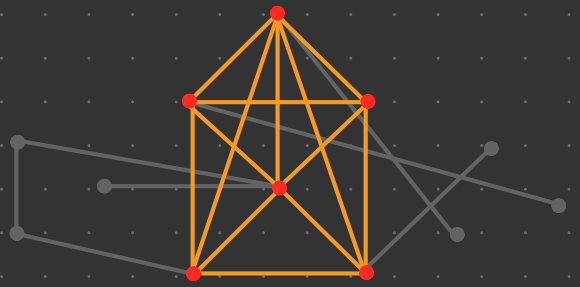
while ∃ vertex :

    remove lowest deg. vertex

return densest subgraph seen.



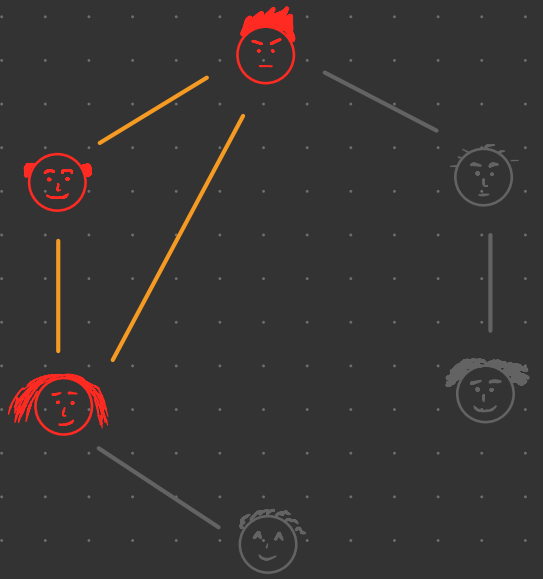
Densest subgraph



while ∃ vertex :

    remove lowest deg. vertex

return densest subgraph seen.



This algorithm is a 2-approximation

↖ Charikar '00

Densest subgraph

while $\exists$ vertex :

$v \leftarrow \min_u \deg(u)$

delete v

for all w neighbour of v :

$\deg(w) = \deg(w) - 1$

return densest subgraph seen.

Densest subgraph

while $\exists$ vertex :

$v \leftarrow \min_u \deg(u)$

delete v

for all w neighbour of v :

$\deg(w) = \deg(w) - 1$

return densest subgraph seen.

$\text{initdeg}(u) = \deg(u) \quad \forall u \in V$

while $\exists$ vertex :

$v \leftarrow \min_u \text{initdeg}(u) - \text{deldeg}(u)$

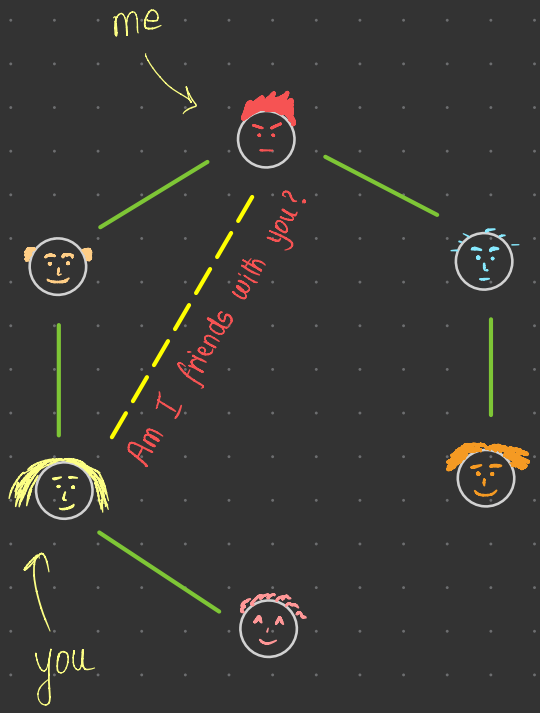
delete v

for all w neighbour of v :

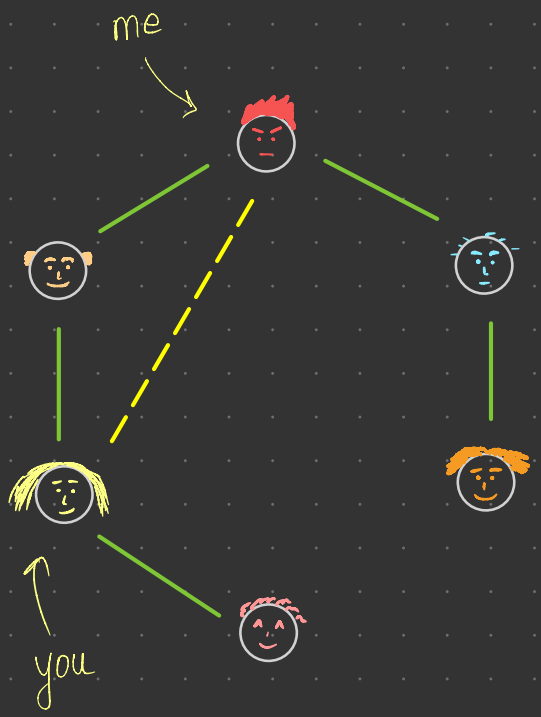
$\text{deldeg}(w) = \text{deldeg}(w) + 1$

return densest subgraph seen.

A nefarious entity wants to know

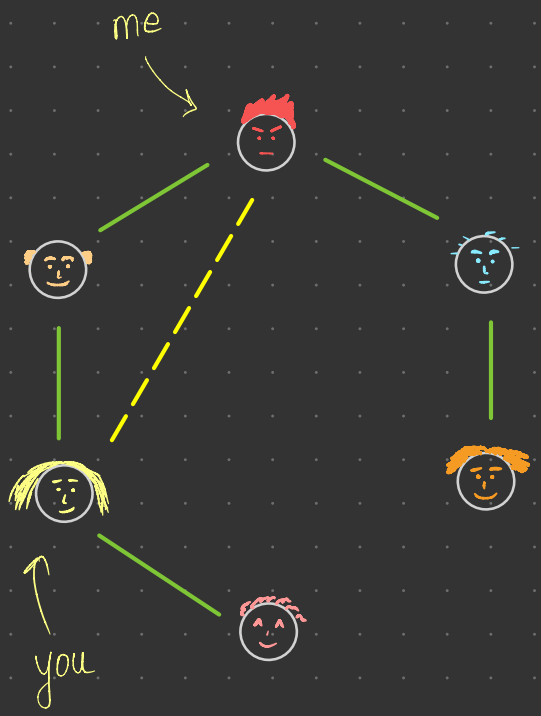


Am I friends with you?



Suppose they know all other friendships (except possibly ours)

Am I friends with you?

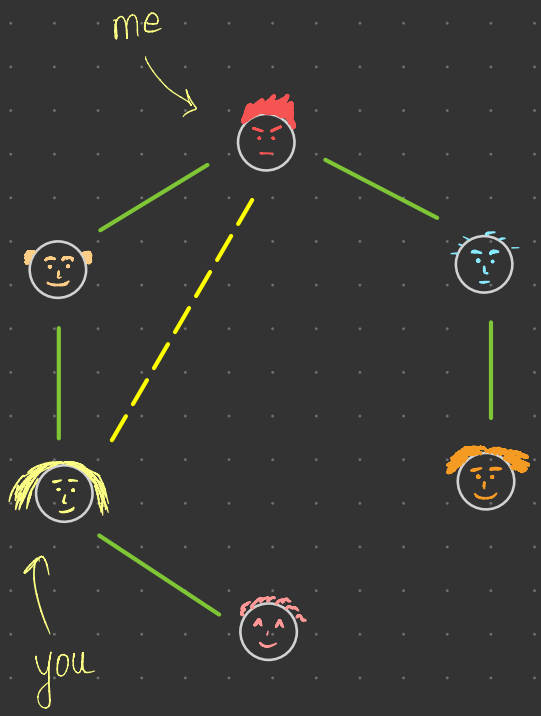


Suppose they know all other friendships (except possibly ours)

They ask me:

"How many friends do you have?"

Am I friends with you?



Suppose they know all other friendships (except possibly ours)

They ask me:

"How many friends do you have?"

3 → They know we are friends

2 → They know we are not

Differential privacy is hiding our friendship

if we are friends : true answer = 3

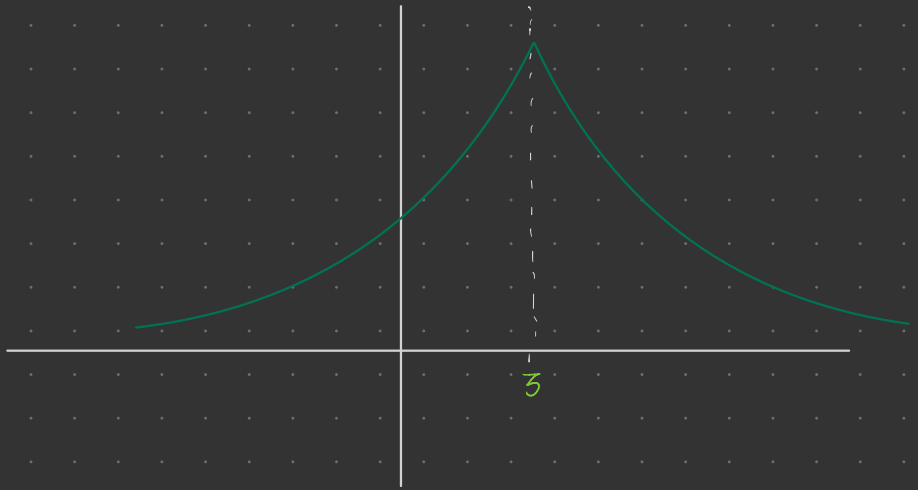
if we are not : true answer = 2

Differential privacy is hiding our friendship

if we are friends : true answer = 3

noisy answer = 3 + noise

if we are not : true answer = 2



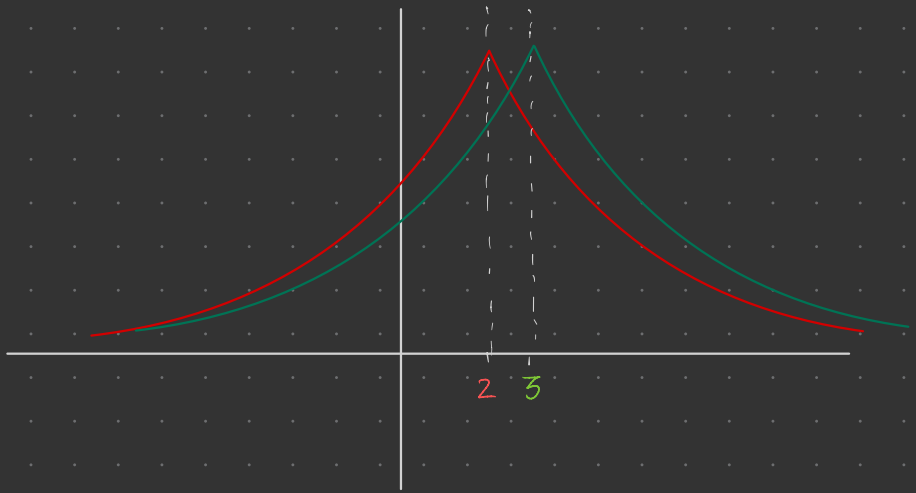
Differential privacy is hiding our friendship

if we are friends : true answer = 3

noisy answer = 3 + noise

if we are not : true answer = 2

noisy answer = 2 + noise



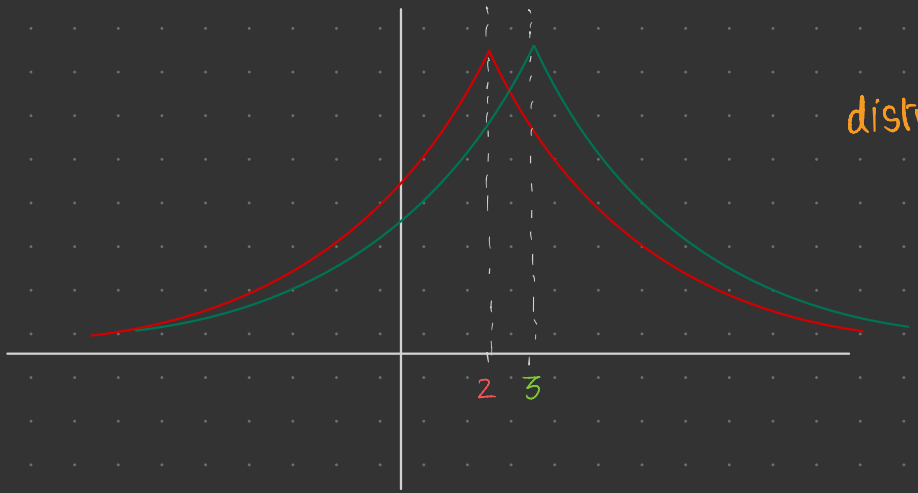
Differential privacy is hiding our friendship

if we are friends : true answer = 3

noisy answer = 3 + noise

if we are not : true answer = 2

noisy answer = 2 + noise



Privacy

|||

distributions are "close"

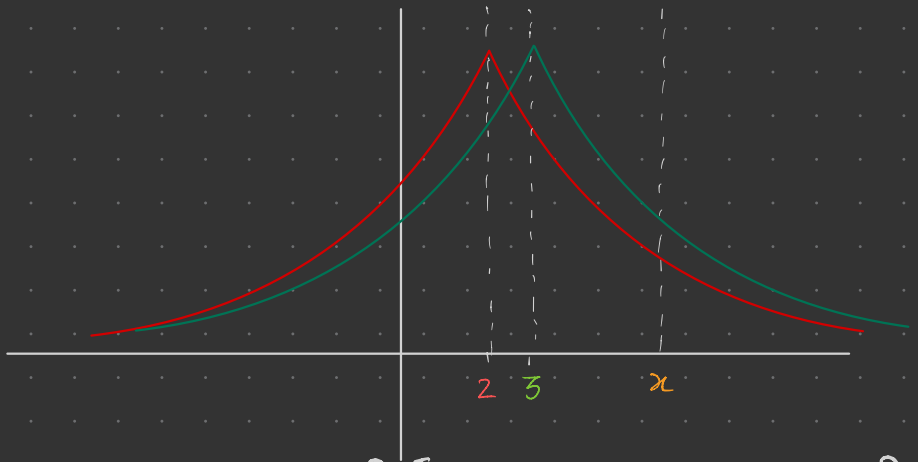
Differential privacy is hiding our friendship

if we are friends : true answer = 3

noisy answer = 3 + noise

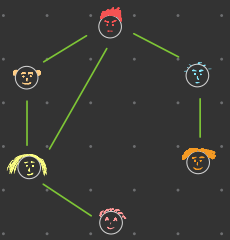
if we are not : true answer = 2

noisy answer = 2 + noise



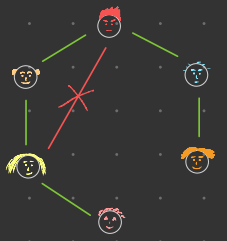
if the given answer is x, $Pr[\text{true answer} = 3] \approx Pr[\text{true answer} = 2]$

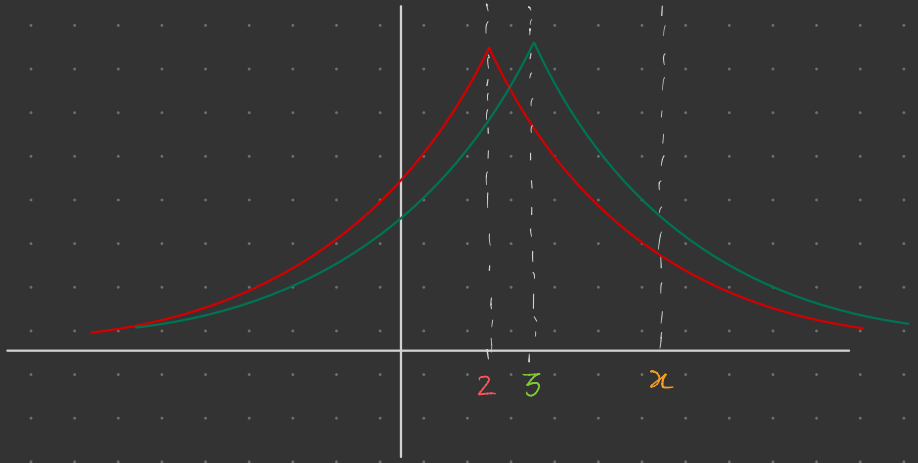
Differential privacy is hiding our friendship



$$\Pr[\mathcal{A}(G) = x] \approx \Pr[\mathcal{A}(G') = x]$$

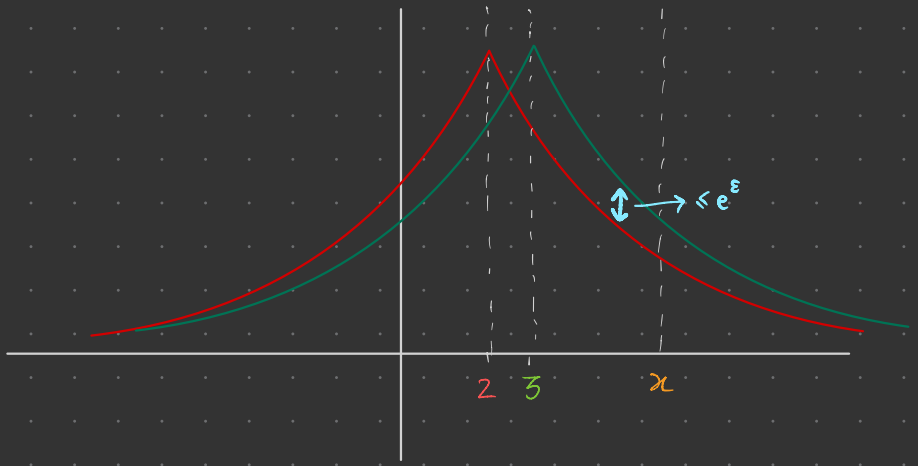
for all x





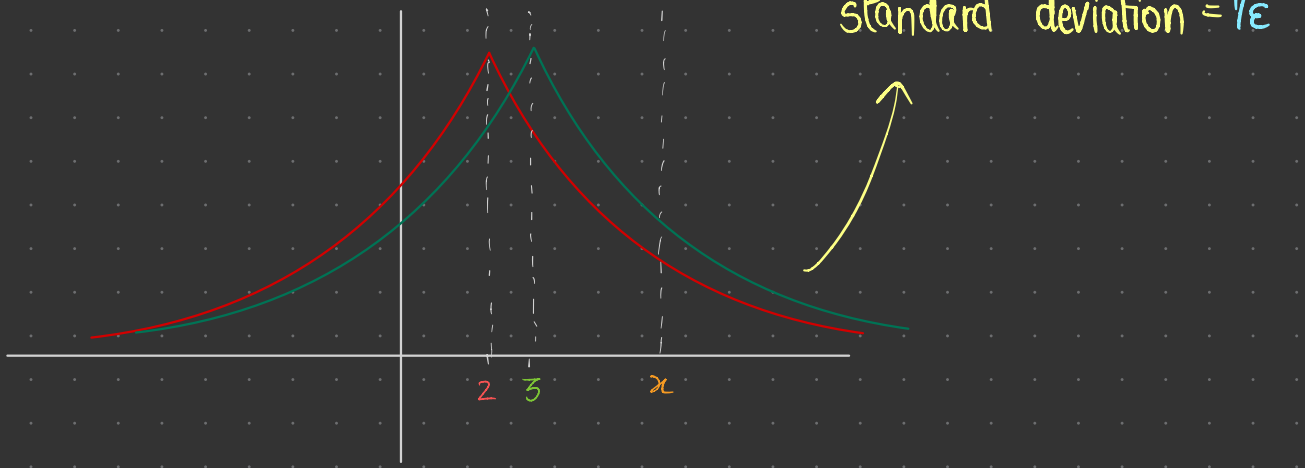
Differential privacy is hiding our friendship

$$e^{-\varepsilon} \cdot \Pr[\mathcal{A}(G') = x] \leq \Pr[\mathcal{A}(G) = x] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(G') = x]$$



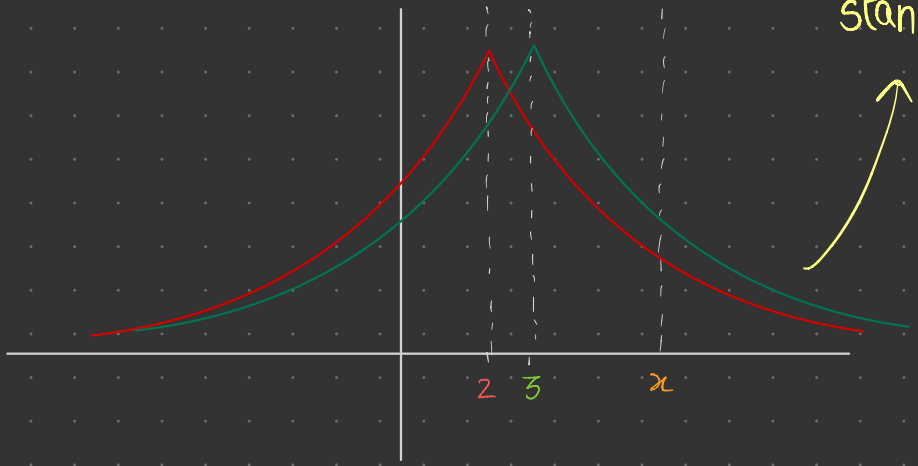
Differential privacy is hiding our friendship

$$e^{-\varepsilon} \cdot \Pr[\mathcal{A}(G') = x] \leq \Pr[\mathcal{A}(G) = x] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(G') = x]$$



Differential privacy is hiding our friendship

$$e^{-\varepsilon} \cdot \Pr[\mathcal{A}(G') = x] \leq \Pr[\mathcal{A}(G) = x] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(G') = x]$$



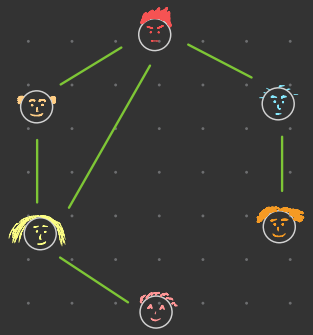
standard deviation = $1/\varepsilon$

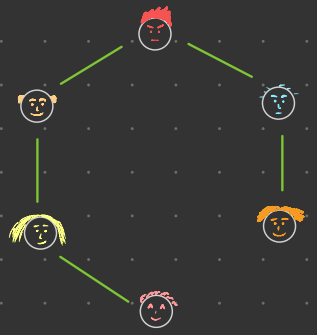
more privacy
$\Downarrow\Downarrow\Downarrow$
less accuracy

How many friends do I have?

if we are friends: $3 + \text{Lap}(1/\varepsilon)$ if we are not: $2 + \text{Lap}(1/\varepsilon)$

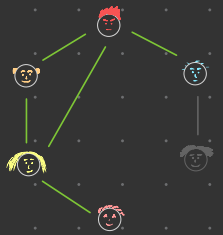
e^{ε}-close

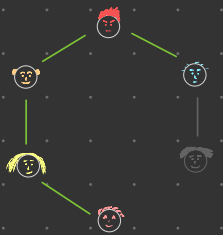




How many friends do I have?

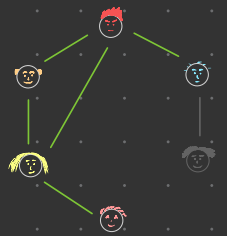
	if we are friends :	if we are not :
Round 1	$3 + \mathrm{Lap}(1/\varepsilon)$	$\leftarrow e^{\varepsilon}$-close $\rightarrow$ $2 + \mathrm{Lap}(1/\varepsilon)$
Round 2	$3 + \mathrm{Lap}(1/\varepsilon)$	$\leftarrow e^{\varepsilon}$-close $\rightarrow$ $2 + \mathrm{Lap}(1/\varepsilon)$

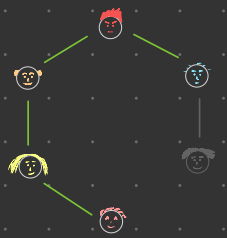




How many friends do I have?

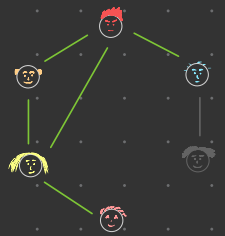
	if we are friends :	if we are not :	
Round 1	$3 + \mathrm{Lap}(1/\varepsilon)$	$2 + \mathrm{Lap}(1/\varepsilon)$	$e^{2\varepsilon}$ - close
Round 2	$3 + \mathrm{Lap}(1/\varepsilon)$	$2 + \mathrm{Lap}(1/\varepsilon)$	

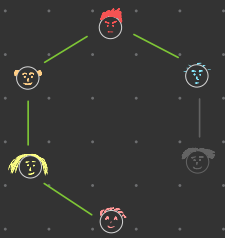




How many friends do I have?

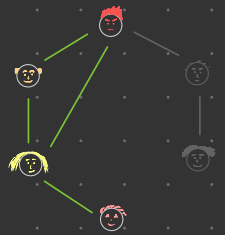
	if we are friends :	if we are not :	
Round 1	$3 + \mathrm{Lap}(2/\varepsilon)$	$2 + \mathrm{Lap}(2/\varepsilon)$	e^{ε} - close
Round 2	$3 + \mathrm{Lap}(2/\varepsilon)$	$2 + \mathrm{Lap}(2/\varepsilon)$	

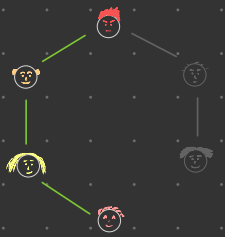




How many friends do I have?

	if we are friends :	if we are not :
Round 1	$3 + \mathrm{Lap}(3/\varepsilon)$	$2 + \mathrm{Lap}(3/\varepsilon)$
Round 2	$3 + \mathrm{Lap}(3/\varepsilon)$	$2 + \mathrm{Lap}(3/\varepsilon)$
Round 3	$2 + \mathrm{Lap}(3/\varepsilon)$	$1 + \mathrm{Lap}(3/\varepsilon)$

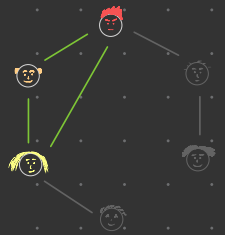


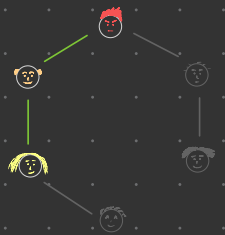


How many friends do I have?

$T = 4$

	if we are friends:	if we are not:
Round 1	$3 + \mathrm{Lap}(T/\varepsilon)$	$2 + \mathrm{Lap}(T/\varepsilon)$
Round 2	$3 + \mathrm{Lap}(T/\varepsilon)$	$2 + \mathrm{Lap}(T/\varepsilon)$
Round 3	$2 + \mathrm{Lap}(T/\varepsilon)$	$1 + \mathrm{Lap}(T/\varepsilon)$
Round 4	$2 + \mathrm{Lap}(T/\varepsilon)$	$1 + \mathrm{Lap}(T/\varepsilon)$

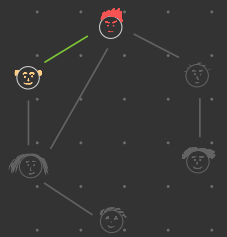


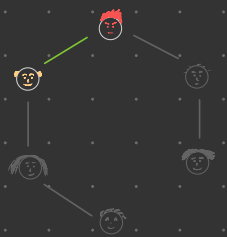


How many friends do I have?

$T = 5$

	if we are friends:	if we are not:
Round 1	$3 + \mathrm{Lap}(T/\varepsilon)$	$2 + \mathrm{Lap}(T/\varepsilon)$
Round 2	$3 + \mathrm{Lap}(T/\varepsilon)$	$2 + \mathrm{Lap}(T/\varepsilon)$
Round 3	$2 + \mathrm{Lap}(T/\varepsilon)$	$1 + \mathrm{Lap}(T/\varepsilon)$
Round 4	$2 + \mathrm{Lap}(T/\varepsilon)$	$1 + \mathrm{Lap}(T/\varepsilon)$
Round 5	$1 + \mathrm{Lap}(T/\varepsilon)$ ← same true answer →	$1 + \mathrm{Lap}(T/\varepsilon)$





How many friends do I have?

$T = 6$

	if we are friends:	if we are not:
Round 1	$3 + \mathrm{Lap}(T/\varepsilon)$	$2 + \mathrm{Lap}(T/\varepsilon)$
Round 2	$3 + \mathrm{Lap}(T/\varepsilon)$	$2 + \mathrm{Lap}(T/\varepsilon)$
Round 3	$2 + \mathrm{Lap}(T/\varepsilon)$	$1 + \mathrm{Lap}(T/\varepsilon)$
Round 4	$2 + \mathrm{Lap}(T/\varepsilon)$	$1 + \mathrm{Lap}(T/\varepsilon)$
Round 5	$1 + \mathrm{Lap}(T/\varepsilon)$	$1 + \mathrm{Lap}(T/\varepsilon)$
Round 6	$0 + \mathrm{Lap}(T/\varepsilon)$	$0 + \mathrm{Lap}(T/\varepsilon)$

$T = n \Rightarrow$ scale of noise $\approx$ scale of answer

How many friends do I have?

$T = 6$

	if we are friends:	if we are not:
Round 1	$3 + \text{Lap}(T/\varepsilon)$	$2 + \text{Lap}(T/\varepsilon)$
Round 2	$3 + \text{Lap}(T/\varepsilon)$	$2 + \text{Lap}(T/\varepsilon)$
Round 3	$2 + \text{Lap}(T/\varepsilon)$	$1 + \text{Lap}(T/\varepsilon)$
Round 4	$2 + \text{Lap}(T/\varepsilon)$	$1 + \text{Lap}(T/\varepsilon)$
Round 5	$1 + \text{Lap}(T/\varepsilon)$	$1 + \text{Lap}(T/\varepsilon)$
Round 6	$0 + \text{Lap}(T/\varepsilon)$	$0 + \text{Lap}(T/\varepsilon)$

reduce # rounds?

↑

Dhulipala et al. '22

$T = n \Rightarrow$ scale of noise $\approx$ scale of answer

How many friends do I have? $T = 6$

	if we are friends:	if we are not:	
Round 1	$3 + \text{Lap}(T/\varepsilon)$	$2 + \text{Lap}(T/\varepsilon)$	both differ by 1
Round 2	$3 + \text{Lap}(T/\varepsilon)$	$2 + \text{Lap}(T/\varepsilon)$	
Round 3	$2 + \text{Lap}(T/\varepsilon)$	$1 + \text{Lap}(T/\varepsilon)$	
Round 4	$2 + \text{Lap}(T/\varepsilon)$	$1 + \text{Lap}(T/\varepsilon)$	
Round 5	$1 + \text{Lap}(T/\varepsilon)$	$1 + \text{Lap}(T/\varepsilon)$	both same
Round 6	$0 + \text{Lap}(T/\varepsilon)$	$0 + \text{Lap}(T/\varepsilon)$	

$T = n \Rightarrow$ scale of noise $\approx$ scale of answer

Did I lose a friend in round t?

	if we are friends	if we are not
Round 1	NO	NO
Round 2	NO	NO
Round 3	YES	YES
Round 4	NO	NO
Round 5	YES	NO
Round 6	YES	YES

Did I lose a friend in round t?

	if we are friends	if we are not
Round 1	0	0
Round 2	0	0
Round 3	1	1
Round 4	0	0
Round 5	1	0
Round 6	1	1

How many friends do I have?

$initdeg(u) = deg(u) \quad \forall u \in V$

while $\exists$ vertex :

$v \leftarrow \min_u initdeg(u) - deldeg(u)$

delete v

for all w neighbour of v :

$deldeg(w) = deldeg(w) + 1$

return densest subgraph seen.

How many friends do I have?

$\text{initdeg}(u) = \deg(u) \quad \forall u \in V$ → privatize with $\text{Lap}(2/\varepsilon)$

while $\exists$ vertex :

$v \leftarrow \min_u \text{initdeg}(u) - \text{deldeg}(u)$

delete v

for all w neighbour of v :

$\text{deldeg}(w) = \text{deldeg}(w) + 1$ → privatize with

$1+1 = ?$
$1+1+0 = ?$
$1+1+0+1 = ?$
$1+1+0+1+0 = ?$

private streaming sums

return densest subgraph seen.

How many friends do I have?

$\text{initdeg}(u) = \deg(u) \quad \forall u \in V$ → privatize with $\text{Lap}(2/\varepsilon)$

while $\exists$ vertex :

$v \leftarrow \min_u \text{initdeg}(u) - \text{deldeg}(u)$

delete v

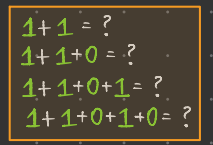
for all w neighbour of v :

$\text{deldeg}(w) = \text{deldeg}(w) + 1$ → privatize with

return densest subgraph seen.

Dwork et al. '10 & Chan et al. '11

only need noise scale $O(\log^2 T)$!



private streaming sums

How many friends do I have?

$\text{initdeg}(u) = \deg(u) \quad \forall u \in V$ → privatize with $\text{Lap}(2/\varepsilon)$

while $\exists$ vertex :

$v \leftarrow \min_u \text{initdeg}(u) - \text{deldeg}(u)$

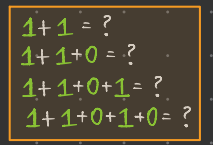
delete v

for all w neighbour of v :

$\text{deldeg}(w) = \text{deldeg}(w) + 1$ → privatize with

Binary Tree Mechanism **BTM**

only need noise scale $O(\log^2 T)$!



private streaming sums

return densest subgraph seen.

Accurate Private Dense Subgraphs

Algorithm for 2-approx densest subgraph with $O(\log^2 n)$ additive error.

→ 1-approx k-core

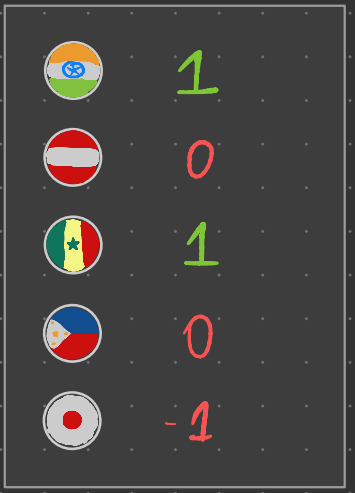
→ packing & inner product lower bounds


All other passports
Fast Track
by invitation only
UK Border
This way

27 Aug 2024

27 Aug	+1	–	+1	–	-1
28 Aug					
29 Aug					
30 Aug					

COUNT



COUNT DISTINCT = 2

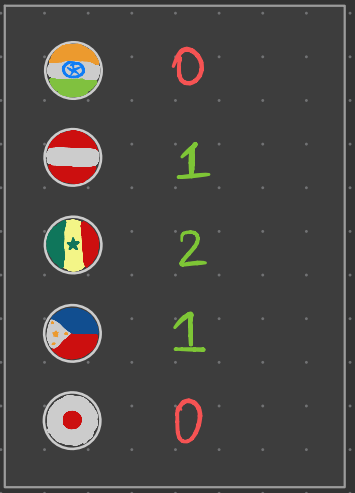
↑

nationalities with count > 0

↖ Jain et al. '23

27 Aug	+1	–	+1	–	-1
28 Aug	-1	+1	+1	+1	+1
29 Aug					
30 Aug					

COUNT



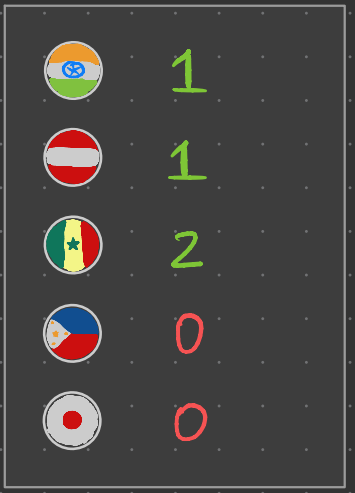
COUNT DISTINCT = 3

↑

nationalities with count > 0

27 Aug	+1	–	+1	–	-1
28 Aug	-1	+1	+1	+1	+1
29 Aug	+1	–	–	-1	–
30 Aug					

COUNT



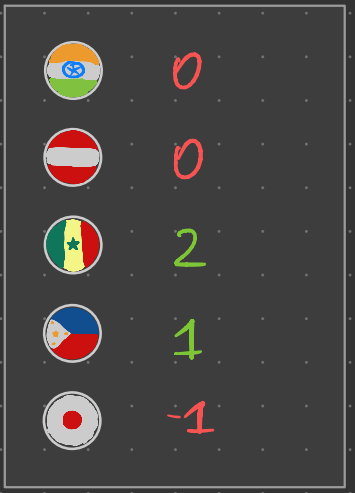
COUNT DISTINCT = 3

↑

nationalities with count > 0

27 Aug	+1	–	+1	–	-1
28 Aug	-1	+1	+1	+1	+1
29 Aug	+1	–	–	-1	–
30 Aug	-1	-1	–	+1	-1

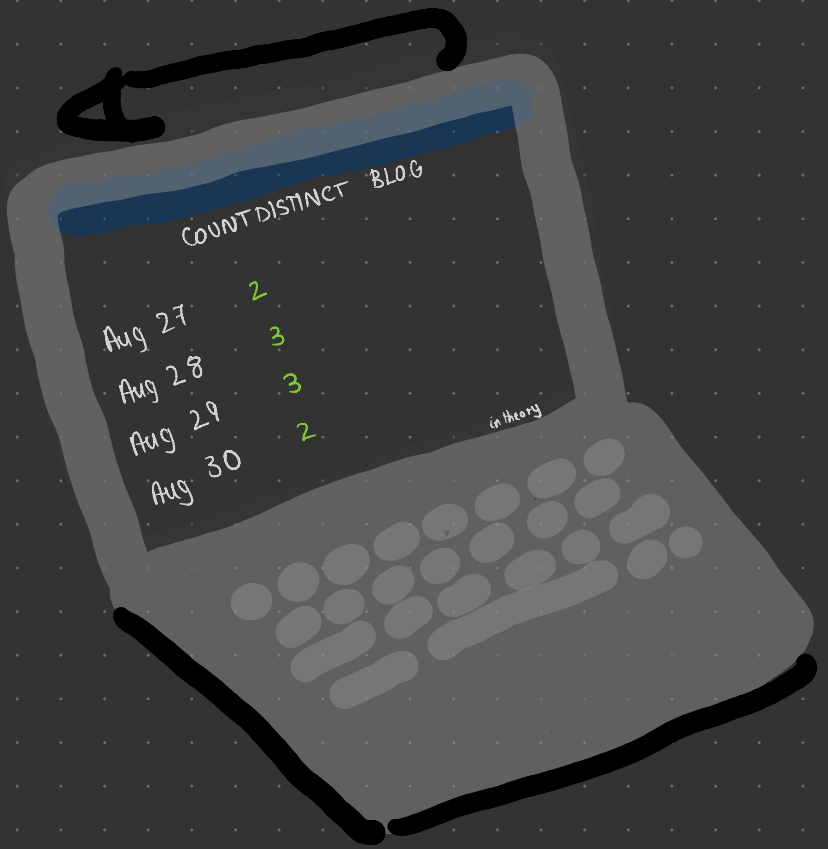
COUNT



COUNT DISTINCT = 2

↑

nationalities with count > 0


COUNT DISTINCT BLOG
Aug 27 2
Aug 28 3
Aug 29 3
Aug 30 2
in theory

adversary knows everyone elses itinerary

wants to figure out if I am travelling by checking the blog

adversary knows everyone elses itinerary

wants to figure out if I am travelling by checking the blog

update in border agent's blog

when I choose to make the trip

when I choose to stay at home

27 Aug	+1	–	+1	–	-1	–	–	+1	–	-1

COUNT DISTINCT = 2

COUNT DISTINCT = 1

Privacy across many days

27 Aug : make trip : $2 + \text{Lap}(1/\varepsilon)$

stay home : $1 + \text{Lap}(1/\varepsilon)$

} e^{ε}-close

Privacy across many days

27 Aug : make trip : $2 + \text{Lap}(1/\varepsilon)$
stay home : $1 + \text{Lap}(1/\varepsilon)$
} e^{ε}-close

29 Aug : make trip : $3 + \text{Lap}(1/\varepsilon)$
stay home : $2 + \text{Lap}(1/\varepsilon)$
} e^{ε}-close

Privacy across many days

27 Aug : make trip : $2 + \text{Lap}(1/\varepsilon)$
stay home : $1 + \text{Lap}(1/\varepsilon)$
} e^{ε}-close

29 Aug : make trip : $3 + \text{Lap}(1/\varepsilon)$
stay home : $2 + \text{Lap}(1/\varepsilon)$
} e^{ε}-close

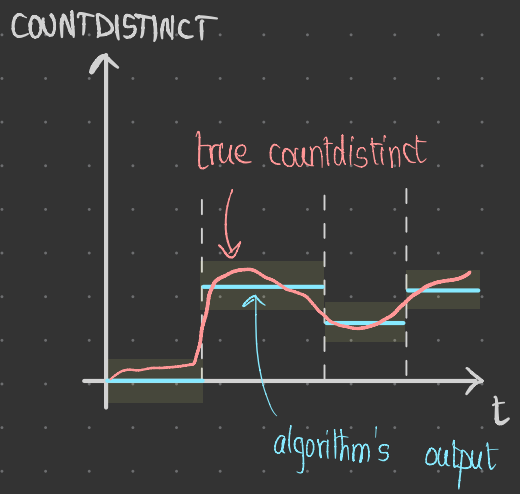
$$\text{Noise} \propto \frac{\#\text{outputs of algorithm}}{\varepsilon}$$

Periodic Recomputation

```
if |COUNTDISTINCT - ESTIMATE| > Threshold :
        recompute ESTIMATE
        start new private if
else :
        output ESTIMATE
```

Periodic Recomputation

```
if |COUNTDISTINCT - ESTIMATE| > Threshold :
        recompute ESTIMATE
        start new private if
else :
        output ESTIMATE
```



						COUNT	FLIP
27 Aug	+1	–	+1	–	−1	0	0
28 Aug	−1	+1	+1	+1	+1	1	1
29 Aug	+1	–	–	−1	–	0	2
30 Aug	−1	−1	–	+1	−1	1	3

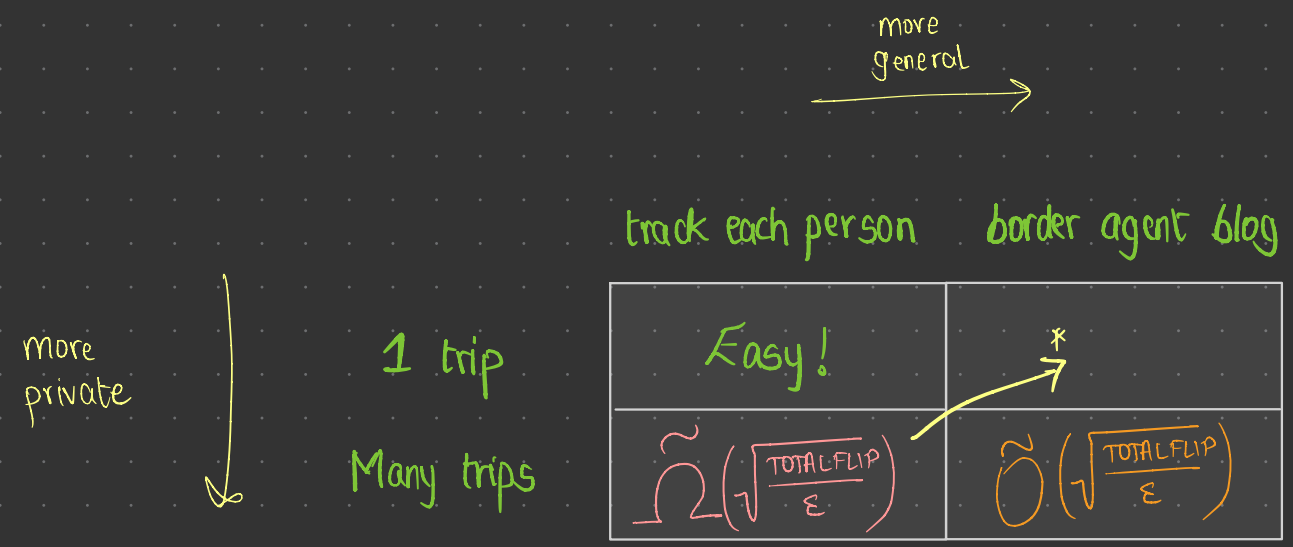
FLIP ( ) = # times COUNT ( ) goes from >0 to =0 or vice-versa

						COUNT	FLIP
27 Aug	+1	–	+1	–	−1	0	0
28 Aug	−1	+1	+1	+1	+1	1	1
29 Aug	+1	–	–	−1	–	0	2
30 Aug	−1	−1	–	+1	−1	1	3

FLIP ( ) = # times COUNT ( ) goes from >0 to $=0$ or vice-versa

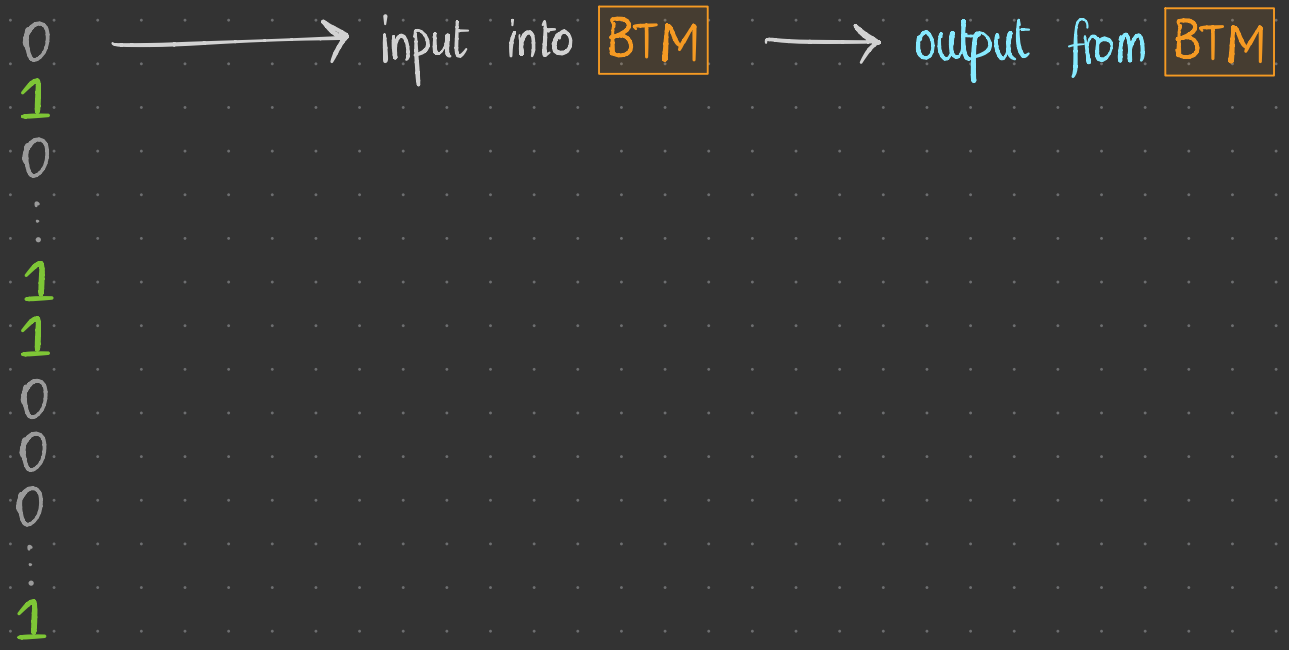
MAXFLIP ↑ Jain et al. '23

TOTALFLIP ↑ ours

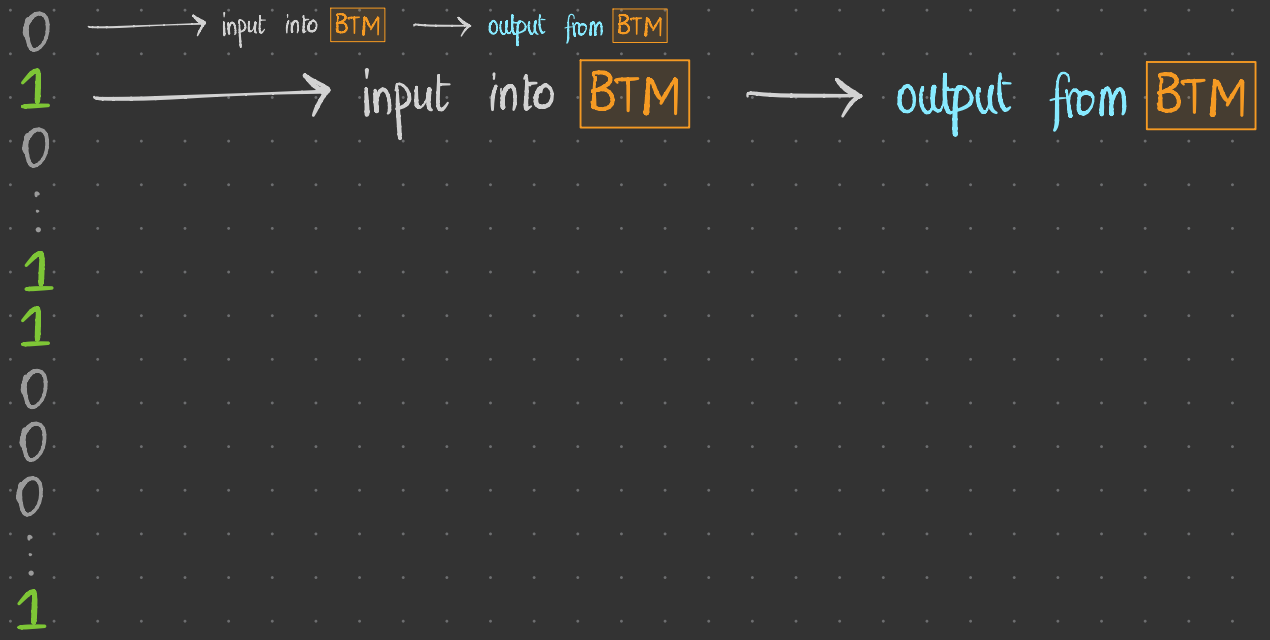


* for algorithms that only depend on COUNTDISTINCT values

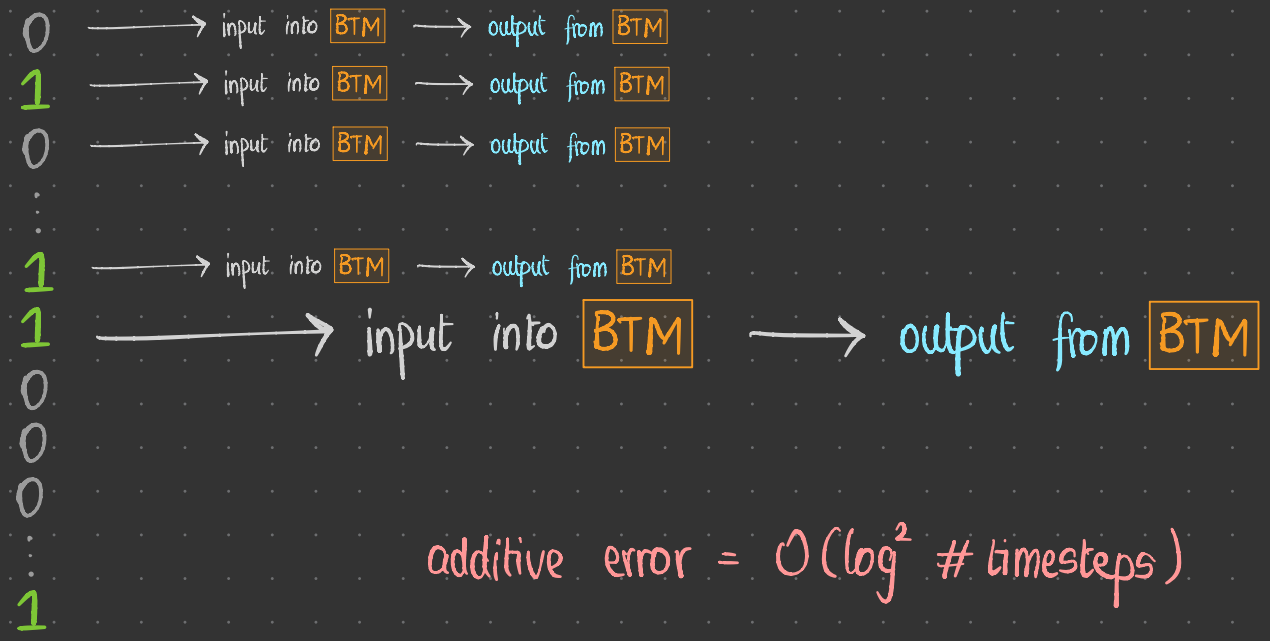
Combining streaming sums with if statements



Combining streaming sums with if statements

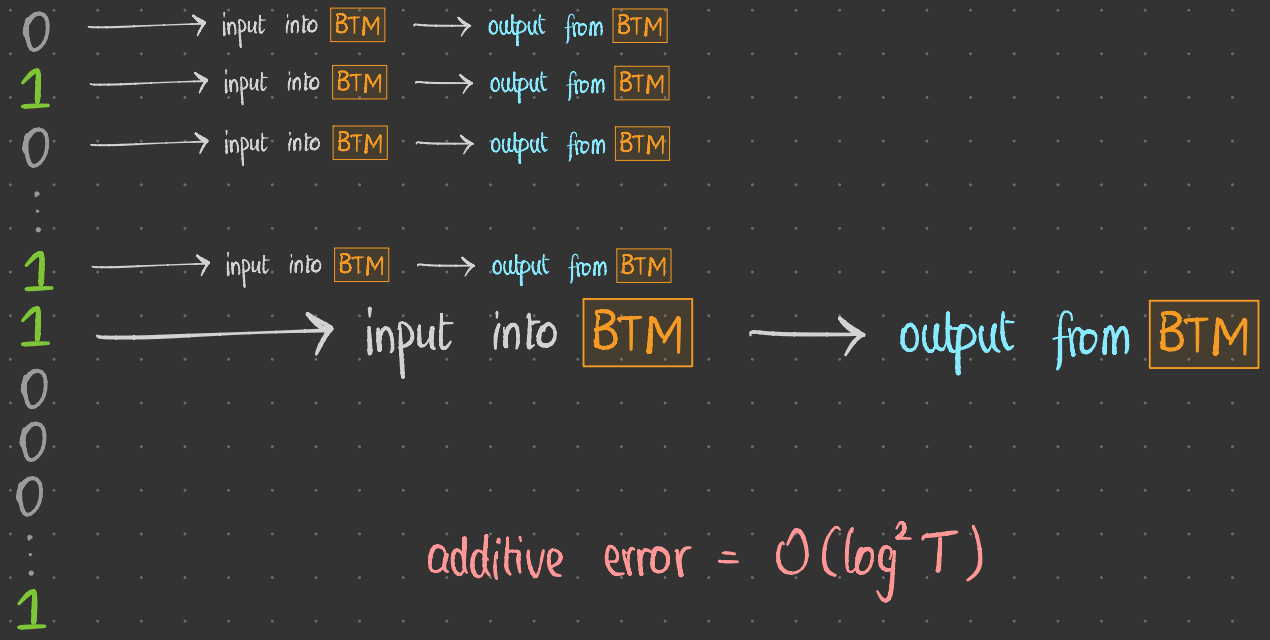


Combining streaming sums with if statements



additive error = $O(\log^2 \#\text{timesteps})$

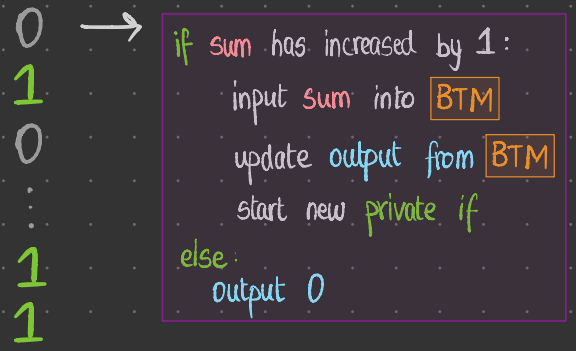
Combining streaming sums with if statements



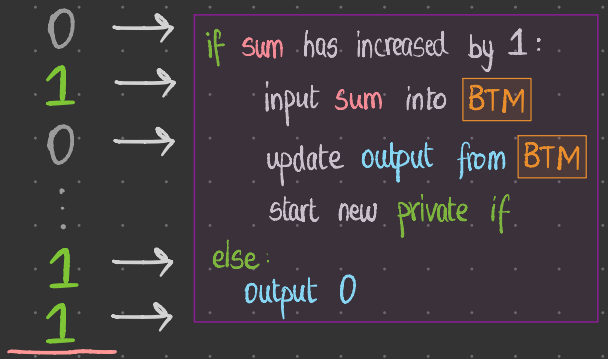
additive error = $O(\log^2 T)$

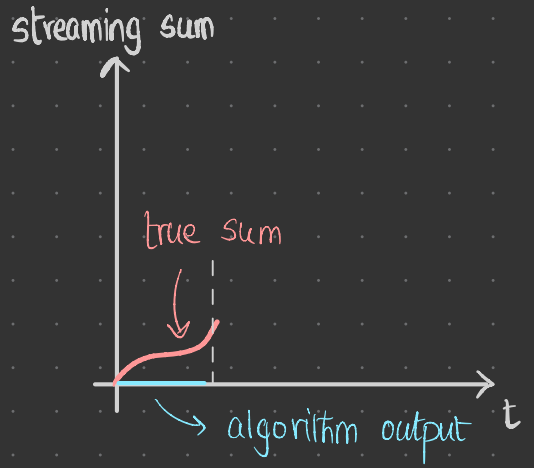
Combining streaming sums with if statements

↳ Dwork et al. 2015



Combining streaming sums with if statements





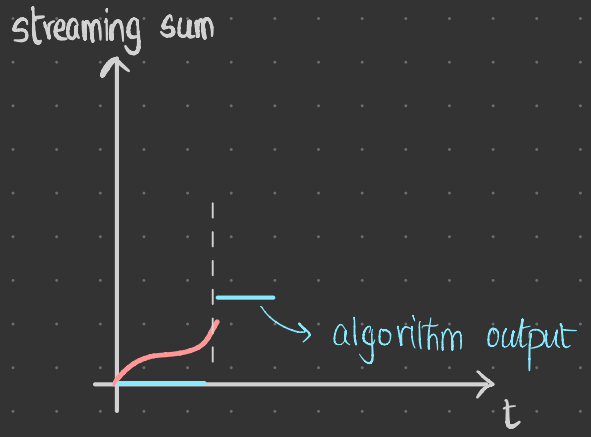
Combining streaming sums with if statements

0 → 1 → 0 → ⋮ → 1 → 1

if sum has increased by 1:
    input sum into BTM
    update output from BTM
    start new private if
else:
    output 0

0 → 0 → 0 → ⋮ → 1

if sum has increased by 1:
    input sum into BTM
    update output from BTM
    start new private if
else:
    output last output of BTM



Combining streaming sums with if statements

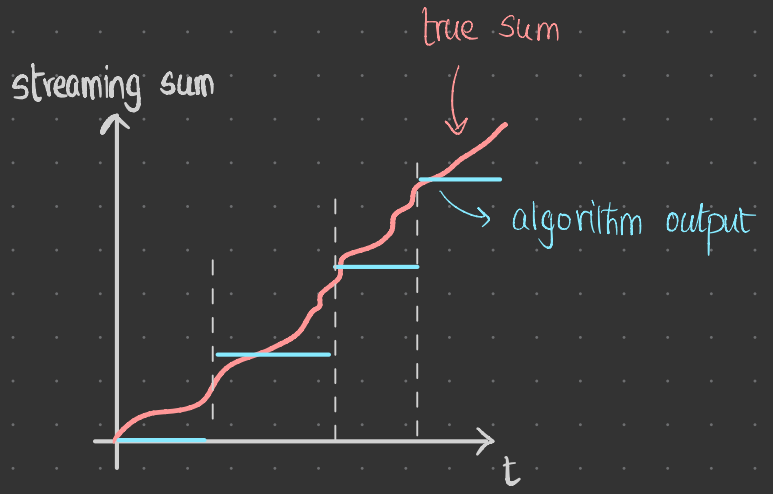
0 → 1 → 0 → ⋮ → 1 → 1

if sum has increased by 1:
    input sum into BTM
    update output from BTM
    start new private if
else:
    output 0

0 → 0 → 0 → ⋮ → 1 →

if sum has increased by 1:
    input sum into BTM
    update output from BTM
    start new private if
else:
    output last output of BTM

⋮



Combining streaming sums with if statements

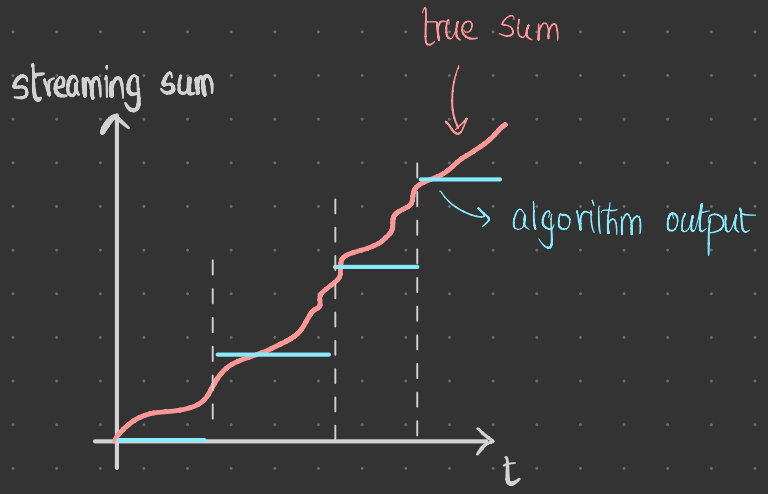
0 → 1 → 0 → ⋮ → 1 → 1

if sum has increased by 1:
    input sum into BTM
    update output from BTM
    start new private if
else:
    output 0

0 → 0 → 0 → ⋮ → 1

if sum has increased by 1:
    input sum into BTM
    update output from BTM
    start new private if
else:
    output last output of BTM

⋮



additive error = $O(\log^2 \#\text{ones} + \log T)$

Combining streaming sums with if statements

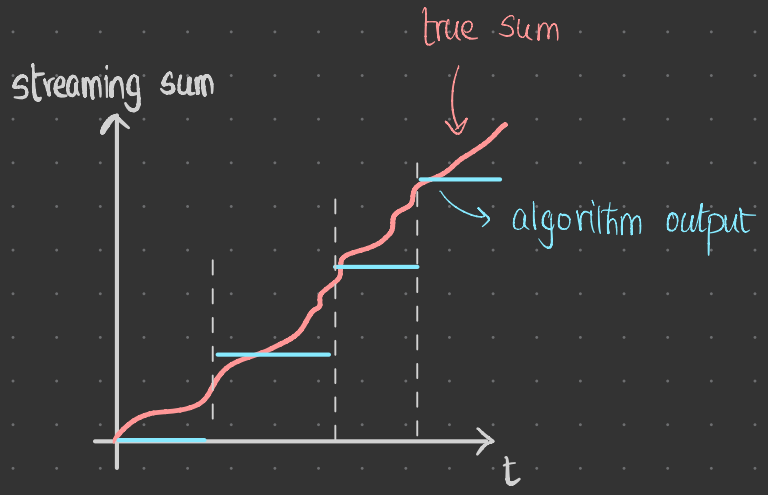
0 → 1 → 0 → ⋮ → 1 → 1

if sum has increased by 1:
    input sum into BTM
    update output from BTM
    start new private if
else:
    output 0

0 → 0 → 0 → ⋮ → 1

if sum has increased by 1:
    input sum into BTM
    update output from BTM
    start new private if
else:
    output last output of BTM

⋮



additive error = $O(\log^2 n + \log T)$

vs

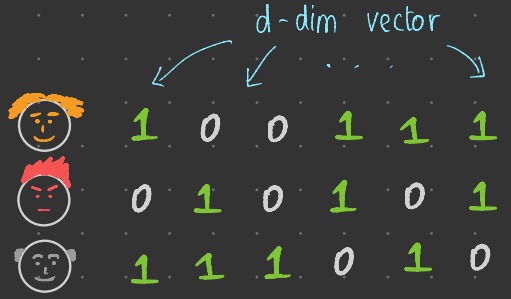
$\Omega(\log T)$

The multi-dimensional setting

The multi-dimensional setting

Additive Error $= O(\log^2 n + \log T)$

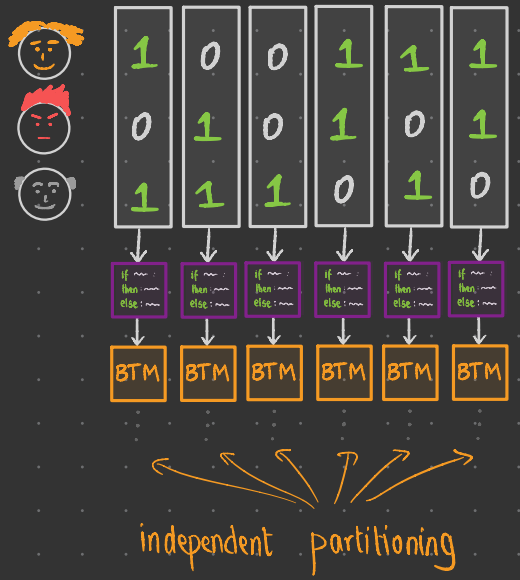
The multi-dimensional setting



$$\text{Additive Error} = O(d \log^2 n_{max} + d \log T)$$

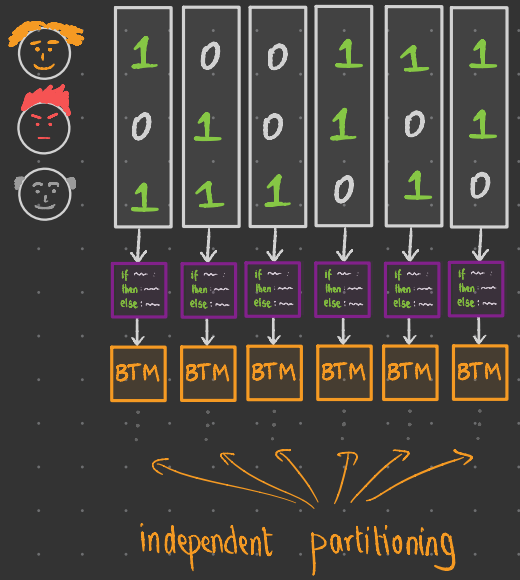
n_{max}: maximum column sum

The multi-dimensional setting



$$\text{Additive Error} = O(d \log^2 n_{max} + d \log T)$$

The multi-dimensional setting

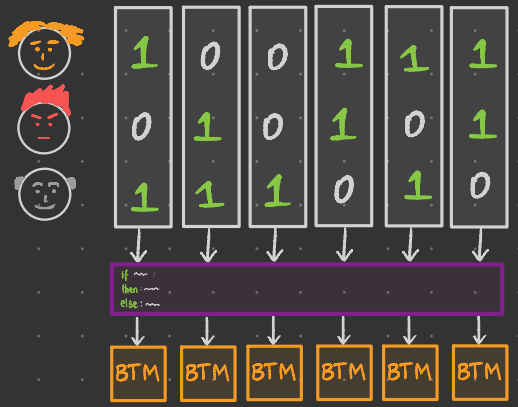


$$\text{Additive Error} = O(d \log^2 n_{max} + d \log T)$$

vs

$$\Omega(d + \log T)$$

The multi-dimensional setting

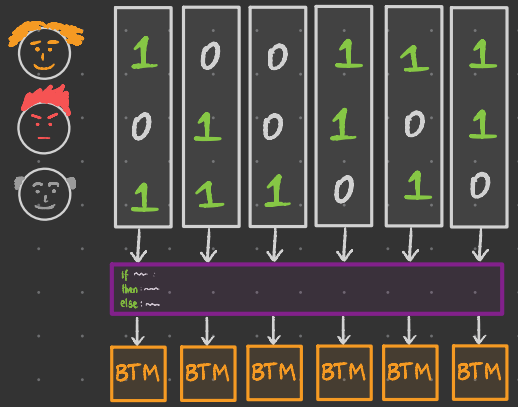


$$\text{Additive Error} = O(d \log^2 n_{max} + \log T)$$

vs

$$\Omega(d + \log T)$$

The multi-dimensional setting

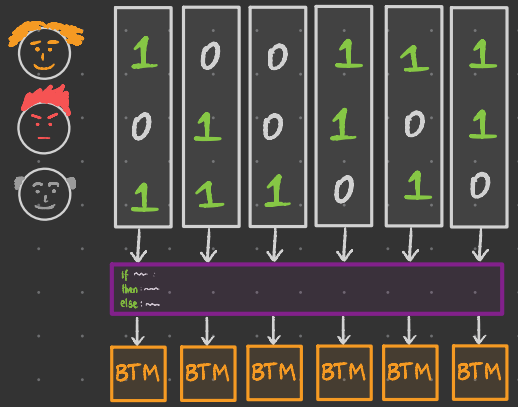


$$\text{Additive Error} = O(d \log^2 n_{max} + \log T)$$

vs

$$\Omega(d + \log T)$$

The multi-dimensional setting



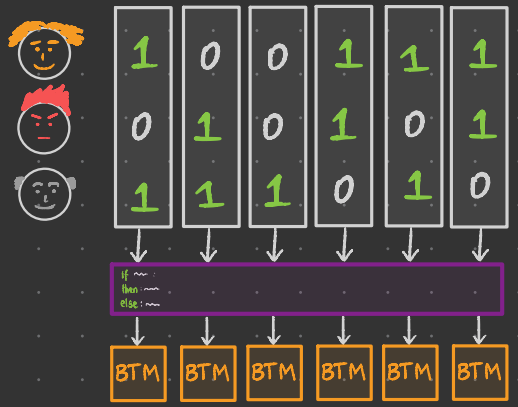
Additive Error $= O(d \log^2 n_{max} + \log T)$

vs

$\Omega(d + \log T)$

What if I only want minimum column sum?

The multi-dimensional setting



Additive Error $= O(d \log^2 n_{max} + \log T)$?

n_{min}

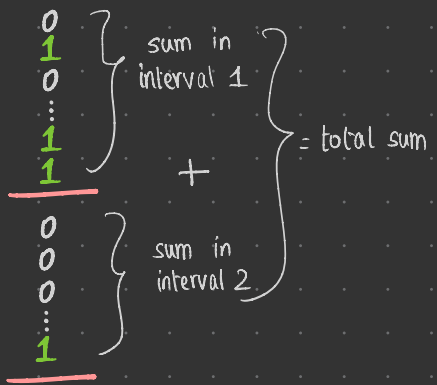
vs

$\Omega(d + \log T)$

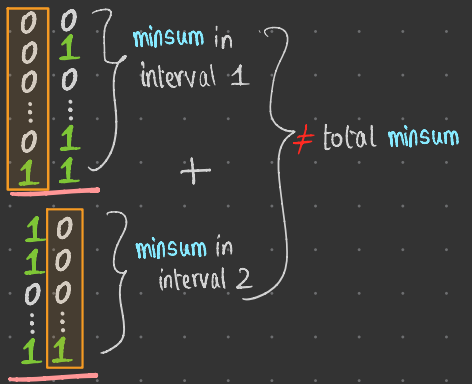
What if I only want minimum column sum?

Minimum column sum

SINGLE DIMENSION

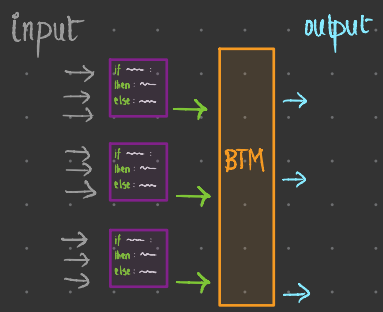


MULTIPLE DIMENSIONS

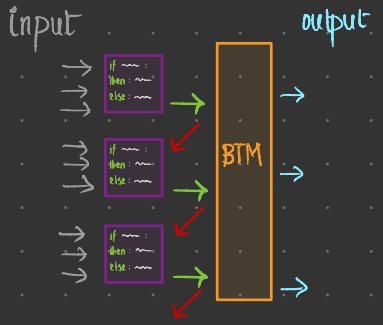


Minimum column sum

PRIOR WORK



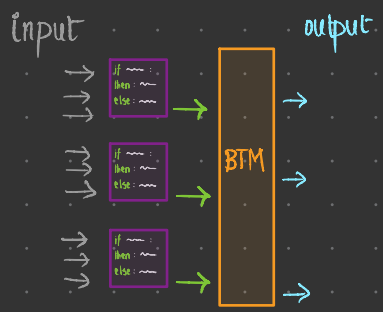
OURS



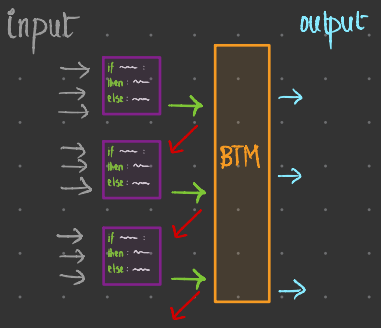
circular dependencies !

Minimum column sum

PRIOR WORK



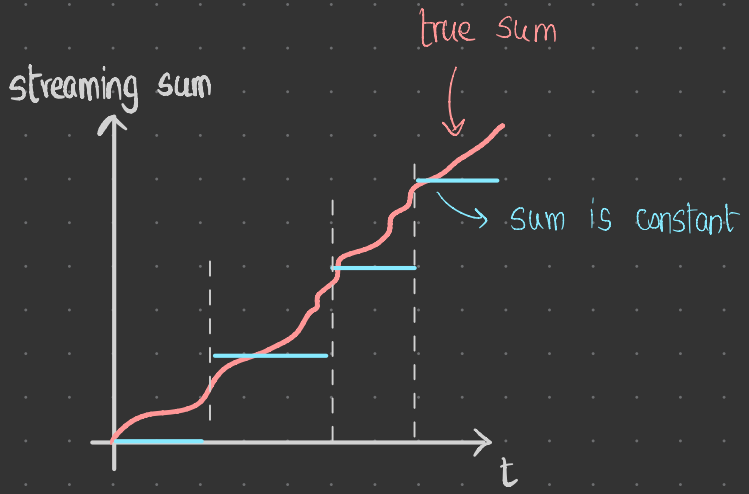
OURS



circular dependencies ! can be fixed

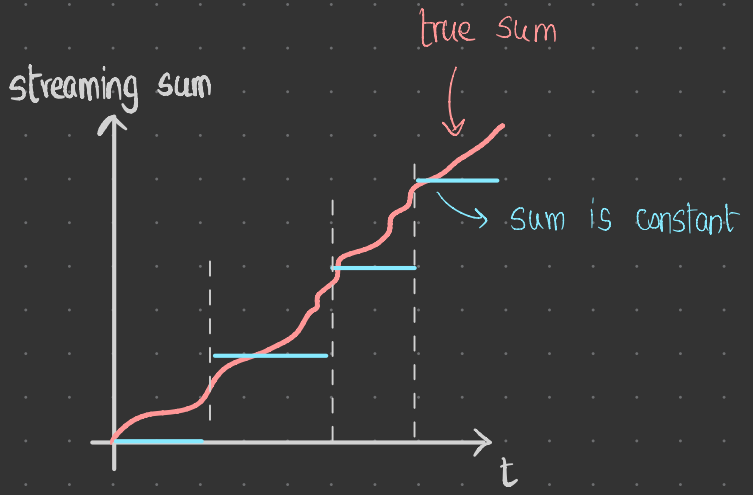
Negative inputs

PRIOR WORK

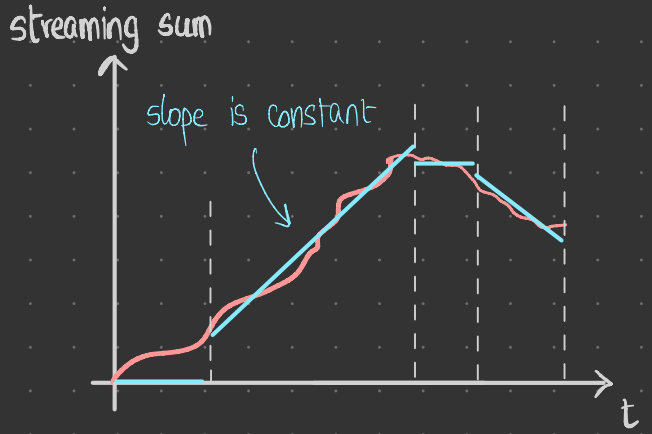


Negative inputs

PRIOR WORK

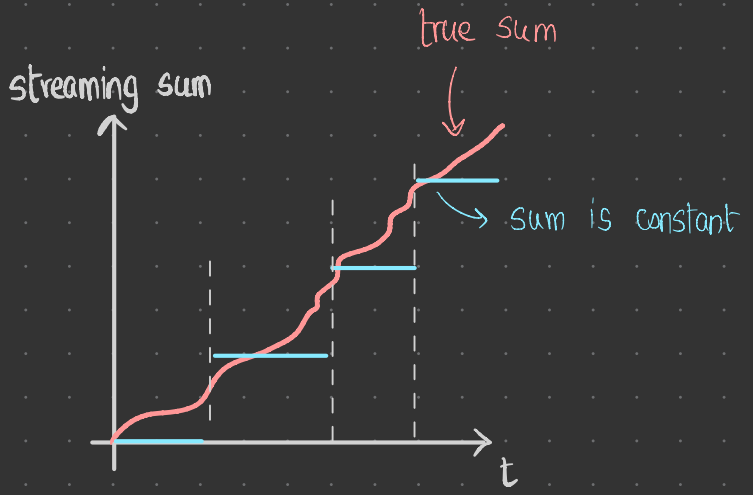


OURS

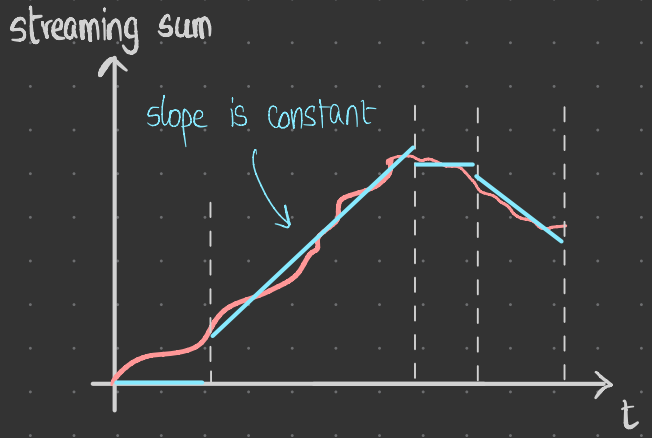


Negative inputs

PRIOR WORK

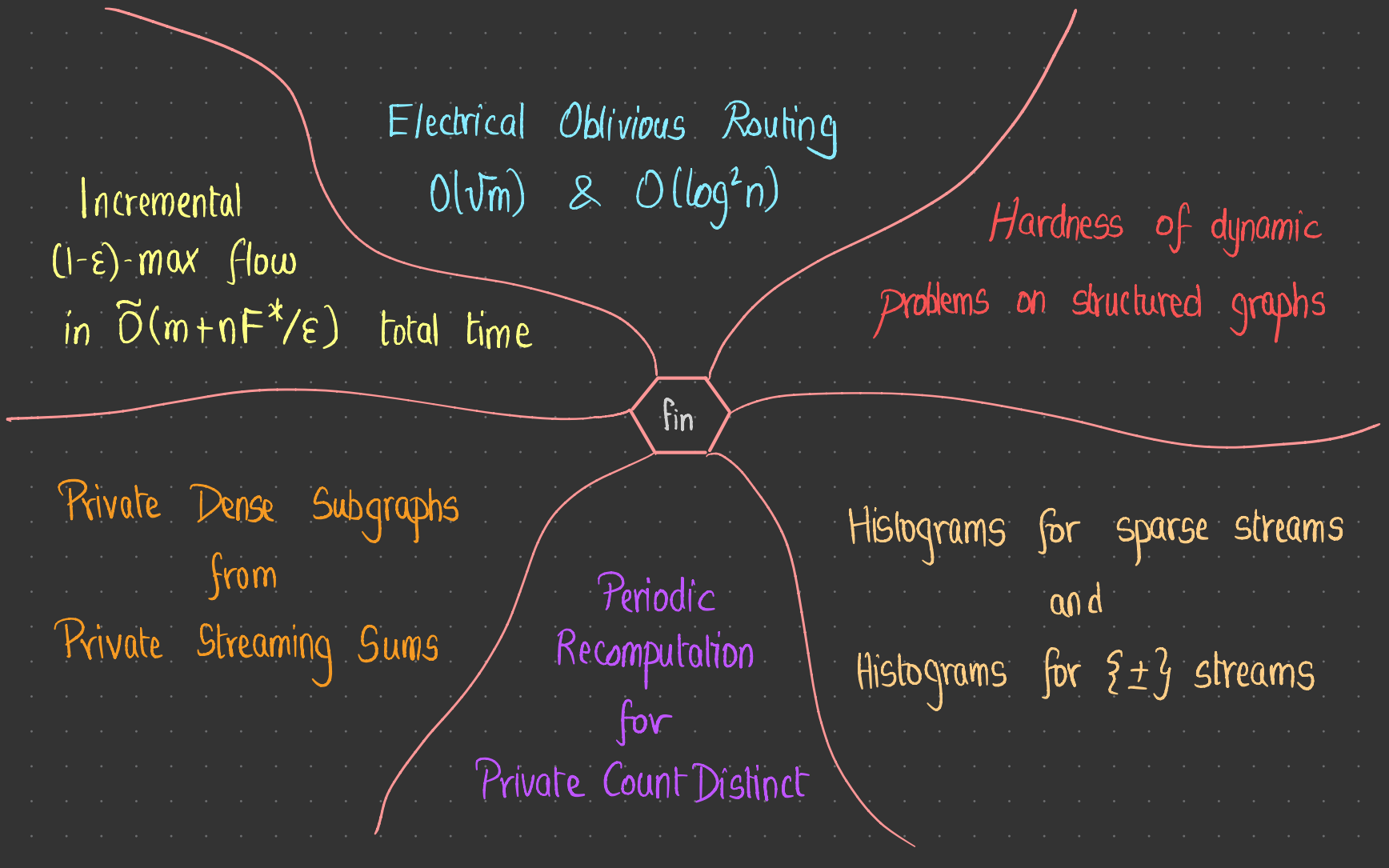


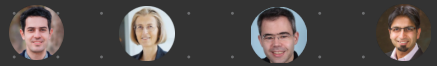
OURS



additive error = $O(\log^2 \#\text{flips} + \log T)$

fin


Electrical Oblivious Routing
$O(\sqrt{m})$ & $O(\log^2 n)$
Incremental
$(1-\varepsilon)$-max flow
in $\tilde{O}(m+nF^*/\varepsilon)$ total time
Hardness of dynamic
Problems on structured graphs
fin
Private Dense Subgraphs
from
Private Streaming Sums
Periodic
Recomputation
for
Private Count Distinct
Histograms for sparse streams
and
Histograms for $\{\pm\}$ streams



Electrical Oblivious Routing

$O(\sqrt{m})$ & $O(\log^2 n)$

Incremental

$(1-\varepsilon)$-max flow

in $\tilde{O}(m+nF^*/\varepsilon)$ total time

Hardness of dynamic

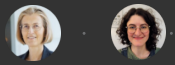
Problems on structured graphs

fin

Private Dense Subgraphs

from

Private Streaming Sums



Periodic

Recomputation

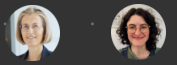
for

Private Count Distinct

Histograms for sparse streams

and

Histograms for $\{\pm\}$ streams



OuMv reductions

vector Matrix vector multiplication

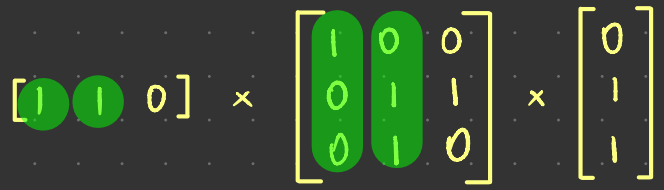
$$\begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

is it $= 0$?

is it > 0 ?

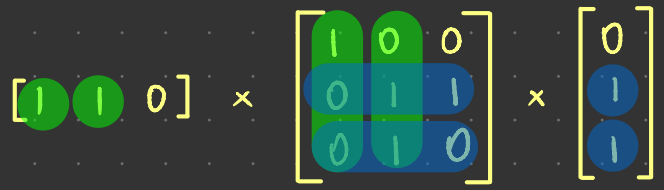
OuMv reductions

vector Matrix vector multiplication



OuMv reductions

vector Matrix vector multiplication



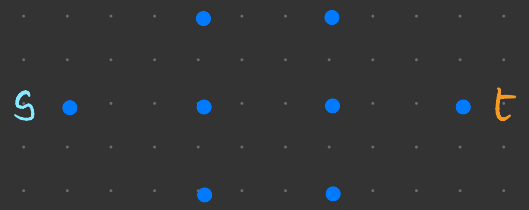
OuMv reductions

vector Matrix vector multiplication

$$\underset{u}{\begin{bmatrix}1 & 1 & 0\end{bmatrix}} \times \underset{M}{\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 1\\ 0 & 1 & 0\end{bmatrix}} \times \underset{v}{\begin{bmatrix}0\\ 1\\ 1\end{bmatrix}}$$

Henzinger et al. '15

$s \to t$ distance



$uMv = 0$ $\quad$ $dist(s,t) = 3$

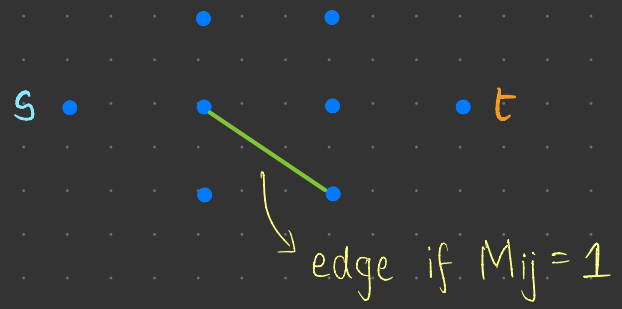
$uMv > 0$ $\quad$ $dist(s,t) \geq 5$

OuMv reductions

vector Matrix vector multiplication

$$\underset{u}{\begin{bmatrix}1 & 1 & 0\end{bmatrix}} \times \underset{M}{\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 1\\ 0 & 1 & 0\end{bmatrix}} \times \underset{v}{\begin{bmatrix}0\\ 1\\ 1\end{bmatrix}}$$

$s \to t$ distance



$uMv = 0$

$uMv > 0$

$dist(s,t) = 3$

$dist(s,t) \geq 5$

OuMv reductions

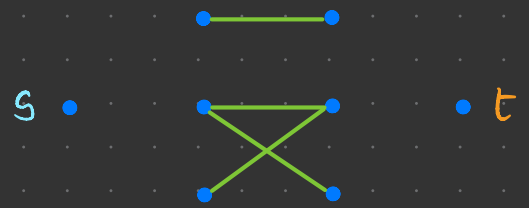
vector Matrix vector multiplication

$$\underset{u}{\begin{bmatrix}1 & 1 & 0\end{bmatrix}} \times \underset{M}{\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 1\\ 0 & 1 & 0\end{bmatrix}} \times \underset{v}{\begin{bmatrix}0\\ 1\\ 1\end{bmatrix}}$$

$uMv = 0$

$uMv > 0$

$s \to t$ distance



$dist(s,t) = 3$

$dist(s,t) \geq 5$

OuMv reductions

vector Matrix vector multiplication

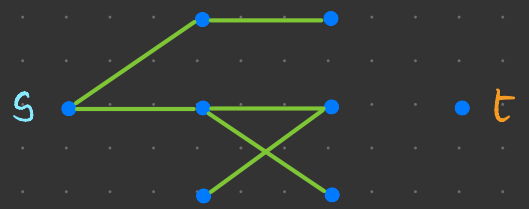
$$\begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

u M v

$uMv = 0$

$uMv > 0$

$s \to t$ distance



$dist(s,t) = 3$

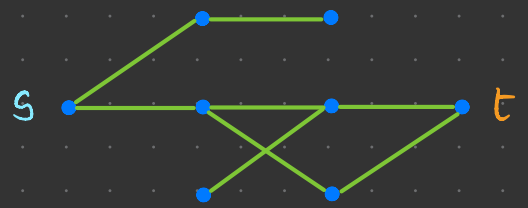
$dist(s,t) \geq 5$

OuMv reductions

vector Matrix vector multiplication

$$\underset{u}{\begin{bmatrix}1 & 1 & 0\end{bmatrix}} \times \underset{M}{\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 1\\ 0 & 1 & 0\end{bmatrix}} \times \underset{v}{\begin{bmatrix}0\\ 1\\ 1\end{bmatrix}}$$

$s \to t$ distance



$uMv = 0$ $\quad$ $dist(s,t) = 3$

$uMv > 0$ $\quad$ $dist(s,t) \geq 5$

OuMv reductions

Online uMv Problem

$(u_1, v_1) \rightarrow$ answer if $u_1 M v_1 = 0$

$(u_2, v_2) \rightarrow$ answer if $u_2 M v_2 = 0$

$\vdots$

$(u_n, v_n) \rightarrow$ answer if $u_n M v_n = 0$

Conjecture: Cannot answer all queries in $O(n^{2.99})$ time.

Dynamic $s \rightarrow t$ distance

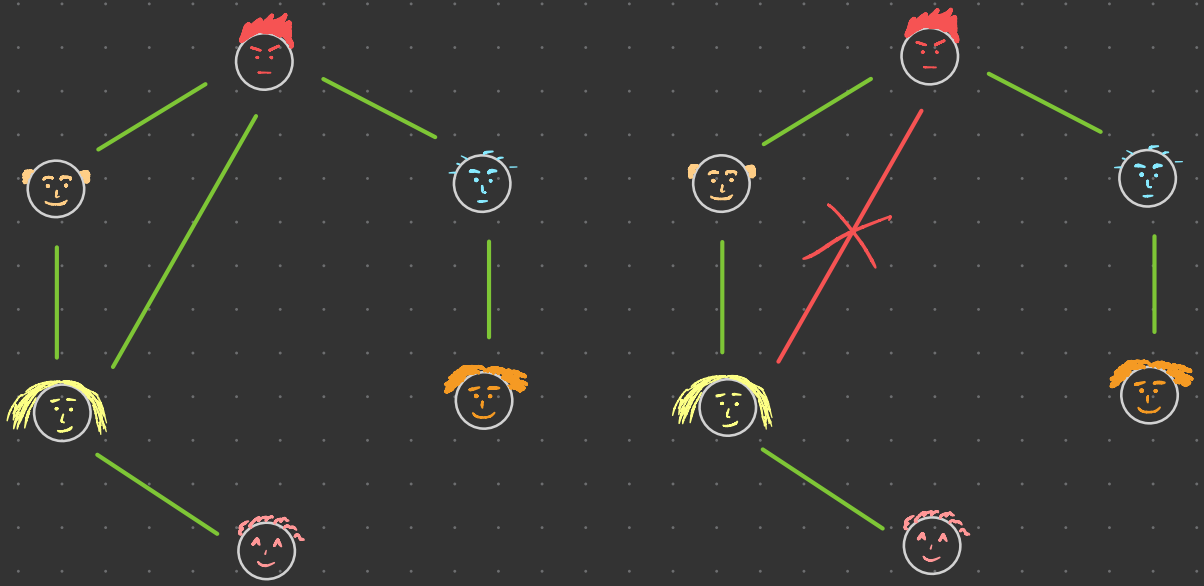
insert edges for u_1, M, v_1

↳ ask if $dist(s,t) = 3$

delete edges for u_1, v_1

insert edges for u_2, v_2

↳ ask if $dist(s,t) = 3$



Privatizing & Dynamizing Algorithm Design

Sricharan AR

advised by : Monika Henzinger + Gramoz Goranci