Quantum query complexity and the adversary bound
Part II: Learning graphs

Alexander Belov
University of Latvia

22nd EWSCS, 5-10 March 2017, Palmse

Learning graphs

Recall the dual adversary bound

minimise $\max_z \sum_{j\in[n]} X_j[\![z,z]\!]$

subject to $\sum_{j:x_j\neq y_j} X_j[\![x,y]\!] = 1$ whenever $f(x)\neq f(y)$;

X_j is a p.s.d. $\mathcal{D}\times\mathcal{D}$ matrix for all $j\in[n]$, Recall the dual adversary bound

minimise $\max_z \sum_{j\in[n]} X_j[\![z,z]\!]$

subject to $\sum_{j:x_j\neq y_j} X_j[\![x,y]\!] = 1$ whenever $f(x)\neq f(y)$;

X_j is a p.s.d. $\mathcal{D}\times\mathcal{D}$ matrix for all $j\in[n]$,

How do we ensure the feasibility condition?

- In general this is difficult,
- but there is a way for functions with short certificates.

Function

$$f : [q]^n \supseteq \mathcal{D} \to \{0,1\}$$

For $x \in f^{-1}(1)$, write out:

$$M_x = \{ S \subseteq [n] \mid x_S \text{ is enough to deduce } f(x) = 1 \}.$$

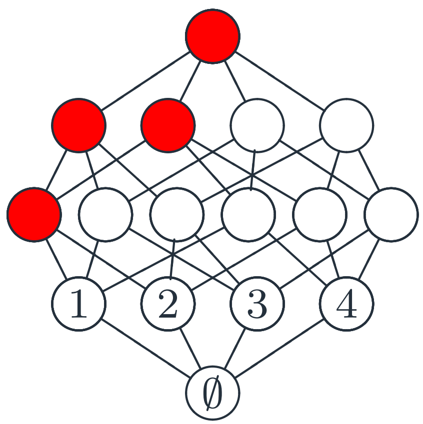
The set of all M_x is the certificate structure of f.

(Interested in inclusion-wise minimal M_x only.)

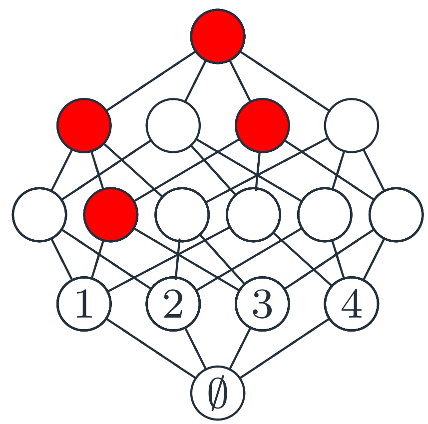
Element Distinctness Problem

Function $f\colon [q]^n \to \{0,1\}$: there are two equal elements.

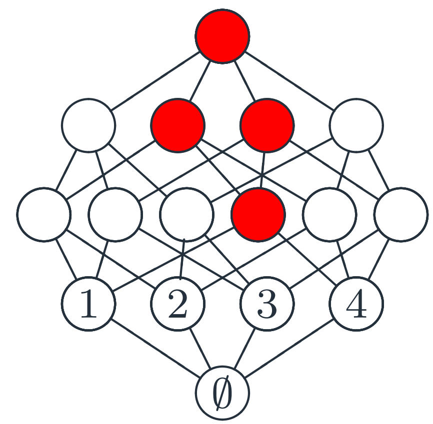
1147



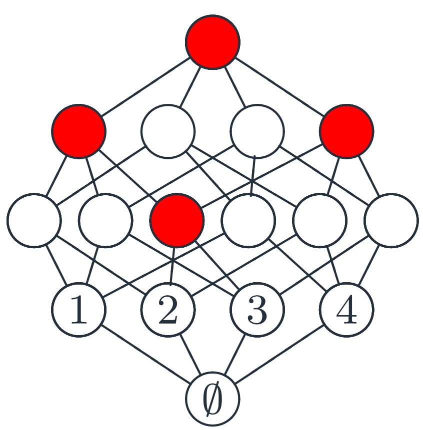
1417



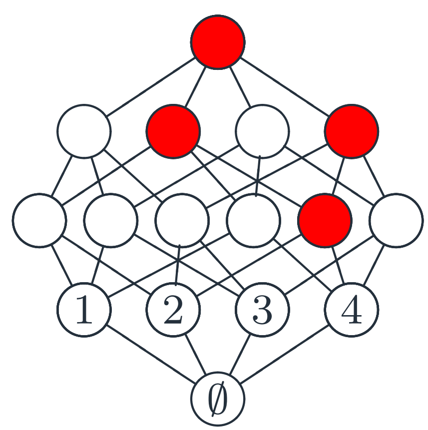
1471



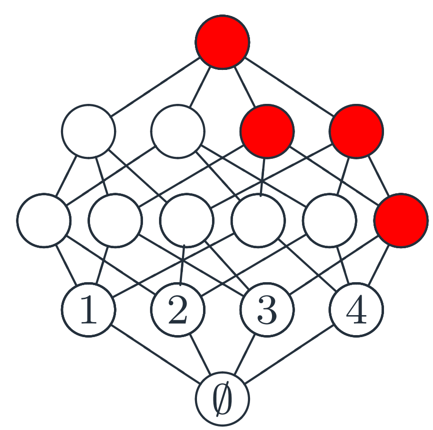
4117



4171



4711



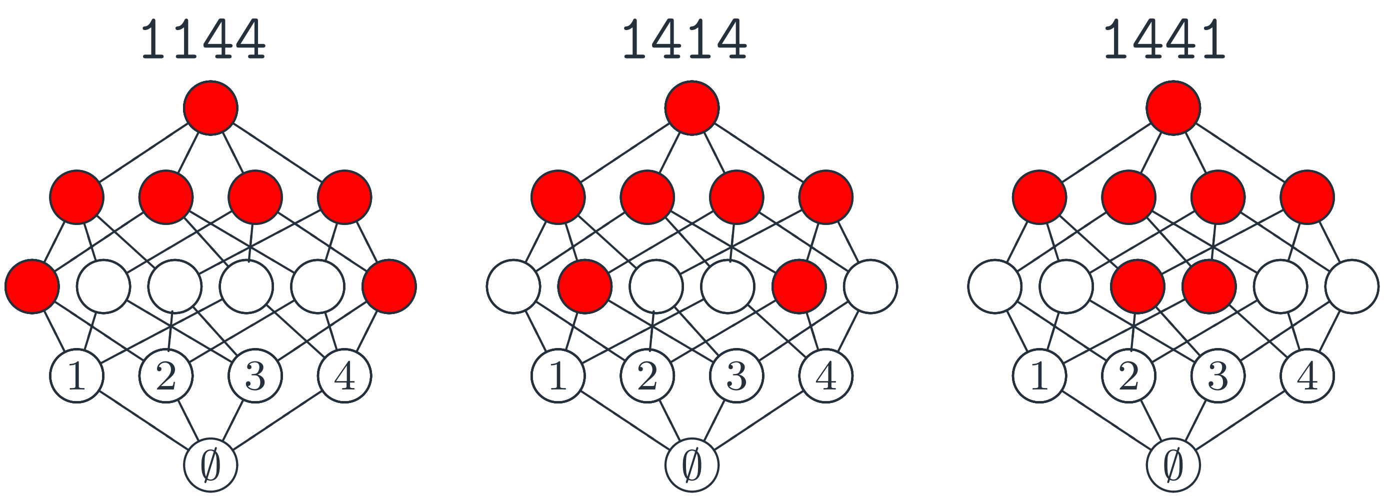
Collision Problem

Distinguish between two cases:

Negative: each symbol in the input string is unique; or

Positive: each symbol has exactly two appearances.

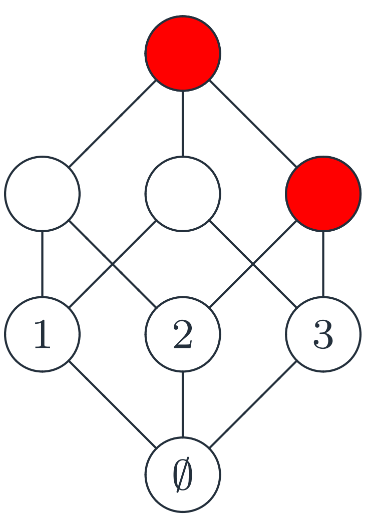
E.g., negative input: 2746 and three variants of positive inputs:



How do we ensure the feasibility condition?

$$\sum_{j: x_j \neq y_j} X_j[\![x, y]\!] = 1 \qquad \text{whenever } f(x) \neq f(y);$$

Element distinctness on positive input $x = 122$.

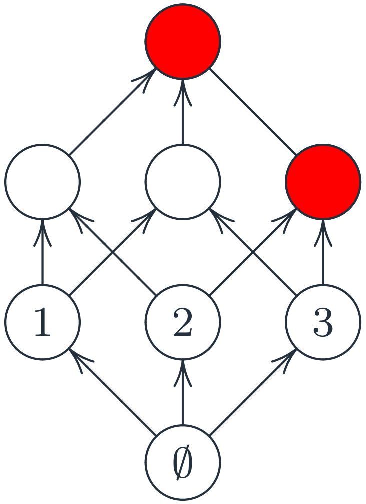


How do we ensure the feasibility condition?

$$\sum_{j : x_j \neq y_j} X_j[\![x, y]\!] = 1 \qquad \text{whenever } f(x) \neq f(y);$$

Element distinctness on positive input $x = 122$.

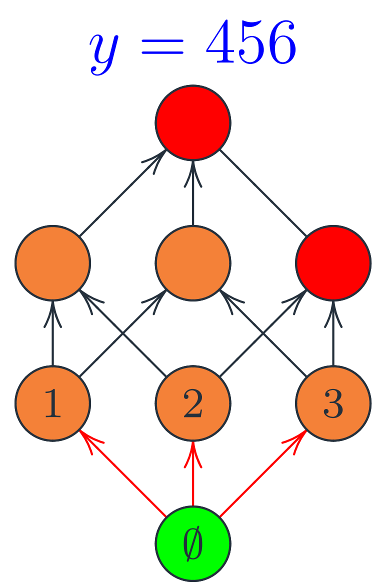
Define flow $p_e(x)$ of value 1 from $\emptyset$ to M_x.

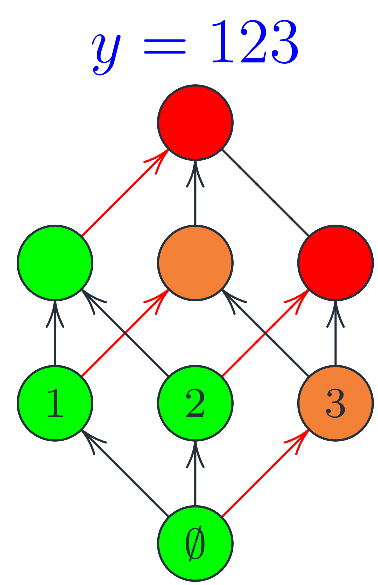


How do we ensure the feasibility condition?

$$\sum_{j:x_j \neq y_j} X_j[\![x,y]\!] = 1 \qquad \text{whenever } f(x) \neq f(y);$$

Element distinctness on positive input $x = 122$.



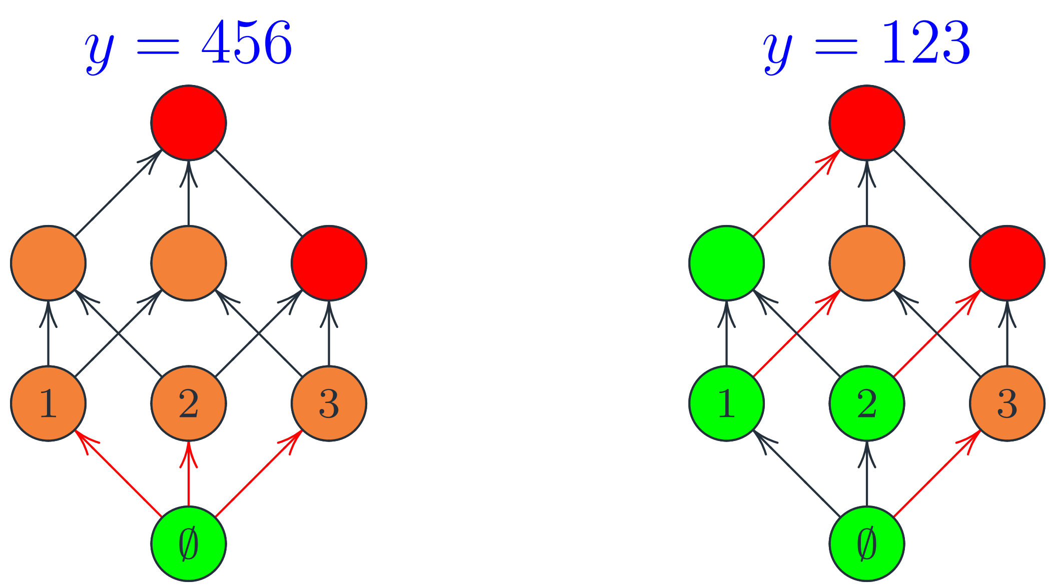


How do we ensure the feasibility condition?

$$\sum_{j: x_j \neq y_j} X_j[\![x,y]\!] = 1 \qquad \text{whenever } f(x) \neq f(y);$$

Element distinctness on positive input $x = 122$.

In all cases, the value of the cut is 1!



For each arc e from S to $S \cup \{j\}$, we define a block-diagonal matrix

$$X_j^e = \sum_\alpha Y_\alpha,$$

where the sum is over all assignments α on S.

Each Y_α is defined as $\psi\psi^*$, where (w_e is the weight of e):

$$\psi[\![z]\!] = \begin{cases} p_e(z)/\sqrt{w_e}, & f(z) = 1, \text{ and } z \text{ satisfies } \alpha; \\ \sqrt{w_e}, & f(z) = 0, \text{ and } z \text{ satisfies } \alpha; \\ 0, & \text{otherwise.} \end{cases}$$

Finally, we define

$$X_j = \sum_{e \text{ loads } j} X_j^e.$$

$$\sum_{j: x_j \neq y_j} X_j[\![x, y]\!] = 1 \qquad \text{whenever } f(x) \neq f(y)$$

Claim. Left-hand side is the value of the cut, and $X_j \succeq 0$.

$$X_j = \sum_{e \text{ loads } j} X_j^e, \qquad X_j^e = \sum_{\alpha} Y_\alpha, \qquad Y_\alpha = \psi\psi^*$$

$$\psi[\![z]\!] = \begin{cases} p_e(z)/\sqrt{w_e}, & f(z) = 1, \text{ and } z \text{ satisfies } \alpha; \\ \sqrt{w_e}, & f(z) = 0, \text{ and } z \text{ satisfies } \alpha; \\ 0, & \text{otherwise.} \end{cases}$$

minimise $\max_z \sum_{j\in[n]} X_j[\![z,z]\!]$

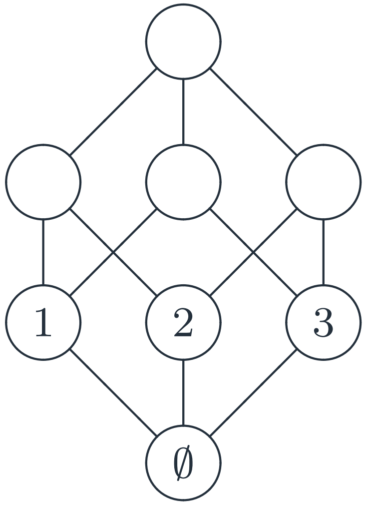
Claim. The objective value is

$$\sum_{j\in[n]} X_j[\![z,z]\!] = \begin{cases} \sum_e w_e, & \text{if } f(z)=0; \\ \sum_e \frac{p_e(z)^2}{w_e}, & \text{if } f(z)=1. \end{cases}$$

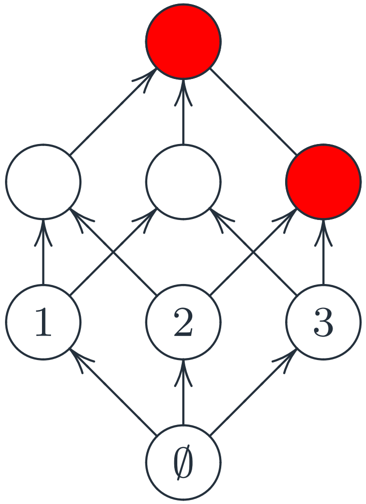
$$X_j = \sum_{e \text{ loads } j} X_j^e, \qquad X_j^e = \sum_\alpha Y_\alpha, \qquad Y_\alpha = \psi\psi^*$$

$$\psi[\![z]\!] = \begin{cases} p_e(z)/\sqrt{w_e}, & f(z)=1, \text{ and } z \text{ satisfies } \alpha; \\ \sqrt{w_e}, & f(z)=0, \text{ and } z \text{ satisfies } \alpha; \\ 0, & \text{otherwise.} \end{cases}$$

- First, for each arc e from S to $S \cup \{j\}$, define its weight w_e.



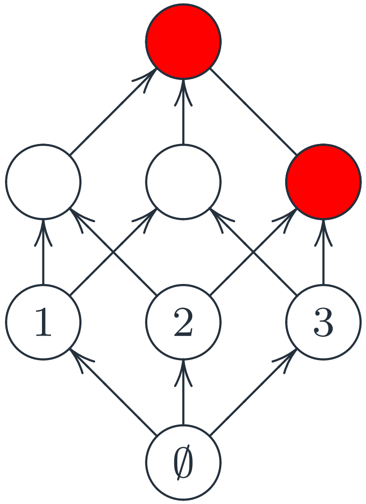
- First, for each arc e from S to $S \cup \{j\}$, define its weight w_e.
- Next, for each M_x in the certificate structure, define flow $p_e(x)$ of value 1 from $\emptyset$ to M_x.



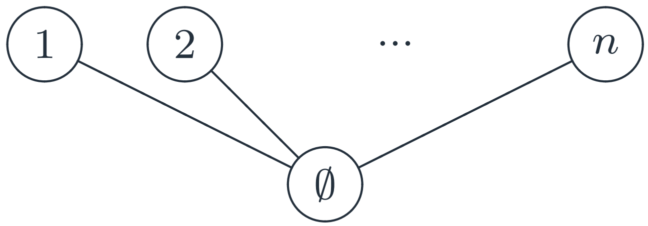
- First, for each arc e from S to $S \cup \{j\}$,
  define its weight w_e.
- Next, for each M_x in the certificate structure,
  define flow $p_e(x)$ of value 1 from $\emptyset$ to M_x.

The complexity of the learning graph is

$$\max\left\{\sum_e w_e, \max_{x \in f^{-1}(1)} \frac{p_e(x)^2}{w_e}\right\}.$$

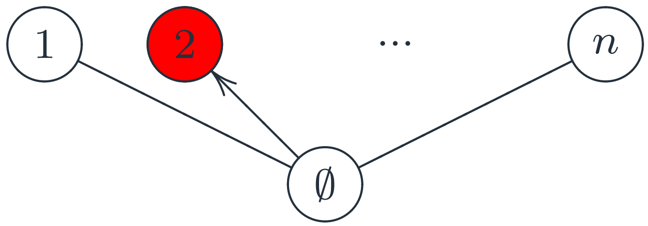


- For each arc from $\emptyset$ to $\{j\}$, its weight is $w_e = \frac{1}{\sqrt{n}}$.



Example: OR function

- For each arc from $\emptyset$ to $\{j\}$, its weight is $w_e = \frac{1}{\sqrt{n}}$.
- For each M_x, the flow goes to $a \in M_x$.

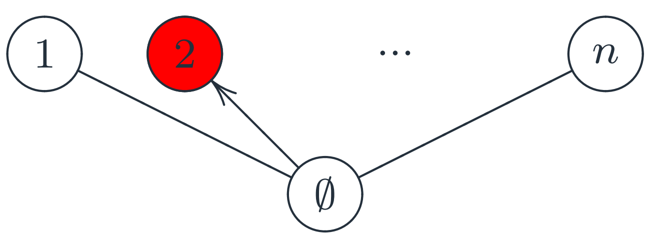


Example: OR function

- For each arc from $\emptyset$ to $\{j\}$, its weight is $w_e = \frac{1}{\sqrt{n}}$.
- For each M_x, the flow goes to $a \in M_x$.

The complexity of the learning graph is

$$\max\left\{\sum_e w_e, \max_{x \in f^{-1}(1)} \frac{p_e(x)^2}{w_e}\right\} = \max\left\{n \cdot \frac{1}{\sqrt{n}}, \frac{1}{1/\sqrt{n}}\right\} = \sqrt{n}.$$



Symmetry

Alternative Description

- Randomized procedure for loading values of variables.
- For each positive input: the goal is to load a 1-certificate.

For OR function:
(x is a positive input, and $\{a\}$ is a 1-certificate)

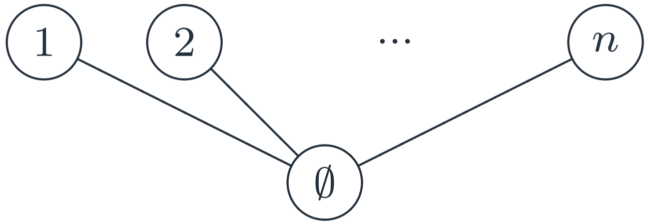
I:	Load a

- We start in the empty set $\emptyset$.
- We divide in a number of stages.
- On each stage we load a number of variables: a transition.
- Length of the transition: number of loaded variables.

I:	Load a

- Define the set of transitions as the union over all inputs:

I:	From $\emptyset$ to $\{j\}$ for all $j \in [n]$



Symmetry Assumption:

- On each stage, the length of each transition is the same.
- On each stage, the number of taken transitions is the same for all inputs, all taken with the same probability.

Then, there exists a learning graph with complexity

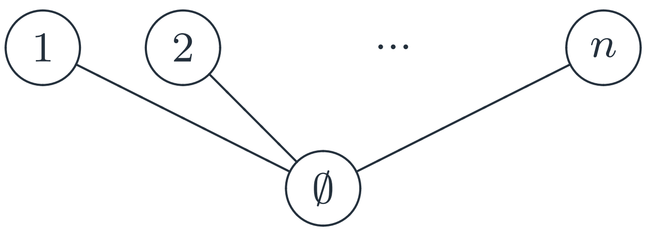
$$\sum_i L_i \sqrt{T_i},$$

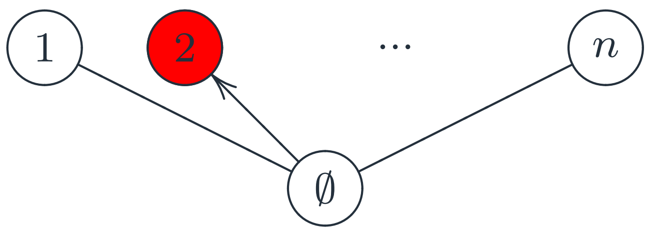
where

Length	L_i:	Number of variables loaded on the stage i
Speciality	T_i:	(Number of transitions on the stage i) / (Number of transitions used for one input)

Symmetry Assumption:

- On each stage, the length of each transition is the same.
- On each stage, the number of taken transitions is the same for all inputs, all taken with the same probability.





	Transitions	Used	Length	Speciality
I:	From $\emptyset$ to $\{j\}$	$j = a$	1	n

Complexity: $\sum_i L_i \sqrt{T_i} = \sqrt{n}$.

Element Distinctness

Given $x_1, \ldots, x_n \in [q]$,
detect whether there exist $a \neq b$ such that $x_a = x_b$.

Certificate: $\{a, b\}$.
Goal: To load two specific elements a and b.

I:	Load a
II:	Load b

	Transitions	Used	Length	Speciality
I:	From $\emptyset$ to $\{i\}$	$i = a$	1	n
II:	From $\{i\}$ to $\{i, j\}$	$i = a, j = b$	1	n^2

Complexity: $\sum_i L_i \sqrt{T_i} = n$.

Naïve learning graph

	Transitions	Used	Length	Speciality
I:	From $\emptyset$ to $\{i\}$	$i = a$	1	n
II:	From $\{i\}$ to $\{i, j\}$	$i = a, j = b$	1	n^2

The second stage is a bottleneck.

Can we improve length?

Can we improve speciality?

I:	Load r elements not from $\{a, b\}$
II:	Load a
III:	Load b

	Transitions	Used	Length	Speciality
I:	From $\emptyset$ to S of r elements	$a, b \notin S$	r	1
II:	From S to $S \cup \{j\}$ for $\|S\| = r$ and $j \notin S$	$a, b \notin S, j = a$	1	n
III:	From S to $S \cup \{j\}$ for $\|S\| = r+1$ and $j \notin S$	$a \in S, j = b$	1	n^2/r

Complexity:

$$\sum_i L_i \sqrt{T_i} = O\big(r + \sqrt{n} + n/\sqrt{r}\big) = O(n^{2/3})$$

when $r = n^{2/3}$.

I:	Load r elements not from $\{a, b\}$
II:	Load a
III:	Load b

Main idea: Before loading b, a is hidden among the r previously loaded elements.

> Where does a man hide a leaf? In the forest.
> But what does he do if there is no forest?..
> He grows a forest to hide it in.
>
> Gilbert Keith Chesterton

A similar algorithm solves any problem with 1-certificate complexity $k = O(1)$.

Let $a_1, \ldots, a_k$ be a 1-certificate.

I:	Load r elements not from $\{a_1, a_2, \ldots, a_k\}$
II.1:	Load a_1
	$\vdots$
II.k:	Load a_k

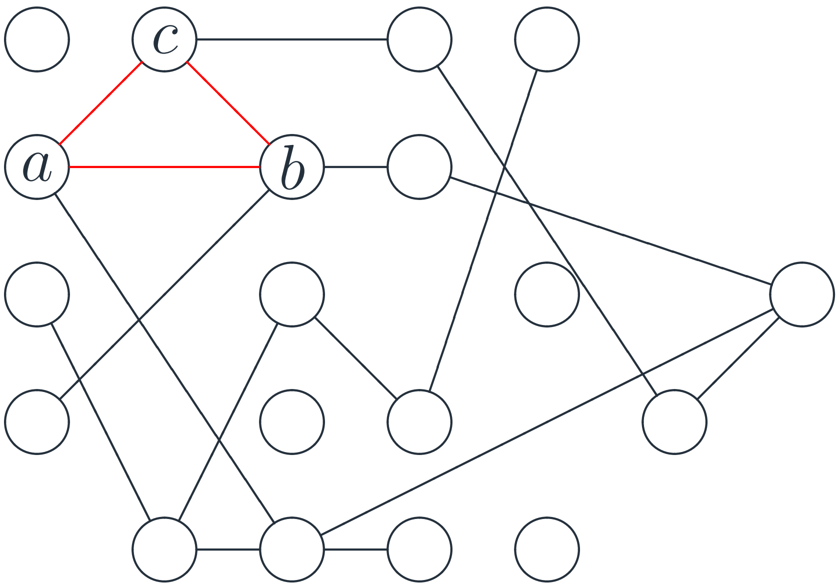
- Complexity is $O(n^{k/(k+1)})$.

Triangle Detection

Given $x_{i,j} \in \{0,1\}$, with $1 \leq i < j \leq n$, detect whether there exist $1 \leq a < b < c \leq n$ such that

$$x_{a,b} = x_{a,c} = x_{b,c} = 1.$$

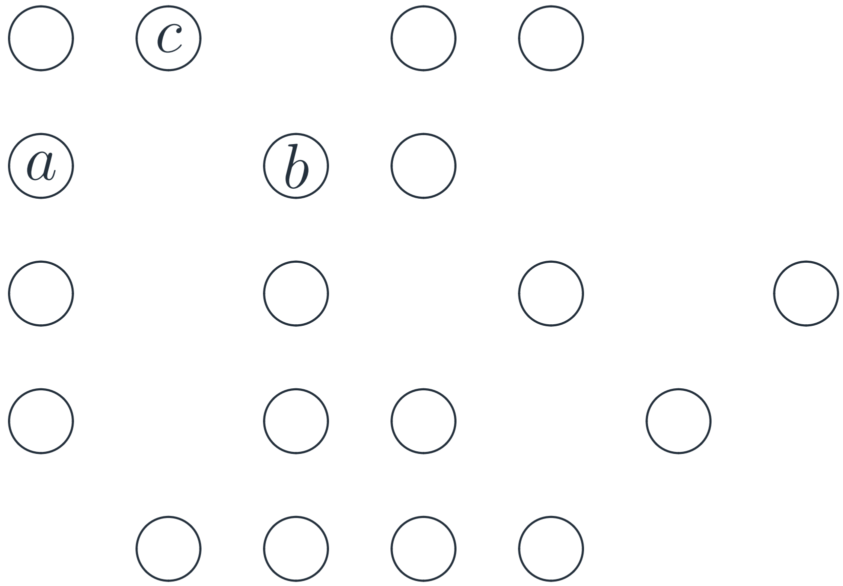
NB: the number of input variables is $\Theta(n^2)$.



Given $x_{i,j} \in \{0,1\}$, with $1 \leq i < j \leq n$, detect whether there exist $1 \leq a < b < c \leq n$ such that

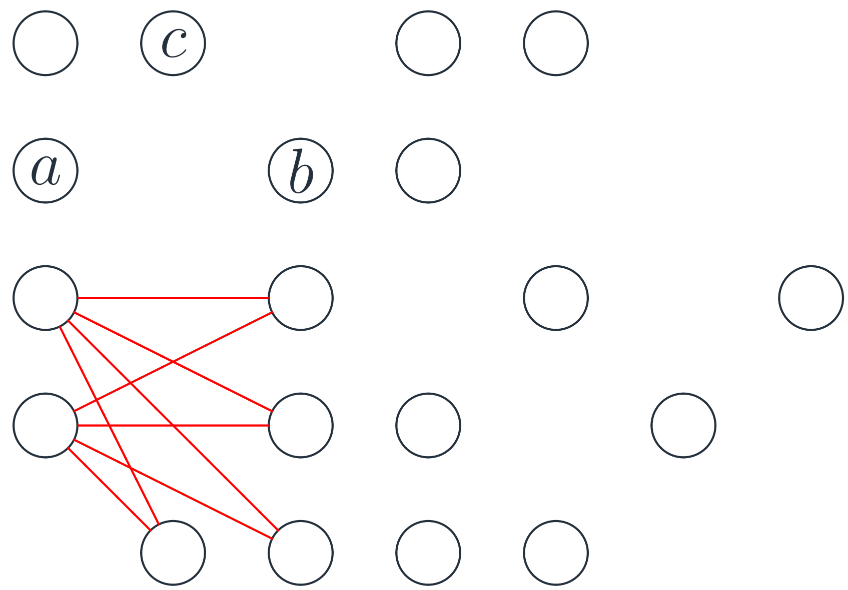
$$x_{a,b} = x_{a,c} = x_{b,c} = 1.$$

- We can use the learning graph of the last section with complexity
$$(n^2)^{3/4} = n^{3/2}.$$
- But we can do better.



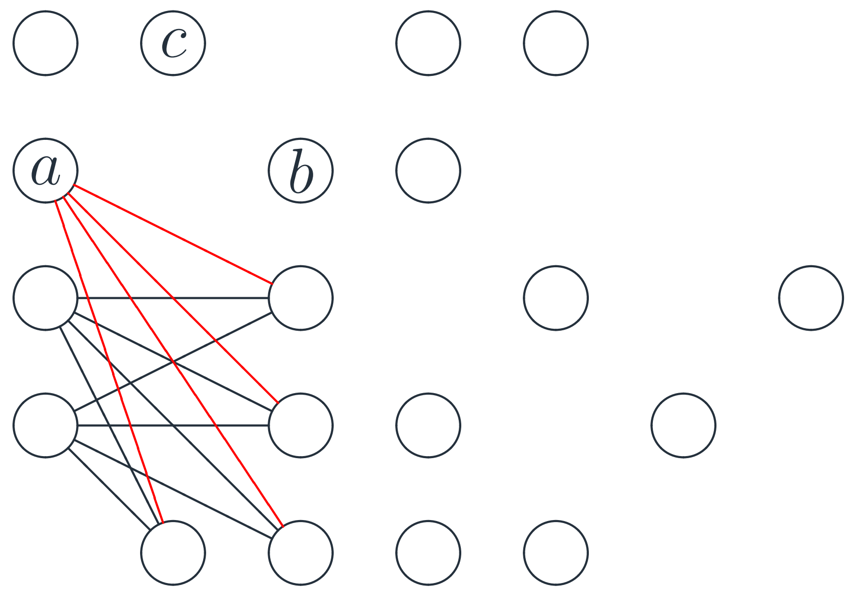
In the beginning nothing is loaded.

Continue as follows...



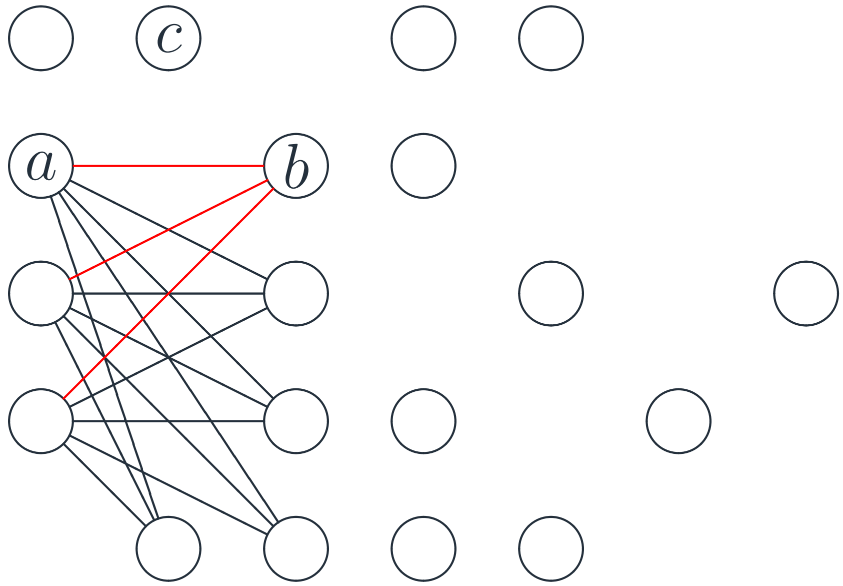
I: Take disjoint $A, B \subseteq [n] \setminus \{a, b, c\}$ of sizes $n^{4/7}$ and $n^{5/7}$, and load all edges between A and B

Length: $|A||B| = n^{9/7}$
Speciality: 1
Complexity: $n^{9/7}$



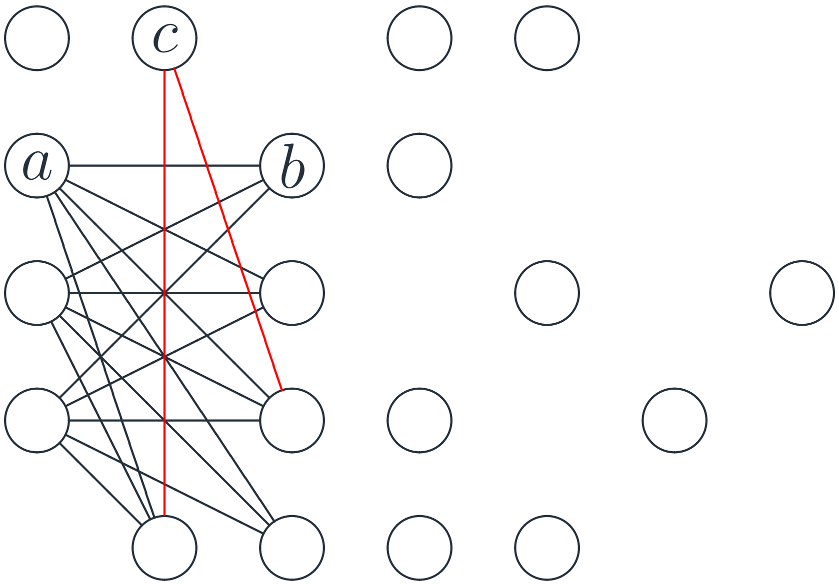
II: Add a to A and load all edges between a and B

Length:	$\lvert B\rvert = n^{5/7}$
Speciality:	n
Complexity:	$n^{17/14}$



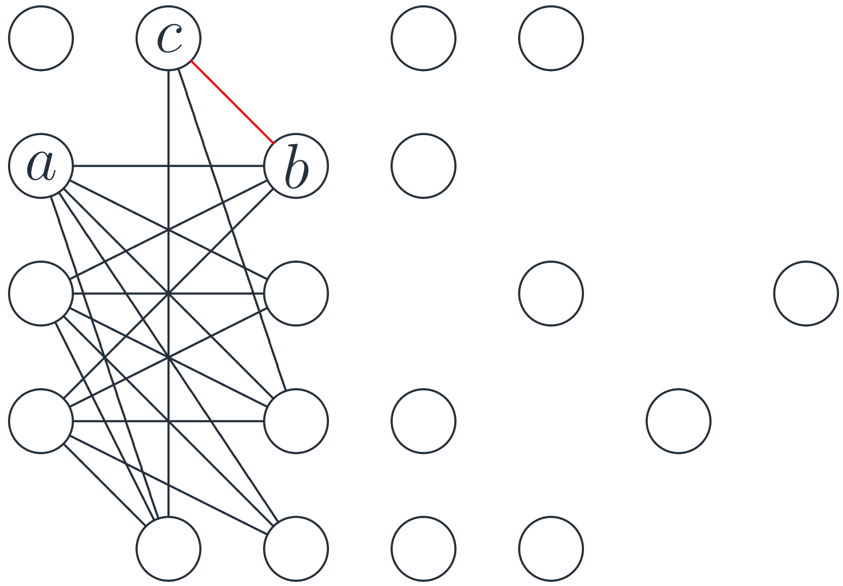
III: Add b to B and load all edges between b and A

Length: $|A| = n^{4/7}$
Speciality: $n^2/|A| = n^{10/7}$
Complexity: $n^{9/7}$



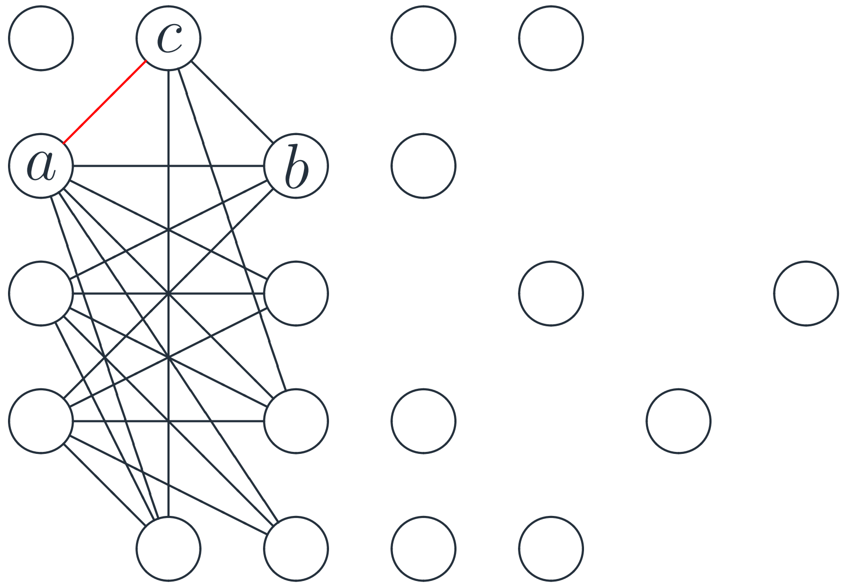
IV: Load $\ell = n^{3/7}$ edges connecting c to vertices in B, but b

Length: $\ell = n^{3/7}$
Speciality: $n^3/(|A||B|) = n^3/(n^{4/7}n^{5/7}) = n^{12/7}$
Complexity: $n^{9/7}$



V: Load edge bc

Length: 1
Speciality: $n^3/|A| = n^3/n^{4/7} = n^{17/7}$
Complexity: $n^{17/14}$



VI: Load edge ac

Length: 1
Speciality: $n^3/\ell = n^{18/7}$
Complexity: $n^{9/7}$

I: Take disjoint $A, B \subseteq [n] \setminus \{a, b, c\}$ of sizes $n^{4/7}$ and $n^{5/7}$ and load all edges between A and B

II: Add a to A and load all edges between a and B

III: Add b to B and load all edges between b and A

IV: Load $\ell = n^{3/7}$ edges connecting c to elements in B, but b

V: Load edge bc

VI: Load edge ac

Stage	I	II	III	IV	V	VI
Length	$n^{9/7}$	$n^{5/7}$	$n^{4/7}$	$n^{3/7}$	1	1
Speciality	1	n	$n^{10/7}$	$n^{12/7}$	$n^{17/7}$	$n^{18/7}$
Complexity	$n^{9/7}$	$n^{17/14}$	$n^{9/7}$	$n^{9/7}$	$n^{17/14}$	$n^{9/7}$

Total complexity: $O(n^{9/7})$.

Thank you!