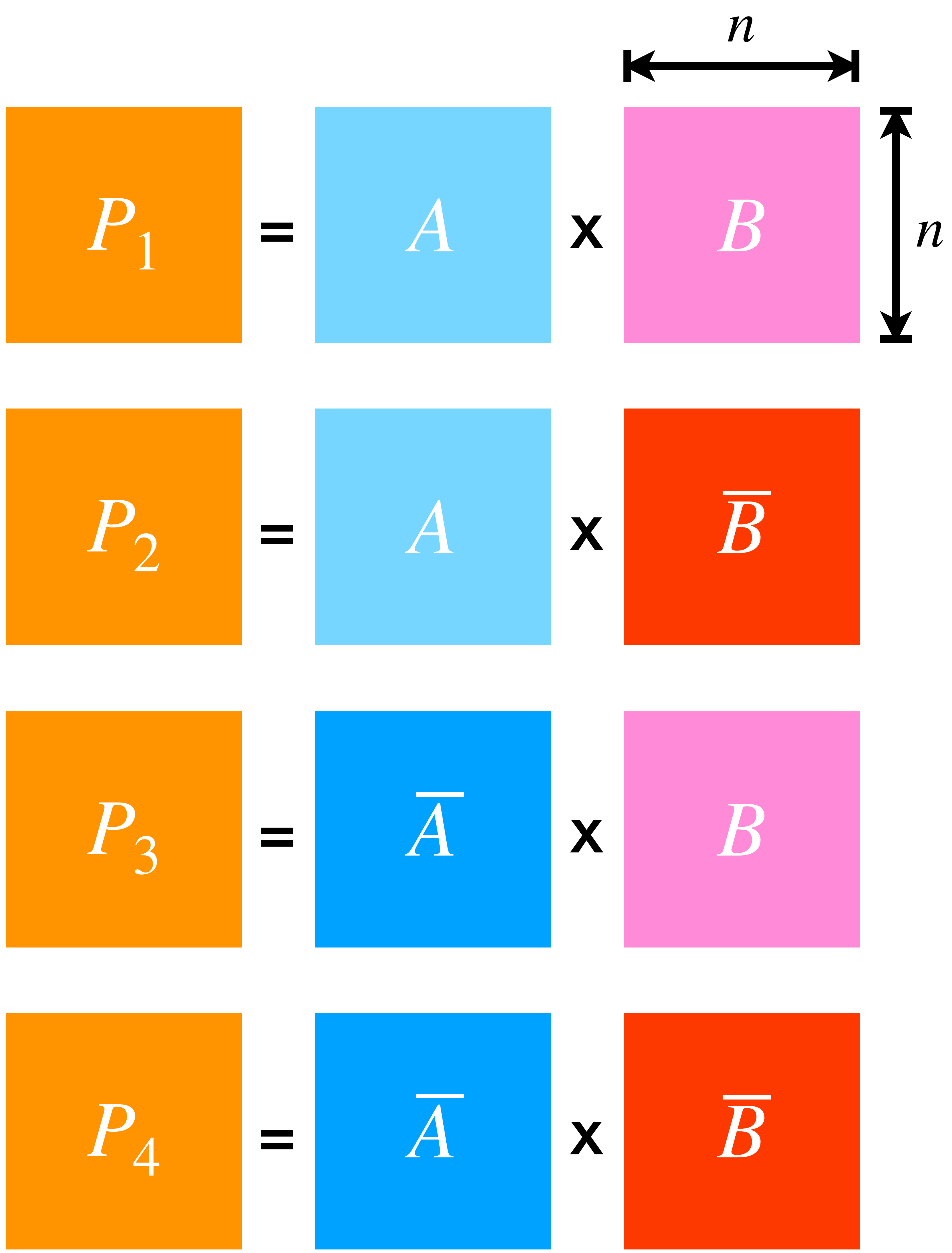
Verifying the Product of Generalized Boolean Matrix Multiplication and Its Applications to Detect Small Subgraphs

Wing-Kai Hon, Meng-Tsung Tsai, and Hung-Lung Wang

WADS 2023

NTHU, Taiwan
Academia Sinica, Taiwan
NTNU, Taiwan

Problem Definition



Input:

(1) two n by n Boolean matrices A and B

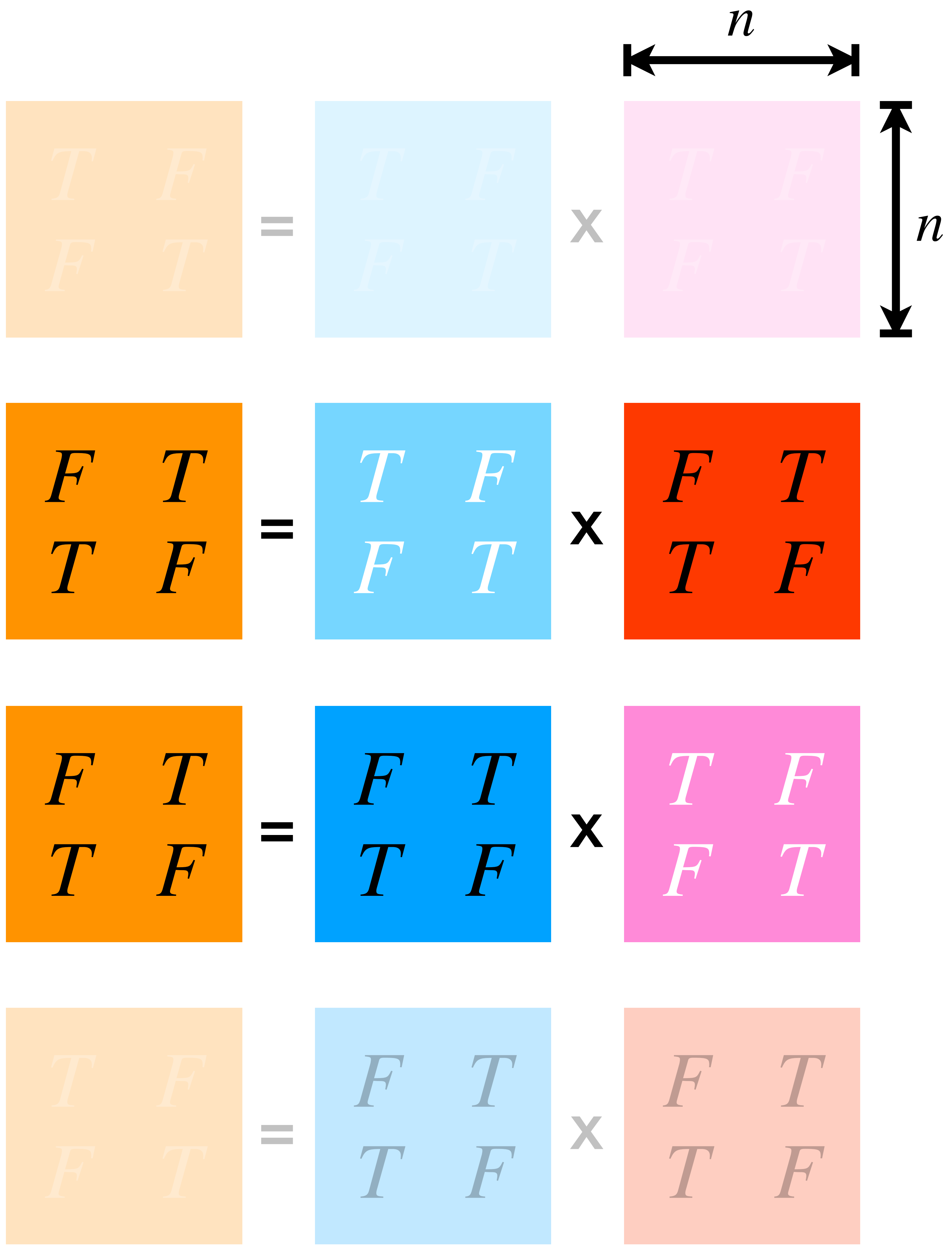
(2) a non-empty subset S of $\{P_1, P_2, P_3, P_4\}$ (given as symbols rather than the explicit matrices)

Output:

"Yes", if the entry-wise logical-and of all matrices in S contains only False entries;

"No", otherwise.

Sanity Check (1/2)



Input:

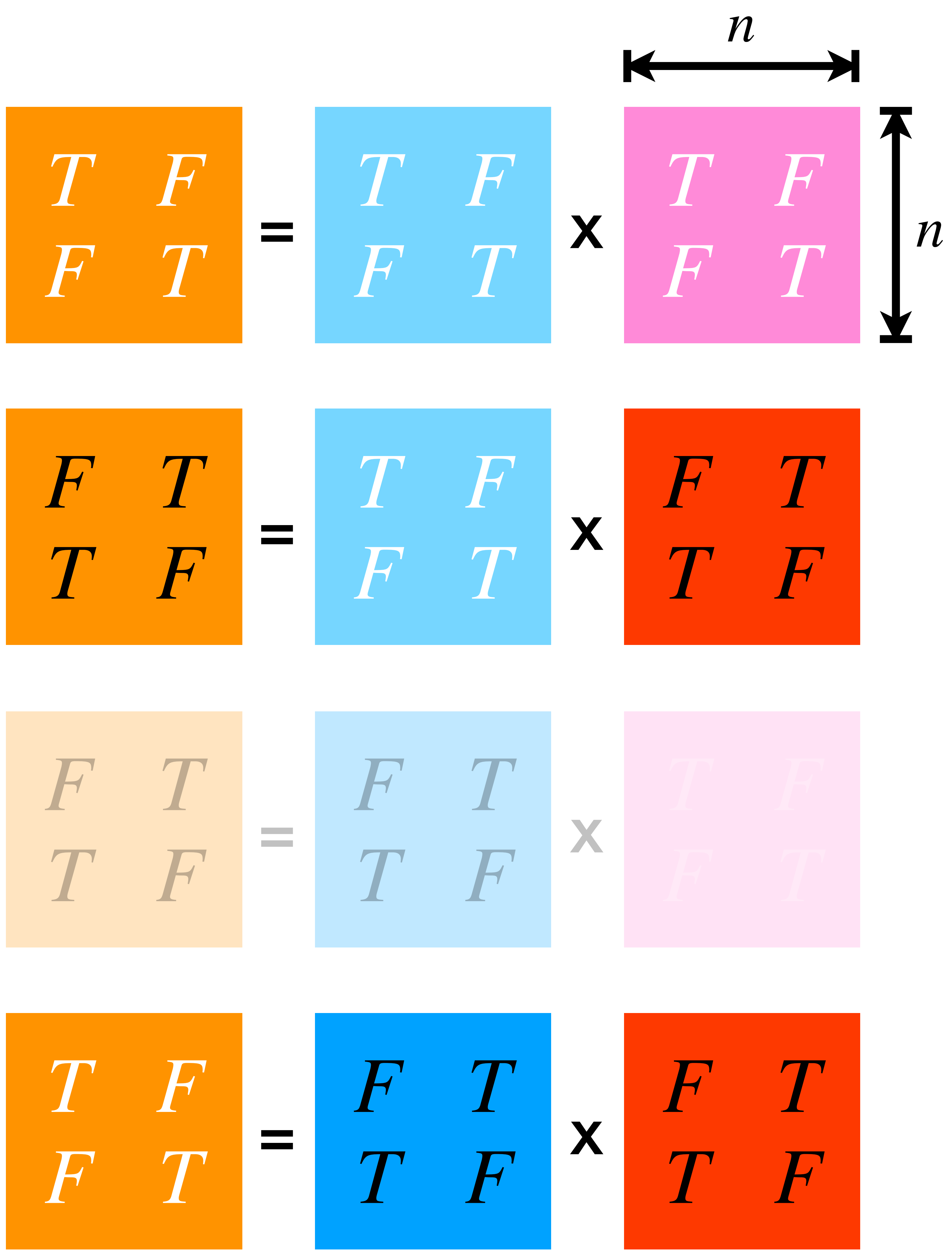
(1) two n by n Boolean matrices A and B

(2) $S = \{P_2, P_3\}$ (given as symbols rather than the explicit matrices)

Output:

"No" because P_2 & $P_3 = \begin{bmatrix} F & T \\ T & F \end{bmatrix}$.

Sanity Check (2/2)



Input:

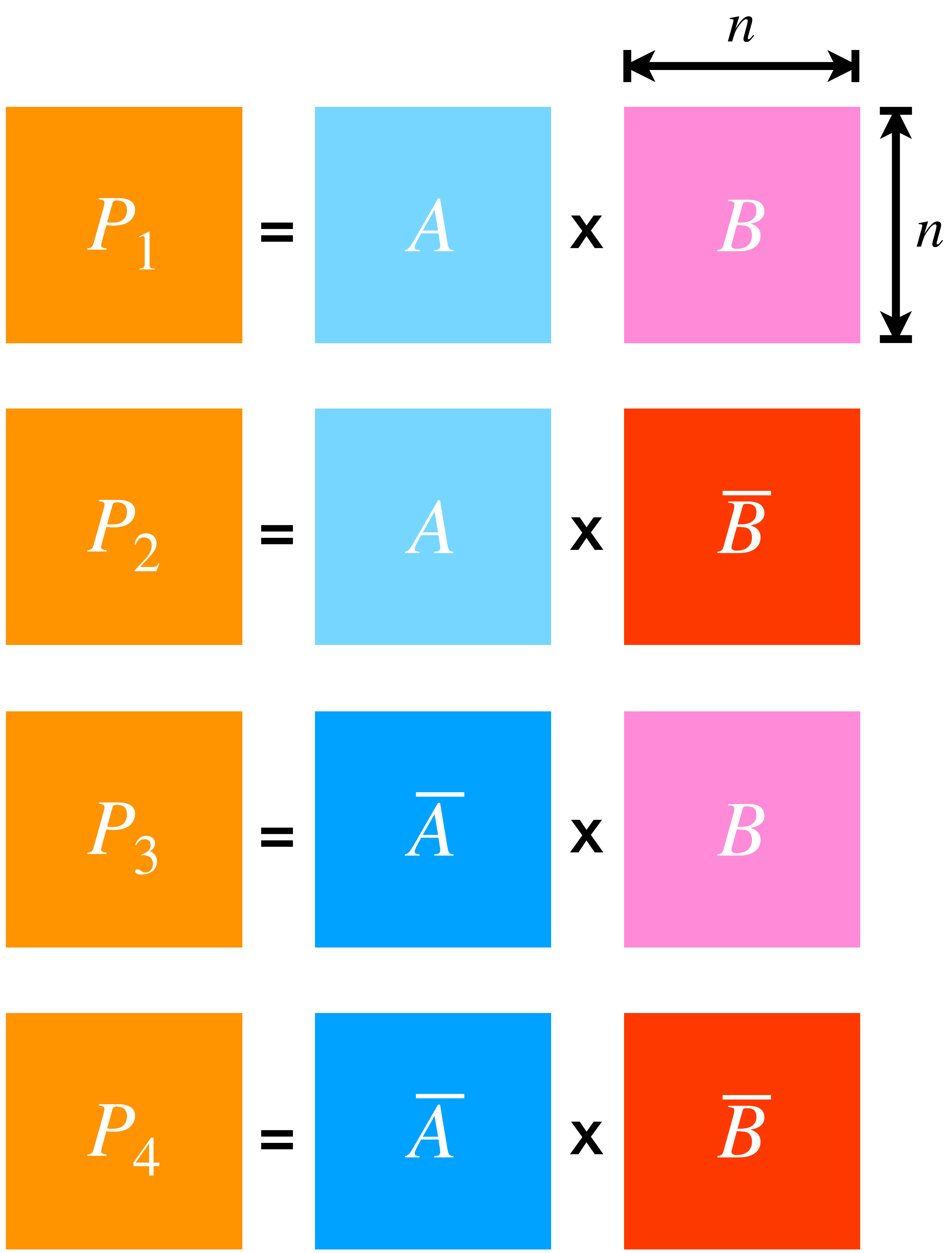
(1) two n by n Boolean matrices A and B

(2) S = $\{P_1, P_2, P_4\}$ (given as symbols rather than the explicit matrices)

Output:

"Yes" because $P_1 \,\&\, P_2 \,\&\, P_4 = \begin{bmatrix} F & F \\ F & F \end{bmatrix}$.

Main Message



Input:

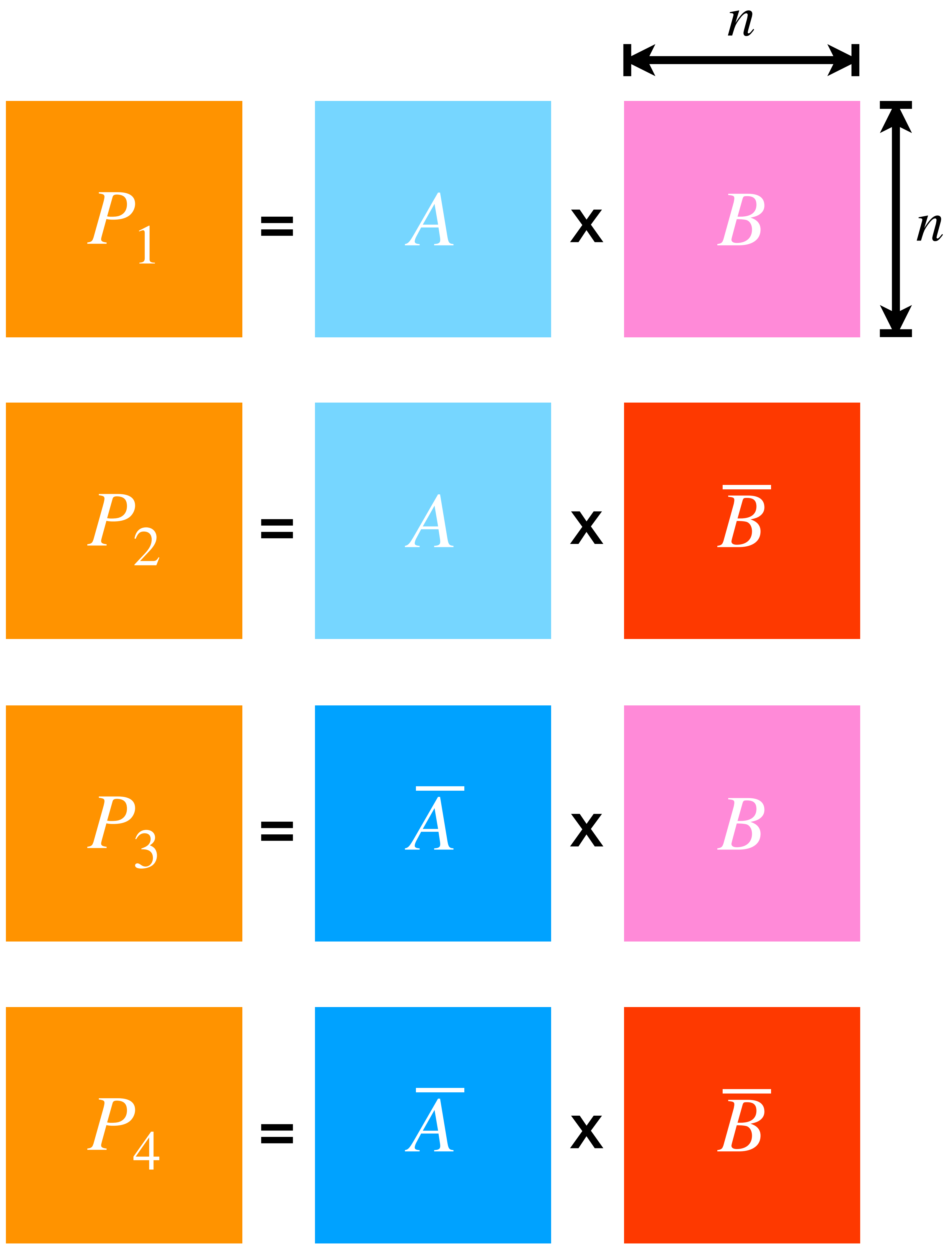
(1) two n by n Boolean matrices A and B

(2) a non-empty subset S of $\{P_1, P_2, P_3, P_4\}$ (given as symbols rather than the explicit matrices)

Our Main Theorem.

Verifying whether the product of GBMM contains only False entries can be done in deterministic $O\left(n^2\right)$ time.

Is Our Result Interesting? (1/3)



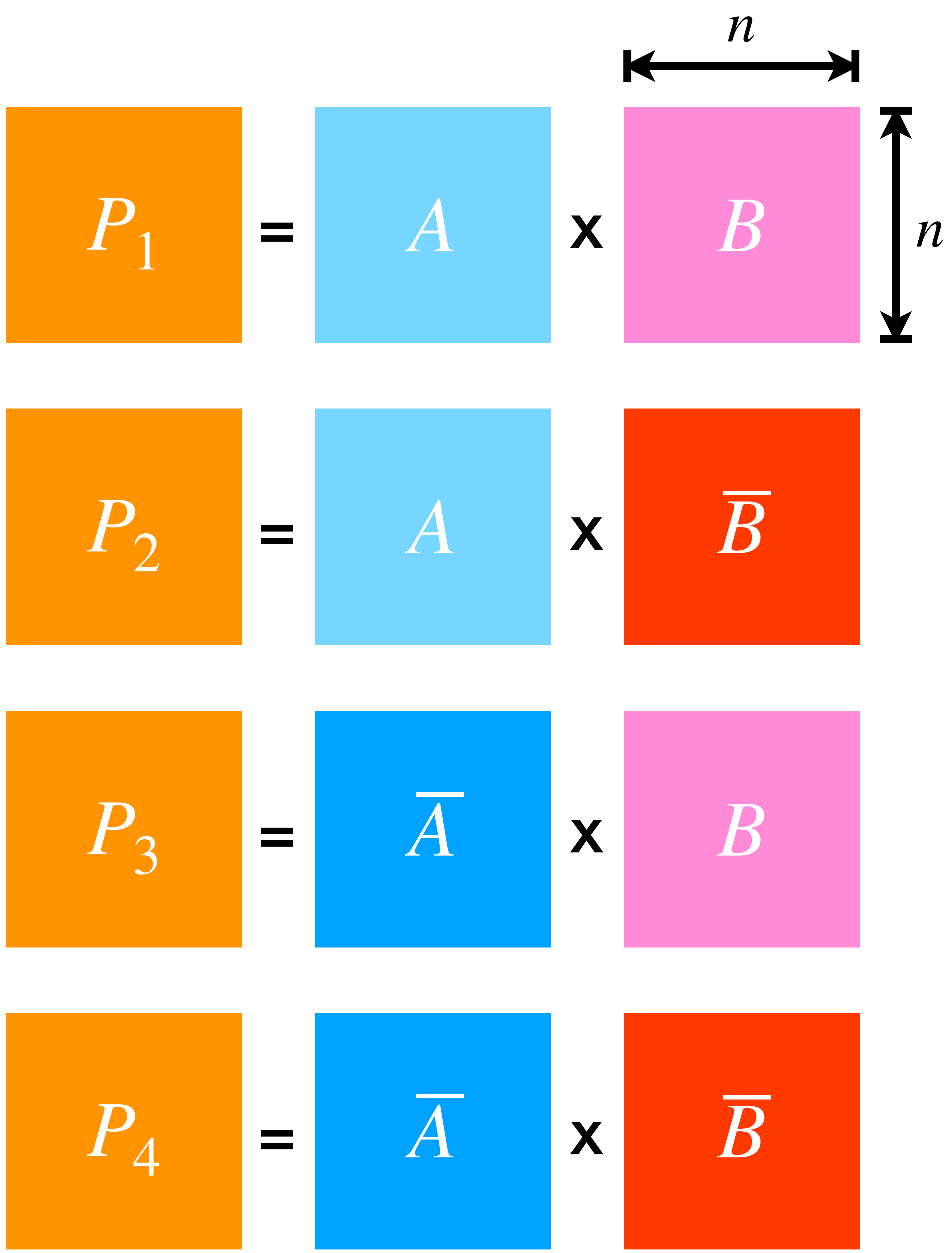
Our Main Theorem.

> Verifying whether the product of GBMM contains only False entries can be done in deterministic $O\left(n^2\right)$ time.

By a reduction from Diameter 2 or 3
[Aingworth, Chekuri, Indyk, and Motwani'99]

> Verifying whether the product of GBMM contains only True entries needs $O\left(n^3\right)$ time by any known combinatorial algorithm.

Is Our Result Interesting? (2/3)



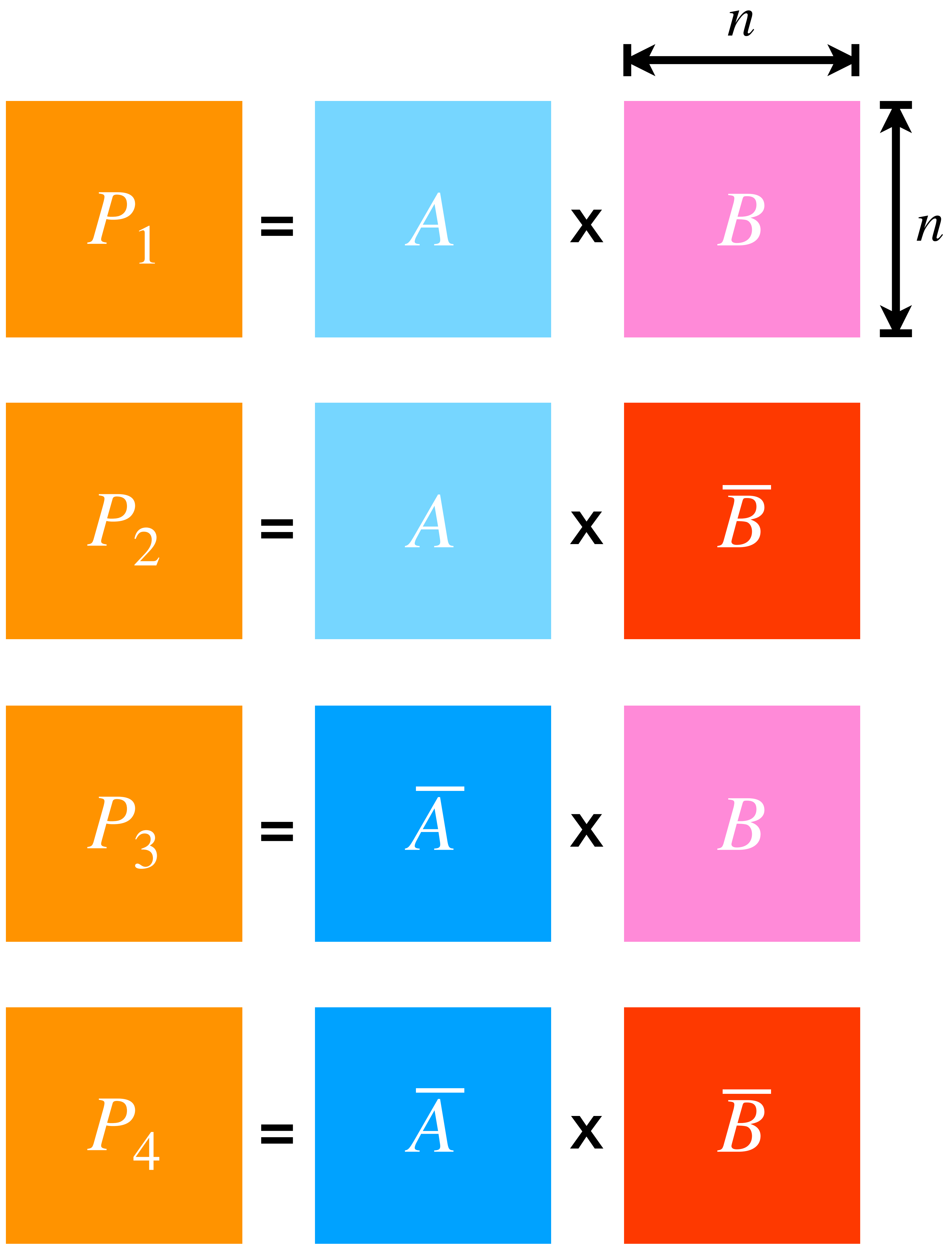
<u>Our Main Theorem</u>.

> Verifying whether the product of GBMM contains only False entries can be done in deterministic $O\left(n^2\right)$ time.

By Freivalds' algorithm [Freivals'77] (noting that it has no known efficient deterministic alternative)

> Verifying whether the product of GBMM contains only False entries can be done in randomized $O\left(n^2\right)$ time.

Is Our Result Interesting? (3/3)



Our Main Theorem.

Verifying whether the product of GBMM contains only False entries can be done in deterministic $O\left(n^2\right)$ time.

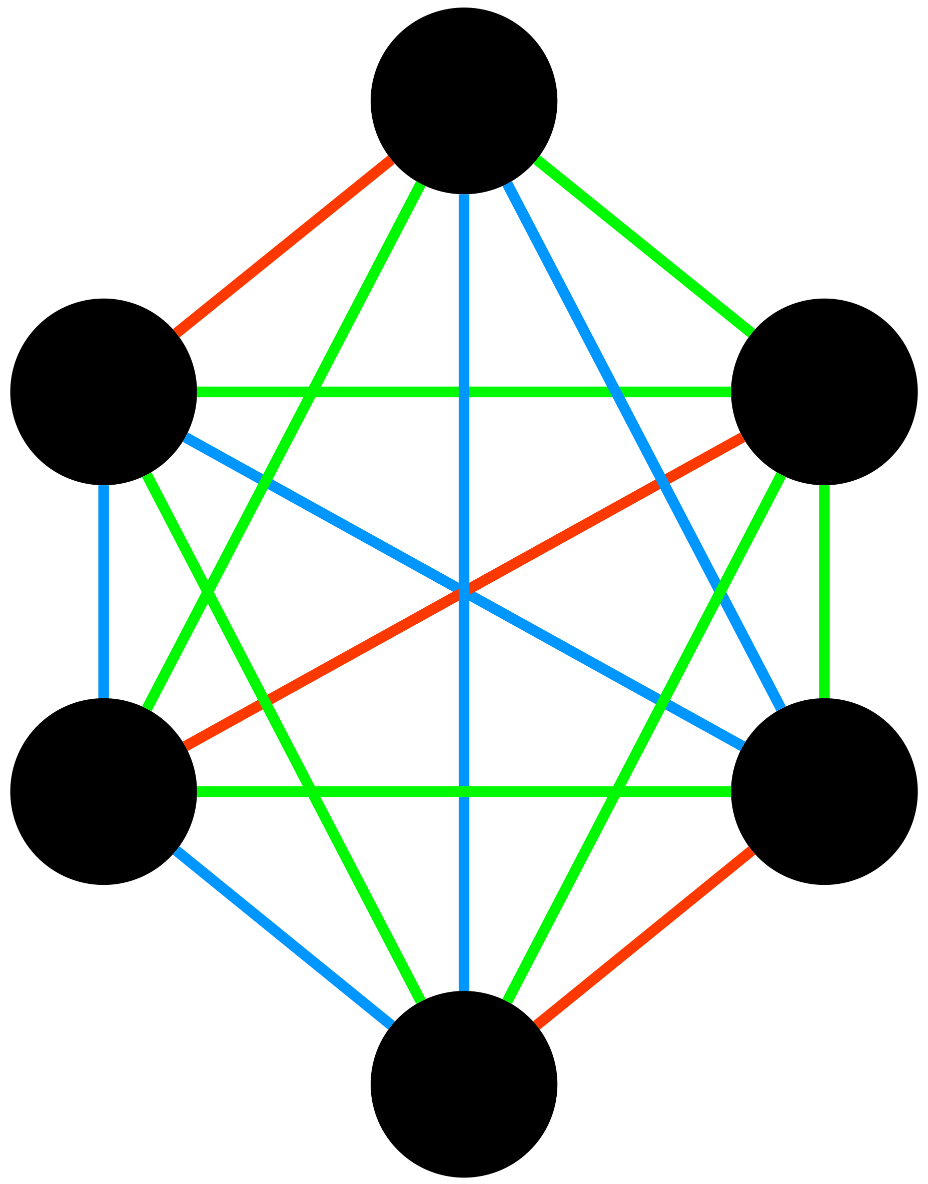
Our main result can be applied to detect the existence of several small subgraphs.

To be introduced in a minute.

Application I

Detecting Designated Colored 4-Cycles

Problem Definition



Fix an edge-colored 4-cycle C.

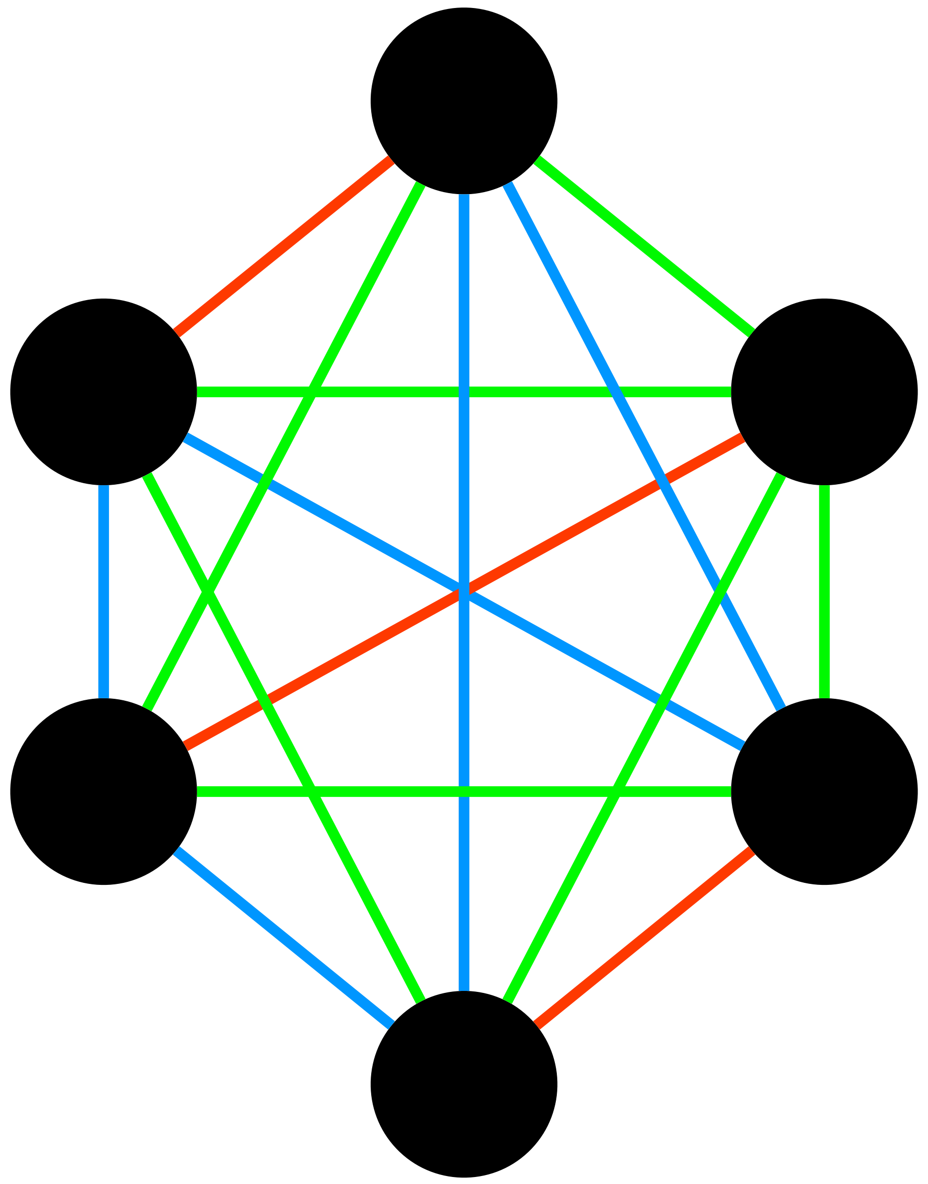
Input: an edge-colored complete graph G.

Output:

“Yes”, G contains C as a subgraph;

“No”, otherwise.

Problem Definition



General (monochromatic) graphs can be thought as 2-edge-colored complete graphs.

Fix an edge-colored 4-cycle C.

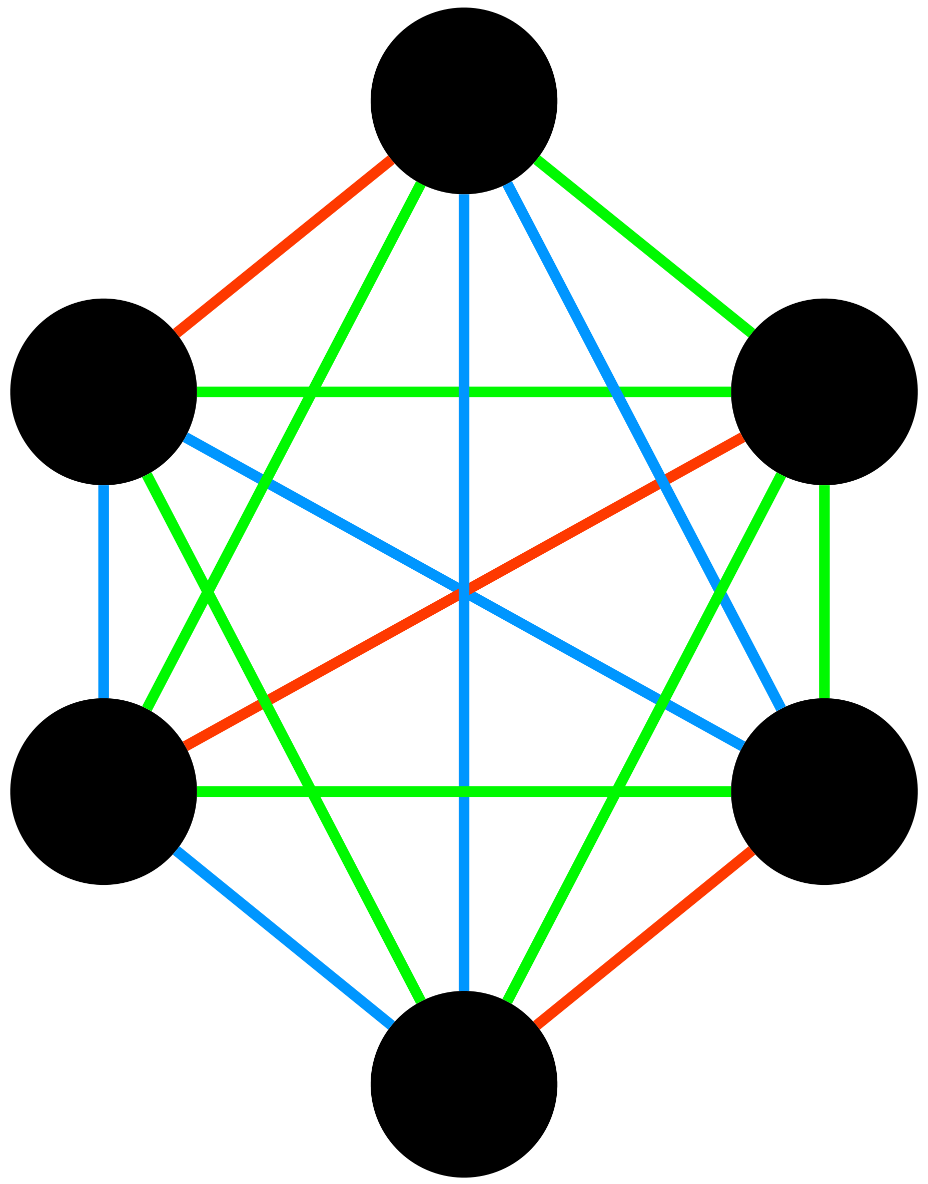
Input: an edge-colored complete graph G.

Output:

"Yes", G contains C as a subgraph;

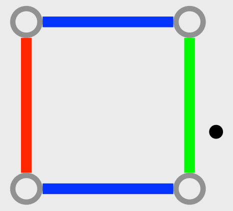
"No", otherwise.

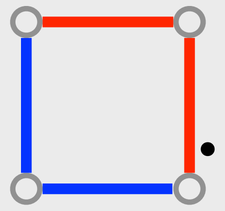
Detecting Different Designated 4-Cycles Can Have Different Complexities, Unconditionally

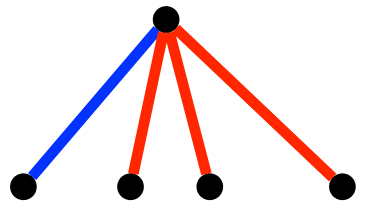
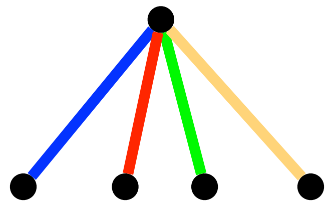
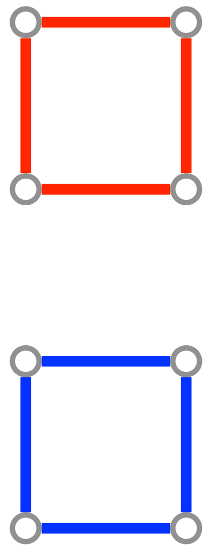
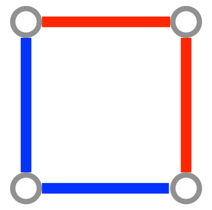
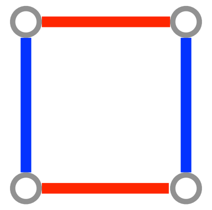
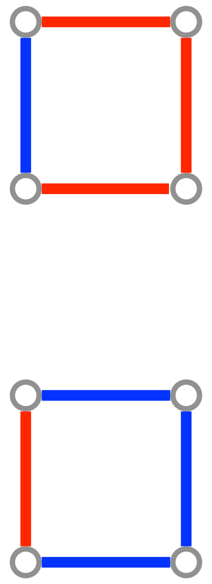
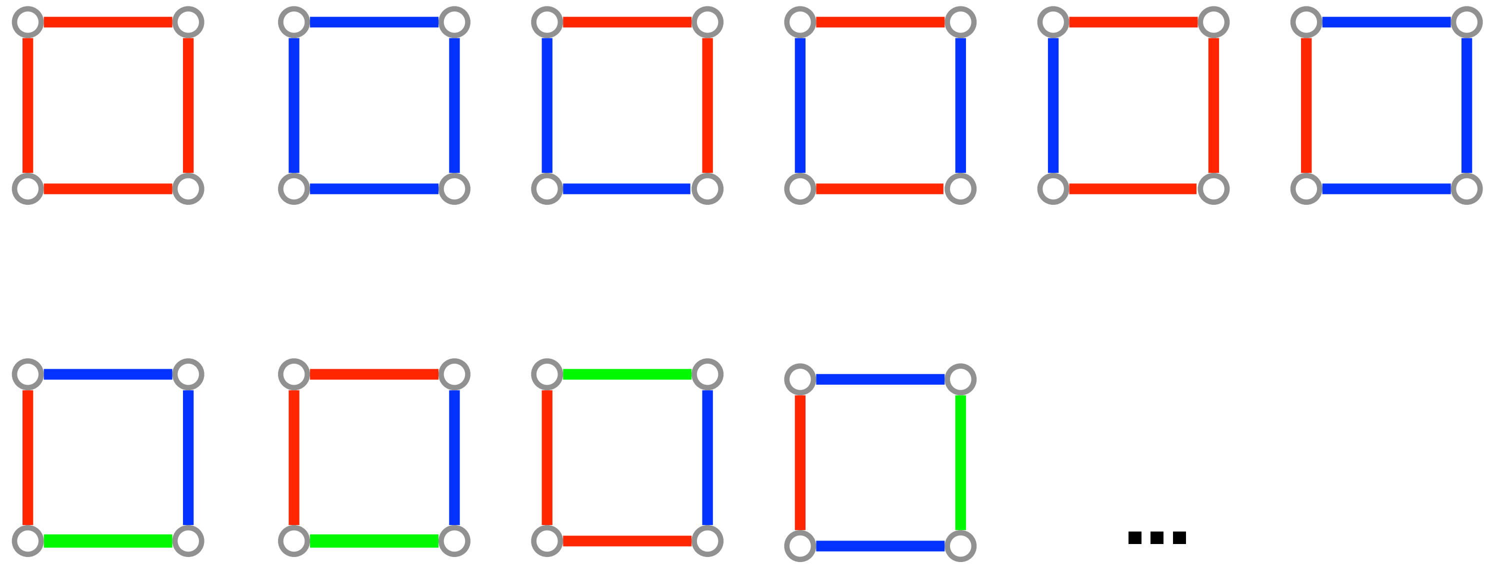


Fix an edge-colored 4-cycle C.

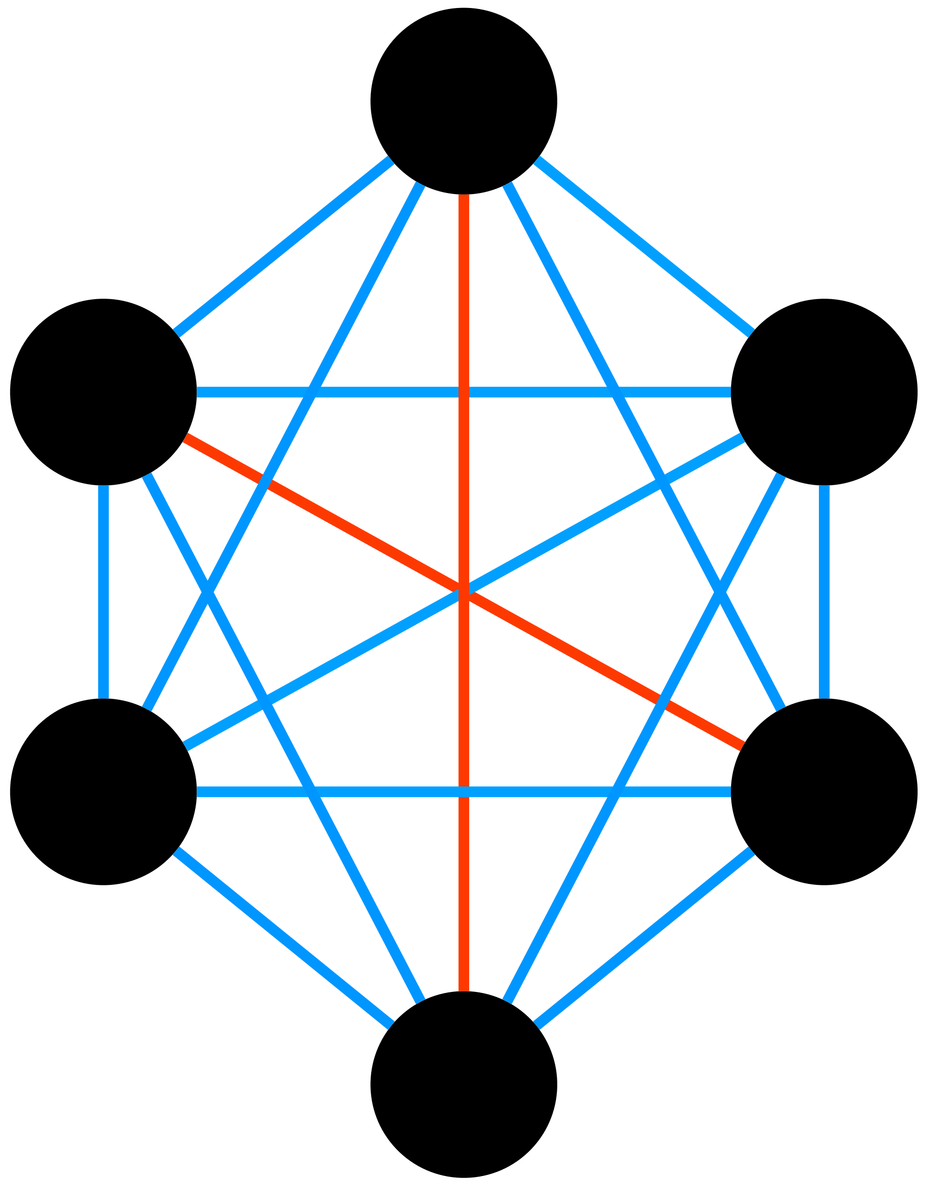
Input: an edge-colored complete graph G.

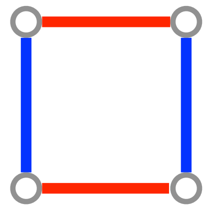
Case I: fix $C =$ . Any single-pass streaming algorithm that detects C requires $\Omega(n^2)$ space.

Case II: fix $C =$ . There is a single-pass streaming algorithm that detects C using $O(n)$ space.

	The number of colors of edges incident to each vertex is at most 2. 					at most 3. 
	The number of edge colors in G is at most 2.				The number of edge colors in G can be more than 2.	
C						
Runtime	Deterministic $O(n^2)$	Deterministic $O(n^2)$	Deterministic $O(n^2)$	Deterministic $O(n^2)$	Randomized $O(n^2)$	Triangle-hard
Approach	Pigeonhole [YZ'97]	Pigeonhole [YZ'97] or Ramsey-type Thm [HTW'23]	Ramsey-type Theorem [LBYY'21]	Ramsey-type Theorem [GKMT'17] [GS'11]	Our Result	Our Result

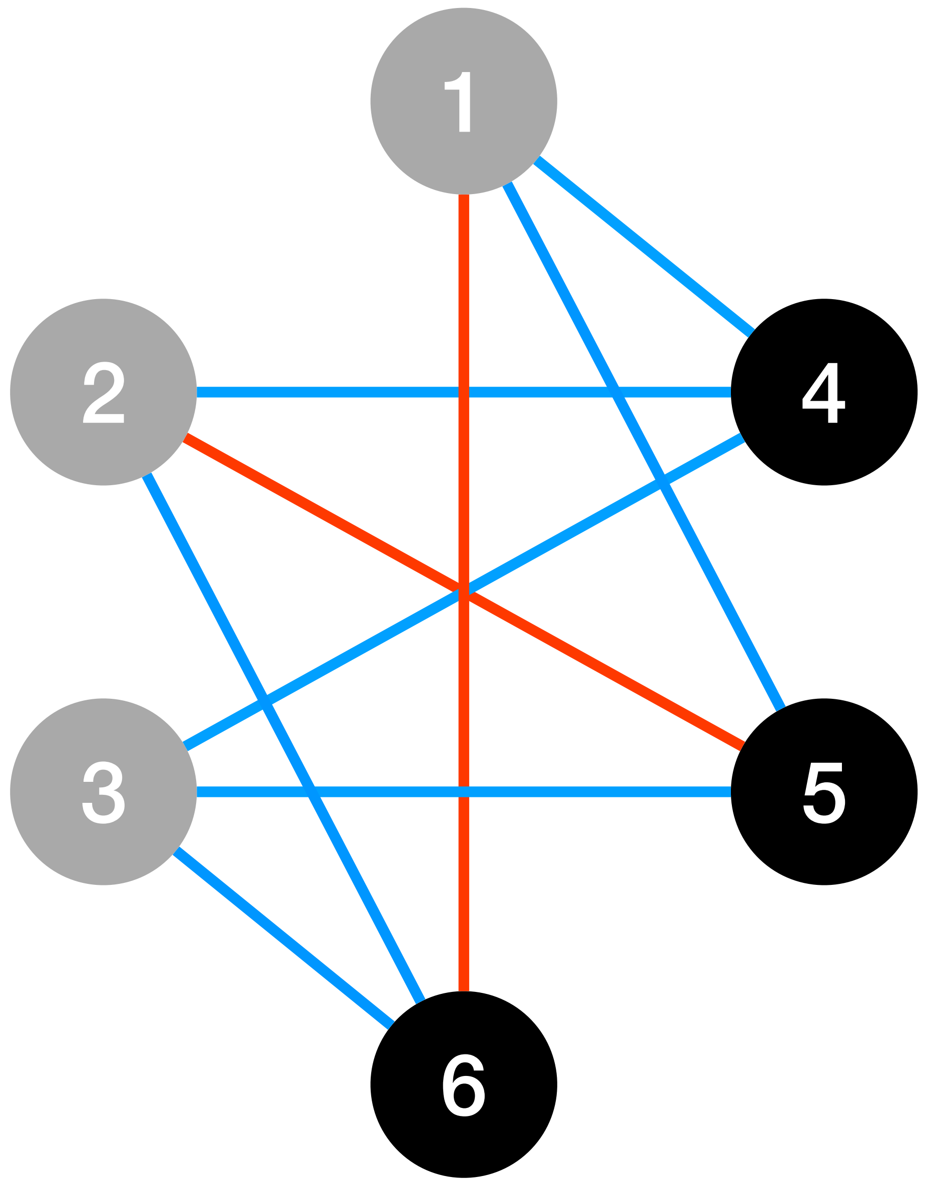
Reduction from Detecting 4-Cycles to GBMM

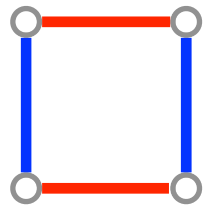


Fix C =  .

Input: an edge-colored complete graph G

Reduction from Detecting 4-Cycles to GBMM

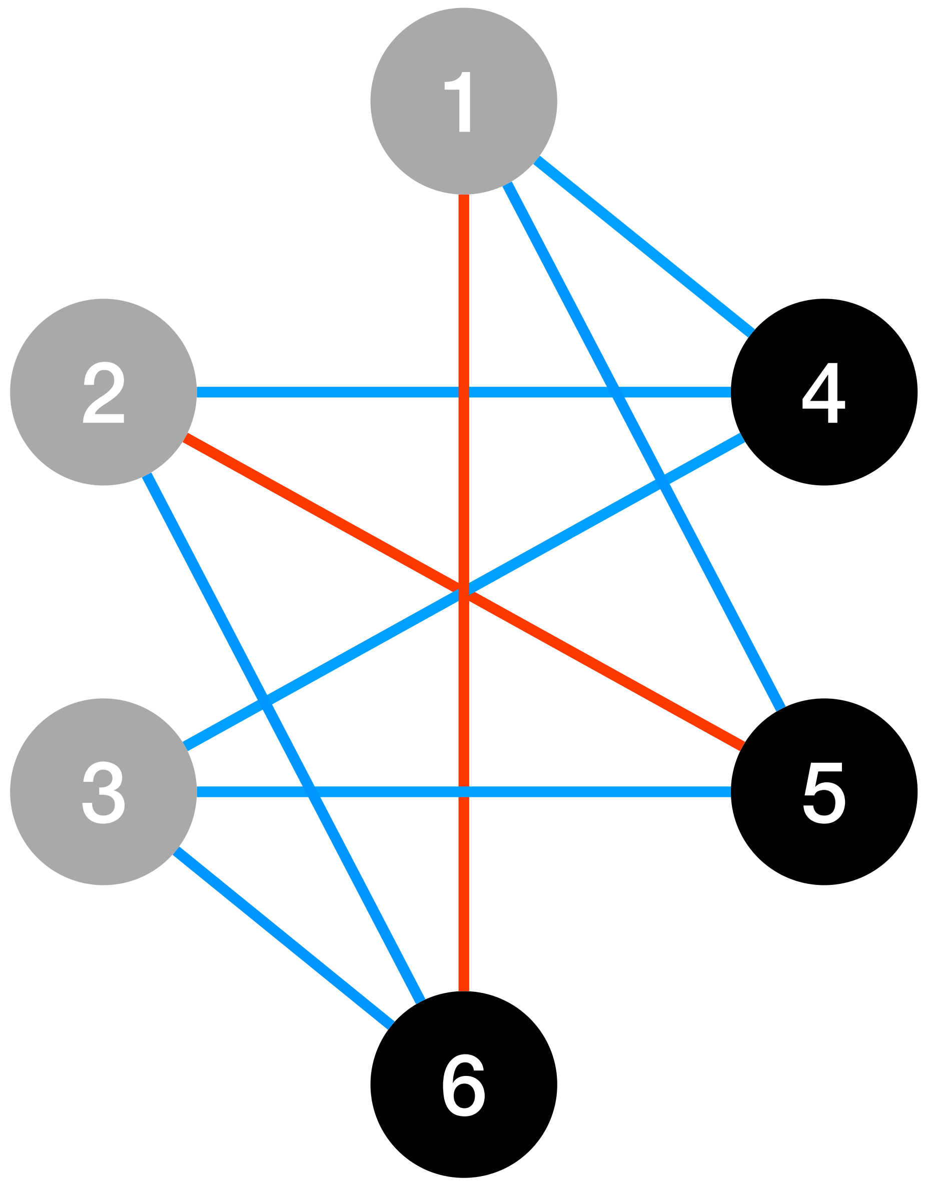


Fix C =  .

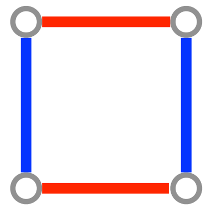
Input: an edge-colored complete graph G

Step 1. Use the color-coding technique to obtain a complete bipartite subgraph.

Reduction from Detecting 4-Cycles to GBMM



A	4	5	6
1	T	T	F
2	T	F	T
3	T	T	T

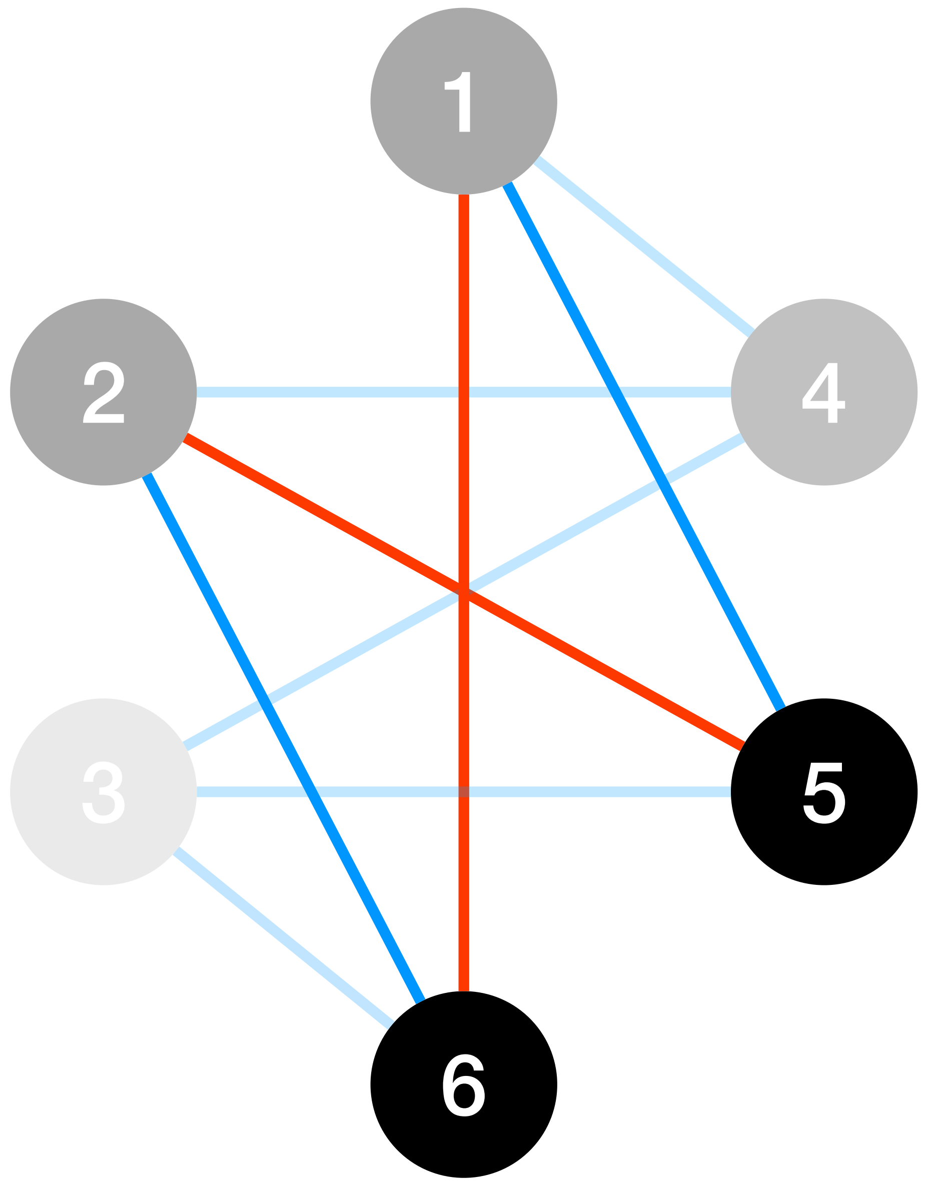
Fix $C =$ .

Input: an edge-colored complete graph G

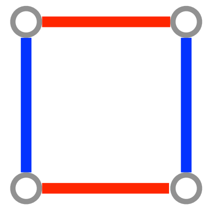
Step 1. Use the color-coding technique to obtain a complete bipartite subgraph.

Step 2. Compute an adjacency matrix A with rows corresponding to one part and columns corresponding to the other. Let $B = A^T$.

Reduction from Detecting 4-Cycles to GBMM



A	4	5	6
1	T	T	F
2	T	F	T
3	T	T	T

Fix $C =$  .

Input: an edge-colored complete graph G

Step 1. Use the color-coding technique to obtain a complete bipartite subgraph.

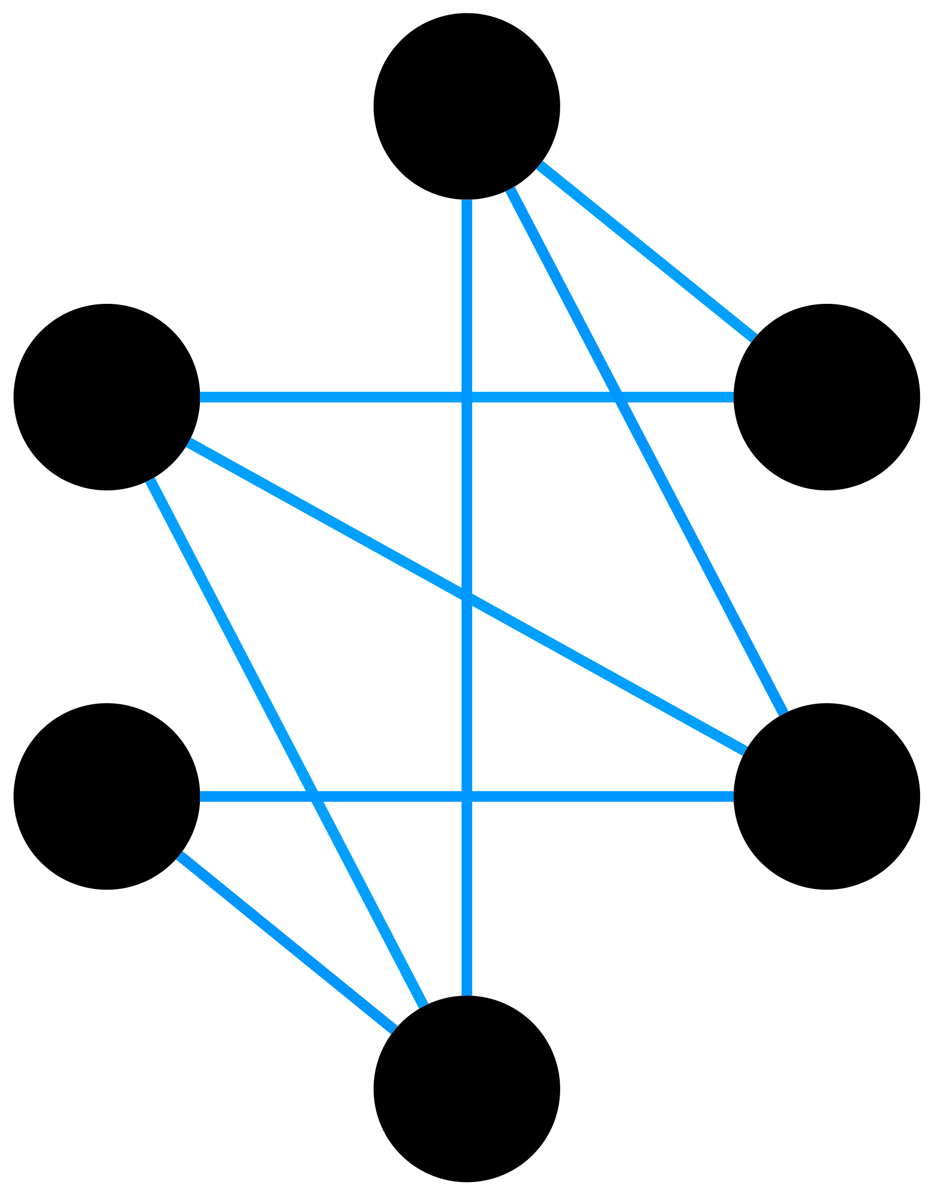
Step 2. Compute an adjacency matrix A with rows corresponding to one part and columns corresponding to the other. Let $B = A^T$.

Step 3. Solve GBMM for $A\overline{B}$ & $\overline{A}B$.

Application II

Detecting Designated Four-Node Induced Subgraphs

Problem Definition



Fix a 4-node graph H.

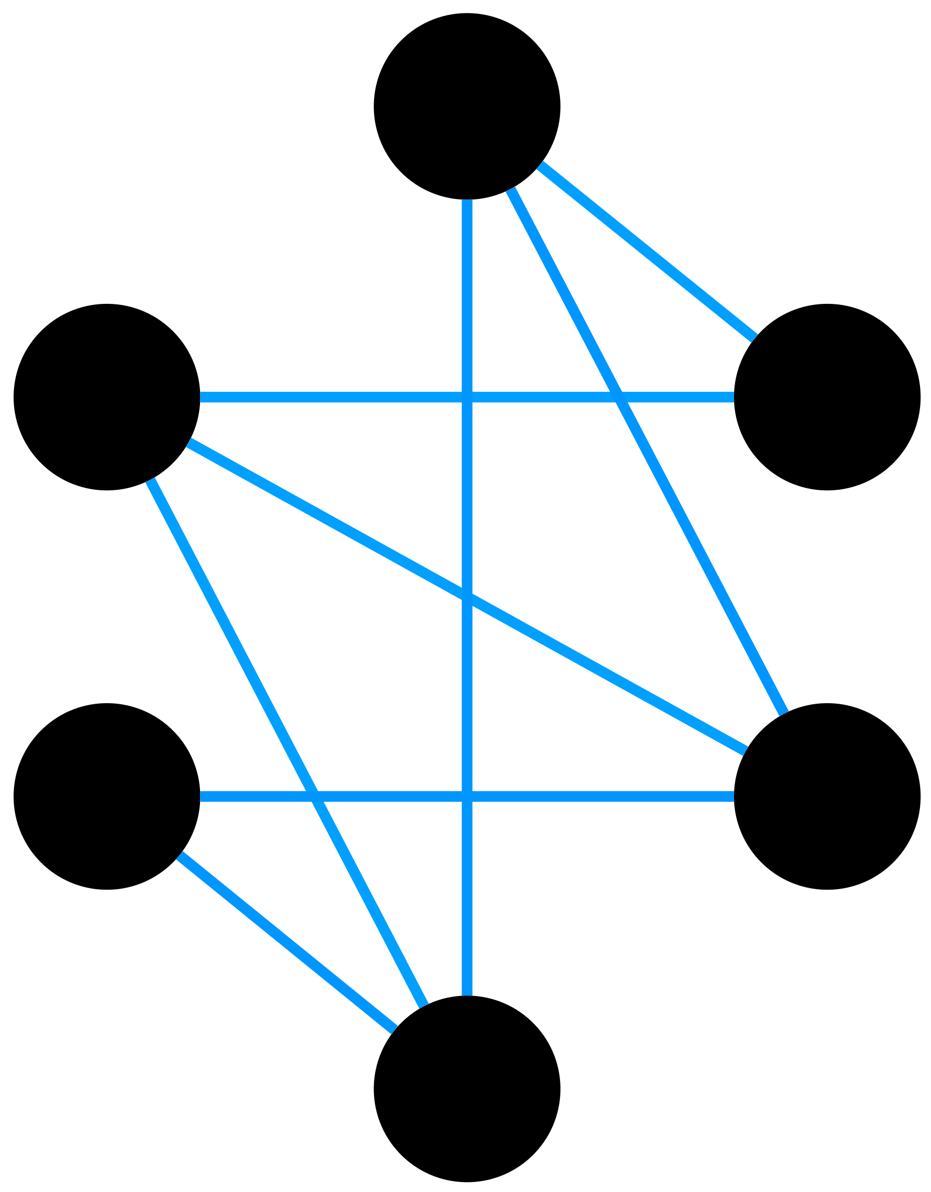
Input: a triangle-free undirected simple graph G.

Output:

"Yes", G contains H as an induced subgraph;

"No", otherwise.

Problem Definition



For each H, with one exception that $H = P_4$, the known best algorithm that solves our problem for general G needs triangle time. [WWWY'15]

Fix a 4-node graph H.

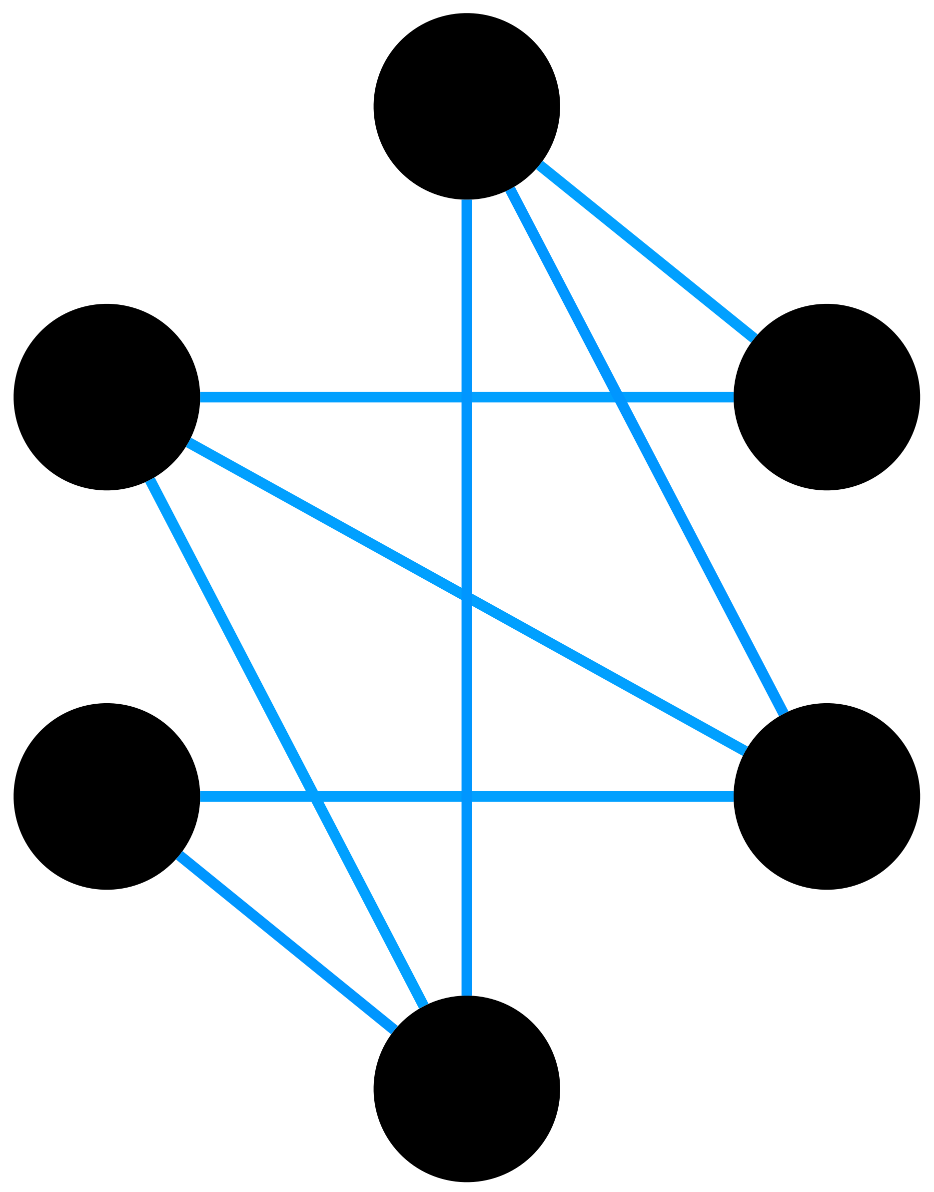
Input: a triangle-free undirected simple graph G.

Output:

"Yes", G contains H as an induced subgraph;

"No", otherwise.

Our Result

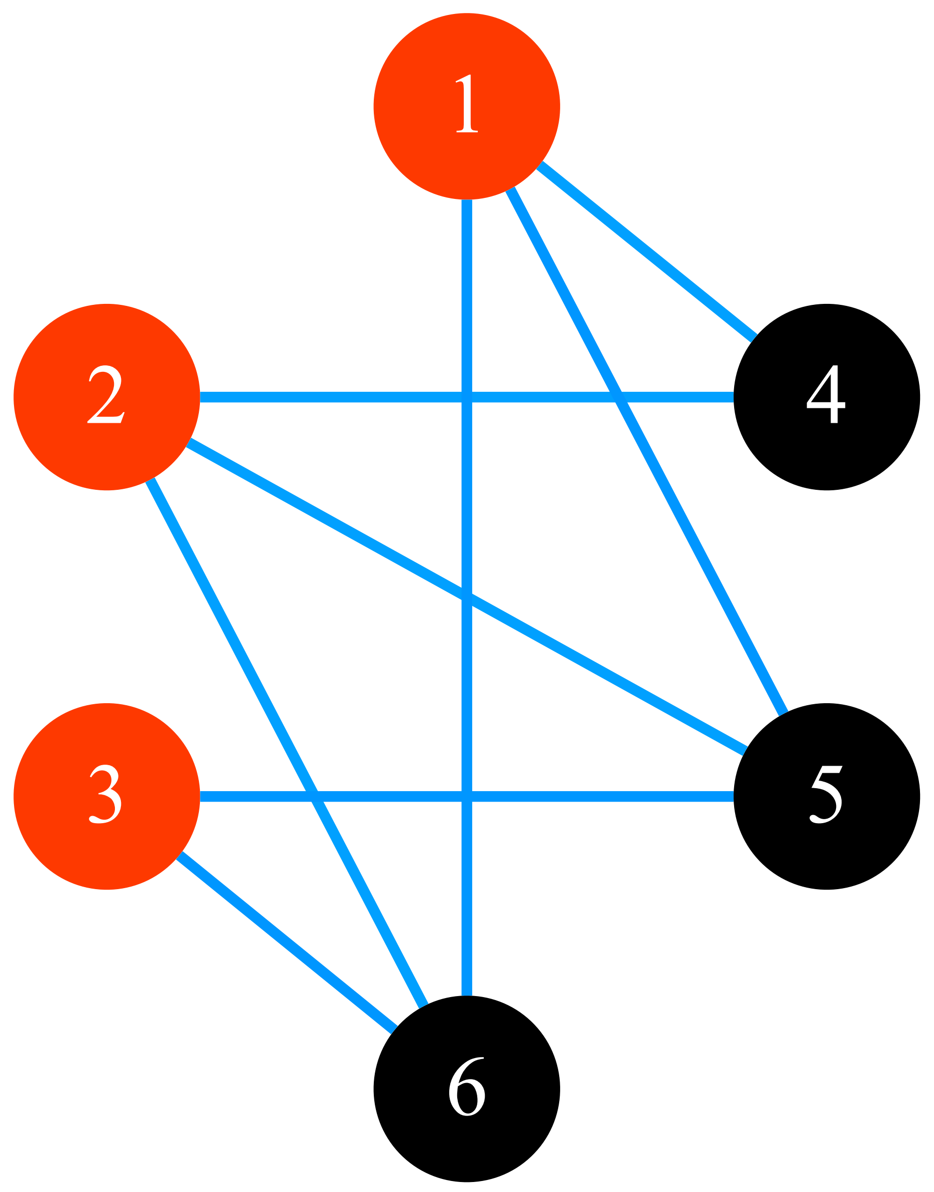


Fix a 4-node graph H.

Input: a triangle-free undirected simple graph G.

For each H, detecting H for triangle-free graphs can be done in randomized $O\left(n^2\right)$ time.

Our Algorithm for a Simple Case: $H = P_4$

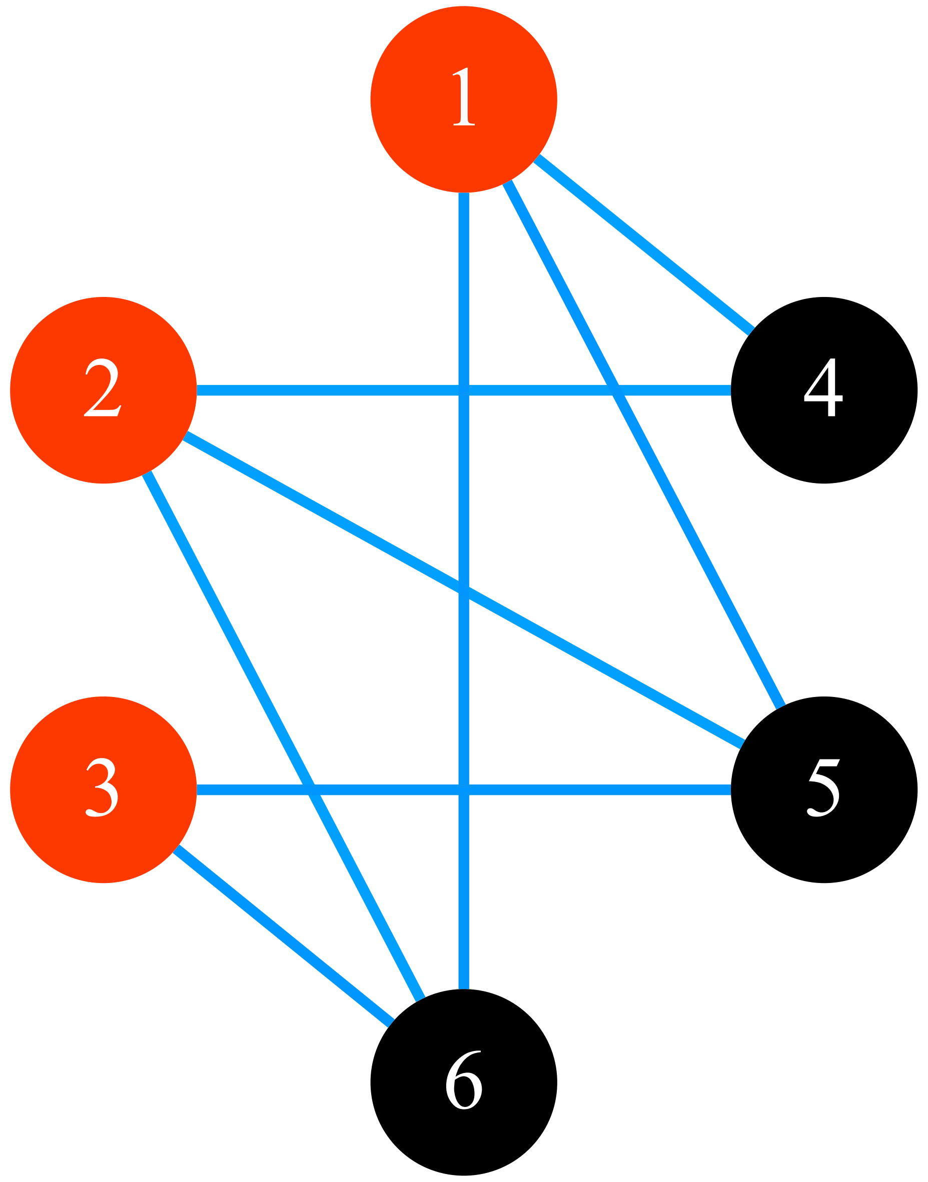


Fix $H = P_4$.

Input: a triangle-free undirected simple graph G.

Step 1. Use the color-coding technique to obtain a bipartite subgraph.

Our Algorithm for a Simple Case: $H = P_4$



A	4	5	6
1	T	T	T
2	T	T	T
3	F	T	T

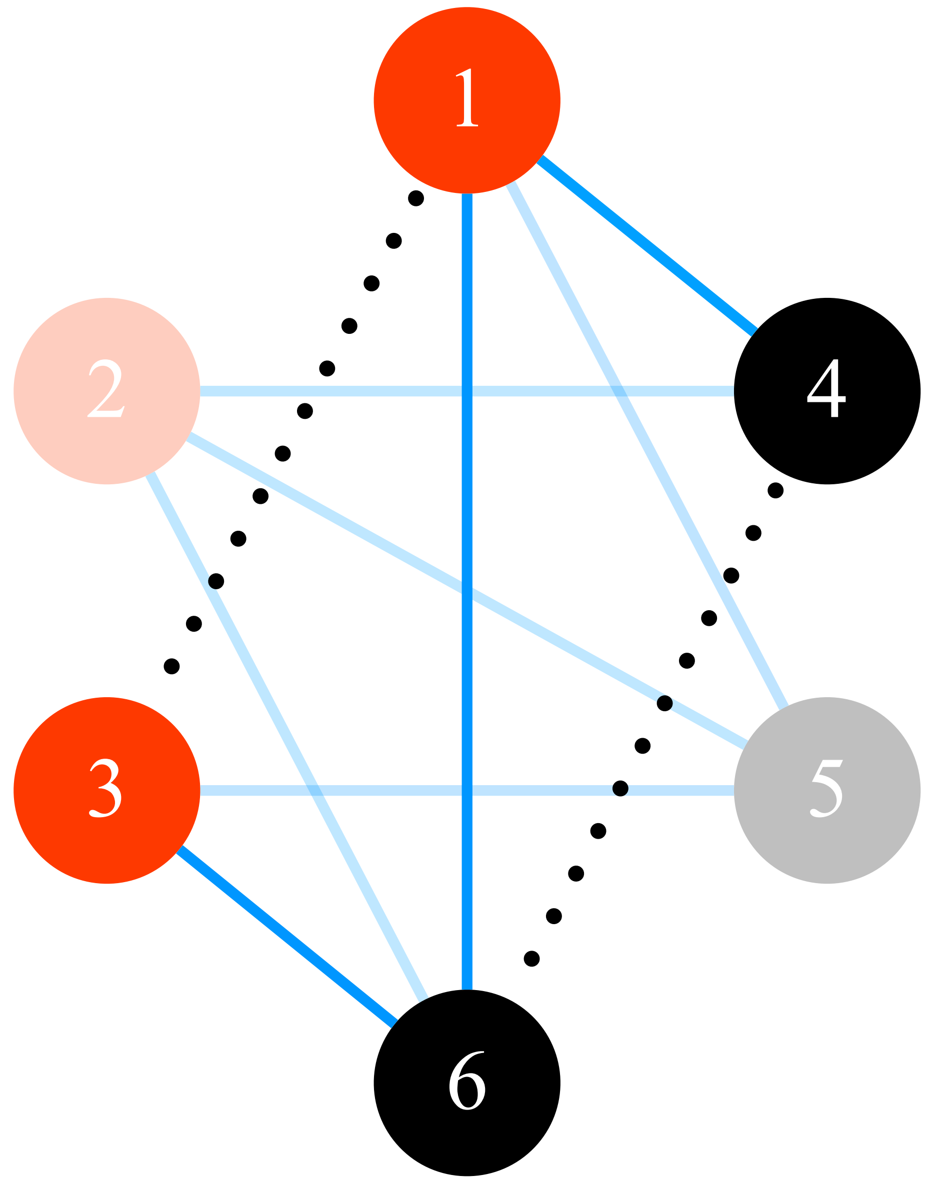
Fix $H = P_4$.

Input: a triangle-free undirected simple graph G.

Step 1. Use the color-coding technique to obtain a bipartite subgraph.

Step 2. Compute an adjacency matrix A with rows corresponding to one part and columns corresponding to the other. Let $B = A^T$.

Our Algorithm for a Simple Case: $H = P_4$



A	4	5	6
1	T	T	T
2	T	T	T
3	F	T	T

Fix $H = P_4$.

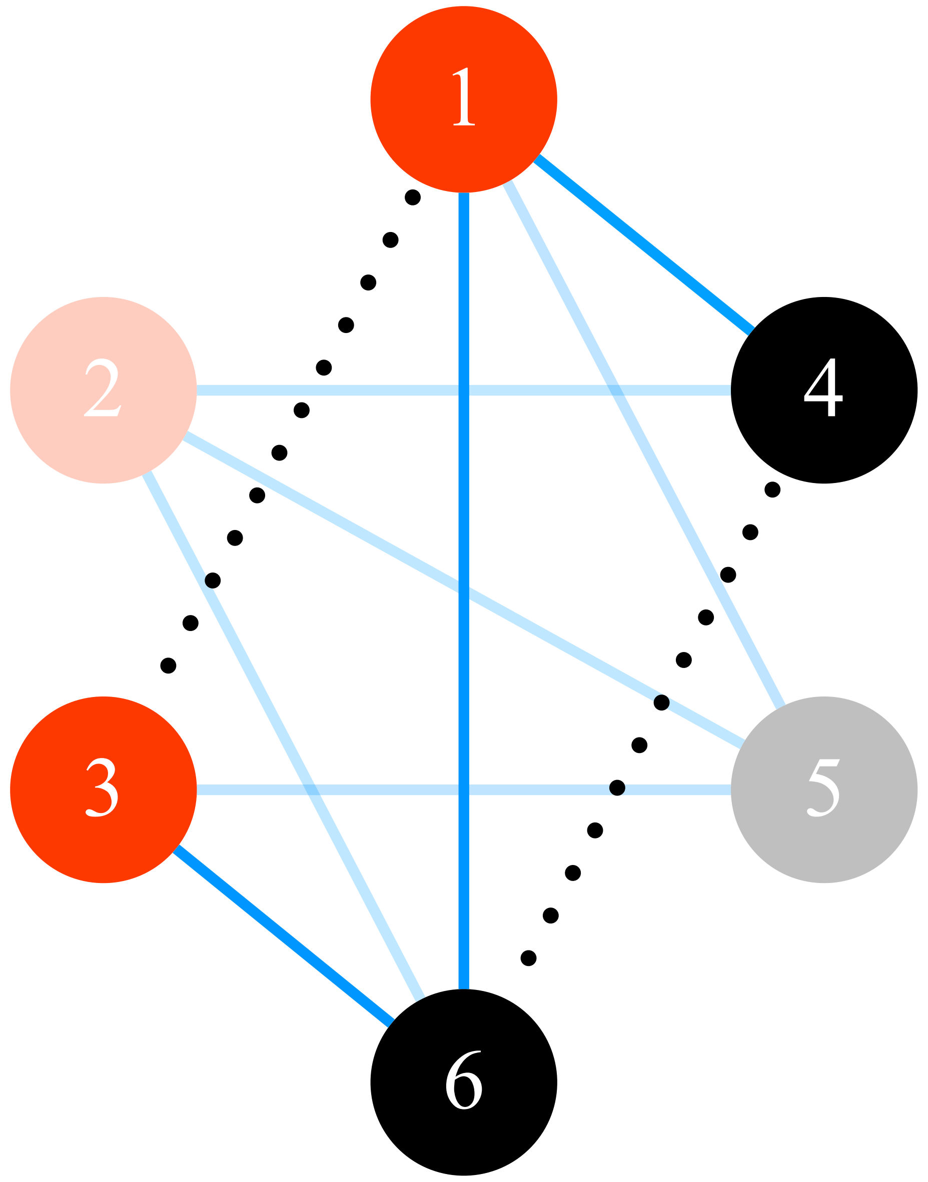
Input: a triangle-free undirected simple graph G.

Step 1. Use the color-coding technique to obtain a bipartite subgraph.

Step 2. Compute an adjacency matrix A with rows corresponding to one part and columns corresponding to the other. Let $B = A^T$.

Step 3. Solve GBMM for $A\overline{B}$ & AB.

Our Algorithm for a Simple Case: $H = P_4$



Fix $H = P_4$.

Input: a triangle-free undirected simple graph G.

Algorithm 2: Computing a factorizing permutation of a cograph

Input: A graph $G = (V, E)$ and an empty stack Q of vertices

Output: A permutation of V that is a factorizing permutation if G is a cograph

begin

$\mathscr{P} = [V]$

1 Choose an arbitrary vertex x of G as $Origin$

if *$Origin$ is an isolated vertex or a universal vertex* **then**

2 recurse on $G[V \setminus \{Origin\}]$

while *there exist some non-singleton parts* **do**

if *$\mathscr{C}_{Origin}$ is not a singleton* **then**

3 Use rule **??** on $\mathscr{C}_{Origin}$ with $Origin$ as pivot

Set $\overline{N}(Origin) \cap \mathscr{C}_{Origin}$ and $N(Origin) \cap \mathscr{C}_{Origin}$ as unused parts

while *there exist unused parts* **do**

Pick an arbitrary unused part $\mathscr{C}$ and an arbitrary vertex $y \in \mathscr{C}$

Set y as the pivot of $\mathscr{C}$

Refine the parts $\mathscr{C}' \neq \mathscr{C}$ of $\mathscr{P}$ with rule **??** using the pivot set $N(y)$

Mark $\mathscr{C}$ as used and the new created subparts without pivot as unused

Let z_l and z_r be the pivots of the nearest non singleton parts to $Origin$ respectively on its left and on its right

4 **if** *z_l is adjacent to z_r* **then** $Origin \leftarrow z_l$ **else** $Origin \leftarrow z_r$

return $\mathscr{P}$

end

Algorithm 5: Recognition test

Input: Let $\sigma = x_1, \ldots, x_n$ be a permutation of the vertex set of a graph G, σ is represented as a doubly linked list.

Output: σ a list of vertices

begin

Let x_0 and x_{n+1} be added to σ (these vertices are dummies which are not twins with any other vertex)

Let z be the current vertex, initially $z \leftarrow x_1$

Let $succ(z)$ (resp. $prec(z)$) be the vertex following (resp. preceding z) in σ

while $z \neq x_{n+1}$ **do**

if *z and $prec(z)$ are twins (true or false) in $G(\sigma)$* **then**

remove $prec(z)$ from σ

else

if *z and $succ(z)$ are twins (true or false) in $G(\sigma)$* **then**

$z \leftarrow succ(z)$

remove $prec(z)$ from σ

else $z \leftarrow succ(z)$

if $|\sigma - \{x_0, x_{n+1}\}| = 1$ **then** **return** G is a cograph **else return** $G(\sigma)$ contains a P_4

end

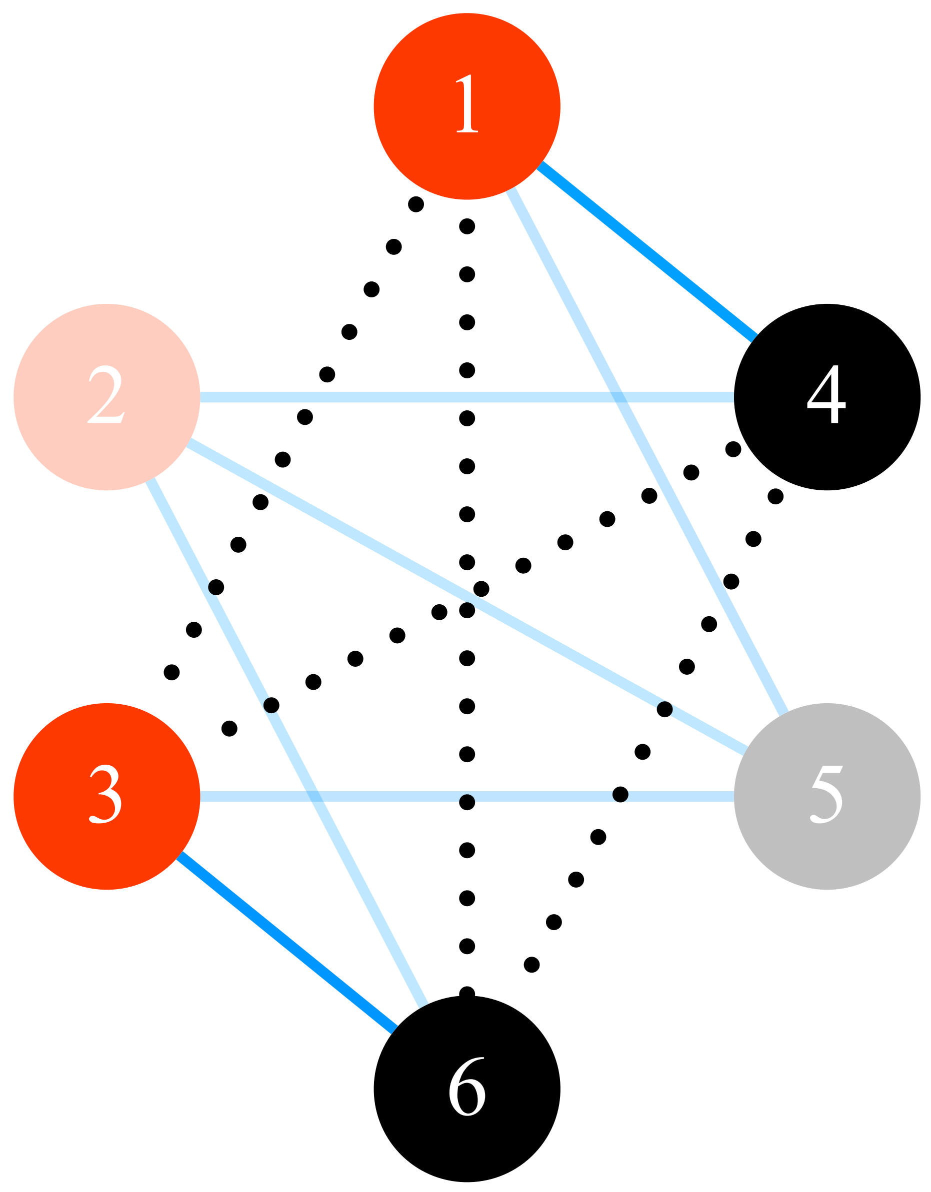
[HP’05]

Deciding whether a general graph contains P_4 as an induced subgraph is equivalent to recognizing cographs.

The implementation of the recognition algorithm is complicated. The simplest one [HP’05] still is lengthy.

Our algorithm is a very simple alternative for triangle-free graph G.

Our Algorithm for the Most Complicated Case: $H = 2K_2$



Fix $H = 2K_2$.

Input: a triangle-free undirected simple graph G.

We need a reduction to 3 different instances of GBMM.

Details are omitted in this talk.

Sketch of Our Deterministic Algorithms

For S = $\{P_1, P_2, P_3\}$

Step 1. Given the input matrices A and B, define an implication graph G_I as a sequence of incremental edge updates.

Step 2. Constructing G_I needs $O(n^3)$ time by the known best combinatorial algorithm [ACIM'99]. We show, however, that identifying all components in G_I after each incremental update can be done in $O(n^2)$ time.

Step 3. We use the information of the components in the dynamic G_I to avoid re-computing the same procedure incurred during the computation of P_1 & P_2 & P_3. Our algorithm runs in deterministic $O(n^2)$ time.

For other S, GBMM can be solved by a simplified variant of our above algorithm.

Recap

Thank you! Any questions?

$$P_1 = A \times B$$

$$P_2 = A \times \bar{B}$$

$$P_3 = \bar{A} \times B$$

$$P_4 = \bar{A} \times \bar{B}$$

Input:

(1) two n by n Boolean matrices A and B

(2) a non-empty subset S of $\{P_1, P_2, P_3, P_4\}$ (given as symbols rather than the explicit matrices)

<u>Our Main Theorem</u>.

Verifying whether the product of GBMM contain only False entries can be done in deterministic $O\left(n^2\right)$ time.