Invitation to fixed-parameter algorithms

Jisu Jeong (Dept. of Math, KAIST)

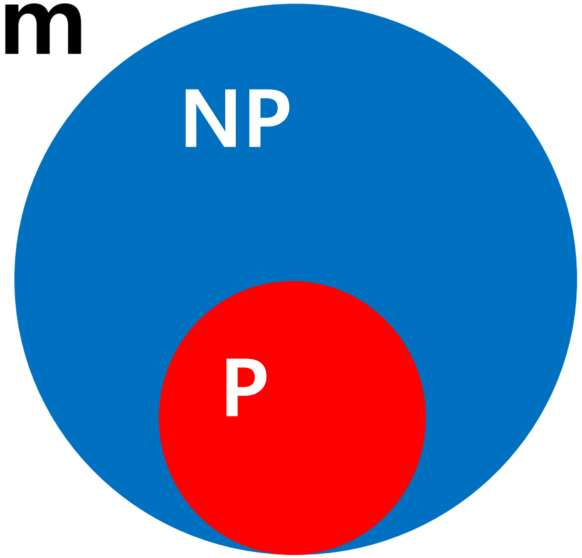
joint work with
Sigve Hortemo Sæther and Jan Arne Telle (Univ. of Bergen, Norway),
Eun Jung Kim (CNRS / Univ. Paris-Dauphine) and Sang-il Oum (KAIST)

FWAC16, Yonsei University
Nov. 11, 2016

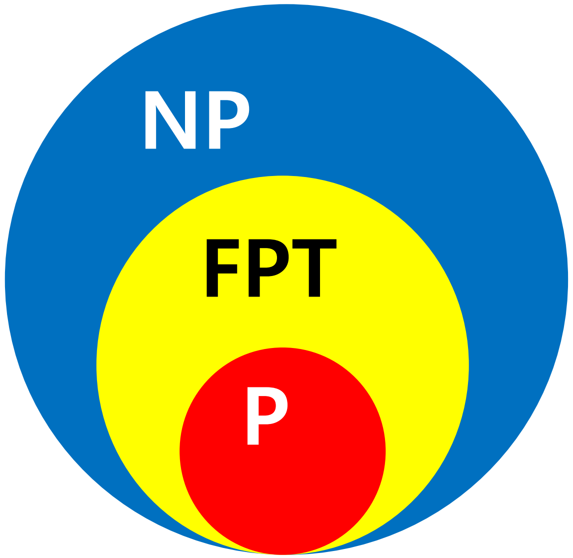
Fixed-Parameter (Tractable) Algorithm

- **P = solve in time poly(n)**
- **NP = verify in time poly(n)**
- One of the Millennium Prize Problems

P⊂NP or P=NP



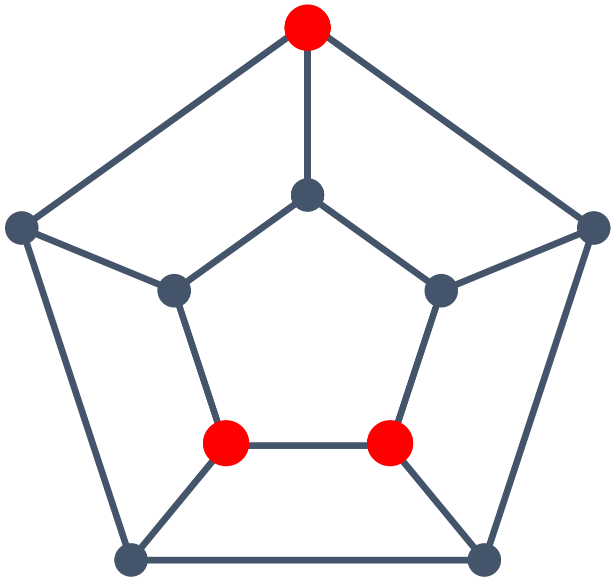
- **FPT = solve in time $f(k) \cdot$ poly(n)**
- k is a parameter



Examples

Dominating Set Problem

Input : a graph G
Question : what is the minimum size of a dominating set in G?



- Dominating set = a set of vertices that dominates all vertices
- Dominating set problem is NP-hard

Examples

Dominating Set Problem

Input : a graph G
Question : what is the minimum size of a dominating set in G?

Fixed-Parameter Algorithm for k-Dominating Set Problem

Input : a planar graph G, an integer k
Parameter : an integer k
Outputs : YES if a dominating set of size k exists
NO otherwise
Time : $2^{O(\sqrt{k})}n$

Examples

Planar Vertex Deletion

Input : a graph G, an integer k
Parameter : an integer k
Outputs : YES if a vertex subset X of size k
such that $G - X$ is planar

Eulerian Deletion

Input : a graph G, an integer k
Parameter : an integer k
Outputs : YES if an edge subset Y of size k
such that $G - Y$ is Eulerian

Examples

Minimum Dominating Set Problem

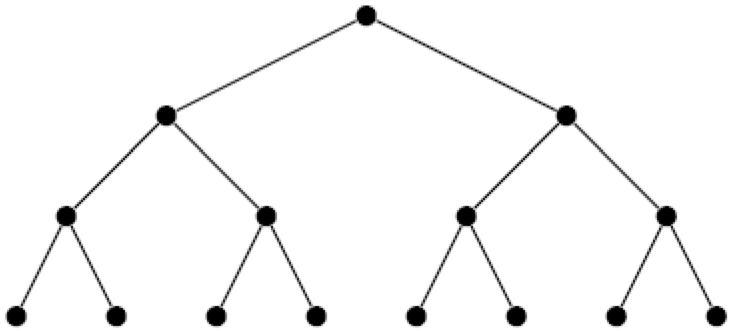
Input : a graph G having tree-width k
Parameter : an integer k
Outputs : the minimum size of a dominating set in G

Theorem (van Rooij, Bodlaender, Rossmanith 2009)

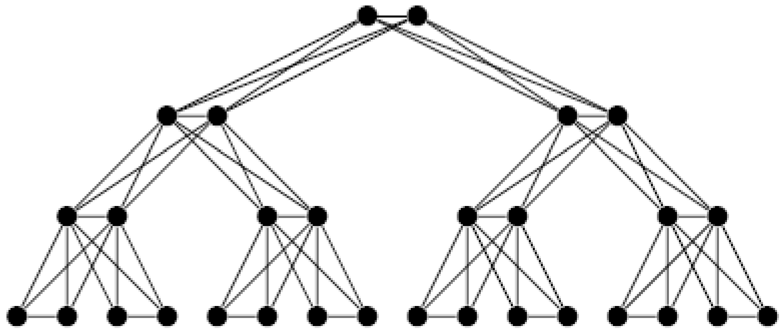
Minimum Dominating Set Problem can be solved in time $O(3^k) \cdot \mathrm{poly}(n)$ if a graph has tree-width k.

Tree-width

- tree-width (Halin 1976, Robertson and Seymour 1984)
  A measure of how "tree-like" the graph is.



tree

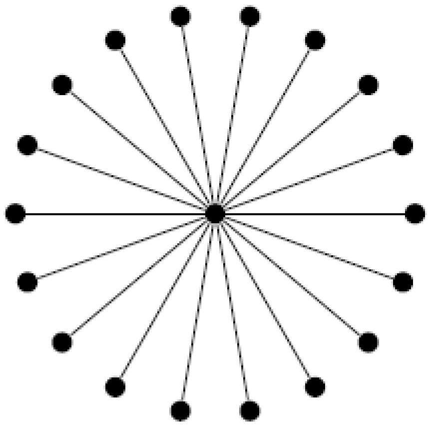


tree-like

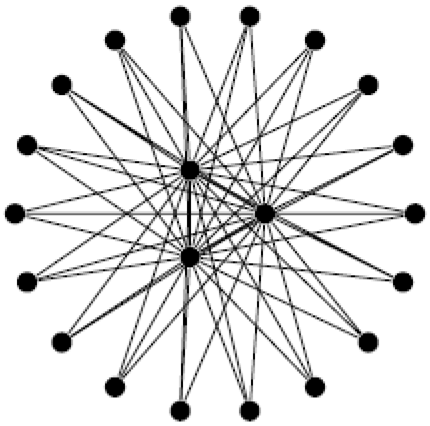
Figures from http://fptschool.mimuw.edu.pl/slides/lec6.pdf

Tree-width

- tree-width (Halin 1976, Robertson and Seymour 1984)
  A measure of how "tree-like" the graph is.



tree

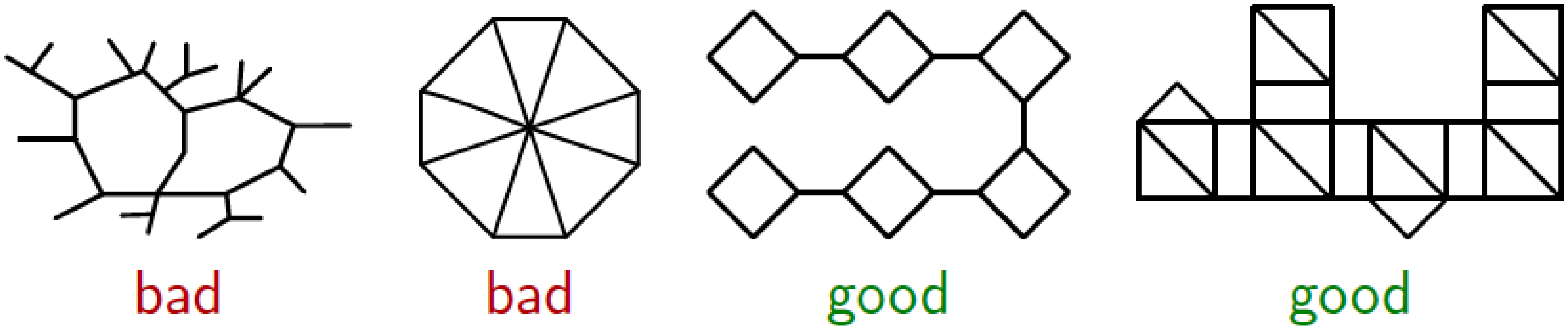


tree-like

Figures from http://fptschool.mimuw.edu.pl/slides/lec6.pdf

Tree-width

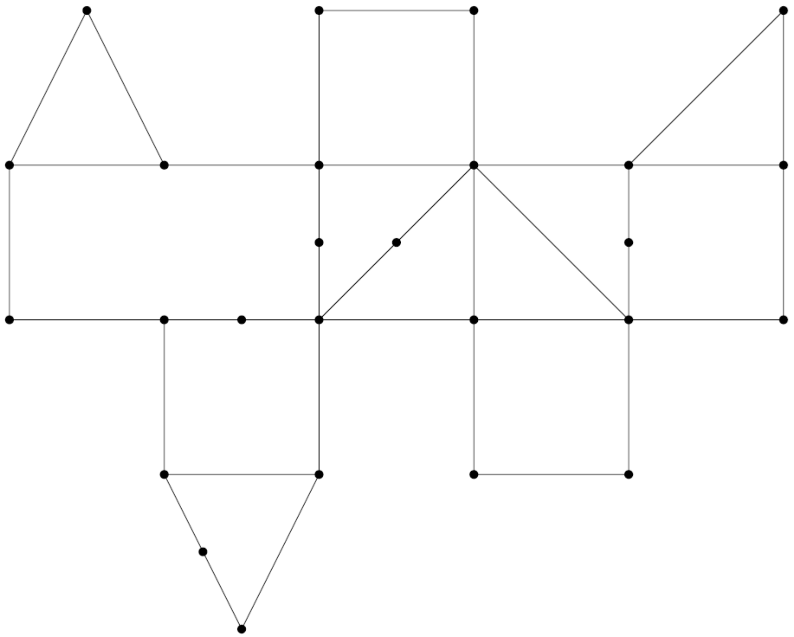
- tree-width (Halin 1976, Robertson and Seymour 1984)
  A measure of how "tree-like" the graph is.

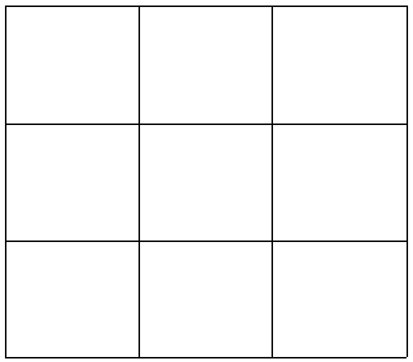


Figures from http://fptschool.mimuw.edu.pl/slides/lec6.pdf

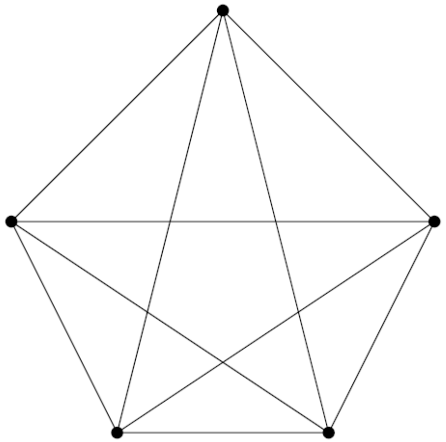
Examples

- tree-width $\leq 1 \Leftrightarrow$ a forest $\Leftrightarrow$ no cycle
- tree-width $\leq 2 \Leftrightarrow$ a series-parallel graph
  $\Leftrightarrow$ no K_4 minor
- The tree-width of a $k \times k$ grid is k.
- The tree-width of K_n is $n - 1$.





4×4 grid



K_5

by Fedor V. Fomin
(http://fptschool.mimuw.edu.pl/slides/lec6.pdf)

Why treewidth?

Very general idea in science: large structures can be understood by breaking them into small pieces

In Computer Science: divide and conquer; dynamic programming

In Graph Algorithms: Exploiting small separators

Lower bounds for tree-width

Using tree-width

Theorem (van Rooij, Bodlaender, Rossmanith 2009)

Minimum Dominating Set Problem can be solved in time $O^*(3^t)$ when a graph has tree-width t.

Theorem (Lokshtanov, Marx, Saurabh 2011)

Minimum Dominating Set Problem cannot be solved in time $O^*((3-\varepsilon)^t)$ where t is the tree-width of the given graph.

Maximum Matching width

Theorem (Vatshelle 2012)

For every graph G,

$$mmw(G) \leq tw(G) + 1 \leq 3\, mmw(G).$$

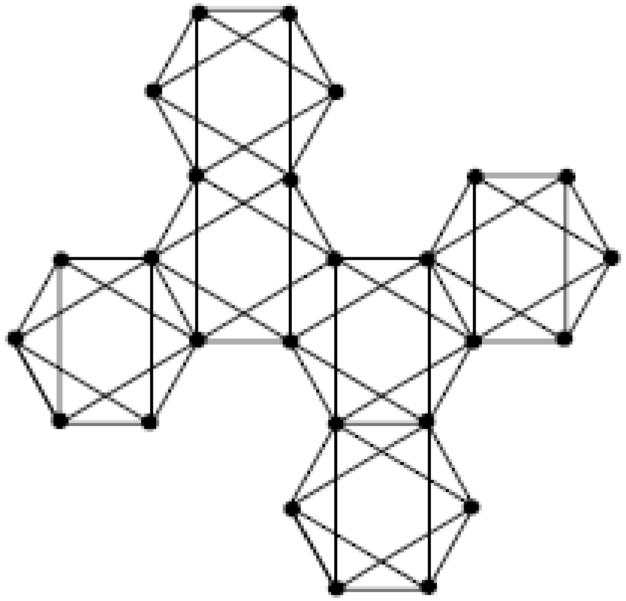
A graph G has bounded tree-width if and only if G has bounded maximum matching width.

Why Maximum Matching width?

Using maximum matching width

Theorem (J., Sæther, Telle IPEC2015)
Minimum Dominating Set Problem can be solved in time $O^*(8^m)$ when a graph has maximum matching width m.

Why Maximum Matching width?



Using tree-width: $O^*(3^t)$
Using mm-width: $O^*(8^m)$

Our algorithm is faster when $8^m < 3^t$, that is,

$$1.893\, mmw(G) < tw(G).$$

Note that for every graph G,

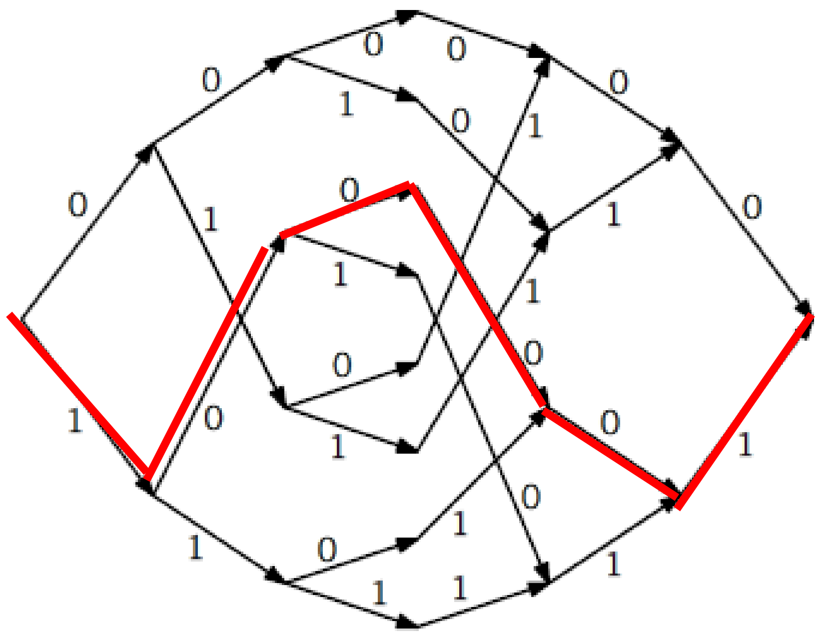
$$mmw(G) \leq tw(G) + 1 \leq 3\, mmw(G).$$

Consider the linear code C that is generated by (100001), (010100), and (001010).

The generator matrix is $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$.

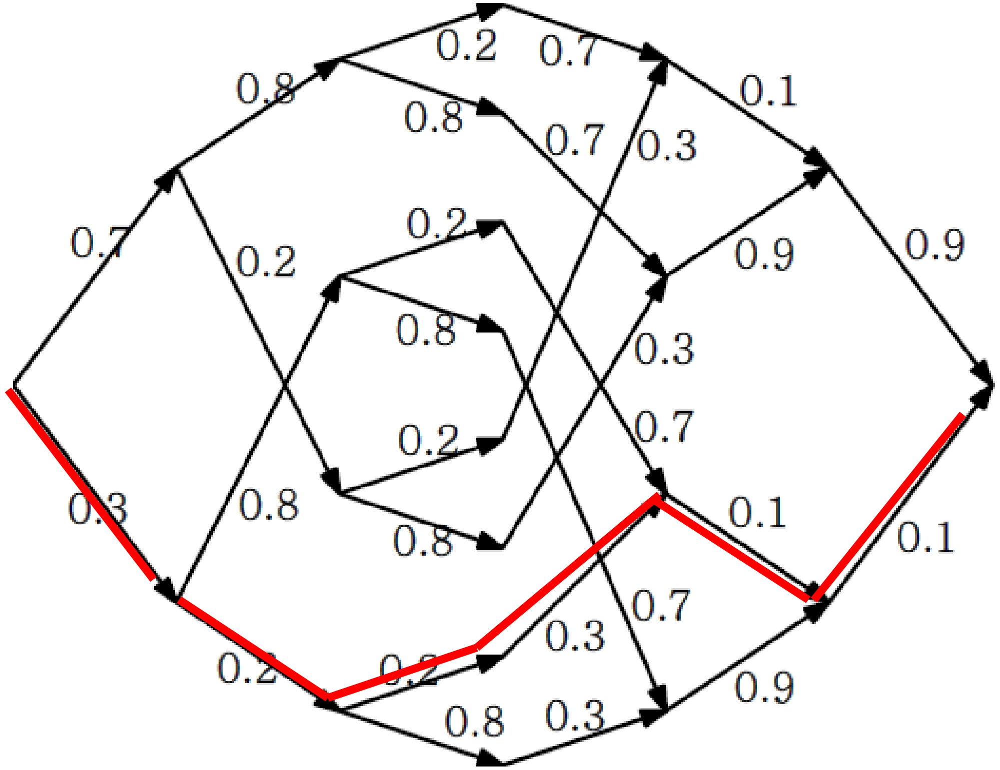
Codewords

{(000000), **(100001)**, (010100), (001010), (110101), (101011), (011110), (111111)}



trellis

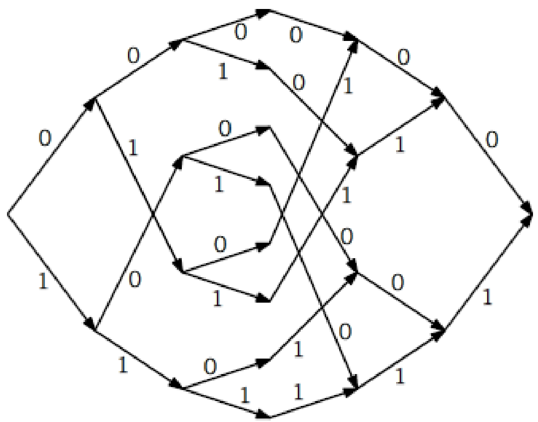
Decode using

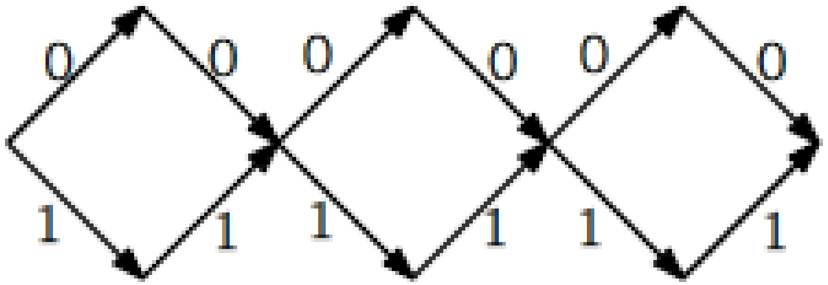


(110101)

(0.7 0.8 0.2 0.7 0.1 0.9)

{(000000), (100001), (010100), (001010), (110101), (101011), (011110), (111111)}





Want to make a better (thinner) trellis

Permute the columns of

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\Downarrow \quad \pi = (1)(4)(5)(2,3,6)$$

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Our Results (J., Kim, Oum SODA2016)

Constructive algorithm for path-width of vectors (SODA2016)

Input : n vectors over F, an integer k

Output : a permutation $v_1, v_2, \dots, v_n$ of n vectors satisfying that for all i,

$$\dim\langle v_1, v_2, \dots, v_i\rangle \cap \langle v_{i+1}, v_{i+2}, \dots, v_n\rangle \le k.$$

Time : FPT $= f(k) \cdot n^3$

$$\begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right) \\ 3 \end{array} \qquad \begin{array}{c} \begin{array}{cccccc} v_1 & v_6 & v_2 & v_4 & v_5 & v_3 \end{array} \\ \left(\begin{array}{ccc|ccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right) \\ 1 \end{array}$$

Actually, we proved a more general statement.

\`vectors' → \`subspaces' (vector = 1-dimensional subspace)

Constructive algorithm for path-width of subspaces

Input : n subspaces over F, an integer k

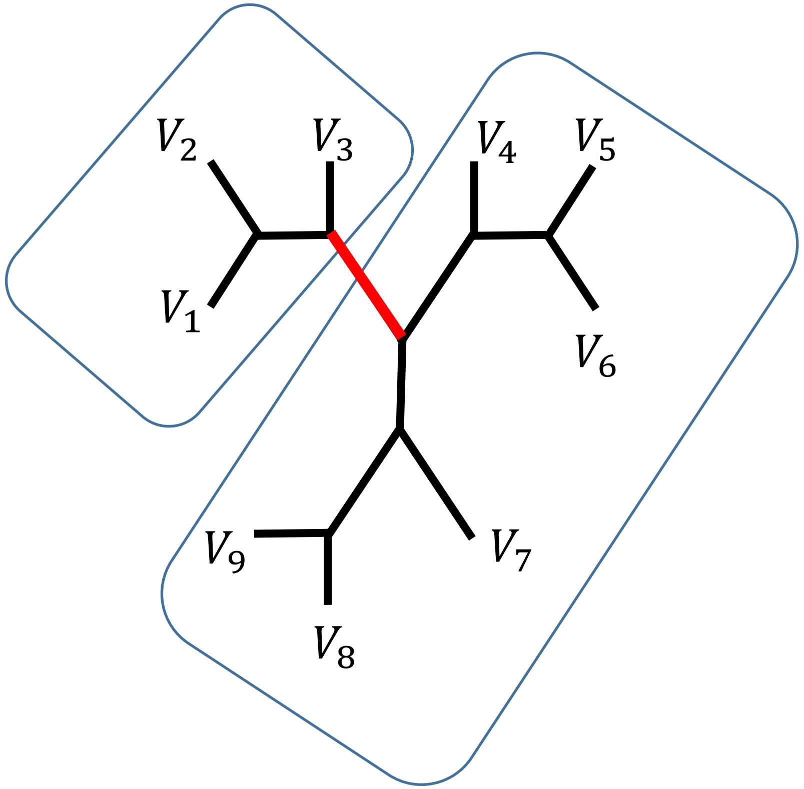
Output : a permutation $V_1, V_2, \dots, V_n$ of n subspaces satisfying that for all i,

$$\dim\langle V_1, V_2, \dots, V_i\rangle \cap \langle V_{i+1}, V_{i+2}, \dots, V_n\rangle \le k.$$

Time : FPT $= f(k) \cdot n^3$

Theorem (J., Kim, Oum 2016+)

Roughly speaking, we can extend our algorithm to the **tree-version**.



$\dim\langle V_1, V_2, V_3\rangle \cap \langle V_4, V_5, \ldots, V_9\rangle \leq k$

Thank you for listening

Proof ideas

1. Dynamic programming 2. Typical sequences 3. Subspace analysis (linear algebra)

Constructive algorithm for path-width of vectors (SODA16)

Input : n vectors over F, an integer k
Output : a permutation $v_1, v_2, \ldots, v_n$ of n vectors satisfying that for all i,

$$\dim\langle v_1, v_2, \ldots, v_i\rangle \cap \langle v_{i+1}, v_{i+2}, \ldots, v_n\rangle \leq k.$$

Time : FPT $= f(k) \cdot n^3$