An introduction

WHY IS EVERYTHING SO HARD?

Thanks to

Virginia V. Williams MIT &
Erik Demaine MIT
for many slides

Todays Talk

REMINDER: MOTIVATION

REMINDER: HOW TO FGC

THE HARD PROBLEMS IN DATABASES

THE REDUCTIONS

Big Goal

- Give definitions of many hardness assumptions
- Give related hardness assumptions that can be used
- Show some simple/classic reductions

Reminder: Fine grained complexity motivation

Motivation

No one would consider an O(n^{100}) time algorithm efficient in practice.

If n is huge, then O(n^2) can also be inefficient.

We have many problems on which we have been stuck on given polynomial times for decades:

- All Pairs Shortest Paths $O(n^3)$ but no $n^{2.99}$
- Longest Common Subsequence $O(n^2)$ but no $n^{1.99}$
- Diameter in sparse graphs $O(n^2)$ but no $n^{1.99}$
- …

Why are we stuck?

Why are we stuck?

We are stuck on many problems from different subareas of CS!

Are we stuck because of *the same reason*?

How do we address this?

Thanks to Virginia & Dream AI

Fine-grained reductions warm up

Problem A on input of size n requires $\tilde{O}(a(n))$ time.

Ignores sub-polynomial factors:
$n^{o(1)}$ e.g. $\lg(n)$, $\lg^2(n)$, …

Solve A with a call to B on an input of size n.

Then B must require $\tilde{O}(a(n))$ time.

Conditional lower bounds.

Intuition: $a(n)$,$b(n)$ are the naive runtimes for A and B. A reducible to B implies that beating the naive runtime for B implies also beating the naive runtime for A.

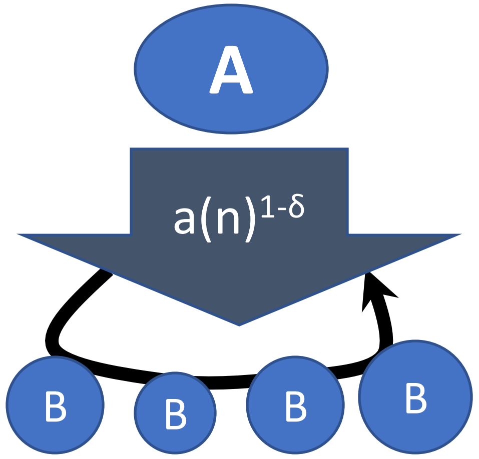
Fine-grained reductions

- A is (a,b)-reducible to B if

  for every $\varepsilon>0$ $\exists$ $\delta>0$, and an $O(a(n)^{1-\delta})$ time algorithm

  that transforms any A-instance of size n to B-instances of size $n_1,\ldots,n_k$ so that $\sum_i b(n_i)^{1-\varepsilon} < a(n)^{1-\delta}$.

- **If B is in $O(b(n)^{1-\varepsilon})$ time, then A is in $O(a(n)^{1-\delta})$ time.**
- Focus on exponents.
- We can build equivalences.



A theory of hardness for polynomial time

Thanks to Virginia

Three central hard problems

3SUM: Given a set S of n integers, are there a,b,c in S with **a+b+c = 0**?

Orthogonal vectors (OV): Given a set S of n vectors in $\{0,1\}^d$, for d = $O(\log^2 n)$ are there u,v in S with $\boldsymbol{u} \cdot \boldsymbol{v} = \mathbf{0}$?

All pairs shortest paths (APSP): given a weighted graph, find the **distance** between every two nodes.

Thanks to Virginia

3SUM: Given a set S of n numbers, are there $a,b,c \in S$ with $\mathbf{a+b+c = 0}$?

- Easy $O(\mathbf{n^2})$ time algorithm
- [BDP'05]: $\sim\mathbf{n^2}/\log^2 n$ time algorithm for integers
- [GP'14] : $\sim\mathbf{n^2}/\log n$ time for real numbers
- Here we'll talk about 3SUM over the integers
- Folklore: one can assume the integers are in $\{-n^3,\ldots,n^3\}$

3SUM Conjecture: 3SUM on n integers in $\{-n^3,\ldots,n^3\}$ requires $n^{2-o(1)}$ time.

Thanks to Virginia

Three central hard problems

3SUM: Given a set S of n integers, are there a,b,c in S with **a+b+c = 0**?

Orthogonal vectors (OV): Given a set S of n vectors in $\{0,1\}^d$, for d = $O(\log^2 n)$ are there u,v in S with $\boldsymbol{u} \cdot \boldsymbol{v} = \mathbf{0}$?

All pairs shortest paths (APSP): given a weighted graph, find the **distance** between every two nodes.

Thanks to Virginia

Orthogonal vectors (OV): Given a set S of n vectors in $\{0,1\}^d$, for d = $O(\log^2 n)$ are there u,v ∈ S with $\mathbf{u} \cdot \mathbf{v} = \mathbf{0}$?

- Easy O($\mathbf{n^2}$ d) time algorithm
- Best known [AWY'15]: $\mathbf{n^{2 - \Theta(1 / \log(d/\log n))}}$

OV Conjecture: OV on n vectors requires $n^{2-o(1)}$ time.

- [W'04]: SETH implies the OV Conjecture.

Thanks to Virginia

Three central hard problems

3SUM: Given a set S of n integers, are there a,b,c in S with **a+b+c = 0**?

Orthogonal vectors (OV): Given a set S of n vectors in $\{0,1\}^d$, for d = $O(\log^2 n)$ are there u,v in S with $\boldsymbol{u \cdot v = 0}$?

All pairs shortest paths (APSP): given a weighted graph, find the **distance** between every two nodes.

Thanks to Virginia

APSP: given a weighted graph, find the **distance** between every two nodes.

Classical problem
Long history

APSP Conjecture:
APSP on n nodes and $O(\log n)$ bit weights requires $n^{3-o(1)}$ time.

Author	Runtime	Year
Fredman	$n^3 \log \log^{1/3} n / \log^{1/3} n$	1976
Takaoka	$n^3 \log \log^{1/2} n / \log^{1/2} n$	1992
Dobosiewicz	$n^3 / \log^{1/2} n$	1992
Han	$n^3 \log \log^{5/7} n / \log^{5/7} n$	2004
Takaoka	$n^3 \log \log^2 n / \log n$	2004
Zwick	$n^3 \log \log^{1/2} n / \log n$	2004
Chan	$n^3 / \log n$	2005
Han	$n^3 \log \log^{5/4} n / \log^{5/4} n$	2006
Chan	$n^3 \log \log^3 n / \log^2 n$	2007
Han, Takaoka	$n^3 \log \log n / \log^2 n$	2012
Williams	$n^3 / \exp(\sqrt{\log n})$	2014

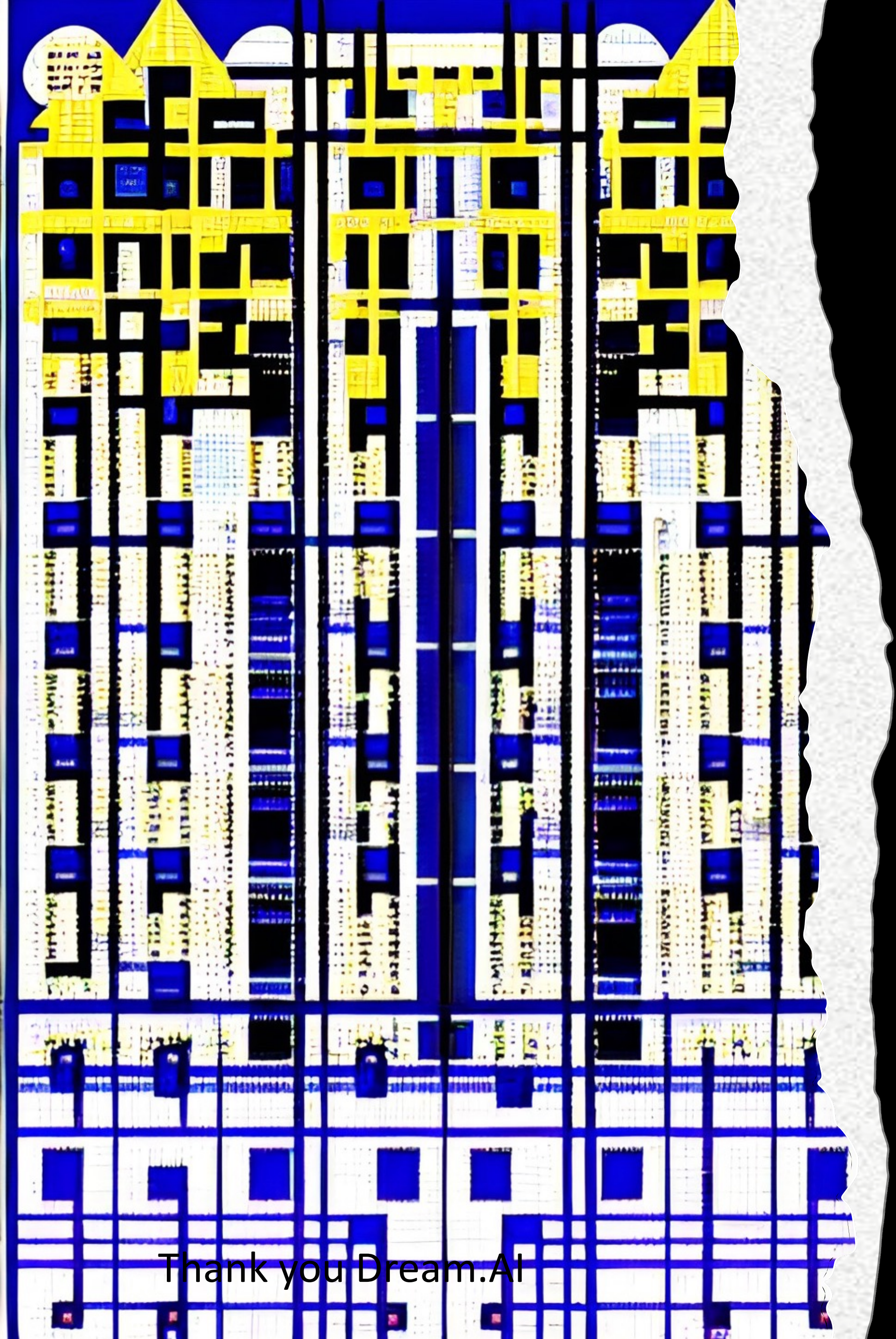
Thanks to Virginia

Problems Used in Databases [From Nofar's Talk]

sBMM	BMM	sTriangle
Triangle	sHyperclique	Hyperclique
VUTD (Vertex-Unbalanced Triangle Detection)	3SUM	Zero-Clique

Problems Used in Databases [From Nofar's Talk]

sBMM	BMM	sTriangle
Triangle	sHyperclique	Hyperclique
VUTD (Vertex-Unbalanced Triangle Detection)	3SUM	Zero-Clique



BMM Hypotheses

- sBMM: Boolean matrices cannot be multiplied in linear time in the number of the 1 entries
- BMM: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$
- Combinatorial BMM: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^{3-\epsilon})$ for $\epsilon > 0$ with any ***combinatorial*** algorithm [WW13]

- Related:
  - OMv [HKNS2015]

OMv and OuMv

Online Matrix vector (**OMv**) takes as input:

- a fixed $M \in \{0,1\}^{n \times n}$ and
- n updates of $\vec{v_i} \in \{0,1\}^n$

After every update must return $\vec{h}$ where $\vec{h}[j] = \min(1, M\vec{v_i}[j])$

[HKNS15]

Hypothesis: OMv requires $n^{3-o(1)}$ time

OuMv takes as input:

- a fixed $M \in \{0,1\}^{n \times n}$ and
- n updates of a *pair* $\vec{u_i}, \vec{v_i} \in \{0,1\}^n$

After every update must return:

$\min(1, \vec{u_i}^T M \vec{v_i})$

[HKNS15]

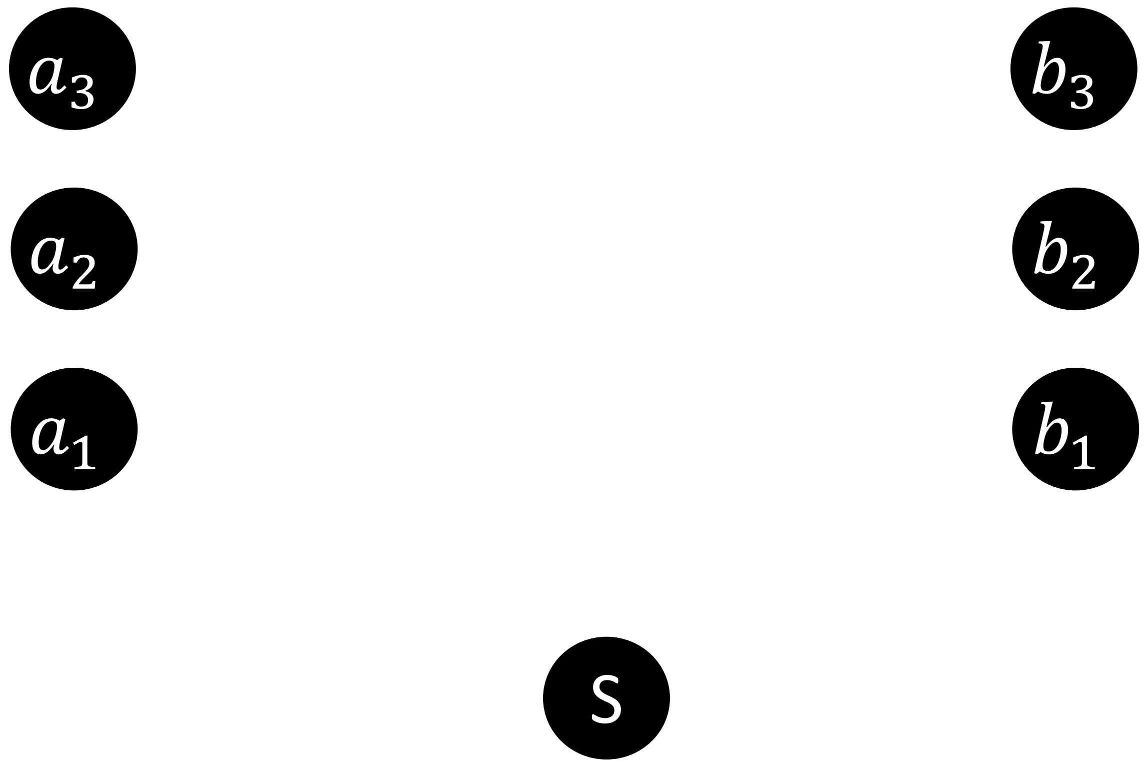
OMv hypothesis → OuMv requires $n^{3-o(1)}$ time

OMv and OuMv Give Dynamic Lower Bounds

- Dynamic graph updates of adding or deleting edges
- We can get queries to determine answers.
- We will show how to get a lower bound on detecting triangles through a single node
- Updates of adding and deleting edges corresponds to adding or deleting rows

Dynamic s-triangle detection
Triangles through a fixed node s

From OuMv we have $\overrightarrow{u_i}^T, M,$ and $\overrightarrow{v_i}$ and need to return min($1, \overrightarrow{u_i}^T M \overrightarrow{v_i}$)

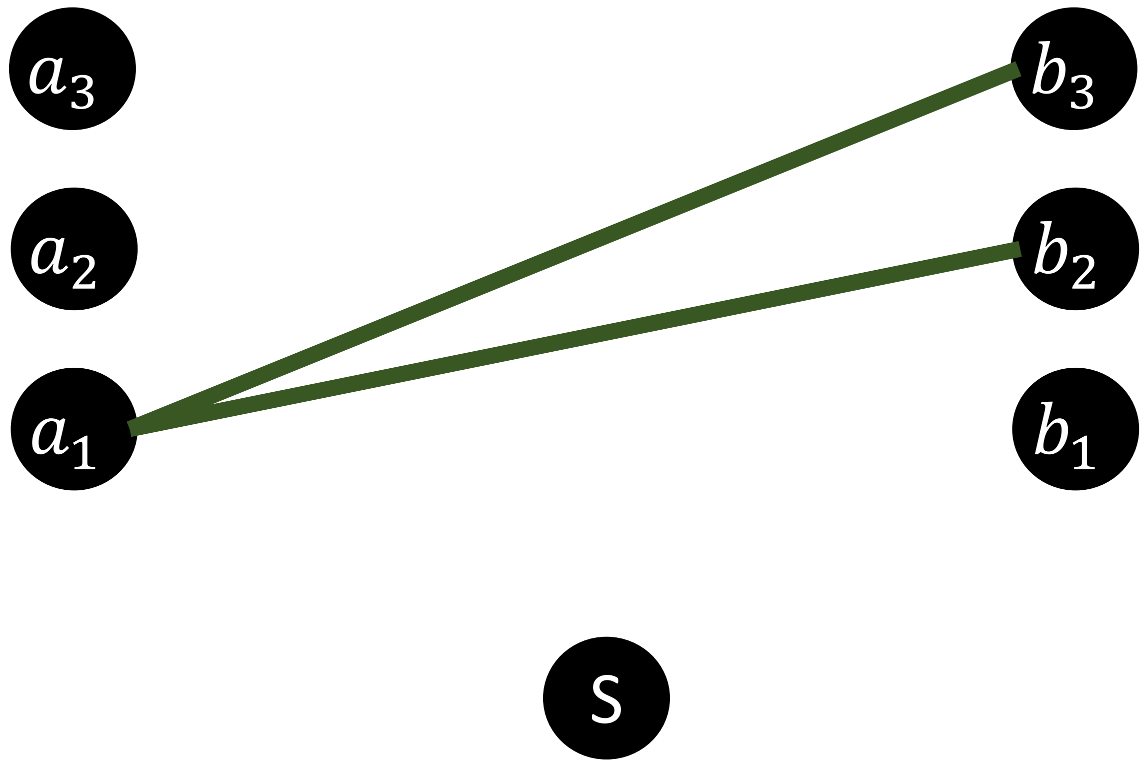


$$\vec{u}_i = < 1,0,1 >$$
$$\vec{v}_i = < 1,1,0 >$$

0	**1**	**1**
0	1	0
1	0	0

Dynamic s-triangle detection
Triangles through a fixed node s

From OuMv we have $\vec{u_i}^T, M,$ and $\vec{v_i}$ and need to return $\min(1, \vec{u_i}^T M \vec{v_i})$

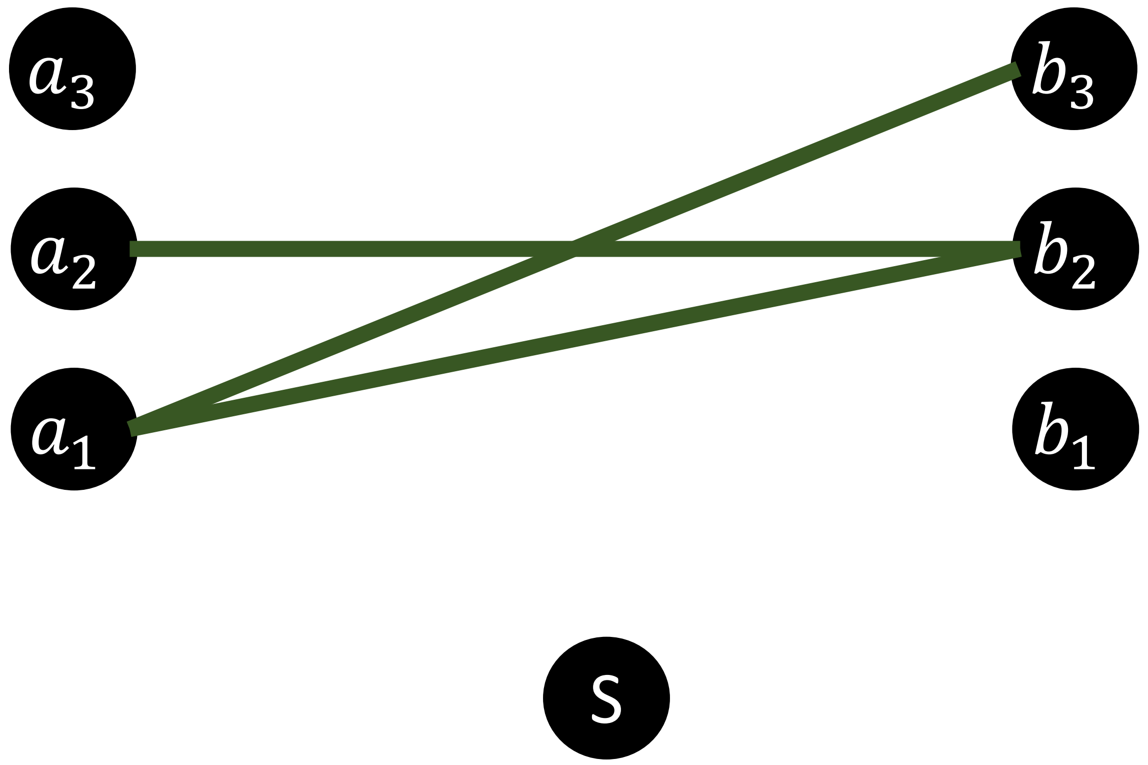


$\vec{u}_i = <1,0,1>$
$\vec{v}_i = <1,1,0>$

0	**1**	**1**
0	1	0
1	0	0

Dynamic s-triangle detection
Triangles through a fixed node s

From OuMv we have $\vec{u_i}^T, M,$ and $\vec{v_i}$ and need to return $\min(1, \vec{u_i}^T M \vec{v_i})$

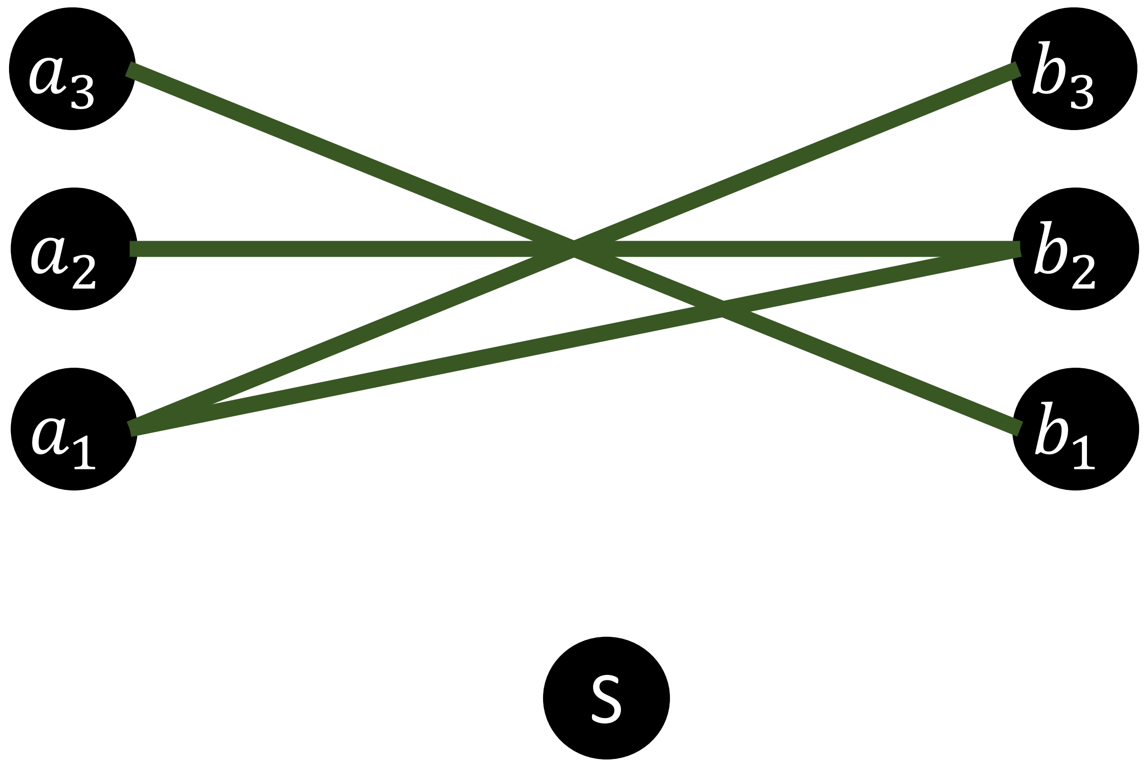


$$\vec{u}_i = < 1,0,1 >$$
$$\vec{v}_i = < 1,1,0 >$$

0	**1**	**1**
0	1	0
1	0	0

Dynamic s-triangle detecting
Triangles through a fixed node s

From OuMv we have $\vec{u_i}^T, M,$ and $\vec{v_i}$ and need to return min($1, \vec{u_i}^T M \vec{v_i}$)

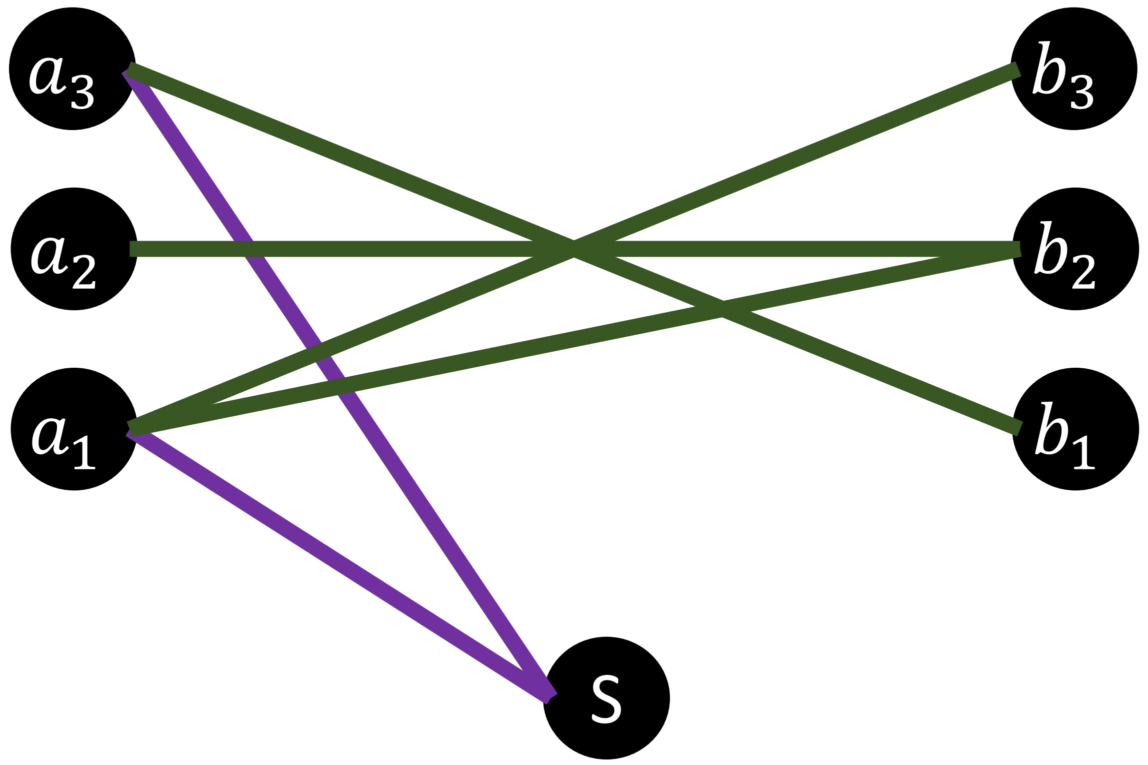


$$\vec{u}_i = < 1,0,1 >$$
$$\vec{v}_i = < 1,1,0 >$$

0	**1**	**1**
0	1	0
1	0	0

Dynamic s-triangle detecting
Triangles through a fixed node s

From OuMv we have $\vec{u_i}^T, M,$ and $\vec{v_i}$ and need to return min($1, \vec{u_i}^T M \vec{v_i}$)

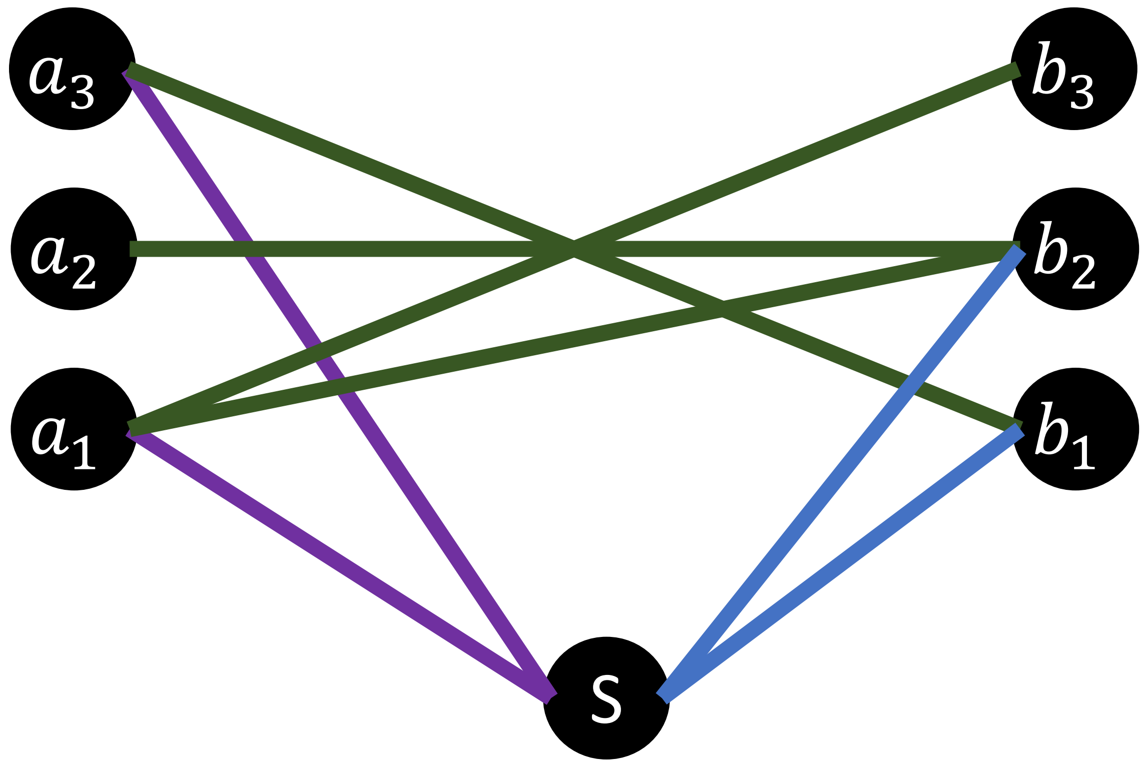


$$\vec{u}_i = <1,0,1>$$
$$\vec{v}_i = <1,1,0>$$

0	**1**	**1**
0	1	0
1	0	0

Dynamic s-triangle detecting
Triangles through a fixed node s

From OuMv we have $\overrightarrow{u_i}^T, M,$ and $\overrightarrow{v_i}$ and need to return $\min(1, \overrightarrow{u_i}^T M \overrightarrow{v_i})$



$$\vec{u}_i = < 1,0,1 >$$
$$\vec{v}_i = < 1,1,0 >$$

0	**1**	**1**
0	1	0
1	0	0

Dynamic s-triangle detecting
Triangles through a fixed node s

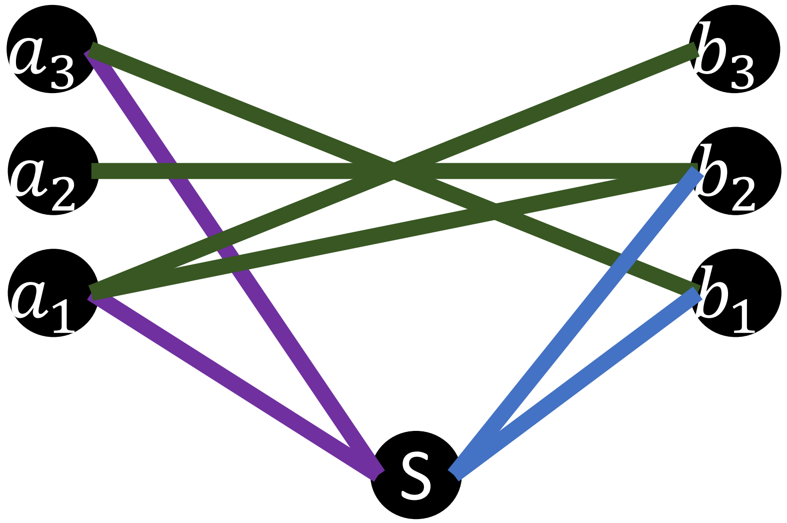
From OuMv we have $\overrightarrow{u_i}^T, M,$ and $\overrightarrow{v_i}$ and need to return min($1, \overrightarrow{u_i}^T M \overrightarrow{v_i}$)

Every pair $(\overrightarrow{u_i}^T, \overrightarrow{v_i})$ corresponds to $\mathrm{O}(n)$ edge updates (deleting or inserting an edge). Total $O(n^2)$.

There are $O(n)$ queries.

OuMv is conjectured to require $O(n^3)$ so either:
updates require $\widetilde{\Omega}(n)$ time
or
queries require $\widetilde{\Omega}(n^2)$ time

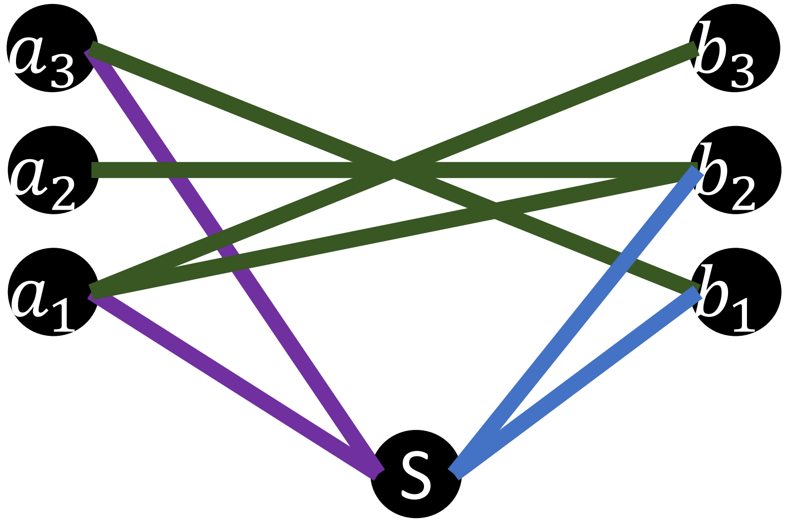
$$\vec{u}_i = < 1,0,1 >$$
$$\vec{v}_i = < 1,1,0 >$$



0	**1**	**1**
0	1	0
1	0	0

Dynamic s-triangle algorithm

- Track count of number of triangles
- Every edge added or deleted between (a_i, b_j)
  - Check if a triangle is formed/deleted and update count
  - $O(1)$
- Every edge added or deleted between (s, a_i)
  - Iterate over all nodes b_j and check if a triangle is formed/deleted update count
  - $O(n)$
- Queries are $O(1)$ (We have a saved count)



An aside

- Listing
- Counting
- Search (find one solution)
- Existence

One detail (not so small)

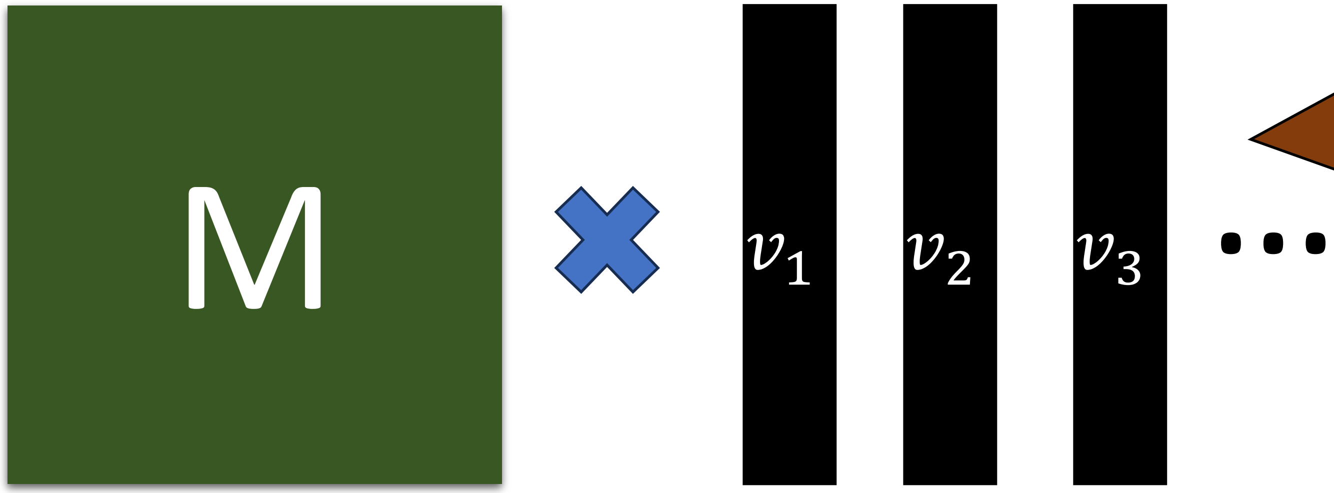
- With listing we give ourselves time to state each solution
- With counting/one instance/existence we don’t give ourselves time per solution
- This can make super-linear hardness (in input AND output) true for counting but not listing
- But, if you judge based on input ONLY then listing is harder than counting

OMv Hypothesis (reminder)

OMv [HKNS2015]

Given an $n \times n$ zero-one matrix M and n vectors of length n $\overrightarrow{v_1}, \overrightarrow{v_2}, \ldots, \overrightarrow{v_n}$ which are given online. After $\overrightarrow{v_i}$ is given we must return $M \cdot \overrightarrow{v_i}$.

Conjecture: No $n^{3-\epsilon}$ algorithm exists for OMv for any $\epsilon > 0$



An average case version of a related problem is hard!

Parity OMv and OuMv

Parity Online Matrix vector (**OMv**) takes as input:

- a fixed $\mathrm{M} \in \{0,1\}^{n \times n}$ and
- n updates of $\vec{v_i} \in \{0,1\}^n$

After every update must return $\vec{h}$ where $\vec{h}[j] = M\vec{v_i}[j] \; mod\ 2$

Parity OuMv takes as input:

- a fixed $\mathrm{M} \in \{0,1\}^{n \times n}$ and
- n updates of a *pair* $\vec{u_i}, \vec{v_i} \in \{0,1\}^n$

After every update must return:

$\vec{u_i}^T M \vec{v_i} \; mod\ 2$

OMv hypothesis → parity OMv requires $n^{3-o(1)}$

OMv hypothesis → parity OuMv requires $n^{3-o(1)}$

Average Case OMv and OuMv [HLS2022]

Average-Case Parity Online Matrix vector (**OMv**) takes as input:

- a **uniformly random** $\mathrm{M} \in \{0,1\}^{n \times n}$ and
- n updates of **uniformly random** $\vec{v_i} \in \{0,1\}^n$

After every update must return $\vec{h}$ where $\vec{h}[j] = M\vec{v_i}[j] \; mod \; 2$

OMv hypothesis → average-case parity OMv requires $n^{3-o(1)}$

Average-Case Parity OuMv takes as input:

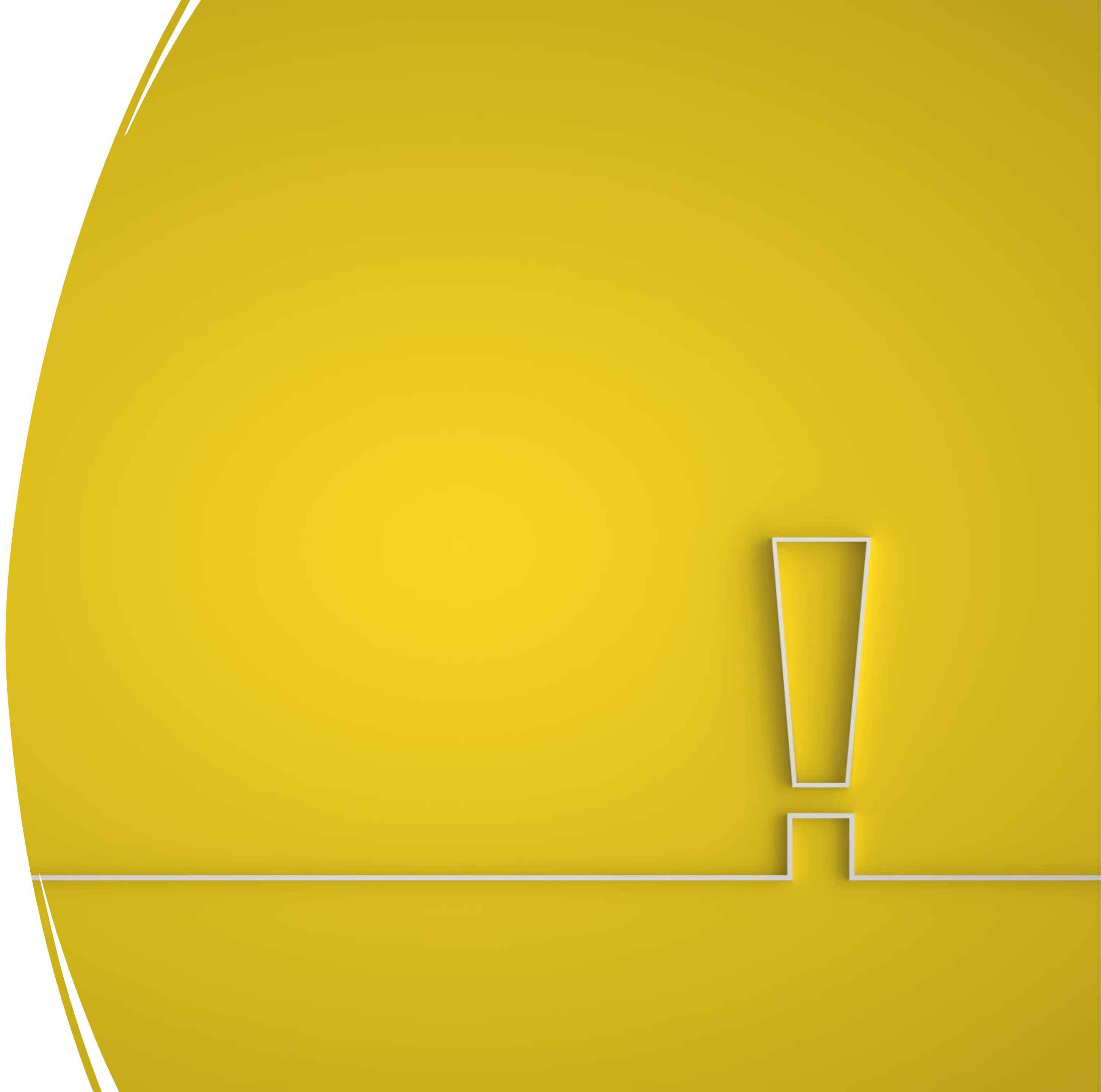
- a **uniformly random** $\mathrm{M} \in \{0,1\}^{n \times n}$ and
- n updates of a *pair* of **uniformly random** $\vec{u_i}, \vec{v_i} \in \{0,1\}^n$

After every update must return:
$\vec{u_i}^T M \vec{v_i} \; mod \; 2$

OMv hypothesis → average-case parity OuMv requires $n^{3-o(1)}$

Counting 5 Length s-t Paths is Hard on Average [HLS2022]

- Many subgraph counting problems are hard even with uniformly random updates
- If you are interested in hardness for random updates to databases we can use existing techniques to show hardness



Triangle Hypotheses

- sTriangle: The existence of a triangle in an undirected graph with m edges cannot be decided in time $O(m)$
- Triangle: The existence of a triangle in an undirected graph with n nodes cannot be decided in time $O(n^2)$
- Alternate triangle: The existence of a triangle in an undirected graph can not be decided in time $O(n^{\omega-\epsilon})$

Fun fact triangle counting is hard in Erdős–Rényi graphs

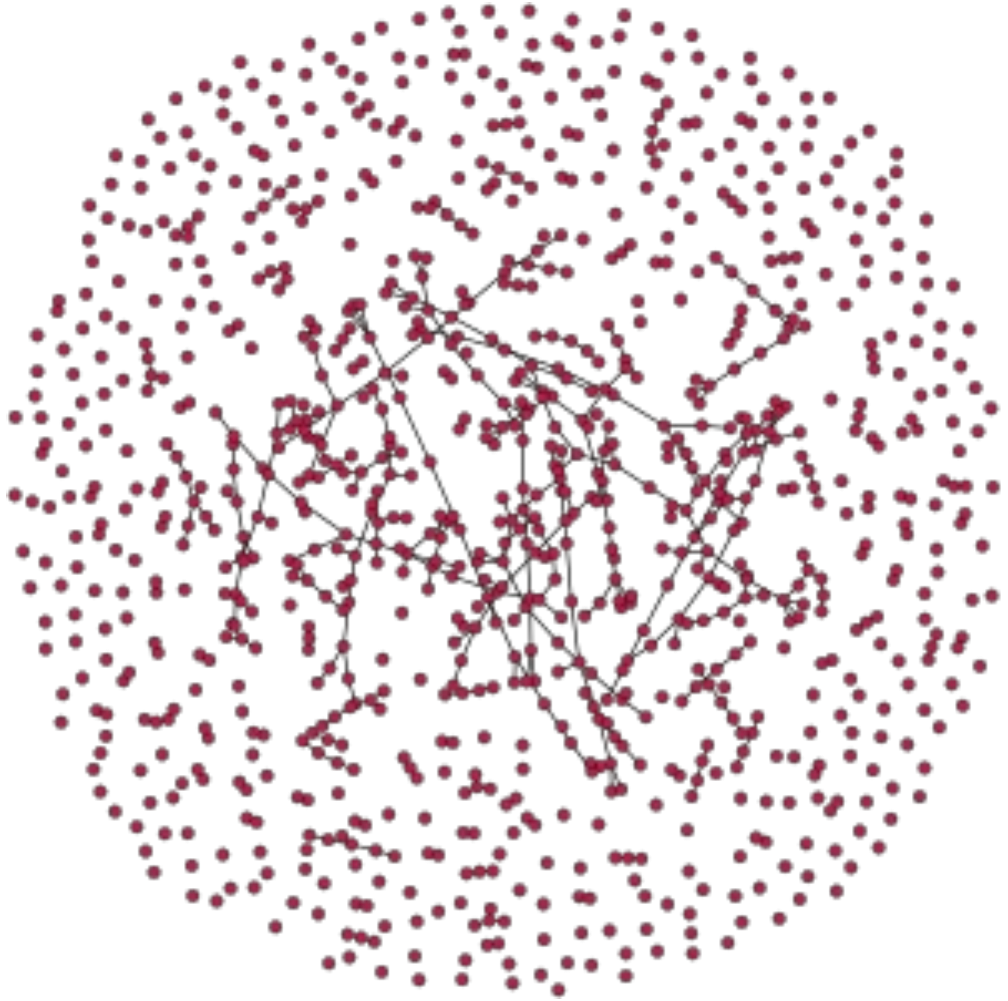
Thank you Dream.AI

Counting k-cliques is hard in Erdős–Rényi graphs [BBB19]

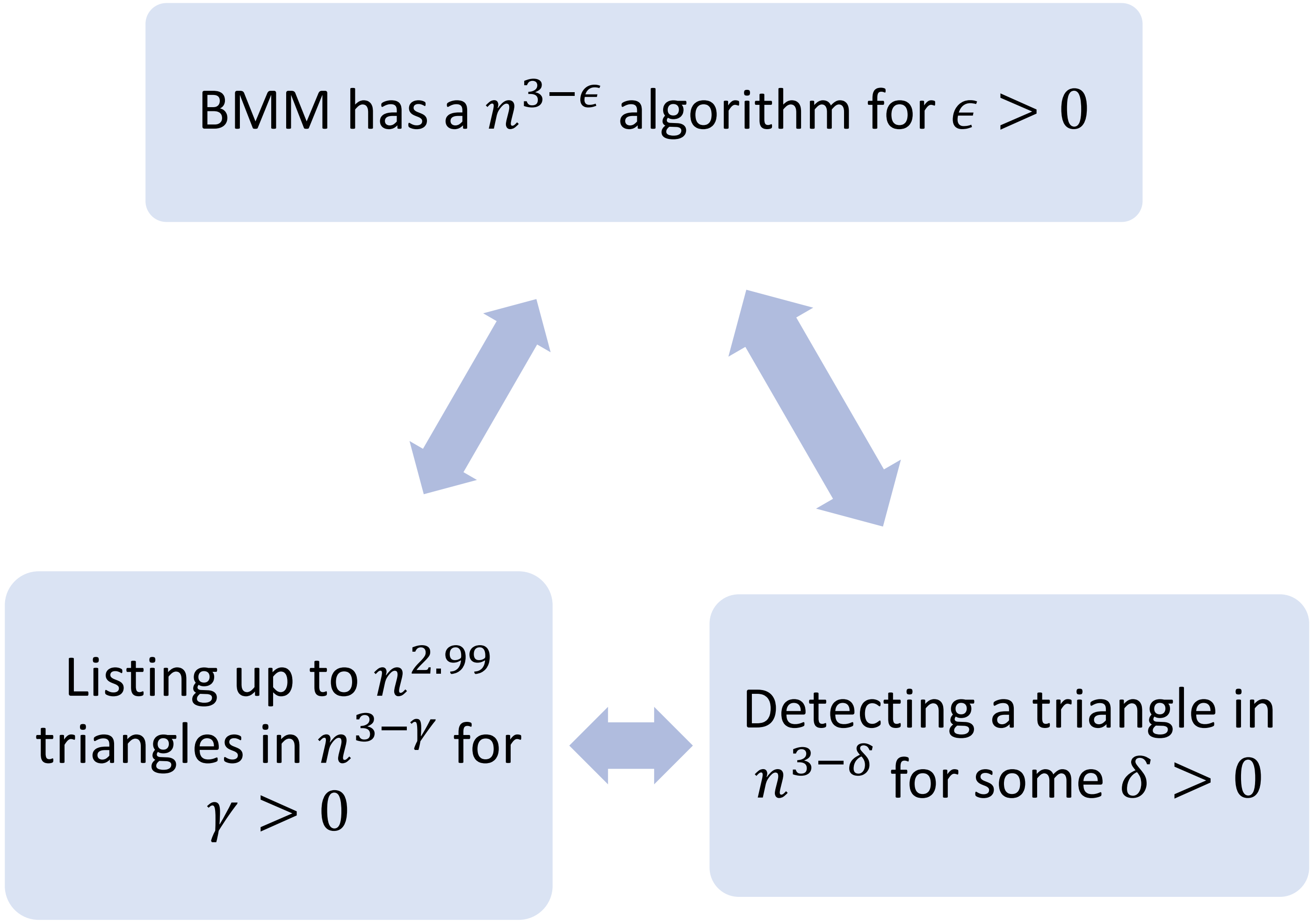
If counting k-cliques in Erdős–Rényi graphs takes n^a time with probability $1 - \lg(n)^{-2k^2}$ then

Algorithm exists for counting k-cliques in worst-case graphs in time $\tilde{O}(n^a)$

If you want to consider the count of the number of joined rows.



Combinatorial Triangle Detection vs Combinatorial BMM [WW13]

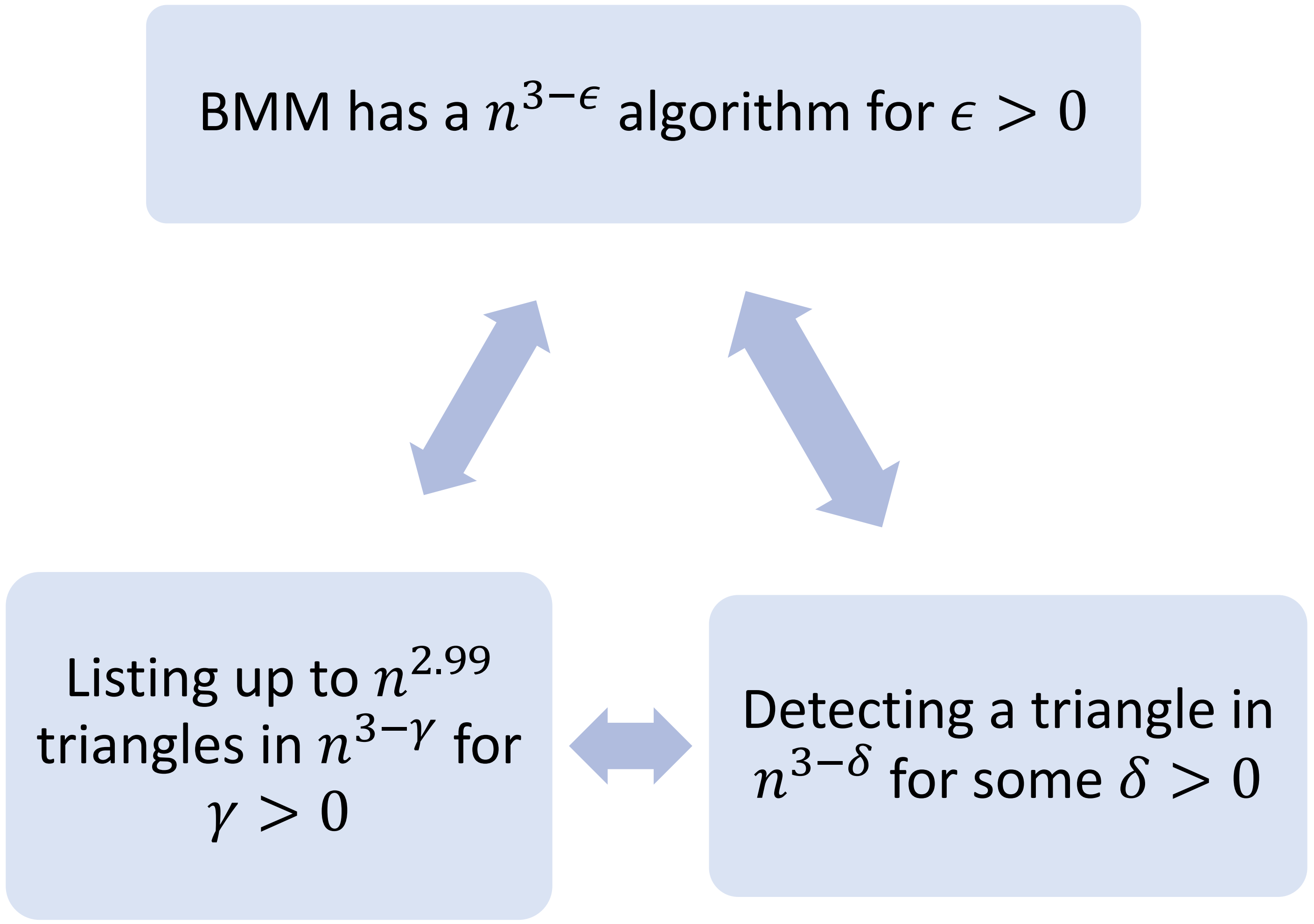


BMM has a $n^{3-\epsilon}$ algorithm for $\epsilon > 0$

Listing up to $n^{2.99}$ triangles in $n^{3-\gamma}$ for $\gamma > 0$

Detecting a triangle in $n^{3-\delta}$ for some $\delta > 0$

Combinatorial Triangle Detection vs Combinatorial BMM [WW13]



BMM has a $n^{3-\epsilon}$ algorithm for $\epsilon > 0$

Listing up to $n^{2.99}$ triangles in $n^{3-\gamma}$ for $\gamma > 0$

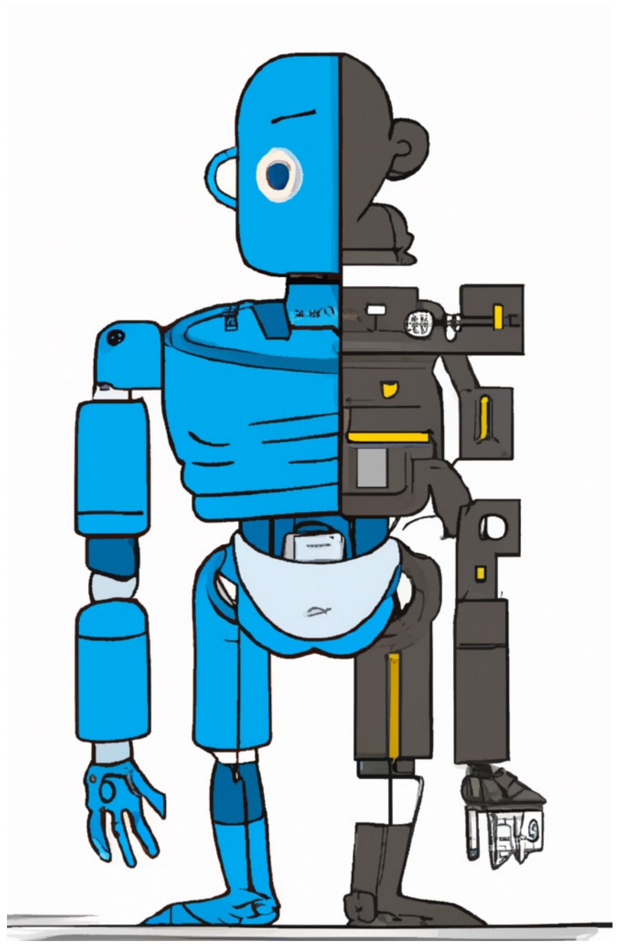
Detecting a triangle in $n^{3-\delta}$ for some $\delta > 0$

3SUM: Given a set S of n numbers, are there $a,b,c \in S$ with $\mathbf{a+b+c = 0}$?

- Easy $O(\mathbf{n^2})$ time algorithm
- [BDP'05]: ~$\mathbf{n^2}/\log^2 n$ time algorithm for integers
- [GP'14] : ~$\mathbf{n^2}/\log n$ time for real numbers
- Here we'll talk about 3SUM over the integers
- Folklore: one can assume the integers are in $\{-n^3,\ldots,n^3\}$

3SUM Conjecture: 3SUM on n integers in $\{-n^3,\ldots,n^3\}$ requires $n^{2-o(1)}$ time. [GO1995]

Thanks to Virginia



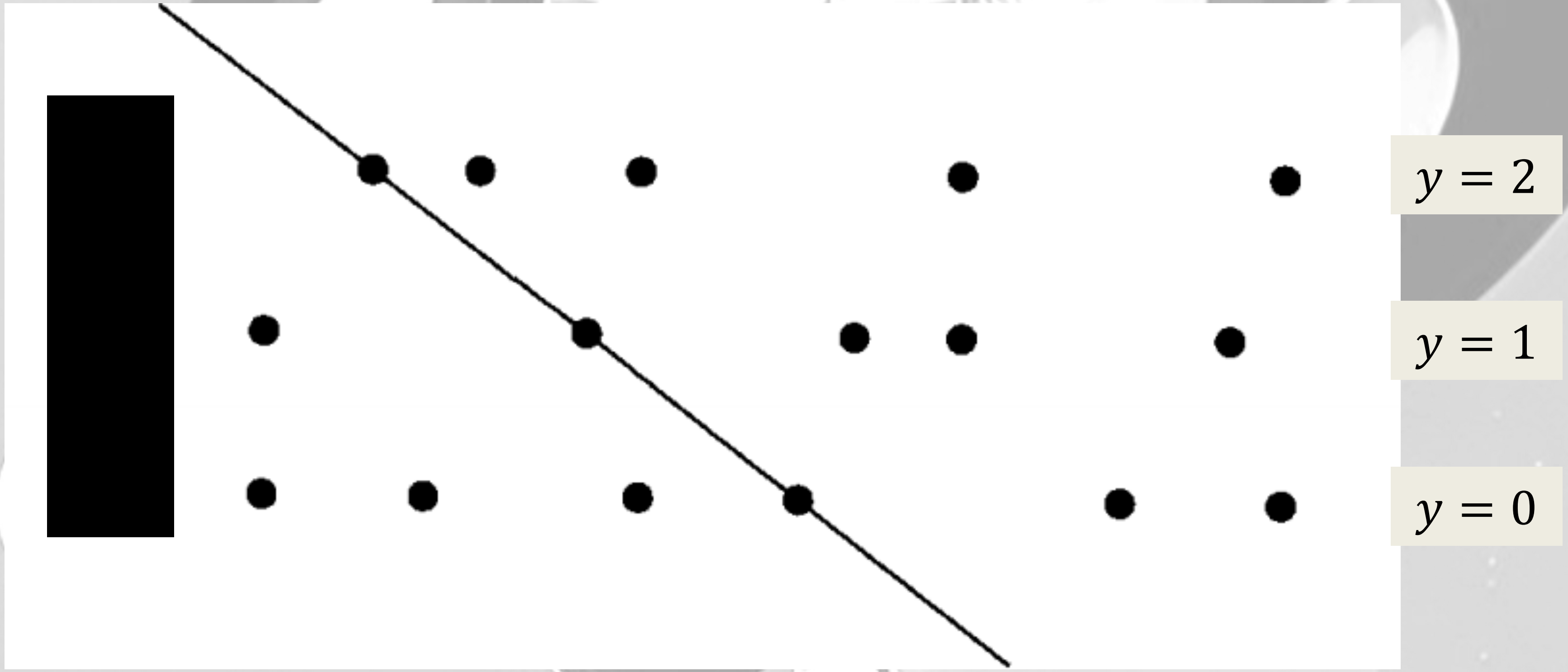
Reductions

We want to show that you can transform a 3-SUM instance into one or many instances of a given problem P.

We want these instances to be small.

We can then argue that you can solve 3-SUM faster than n^2 time if there is a fast algorithm for P.

3SUM′ → GeomBase
[Gajentaan & Overmars 1995]

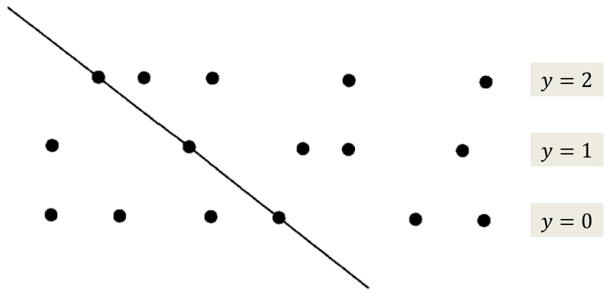


Thanks To Erik Demaine

How to Get to GeomBase?

GeomBase: We are given points (a, b) where $b \in [0,1,2]$. We are then asked if there are any three points that fall on a line with slope ***not equal to zero***. We want to show this is n^2 hard.

3SUM′ → GeomBase
[Gajentaan & Overmars 1995]



3-SUM: Takes as input one list of integers, S, in $[-n^3, n^3]$. We then want to know if there are three numbers that sum to zero.

Pause for creating a reduction!

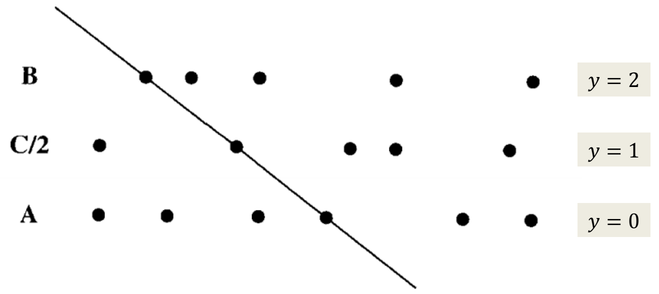
Colorful 3-SUM, or 3SUM': Takes as input three lists of integers, A,B,C , in $[-n^3, n^3]$. We then want to know if there are three numbers one from each list that sum to zero.

Everybody ready….

Pause for creating a reduction!

Getting to GeomBase?

3SUM′ → GeomBase
[Gajentaan & Overmars 1995]



Colorful 3-SUM, or 3SUM' problem takes as input three lists of integers, A,B,C , in $[-n^3, n^3]$. We then want to know if there are three numbers one from each list that sum to zero.

How do we connect 3-SUM to GeomBase? As suggested by the image, we are going to put numbers from list A on the y=0 line, B on the line y=2, and –C/2 on line y=1

How to reduce to GeomBase

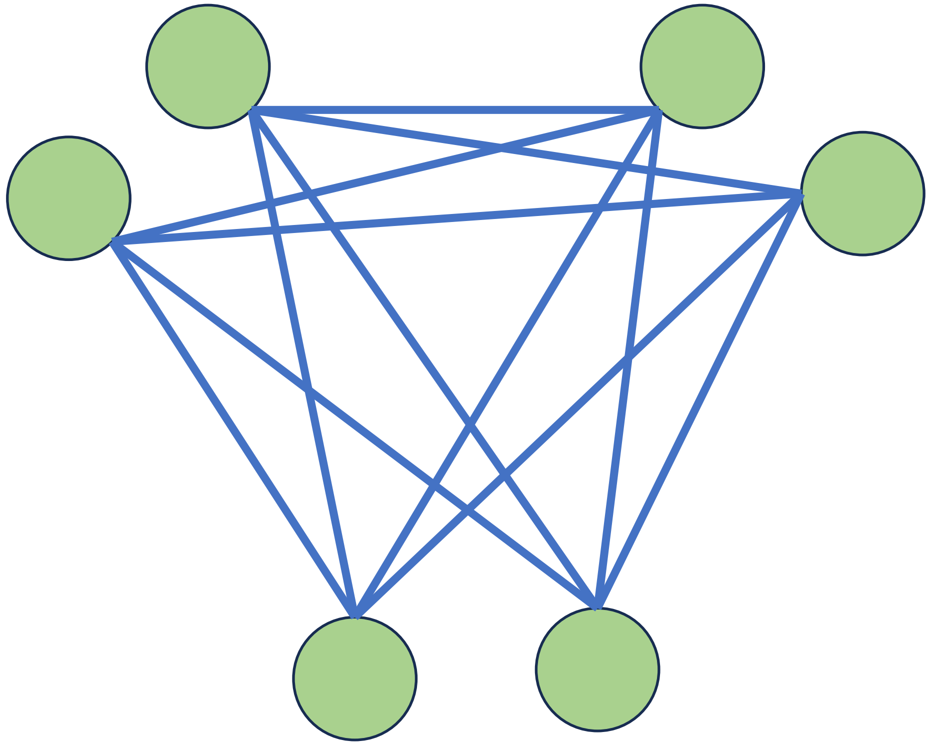
$$a \in A \rightarrow (a, 0)$$
$$b \in B \rightarrow (b, 2)$$
$$c \in C \rightarrow (-c/2, 1)$$

If three points are on a line then $\frac{a+b}{2} = -\frac{c}{2}$

which is equivalent to $a + b + c = 0$

Zero k-Clique

Zero-k-Clique:
$\forall k \geq 3 \ \forall \epsilon > 0$ the existence of a k-clique of weight 0 in a weighted graph with n nodes cannot be decided in time $O(n^{k-\epsilon})$

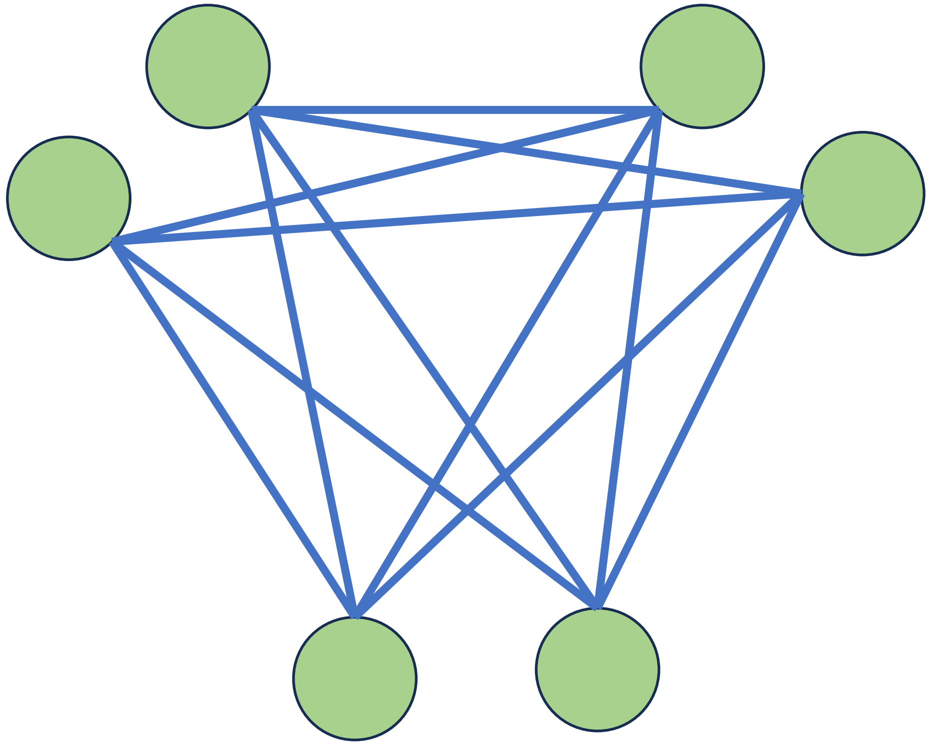


Consider a k-partite graph with weights on the edges in the range $[-n^k, n^k]$.

Zero k-Clique

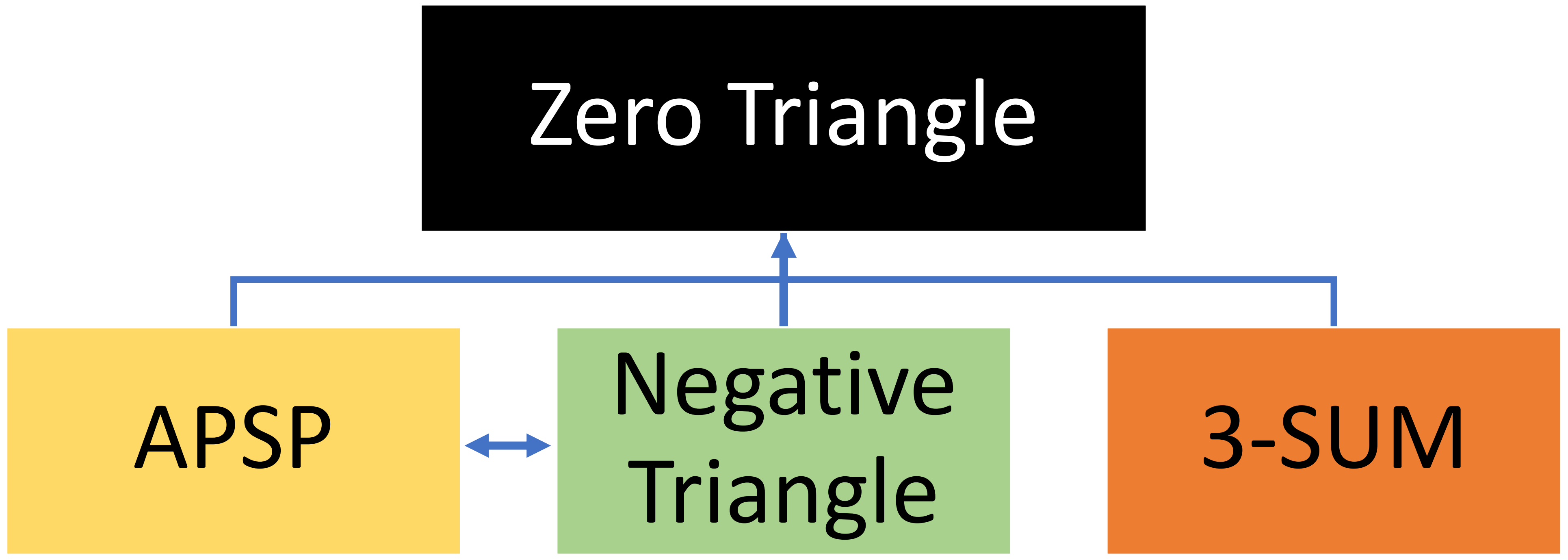
Bonus: Negative Triangle asks if there is a k-clique where the sum of edges is negative. Hypothesized to be $\tilde{\Omega}(n^3)$

Zero-k-Clique:
$\forall k \geq 3 \ \forall \epsilon > \ 0$ the existence of a k-clique of weight 0 in a weighted graph with n nodes cannot be decided in time $O(n^{k-\epsilon})$



Consider a k-partite graph with weights on the edges in the range $[-n^k, n^k]$.

Zero Triangle is SUPER Hard



Negative Triangle

Zero Triangle

3-SUM

The problems

Negative Triangle: You are given a graph G with integer edge weights represented with $O(\lg(n))$ bits. Return true if there are three nodes that form a clique where the sum of the edge weights of that clique is negative.

3SUM: Given a set S of n integers, are there a,b,c in S with **a+b+c = 0**?

Zero Triangle: You are given a graph G with integer edge weights represented with $O(\lg(n))$ bits. Return true if there are three nodes that form a clique where the sum of the edge weights of that clique is zero.

Reducing Negative Triangle to Zero Triangle [WW13]

Negative Triangle: You are given a graph G with integer edge weights represented with $O(\lg(n))$ bits. Return true if there are three nodes that form a clique where the sum of the edge weights of that clique is negative.

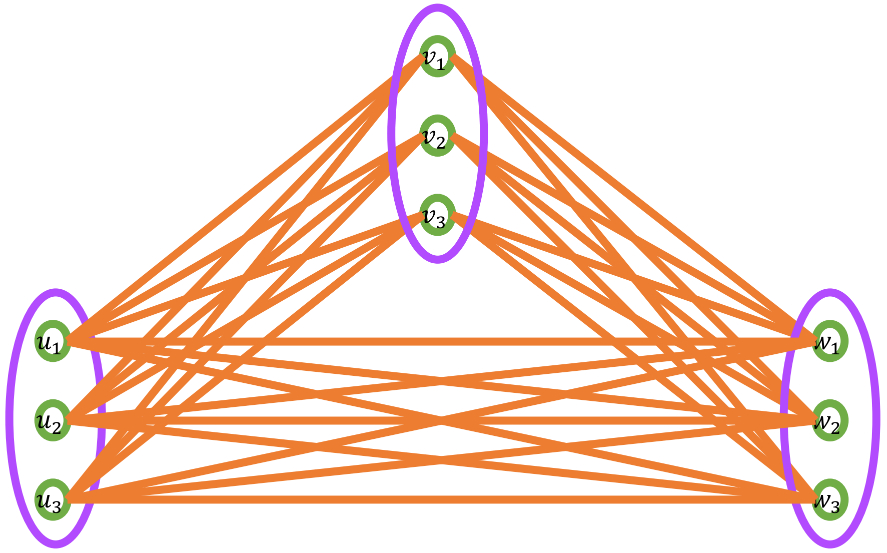
Split each edge set into positive and negative edges. We then create instances for all possible combinations (except all positive) for 7 in total.

$(+,+,+)$ ✗

$(+,+,-)$ ✓

$(+,-,-)$ ✓

$(-,-,-)$ ✓ ✓



Reducing Negative Triangle to Zero Triangle

Negative Triangle: You are given a graph G with integer edge weights represented with $O(\lg(n))$ bits. Return true if there are three nodes that form a clique where the sum of the edge weights of that clique is negative.

If $w(u,v) + w(v,t) + w(t,u) < 0$ then there exists an i such that

$(+,-,-)$

$$\left\lfloor \frac{w(u,v)}{2^i} \right\rfloor + \left\lfloor \frac{w(v,t)}{2^i} \right\rfloor + \left\lfloor \frac{w(t,u)}{2^i} \right\rfloor \in \{-1,-2\}$$

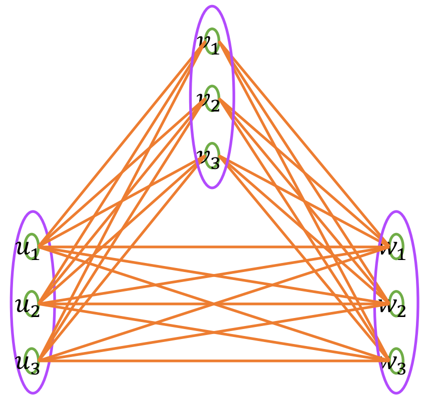
If $w(u,v) + w(v,t) + w(t,u) < 0$ then there exists an i such that

$(+,+,-)$

$$\left\lfloor \frac{w(u,v)}{2^i} \right\rfloor + \left\lfloor \frac{w(v,t)}{2^i} \right\rfloor + \left\lfloor \frac{w(t,u)}{2^i} \right\rfloor = \{-2,-3\}$$

or

$$w(u,v) + w(v,t) + w(t,u) = -1$$

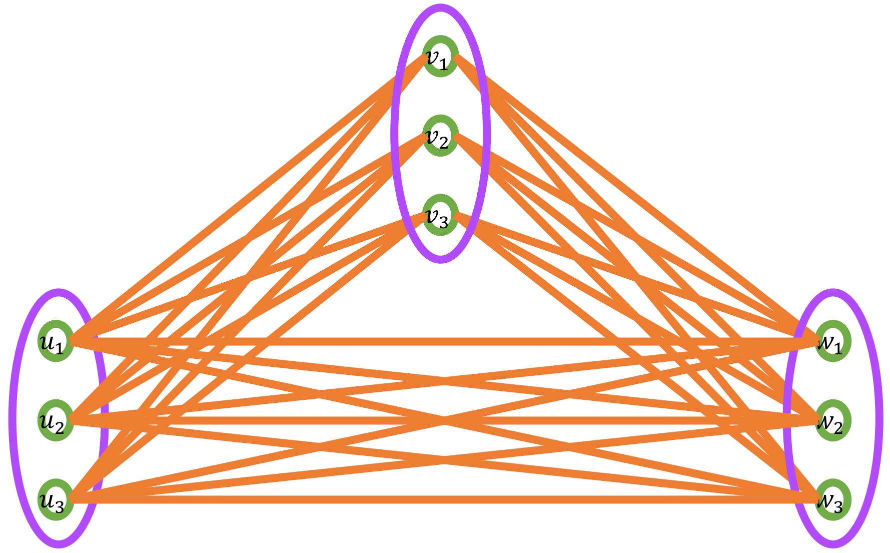


Reducing Negative Triangle to Zero Triangle

Negative Triangle: You are given a graph G with integer edge weights represented with $O(\lg(n))$ bits. Return true if there are three nodes that form a clique where the sum of the edge weights of that clique is negative.

Let $G_i^{+\Delta}$ be a graph where

- $\mathrm{w}_i(u, v)$ in G_i is $\left\lfloor \frac{w(u,v)}{2^i} \right\rfloor$
- $\mathrm{w}_i(v, t)$ in G_i is $\left\lfloor \frac{w(u,v)}{2^i} \right\rfloor$
- $\mathrm{w}_i(t, u)$ in G_i is $\left\lfloor \frac{w(u,v)}{2^i} \right\rfloor + \Delta$

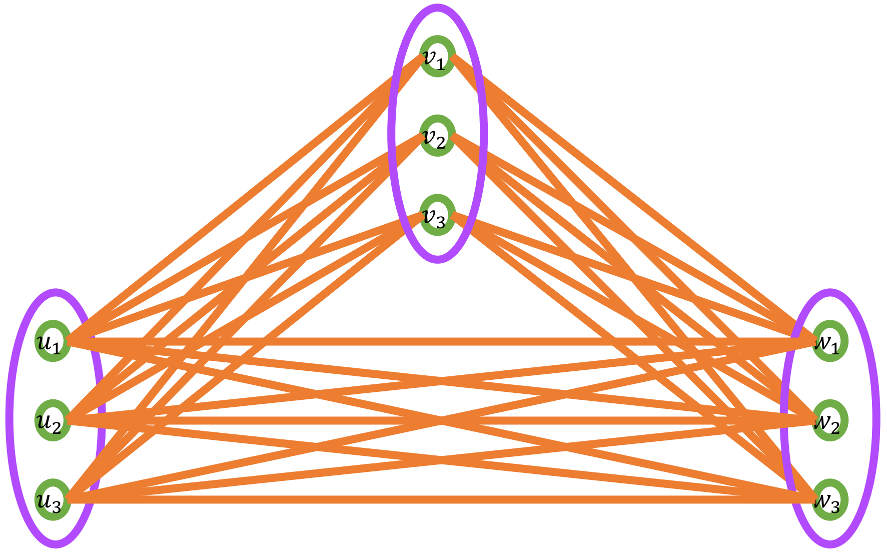


Reducing Negative Triangle to Zero Triangle

Negative Triangle: You are given a graph G with integer edge weights represented with $O(\lg(n))$ bits. Return true if there are three nodes that form a clique where the sum of the edge weights of that clique is negative.

Let $G_i^{+\Delta}$ be a graph where

- $\mathrm{w}_i(u, v)$ in G_i is $\left\lfloor \frac{w(u,v)}{2^i} \right\rfloor$
- $\mathrm{w}_i(v, t)$ in G_i is $\left\lfloor \frac{w(u,v)}{2^i} \right\rfloor$
- $\mathrm{w}_i(t, u)$ in G_i is $\left\lfloor \frac{w(u,v)}{2^i} \right\rfloor + \Delta$

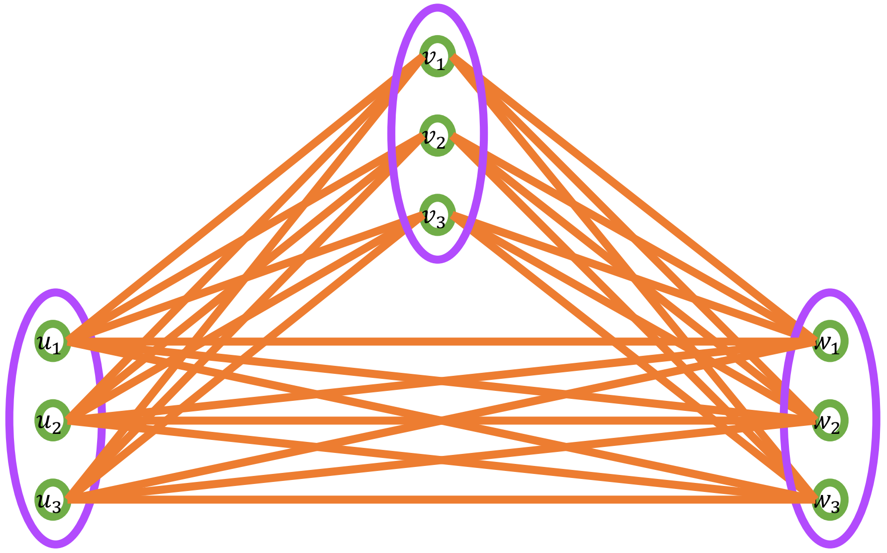


Reducing Negative Triangle to Zero Triangle

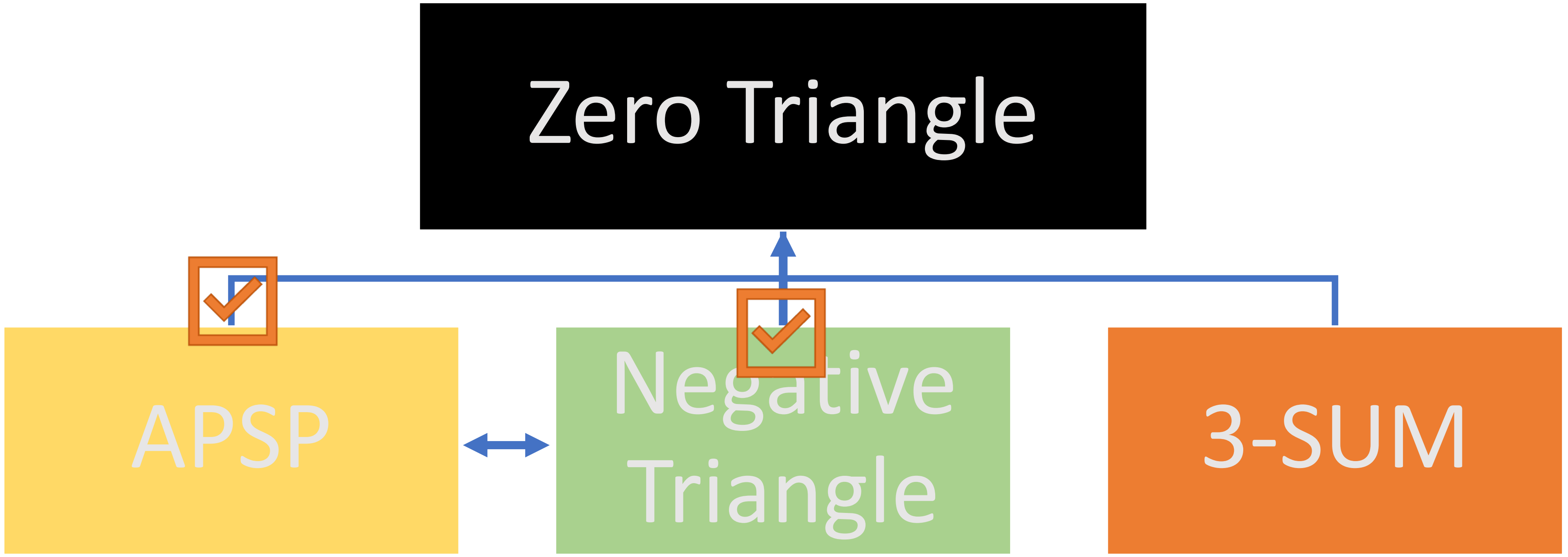
Negative Triangle: You are given a graph G with integer edge weights represented with $O(\lg(n))$ bits. Return true if there are three nodes that form a clique where the sum of the edge weights of that clique is negative.

$(+,-,-)$ For all $O(\lg(n))$ values of i we produce G_i^{+1} and G_i^{+2} and run zero triangle on all these instances.

$(+,+,-)$ For all $O(\lg(n))$ values of i we produce G_i^{+2} & G_i^{+3} and run zero triangle on all these instances.



Zero Triangle solves APSP and Negative Triangle



If zero triangle has an algorithm which runs in $n^{3-\epsilon}$ time for some constant $\epsilon > 0$
then
negative triangle has an algorithm which runs in $\lg(n) \cdot n^{3-\epsilon} + n^2$ time.

Nearly Linear Hash Functions (like magic)

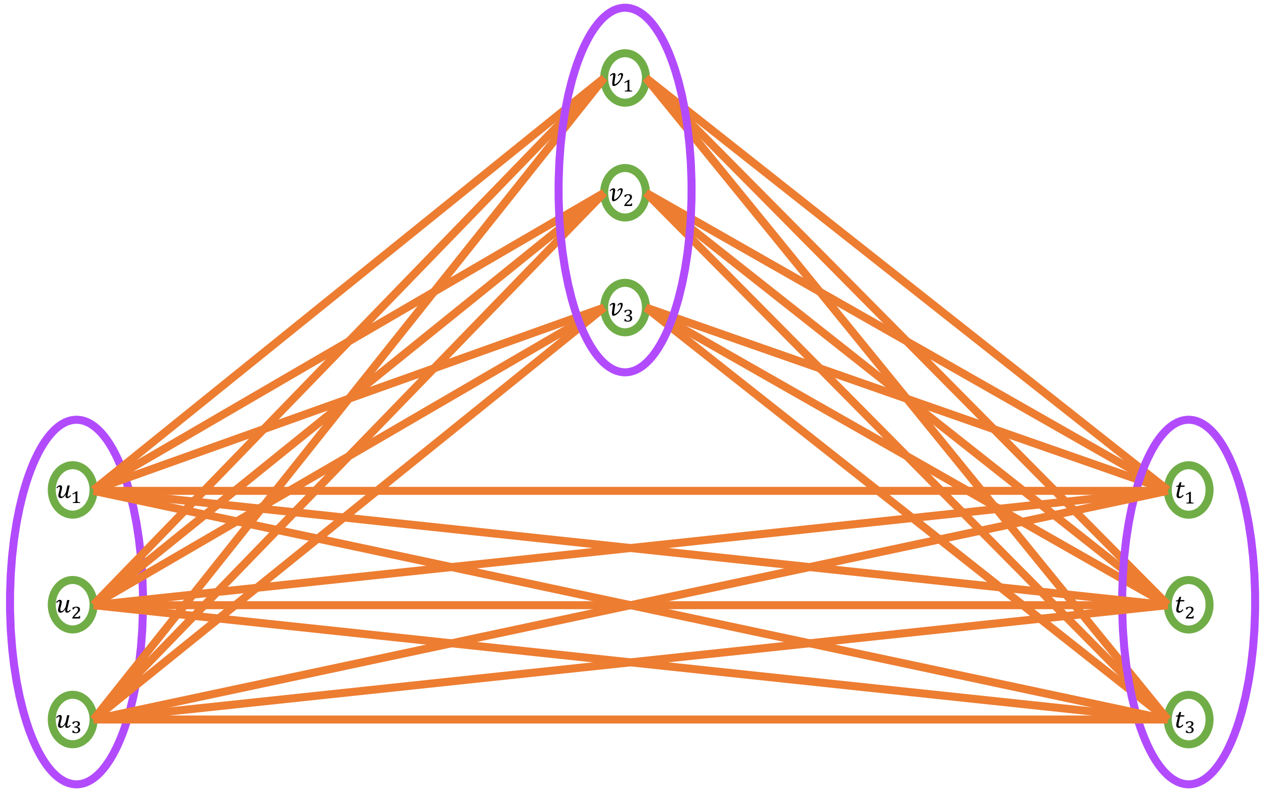
$$a + b + c = 0$$
$$h(a) + h(b) + h(c) = \{0,1,2\}$$

We will use two of these drawn independently h_1, h_2 with range $[-\sqrt{n}, \sqrt{n}]$

Reducing 3-SUM to Zero Triangle

- $w(u_i, v_j)$ is a value where $h_1(i - j)$
- $w(v_j, t_k)$ is a value where $h_1(j - k)$
- $w(t_k, u_i)$ is a value where $h_1(k - i)$

These are sets of size $\sqrt{n}$ how do we decide which elements we use?



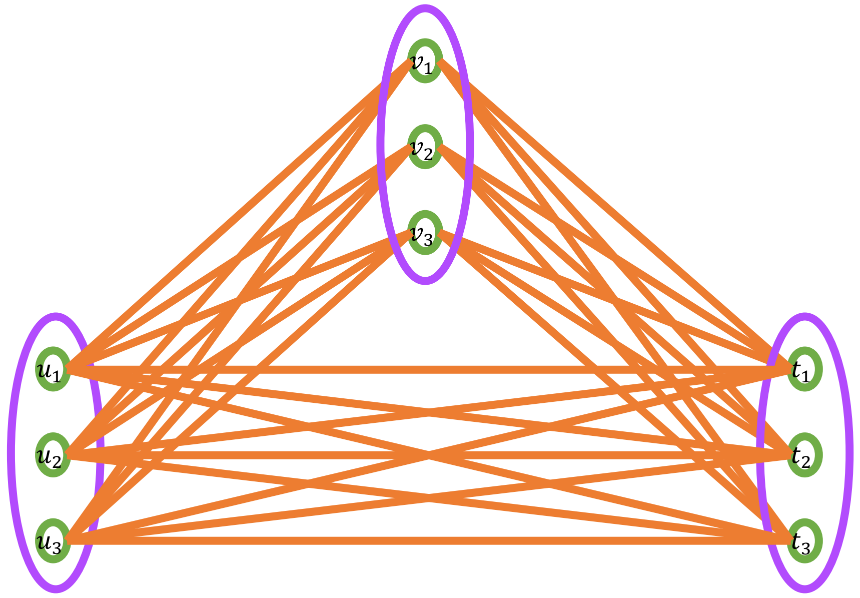
Reducing 3-SUM to Zero Triangle

- $w(u_i, v_j)$ is a value where $h_1(x) = \ i - j$
- $w(v_j, t_k)$ is a value where $h_1(x) = j - k$
- $w(t_k, u_i)$ is a value where $h_1(x) = k - i$ + {0,1,2}

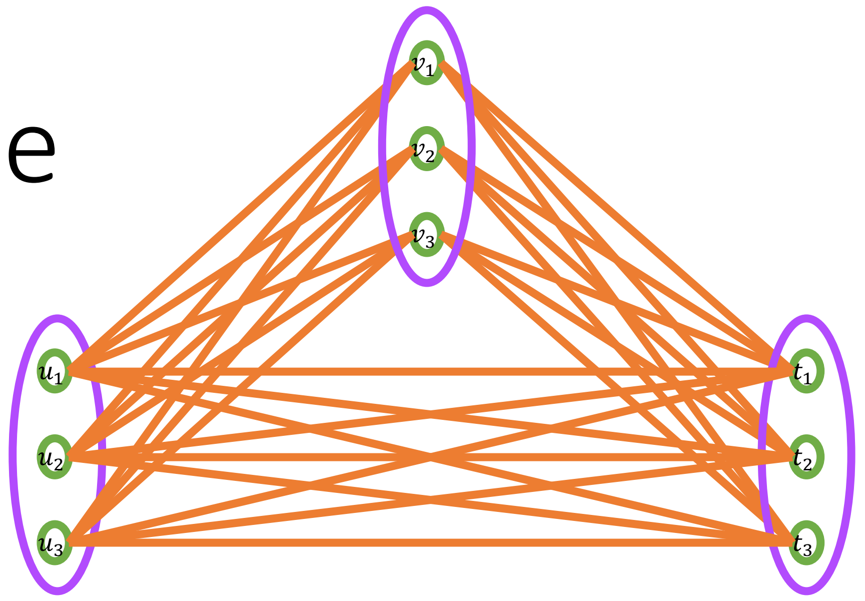
We produce a group of instances for $\ell \in [-\sqrt{n}, \sqrt{n}]$
Now we will use h_2

- $w(u_i, v_j)$ is a value where $h_2(x) = i$
- $w(v_j, t_k)$ is a value where $h_2(x) = \ell$
- $w(t_k, u_i)$ is a value where $h_2(x) = -i - \ell$ + {0,1,2}

So we have sets implied by the value of two hashes



Reducing 3-SUM to Zero Triangle



- $w(u_i, v_j)$ is a value where $h_1(x) = i - j$
- $w(v_j, t_k)$ is a value where $h_1(x) = j - k$
- $w(t_k, u_i)$ is a value where $h_1(x) = k - i + \{0,1,2\}$

We produce a group of instances for $\ell \in [-\sqrt{n}, \sqrt{n}]$
Now we will use h_2

- $w(u_i, v_j)$ is a value where $h_2(x) = i$
- $w(v_j, t_k)$ is a value where $h_2(x) = \ell$
- $w(t_k, u_i)$ is a value where $h_2(x) = -i - \ell + \{0,1$

So we have sets implied by the value of two hashes

If we have

$$h_1(x) = c_1$$
$$h_2(x) = c_2$$

so the expected size is n/R^2 where $R = \sqrt{n}$. So $O(1)$.

To prove it this way fully you need to resolve the issue of unexpectedly large buckets.

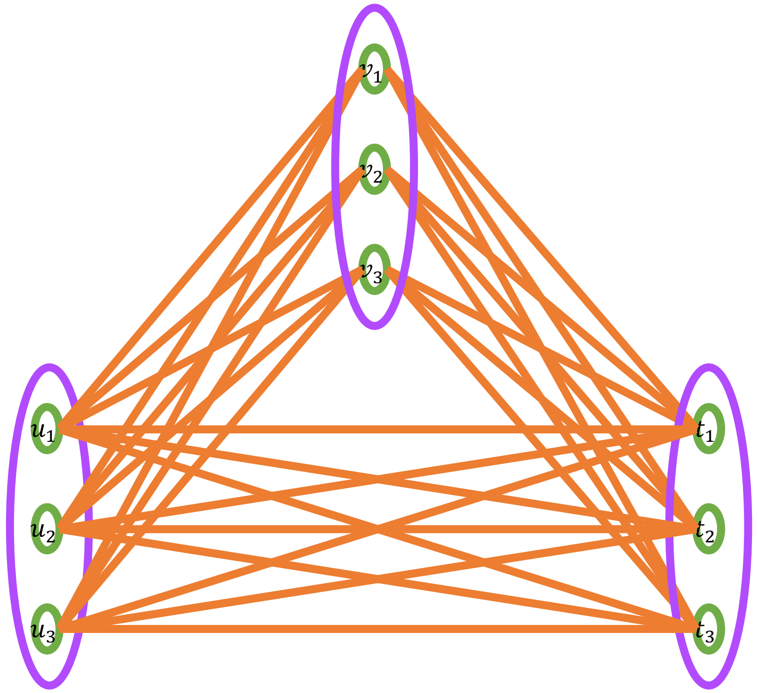
Quick Check In

We will produce a constant number of instances for $\ell \in [-\sqrt{n}, \sqrt{n}]$

Each instance is of size $\sqrt{n}$ nodes and there are $O(\sqrt{n})$ of these.

So if there is an algorithm for zero triangle which runs in $n^{3-\epsilon}$ time

$$O(\sqrt{n} \cdot (\sqrt{n})^{3-\epsilon}) = O\left(n^{2-\epsilon/2}\right)$$



What to do about our buckets?

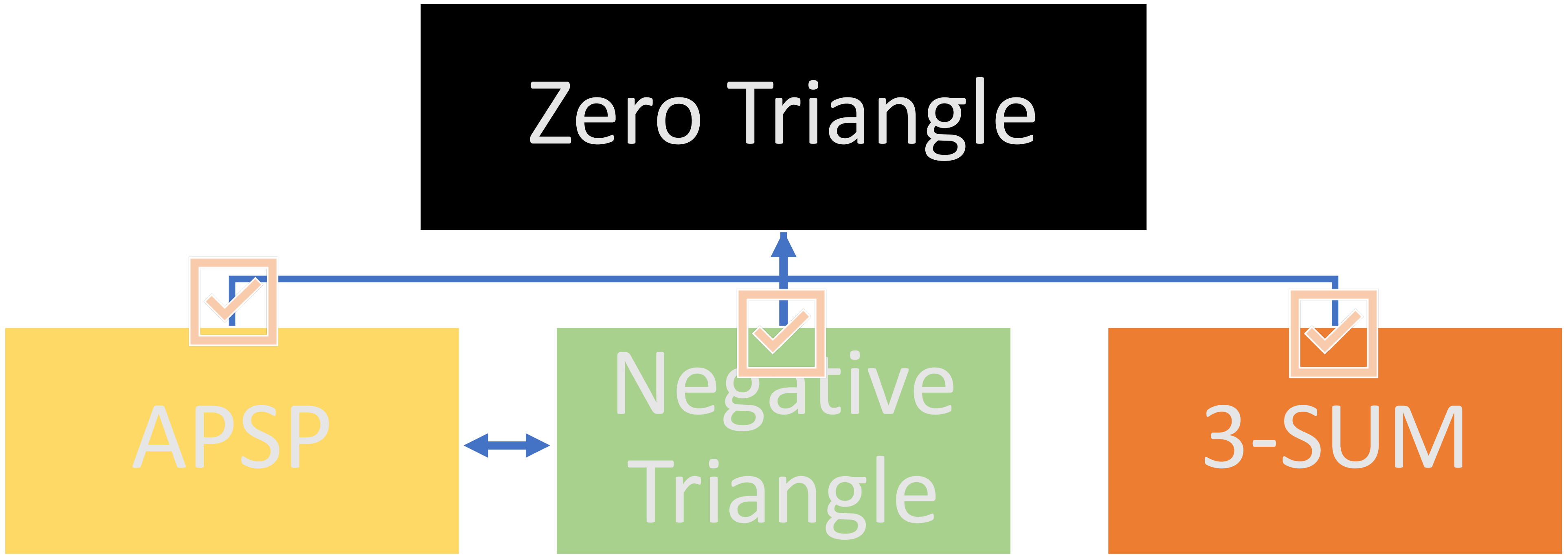
For any bucket that is larger than $\lg^2(n)$ take $O(n)$ all values in that bucket .

Now, for any bucket of size smaller than $\lg^2(n)$ we can simply exhaustively search over all $\lg^6 n$ possible values.

With high probability this results in an algorithm that solves the problem and takes:

$$O\left(\lg(n)n + \lg(n) \cdot n^{2-\epsilon/2}\right)$$

Zero Triangle solves APSP and 3-SUM

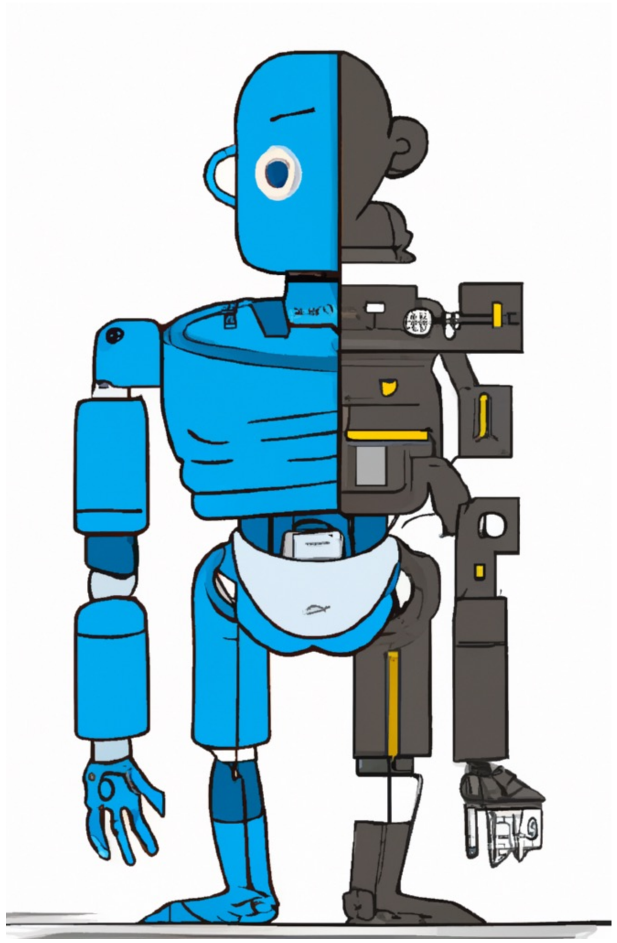


3SUM: Given a set S of n numbers, are there $a,b,c \in S$ with $\mathbf{a+b+c = 0}$?

- Easy O($\mathbf{n^2}$) time algorithm
- [BDP'05]: ~$\mathbf{n^2}/\log^2 n$ time algorithm for integers
- [GP'14] : ~$\mathbf{n^2}/\log n$ time for real numbers
- Here we'll talk about 3SUM over the integers
- Folklore: one can assume the integers are in $\{-n^3,\ldots,n^3\}$

3SUM Conjecture: 3SUM on n integers in $\{-n^3,\ldots,n^3\}$ requires $n^{2-o(1)}$ time. [GO1995]

Thanks to Virginia



Reductions

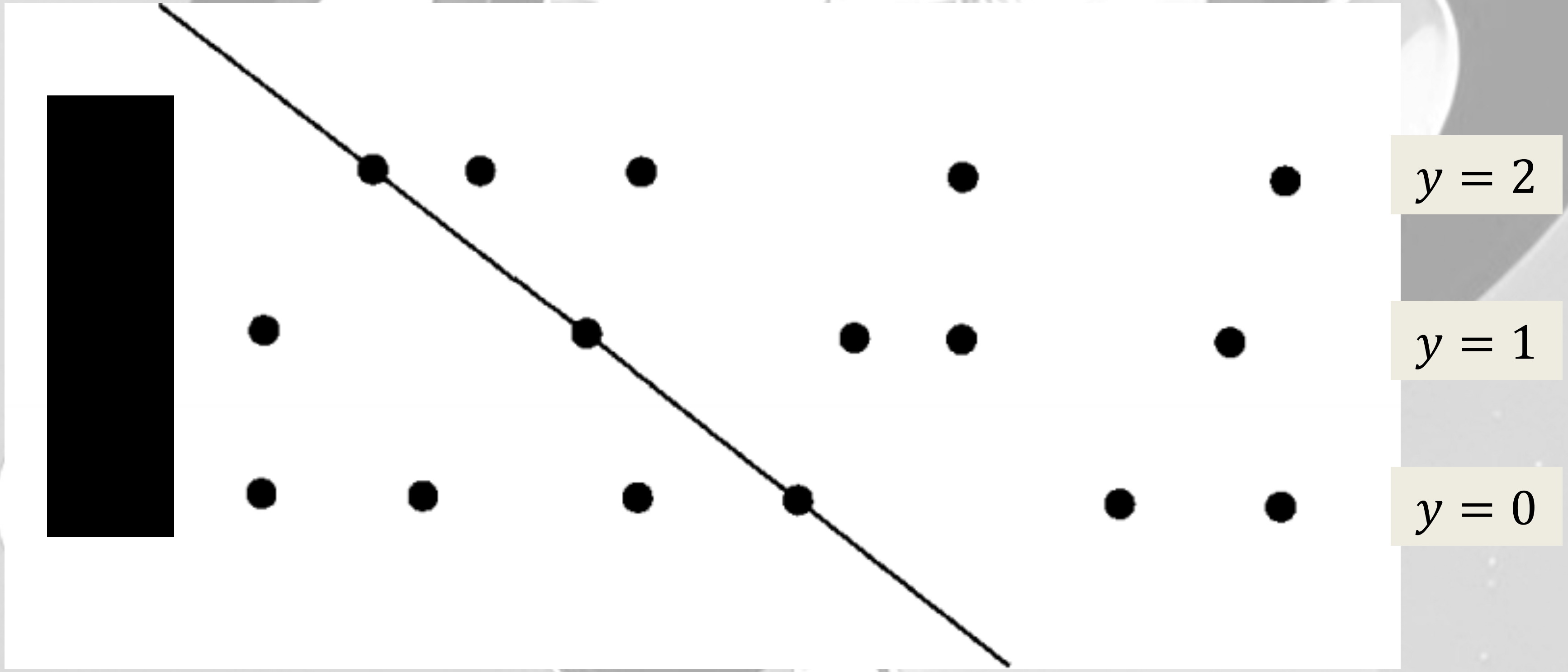
We want to show that you can transform a 3-SUM instance into one or many instances of a given problem P.

We want these instances to be small.

We can then argue that you can solve 3-SUM faster than n^2 time if there is a fast algorithm for P.

$3SUM' \to$ GeomBase

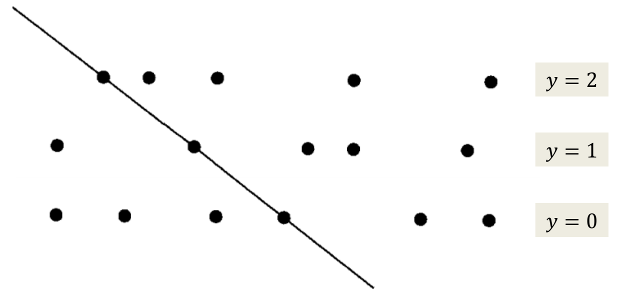
[Gajentaan & Overmars 1995]



Thanks To Erik Demaine

How to Get to GeomBase?

3SUM′ → GeomBase
[Gajentaan & Overmars 1995]



Pause for creating a reduction!

GeomBase: We are given points (a, b) where $b \in [0,1,2]$. We are then asked if there are any three points that fall on a line with slope ***not equal to zero***. We want to show this is n^2 hard.

3-SUM: Takes as input one list of integers, S, in $[-n^3, n^3]$. We then want to know if there are three numbers that sum to zero.

Colorful 3-SUM, or 3SUM’: Takes as input three lists of integers, A,B,C , in $[-n^3, n^3]$. We then want to know if there are three numbers one from each list that sum to zero.

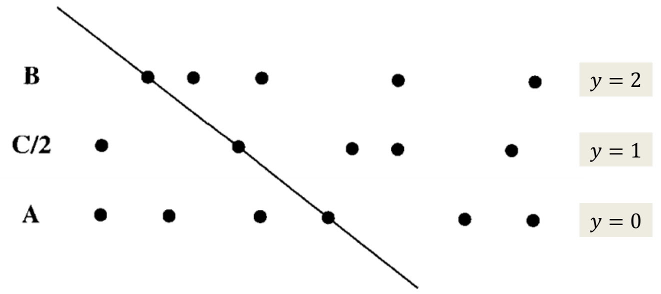
Everybody ready….

Pause for creating a reduction!

Getting to GeomBase?

$3SUM' \rightarrow$ GeomBase

[Gajentaan & Overmars 1995]



Colorful 3-SUM, or 3SUM' problem takes as input three lists of integers, A,B,C , in $[-n^3, n^3]$. We then want to know if there are three numbers one from each list that sum to zero.

How do we connect 3-SUM to GeomBase? As suggested by the image, we are going to put numbers from list A on the y=0 line, B on the line y=2, and –C/2 on line y=1

How to reduce to GeomBase

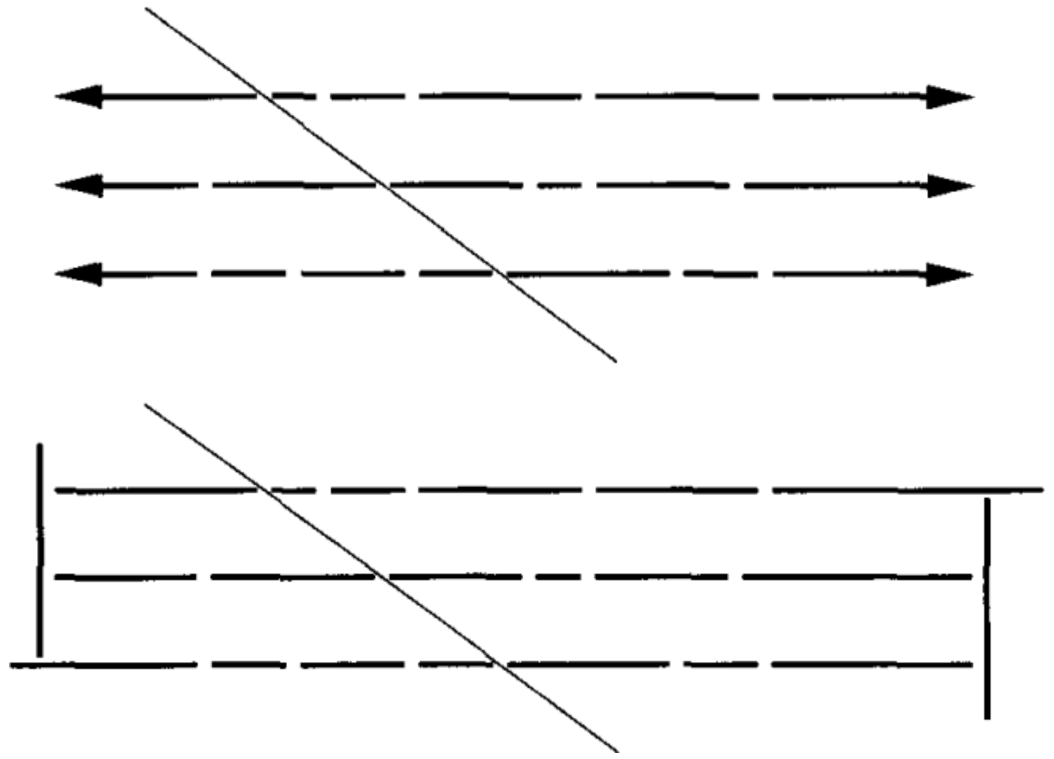
$$a \in A \rightarrow (a, 0)$$
$$b \in B \rightarrow (b, 2)$$
$$c \in C \rightarrow (-c/2, 1)$$

If three points are on a line then $\frac{a+b}{2} = -\frac{c}{2}$

which is equivalent to $a + b + c = 0$

GeomBase → Separator

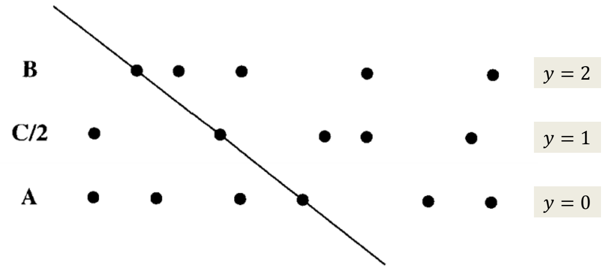
[Gajentaan & Overmars 1995]



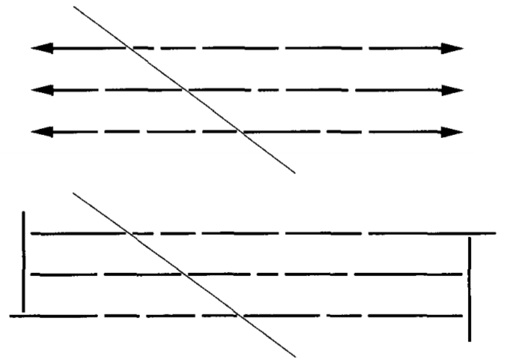
Thanks To Erik Demaine

Separator

3SUM′ → GeomBase
[Gajentaan & Overmars 1995]



GeomBase → Separator
[Gajentaan & Overmars 1995]



Given three lines with zero slope where each line has n gaps. We are asked if there is a line which can go through these gaps which serves as a separator of these lines.

How do you go from GeomBase to Separator?

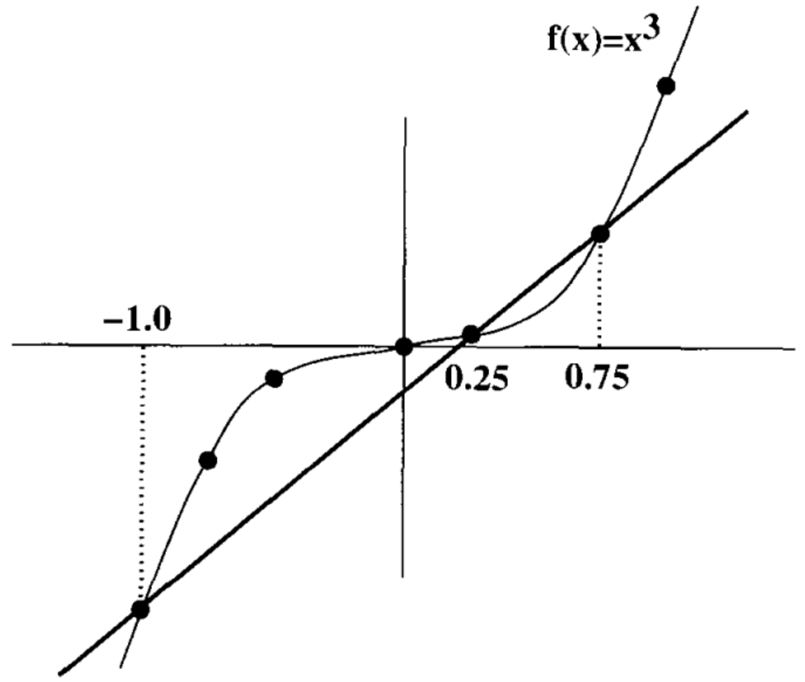
Three Colinear Points

You are given n integer points in the plane.

3-SUM:

Given a set S of n numbers, are there $a, b, c \in S$ where **a+b+c = 0**?

HINT: we are going to take $a \in S$ and create the point (a, a^3).



Three Colinear Points

HINT: we are going to take $a \in S$ and create the point (a, a^3).

When are $(a, a^3), (b, b^3), (c, c^3)$ colinear?

Iff $a + b + c = 0$ [Gajentaan Overmars 94]

Thanks To Erik Demaine

Why?

$$\frac{a-b}{a^3-b^3} = \frac{a-c}{a^3-c^3} = \frac{b-c}{b^3-c^3}$$

Implies $(a-b)(b^3-c^3)-(b-c)(a^3-b^3)=0$ which has roots:

$$c=-a-b, \qquad b=a, \qquad c=a, \qquad c=b$$

But $a \neq b, b \neq c, a \neq c$.

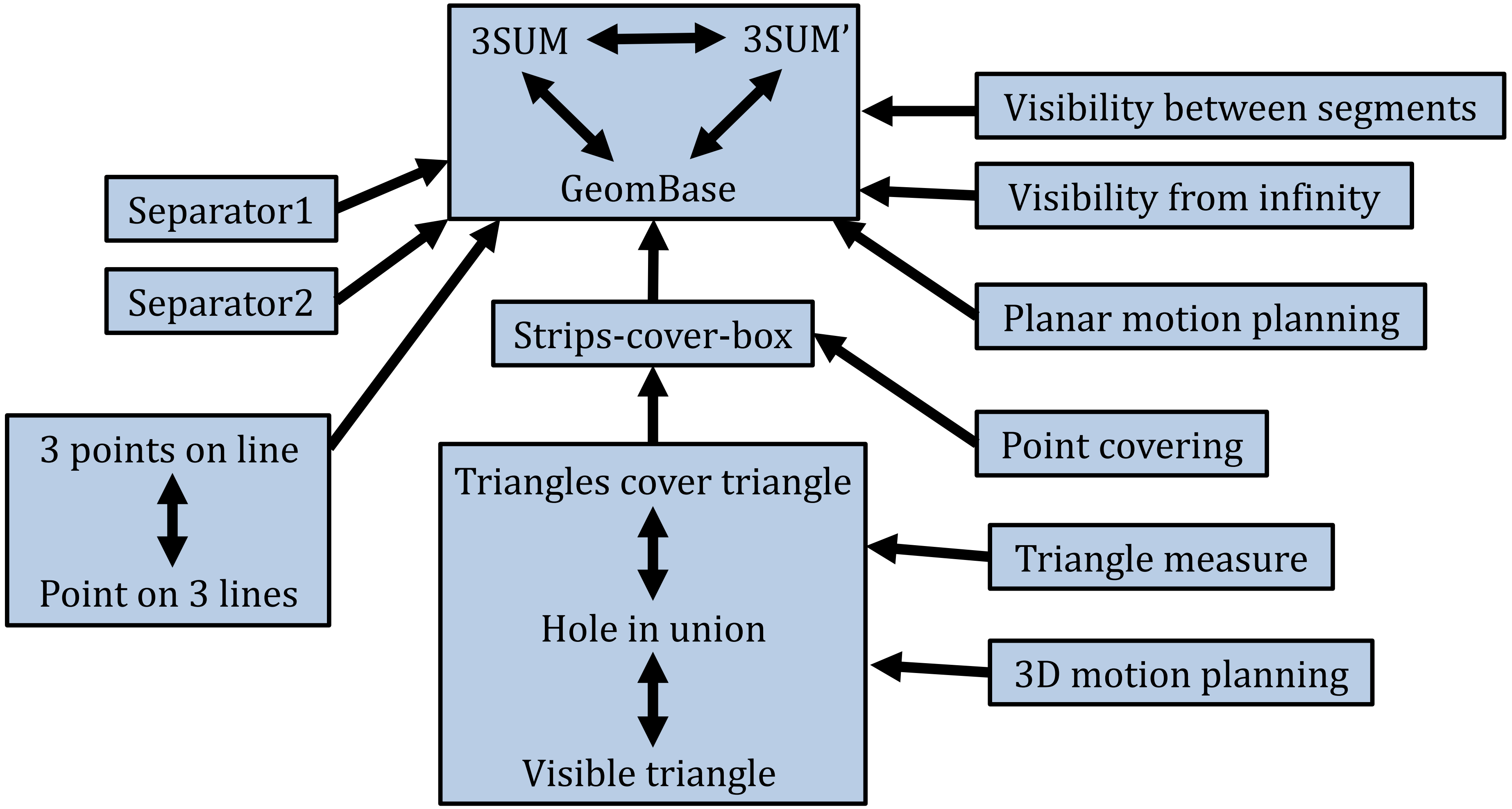
WAIT A MINUITE! (you might be thinking)

What if in S we have two copies of a number, say 1, and a solution like 1,1,-2 exists?

Good catch! Pre-process in $O(n\lg(n))$ time for any duplicate number solutions.

3SUM-hard problems

[Gajentaan & Overmars 1995]



Thanks To Erik Demaine

Why is everything so hard?

We have some answers.

Let us go look at that problem set!