数据挖掘导论

Introduction to Data Mining

Frequent Pattern Mining

数据智能实验室
DATA INTELLIGENCE LABORATORY
DI-LAB
浙江大学
Zhejiang University

Agenda

- Closed and Maximal Pattern Mining
- Frequent Subsequence Mining
- Frequent Subgraph Mining
- Summary

Closed Item Set

- **An item set is closed if 1) it is frequent 2) none of its superset has the same support**

transaction database

1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{b, c, e\}$
10: $\{a, d, e\}$

frequent item sets

0 items	1 item	2 items	3 items
$\emptyset$: 10	$\{a\}$: 7	$\{a, c\}$: 4	$\{a, c, d\}$: 3
	$\{b\}$: 3	$\{a, d\}$: 5	$\{a, c, e\}$: 3
	$\{c\}$: 7	$\{a, e\}$: 6	$\{a, d, e\}$: 4
	$\{d\}$: 6	$\{b, c\}$: 3	
	$\{e\}$: 7	$\{c, d\}$: 4	
		$\{c, e\}$: 4	
		$\{d, e\}$: 4	

Maximal Item Set

- **An item set is maximal if 1) it is frequent 2) none of its superset is frequent**

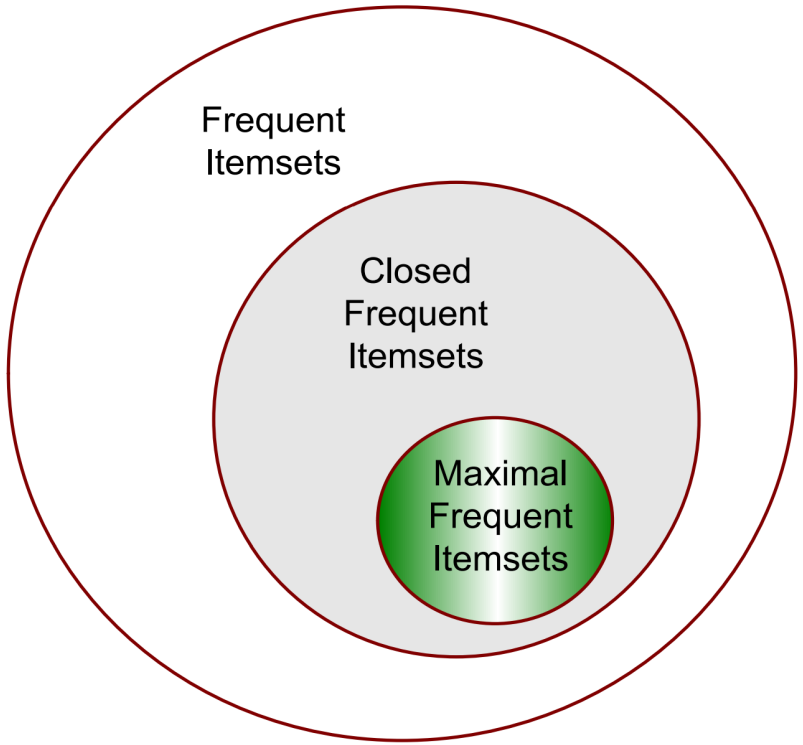
transaction database

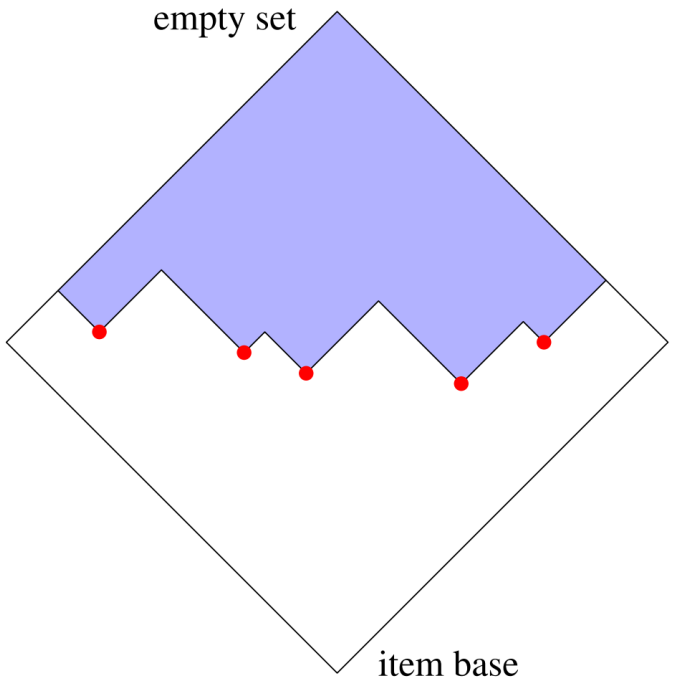
1: $\{a, d, e\}$
2: $\{b, c, d\}$
3: $\{a, c, e\}$
4: $\{a, c, d, e\}$
5: $\{a, e\}$
6: $\{a, c, d\}$
7: $\{b, c\}$
8: $\{a, c, d, e\}$
9: $\{b, c, e\}$
10: $\{a, d, e\}$

frequent item sets

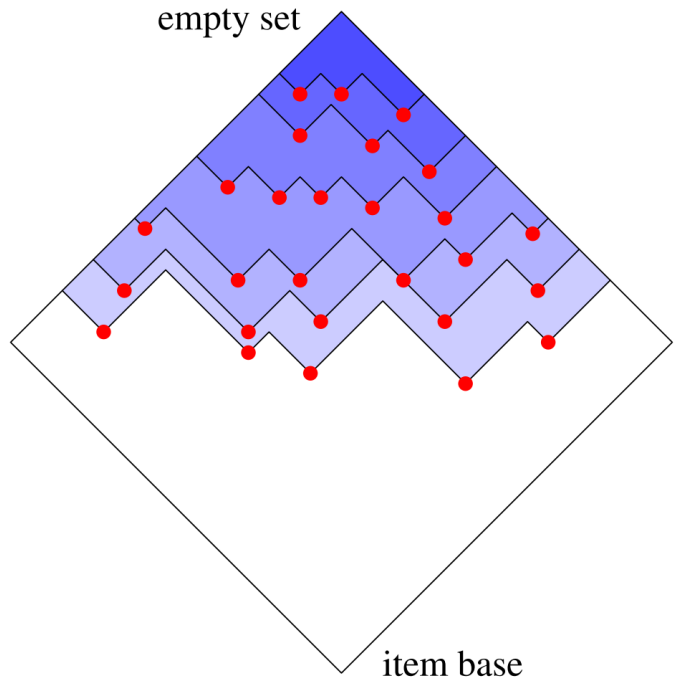
0 items	1 item	2 items	3 items
$\emptyset$: 10	$\{a\}$: 7	$\{a, c\}$: 4	$\{a, c, d\}$: 3
	$\{b\}$: 3	$\{a, d\}$: 5	$\{a, c, e\}$: 3
	$\{c\}$: 7	$\{a, e\}$: 6	$\{a, d, e\}$: 4
	$\{d\}$: 6	$\{b, c\}$: 3	
	$\{e\}$: 7	$\{c, d\}$: 4	
		$\{c, e\}$: 4	
		$\{d, e\}$: 4	

Maximal Pattern





maximal (frequent) item sets



closed (frequent) item sets

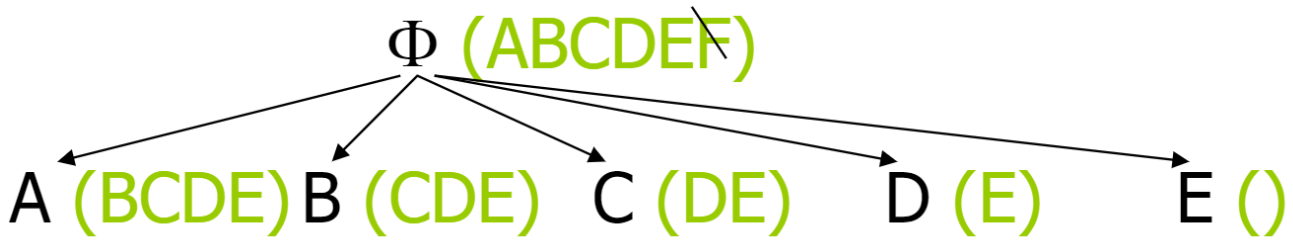
MaxMiner Algorithm

Construct the root node

- ✓ {F} is not frequent and it will not appear in tail(root)
- ✓ The root node has five child nodes

Tid	Items
10	A,B,C,D,E
20	B,C,D,E,
30	A,C,D,F

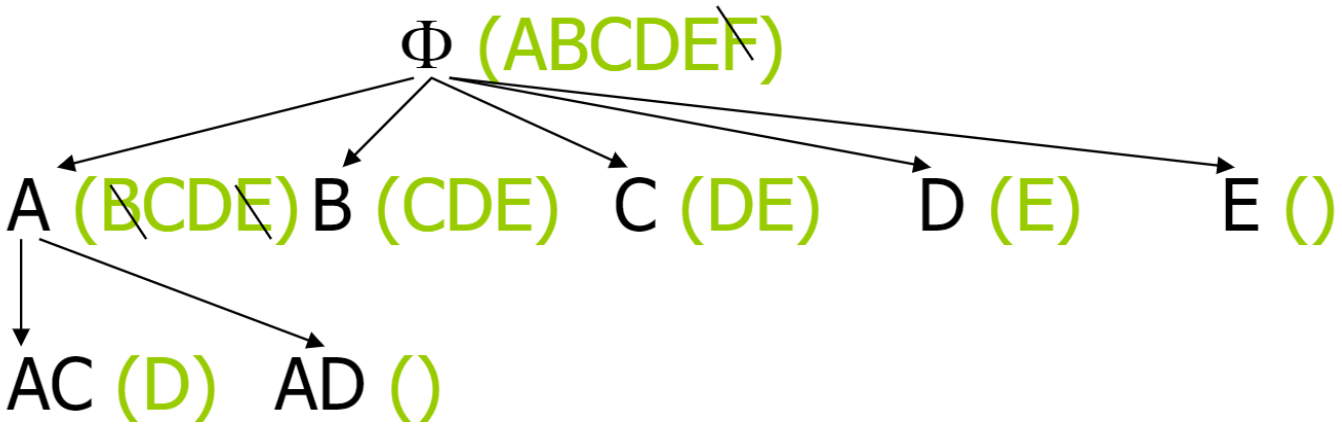
Min_sup=2



MaxMiner Algorithm

Access node A

- ✓ ABCDE is not frequent. We need to expand A
- ✓ AB and AE are not frequent. We only need to generate child nodes AC and AD



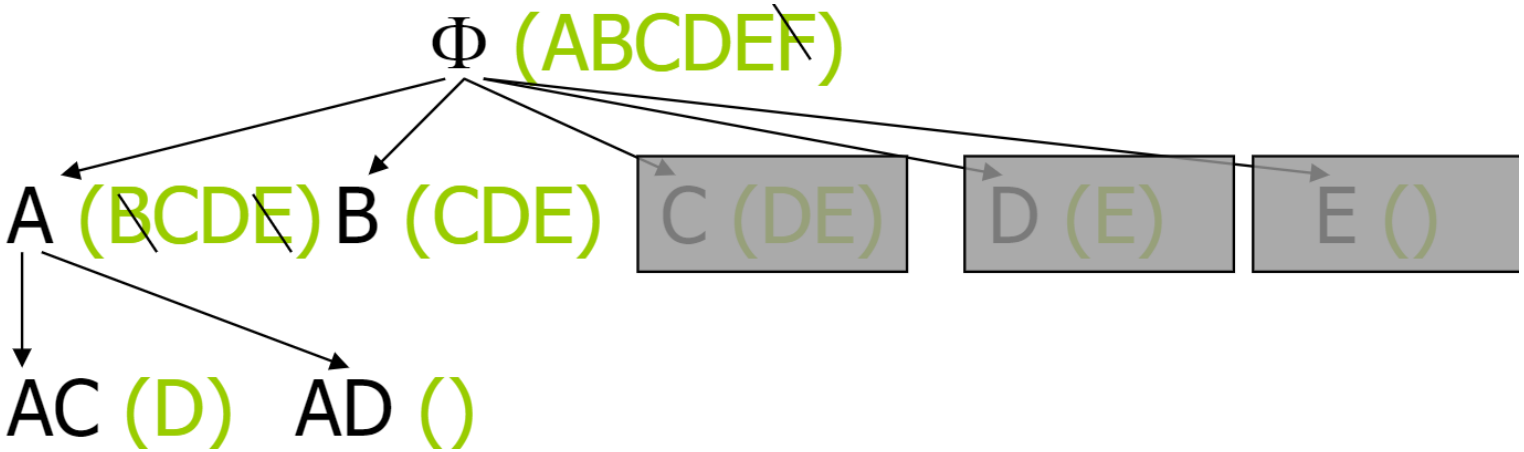
Node A

Items	Frequency
ABCDE	1
AB	1
AC	2
AD	2
AE	1

MaxMiner Algorithm

Access node B

- ✓ BCDE is frequent. Then, it is a maximum item set.
- ✓ Remove the subtree of node B
- ✓ Remove subtrees of nodes C, D and E because CDE, DE and E are subsets of BCDE



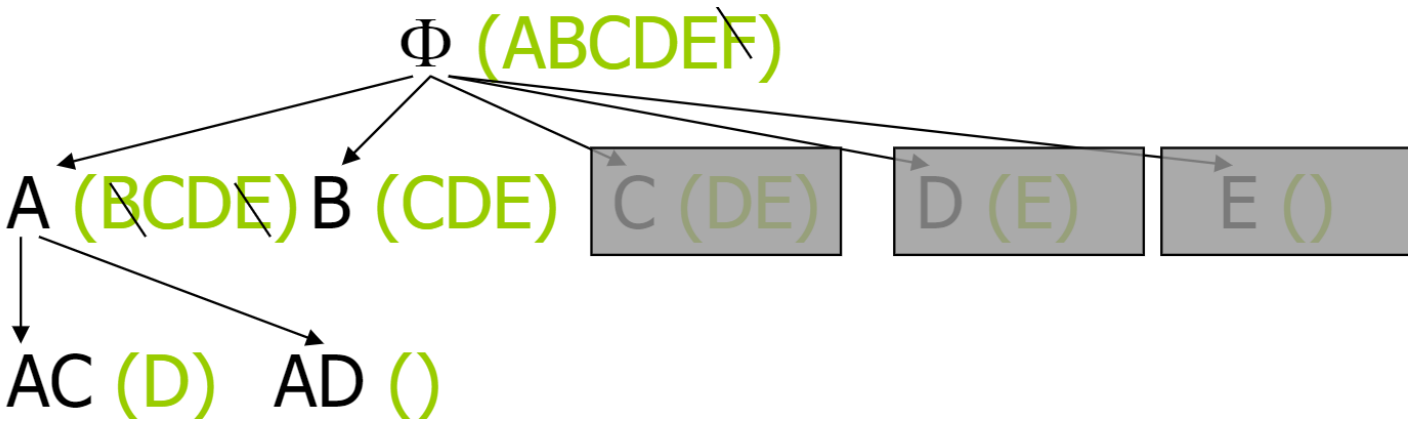
Node B

Items	Frequency
BCDE	2
BC	
BD	
BE	

MaxMiner Algorithm

Access node AC

- ✓ ACD is frequent. Then, it is also a maximum item set.
- ✓ Remove subtree of node AD because AD is a subset of ACD
- ✓ Final results: **BCDE and ACD**



Node AC

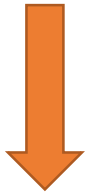
Items	Frequency
ACD	2

Agenda

- Closed and Maximal Pattern Mining
- Frequent Subsequence Mining
- Frequent Subgraph Mining
- Summary

Frequent Subsequence Mining

ID	S_i
1	<ACGCAG>
2	<AGCGAC>
3	<CGCACG>
4	<AGCACG>



Support=2

Frequent sequence set	Patterns
L1	A, C, G
L2	AC, CG, GC, CA, AG
L3	ACG, CGC, GCA, AGC, CAC
L4	CGCA, GCAC, CACG
L5	GCACG

Image source: https://www.dbs.ifi.lmu.de/Lehre/KDD/SS16/skript/3b_SeqPattMining.pdf

SuffixTree

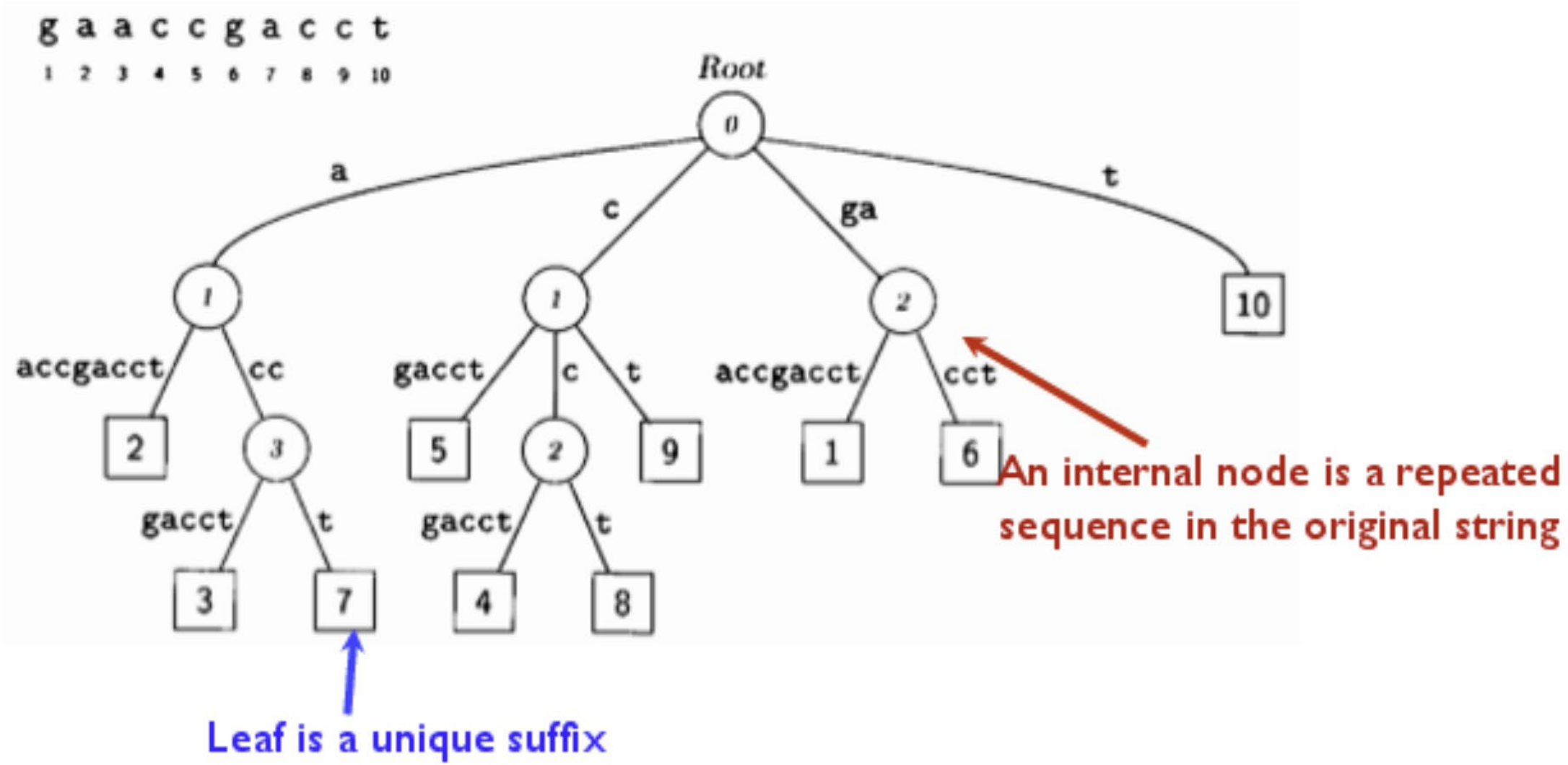


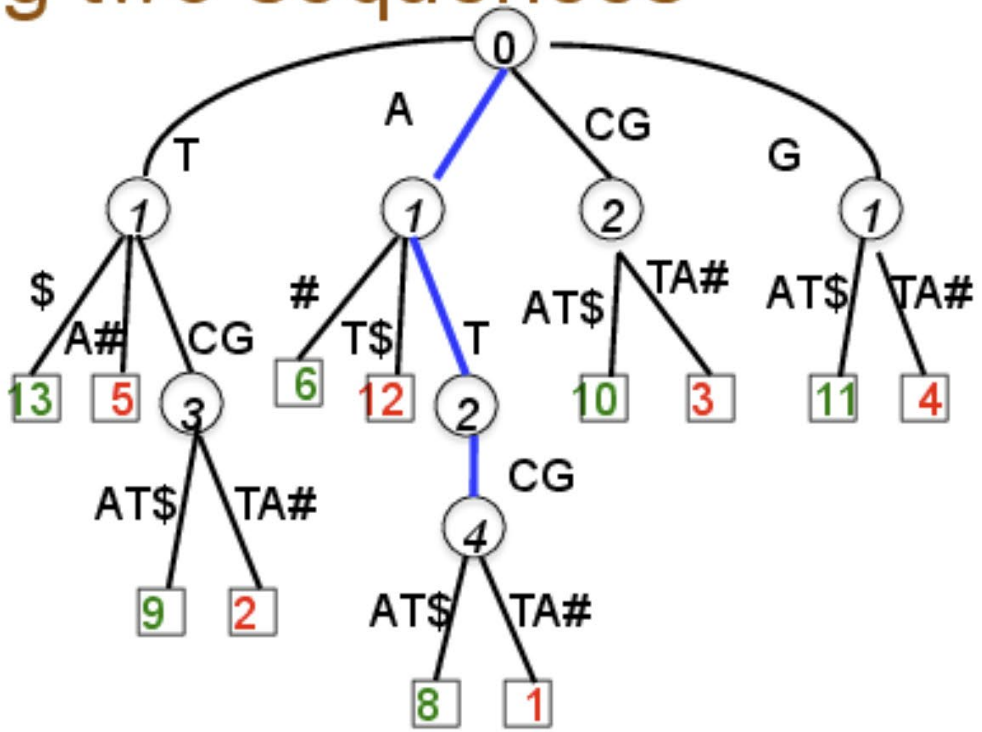
Image source: https://cgi.luddy.indiana.edu/~yye/c343-2019/suffix.php

SuffixTree

Matching two sequences

ATCGTA#
7
A# 6
TA# 5
GTA# 4
CGTA# 3
TCGTA# 2
ATCGTA# 1

ATCGAT$
$ 14
T$ 13
AT$ 12
GAT$ 11
CGAT$ 10
TCGAT$ 9
ATCGAT$ 8



ATCG is the longest common substring

Every unique matching sequence is represented by **an internal node with exactly two child nodes, such that the child nodes are leaf nodes from different sequences**

Image source: https://cgi.luddy.indiana.edu/~yye/c343-2019/suffix.php

PrefixSpan Algorithm

Id	Sequence
S_1	$CAGAAGT$
S_2	$TGACAG$
S_3	$GAAGT$

minSup=3

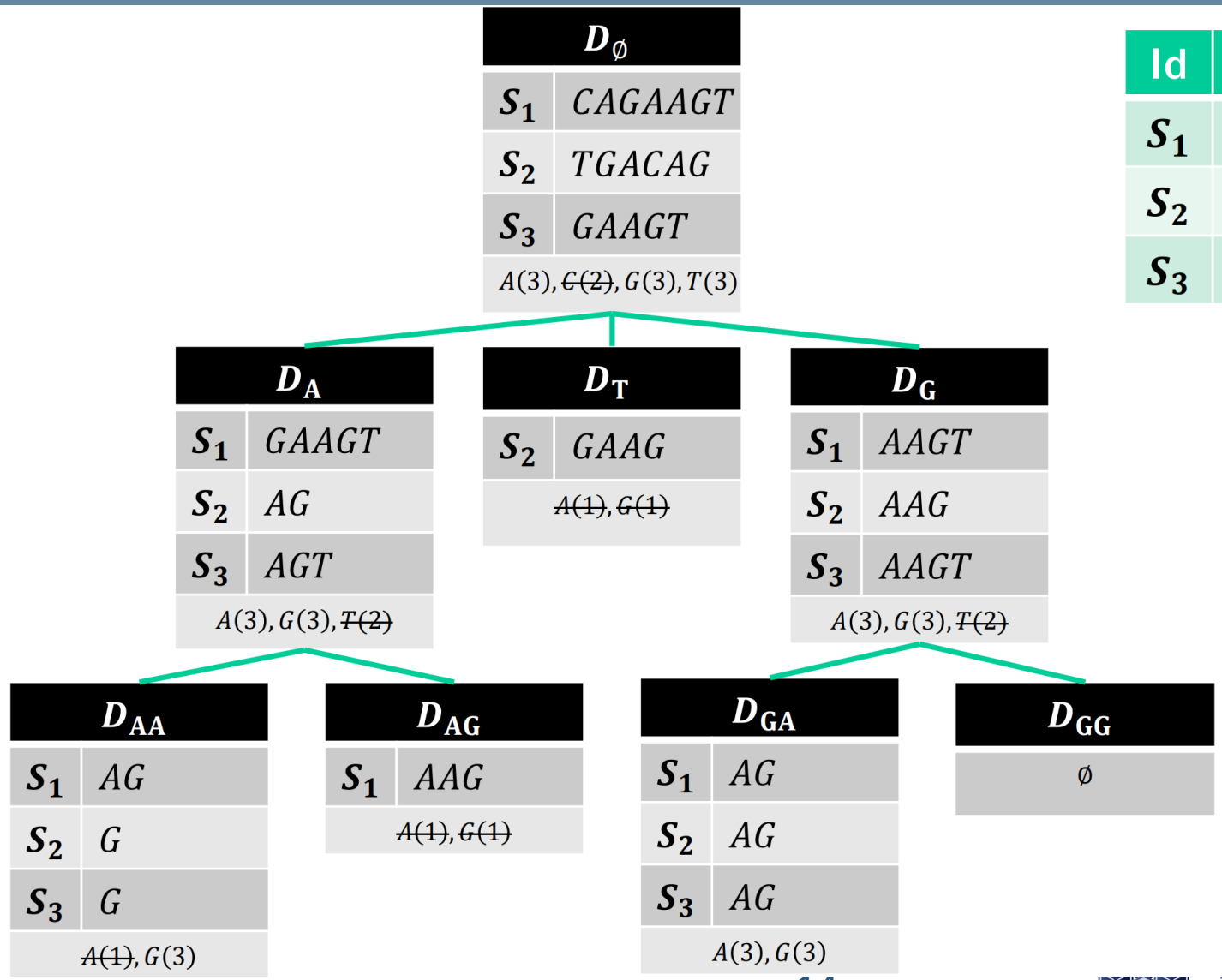


Image source: https://www.dbs.ifi.lmu.de/Lehre/KDD/SS16/skript/3b_SeqPattMining.pdf

PrefixSpan Algorithm

D_{AA}	
S_1	AG
S_2	G
S_3	G
$\cancel{A(1)}, G(3)$	

D_{AG}	
S_1	AAG
$\cancel{A(1)}, \cancel{G(1)}$	

D_{GA}	
S_1	AG
S_2	AG
S_3	AG
$A(3), G(3)$	

D_{GG}
$\emptyset$

D_{AAG}
$\emptyset$

D_{GAA}	
S_1	G
S_2	G
S_3	G
$G(3)$	

D_{GAG}
$\emptyset$

D_{GAAG}
$\emptyset$

Id	Sequence
S_1	$CAGAAGT$
S_2	$TGACAG$
S_3	$GAAGT$

Image source: https://www.dbs.ifi.lmu.de/Lehre/KDD/SS16/skript/3b_SeqPattMining.pdf

Agenda

- Closed and Maximal Pattern Mining
- Frequent Subsequence Mining
- Frequent Subgraph Mining
- Summary

Frequent Subgraph Mining

Given: A graph dataset $GS = \{G_i | i = 0, ..., n\}$, and *minsup*

- Each G_i a labeled, undirected graph (V, E, L, l)
- *minsup* is the minimum support

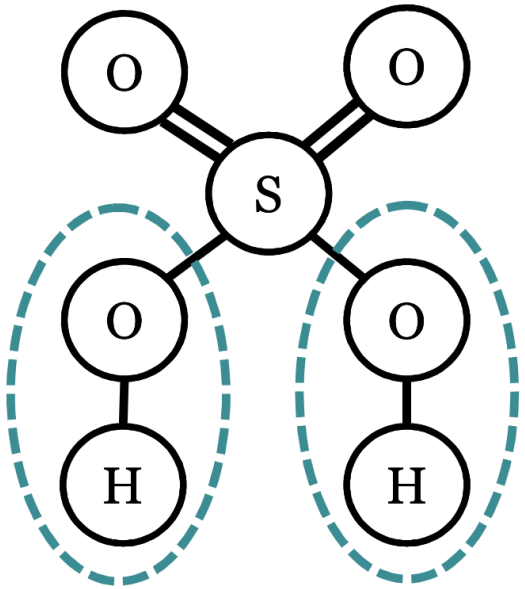
Find: All g s.t. $\sum_{G_i \in GS} \zeta(g, G_i) \geq minsup$

$$\zeta(g, G_i) = \begin{cases} 1 & \textit{if } g \textit{ isomorphic to a subgraph of } G, \\ 0 & \textit{if } g \textit{ not isomorphic to a subgraph of } G \end{cases}$$

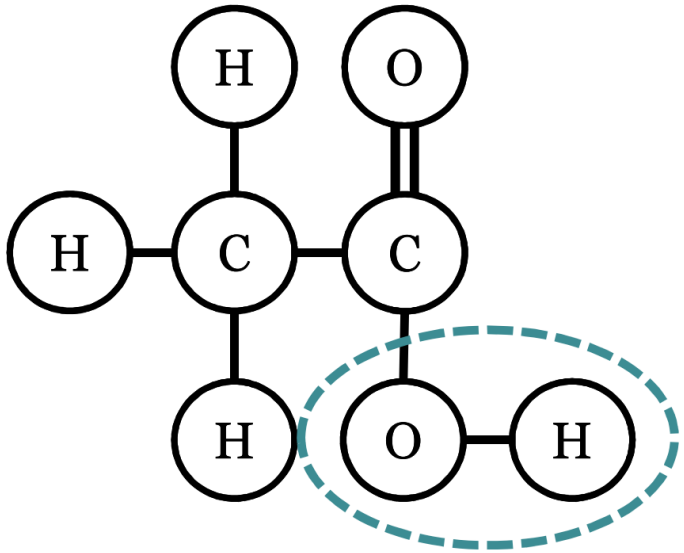
Recall…

O-H present in ¾ inputs

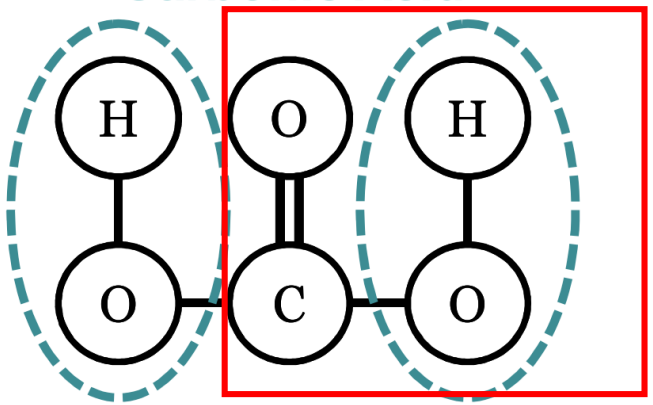
Sulfuric Acid



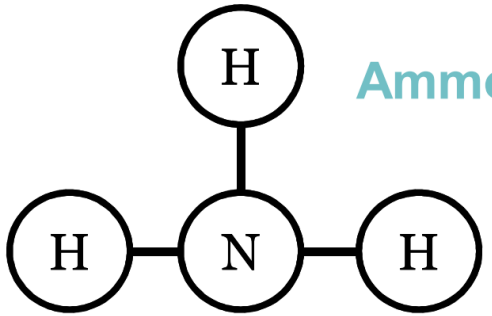
Acetic Acid



Carbonic Acid



Ammonia



Graph Isomorphism

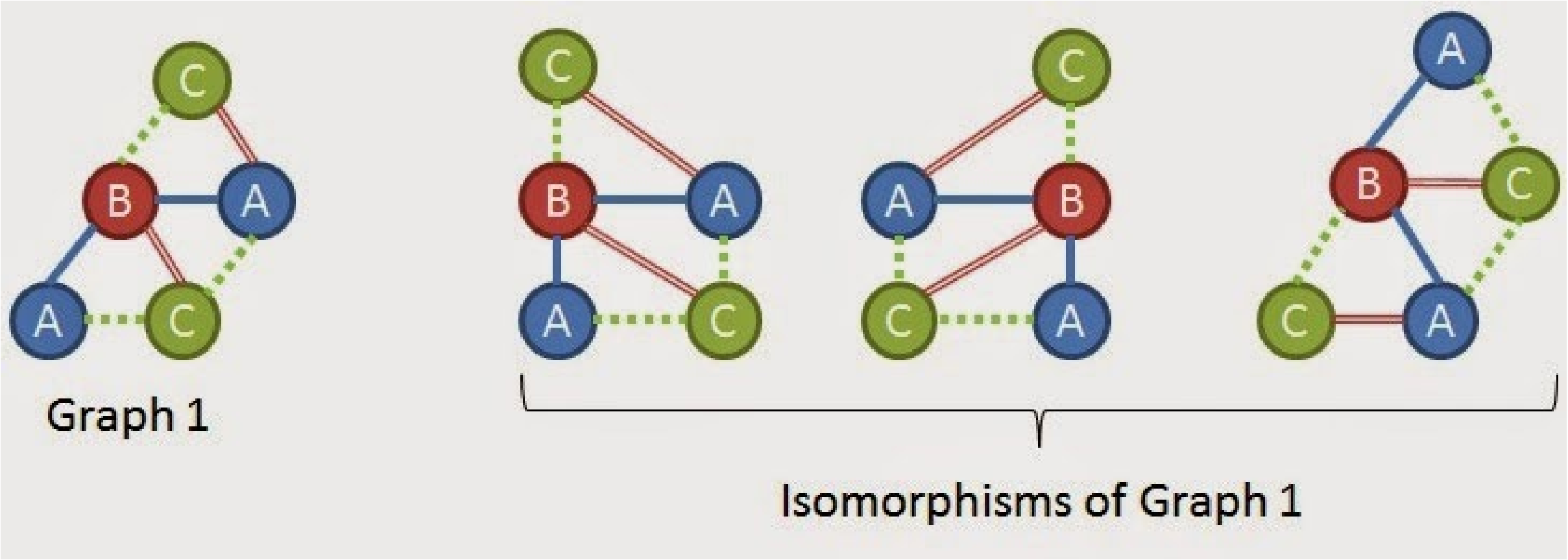
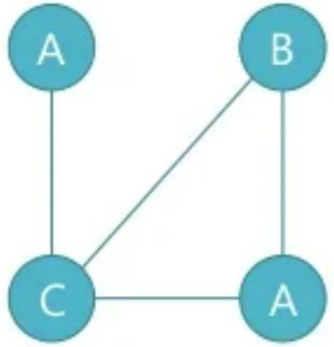
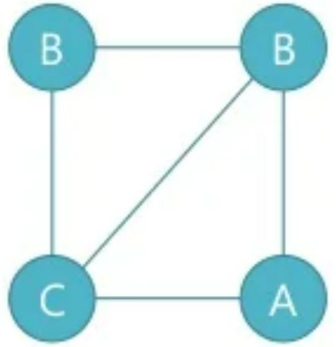


Image from: http://simpledatamining.blogspot.com/2015/03/graph-pattern-mining-gspan-introduction.html

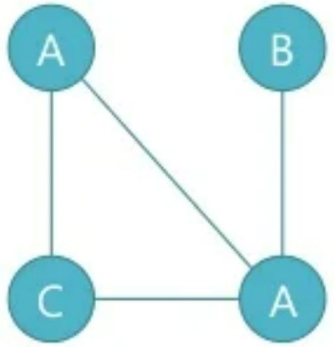
Frequent Subgraph: Another Simple Example



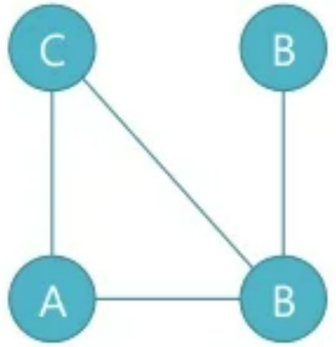
Graph 1



Graph 2

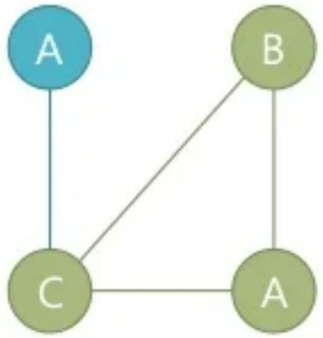


Graph 3

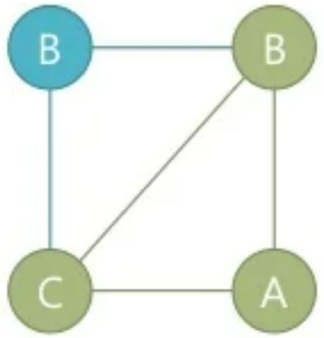


Graph 4

Frequent Subgraph: Another Simple Example

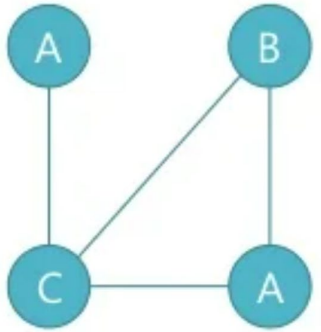


Graph 1

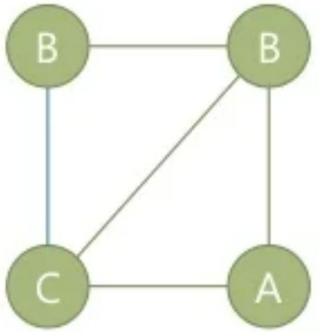


Graph 2

k = 3, frequency: 3, support: 75%

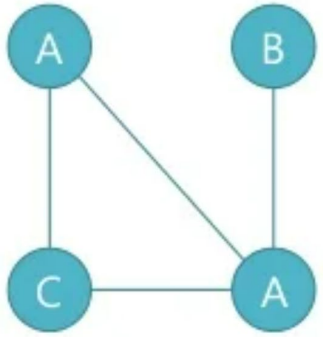


Graph 1

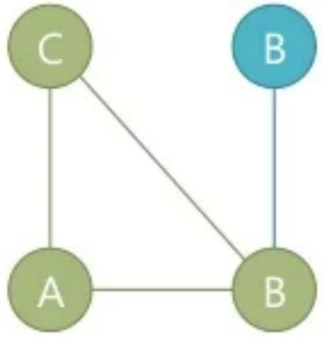


Graph 2

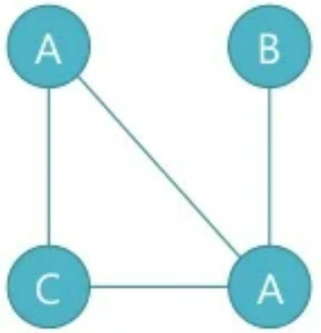
k = 4, frequency: 2, support: 50%



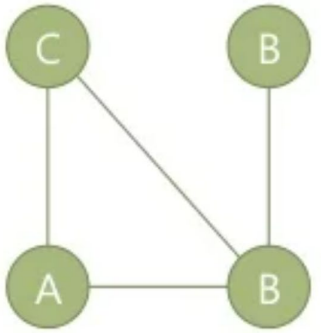
Graph 3



Graph 4



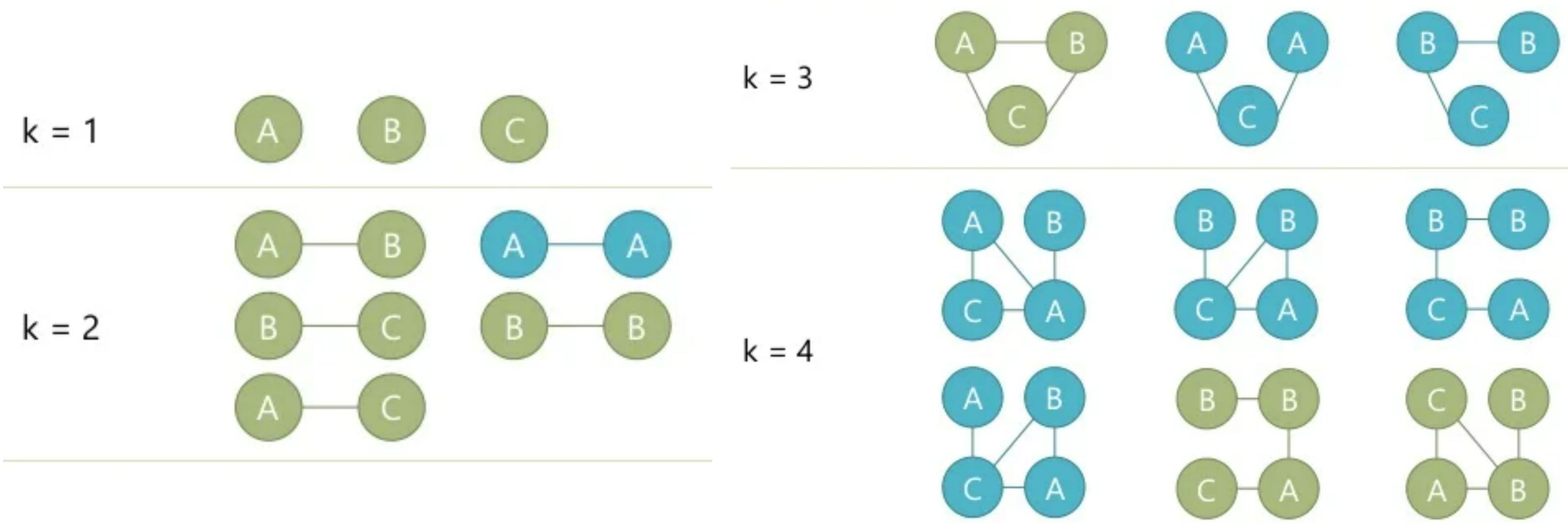
Graph 3



Graph 4

Example source: https://www.slideshare.net/vwchu/data-mining-seminar-graph-mining-and-social-network-analysis

Frequent Subgraph: Another Simple Example

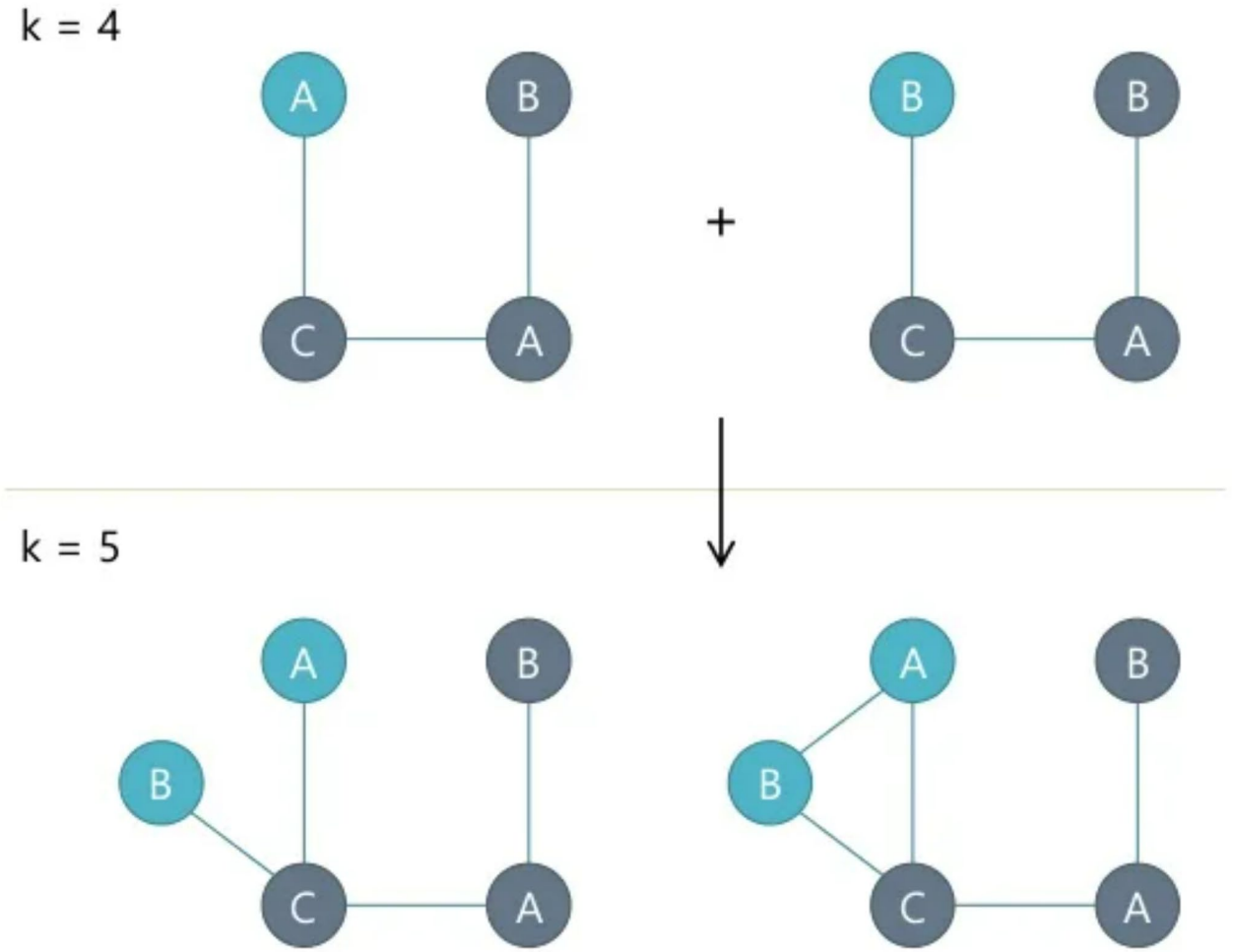


Apriori-based Graph Mining

```
procedure AprioriGraph(D, min_sup, S_k)
1    S_{k+1} ← ∅;
2    foreach frequent g_i ∈ S_k do
3        foreach frequent g_j ∈ S_k do
4            foreach size (k+1) graph g formed by merge(g_i, g_j) do
5                if g is frequent in D and g ∉ S_{k+1} then
6                    insert g into S_{k+1};
7    if s_{k+1} ≠ ∅ then
8        AprioriGraph(D, min_sup, S_{k+1});
9    return;
```

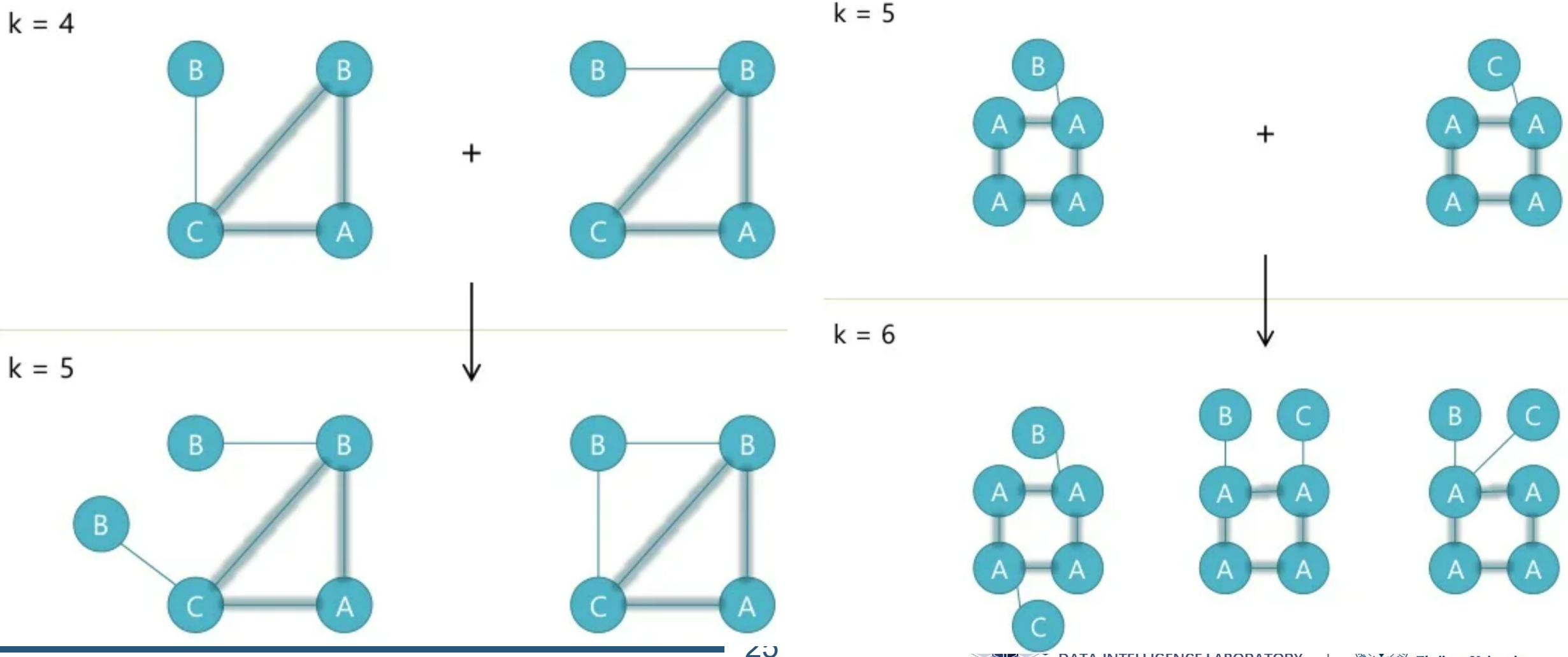
Apriori-based Graph Mining

Vertex-Based candidate generation

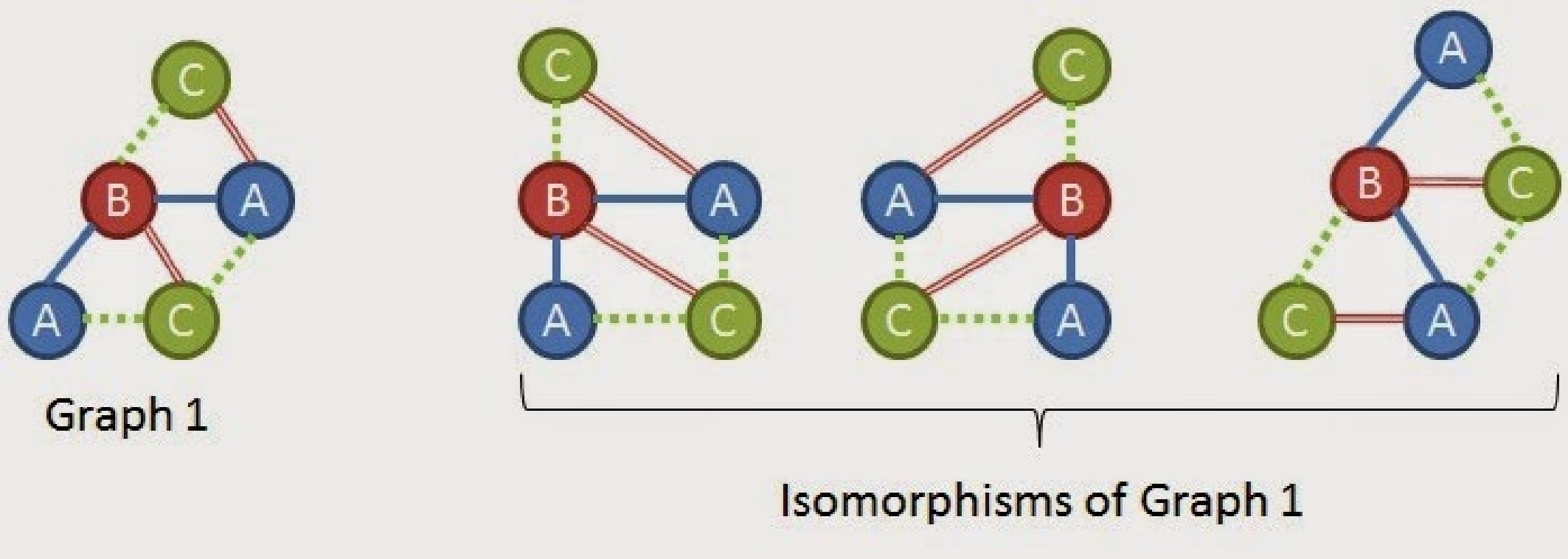


Apriori-based Graph Mining

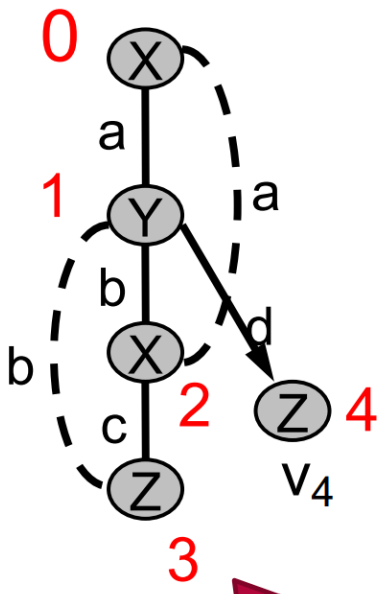
Edge-Based candidate generation



How to Determine Graph Isomorphism?



How to Determine Graph Isomorphism?



Vertex discovery time

Edge#	Code
0	(0,1,X,a,Y)
1	(1,2,Y,b,X)
2	(2,0,X,a,X)
3	(2,3,X,c,Z)
4	(3,1,Z,b,Y)
5	(1,4,Y,d,Z)

DFS-edge: $(i, j, L_i, L_{i,j}, L_j)$

i,j – vertices by discovery time

L_i, L_j - vertex labels of v_i, v_j

$L_{i,j}$ - edge label between v_i, v_j

$i < j$ - forward edge

$i > j$ - backward edge

Image source: https://hpi.de/fileadmin/user_upload/fachgebiete/mueller/courses/graphmining/GraphMining-04-FrequentSubgraph.pdf

How to Determine Graph Isomorphism?

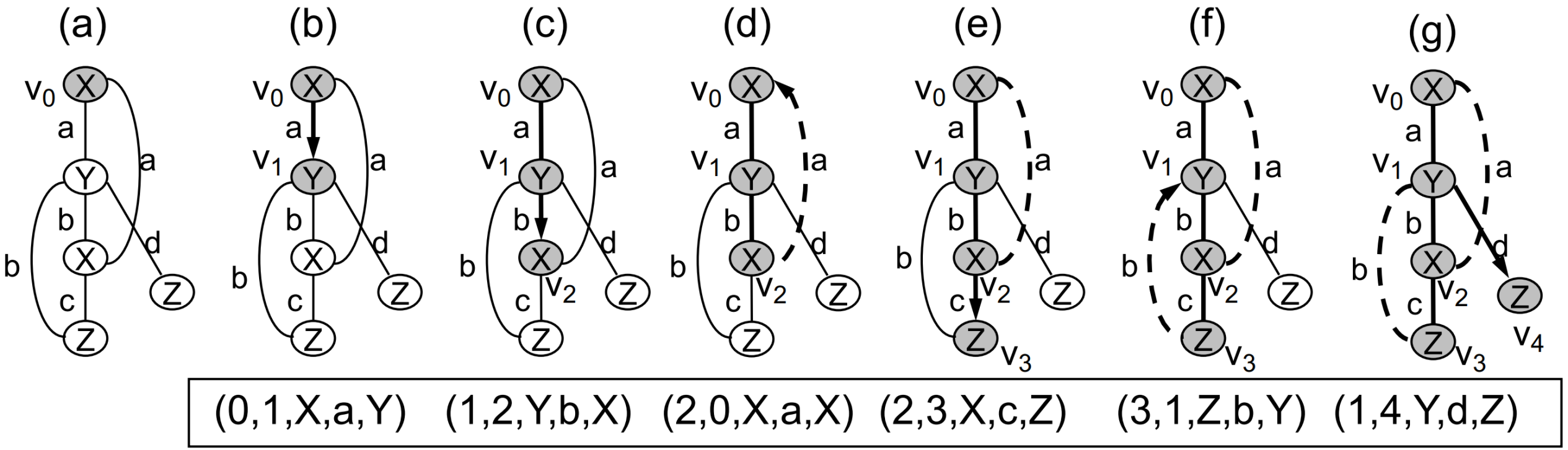


Image source: https://hpi.de/fileadmin/user_upload/fachgebiete/mueller/courses/graphmining/GraphMining-04-FrequentSubgraph.pdf

How to Determine Graph Isomorphism?

	(a)	(b)	(c)
1	(0, 1, X, a, Y)	(0, 1, Y, a, X)	(0, 1, X, a, X)
2	(1, 2, Y, b, X)	(1, 2, X, a, X)	(1, 2, X, a, Y)
3	(2, 0, X, a, X)	(2, 0, X, b, Y)	(2, 0, Y, b, X)
4	(2, 3, X, c, Z)	(2, 3, X, c, Z)	(2, 3, Y, b, Z)
5	(3, 1, Z, b, Y)	(3, 0, Z, b, Y)	(3, 0, Z, c, X)
6	(1, 4, Y, d, Z)	(0, 4, Y, d, Z)	(2, 4, Y, d, Z)

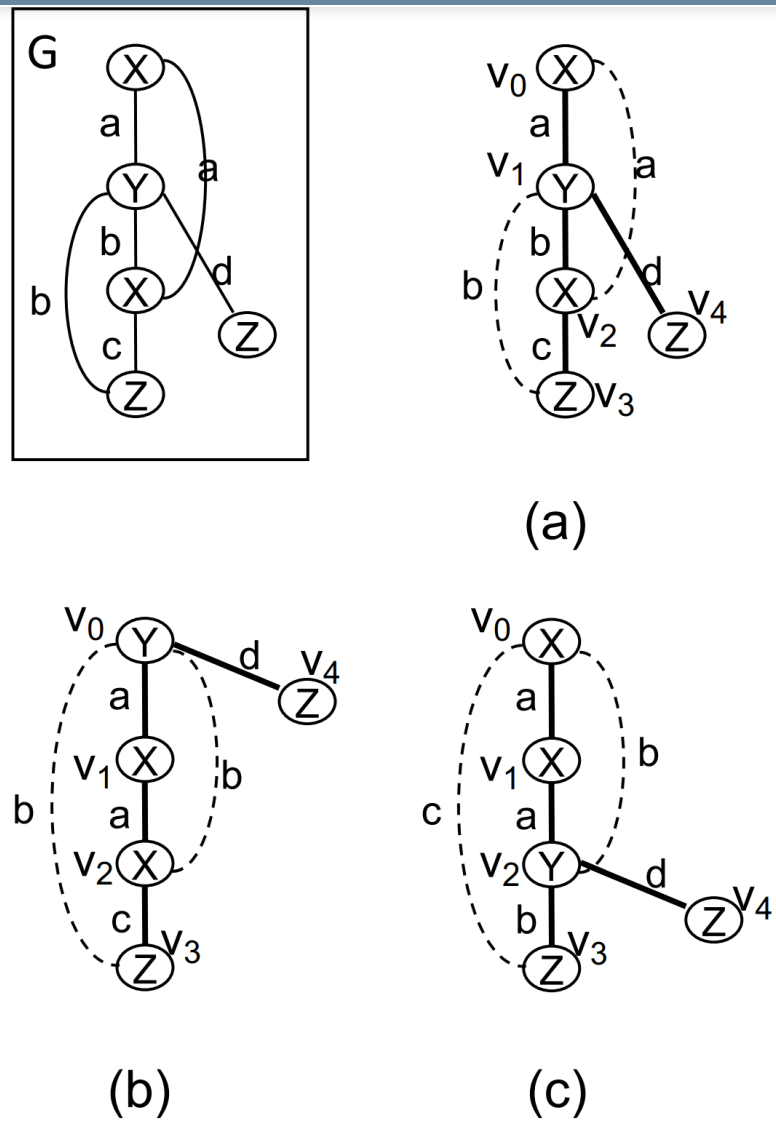


Image source: https://hpi.de/fileadmin/user_upload/fachgebiete/mueller/courses/graphmining/GraphMining-04-FrequentSubgraph.pdf

How to Determine Graph Isomorphism?

Defining $a_t \prec_e b_t$

- $a_t = (i_a, j_a, L_{i_a}, L_{i_a,j_a}, L_{j_a})$
- $b_t = (i_b, j_b, L_{i_b}, L_{i_b,j_b}, L_{j_b})$
- $a_t \prec_e b_t$ if

1. Both are forward edges and
   - $i_b < i_a$ (edge starts from a later visited vertex, why?)
   - $i_b = i_a$ and the labels of *a* are lexicographically less than labels of *b*, in order of tuple
2. Both are backward edges ($i_a = i_b$ for the same reason)
   - $j_a < j_b$ (edge connected to a earlier discovered vertex)
   - $j_a = j_b$ and the edge label of *a* is lexicographically less than the one of *b*
     - Question: Why not the node labels?
3. a_t is backward and b_t is forward

If they are forward edges $j_a = j_b$ because since the previous edges are equal you have discovered a new node

Image source: https://hpi.de/fileadmin/user_upload/fachgebiete/mueller/courses/graphmining/GraphMining-04-FrequentSubgraph.pdf

How to Determine Graph Isomorphism?

Min DFS-Code

	(a)	(b)	(c)
1	(0, 1, X, a, Y)	(0, 1, Y, a, X)	**(0, 1, X, a, X)**
2	(1, 2, Y, b, X)	(1, 2, X, a, X)	**(1, 2, X, a, Y)**
3	(2, 0, X, a, X)	(2, 0, X, b, Y)	**(2, 0, Y, b, X)**
4	(2, 3, X, c, Z)	(2, 3, X, c, Z)	**(2, 3, Y, b, Z)**
5	(3, 1, Z, b, Y)	(3, 0, Z, b, Y)	**(3, 0, Z, c, X)**
6	(1, 4, Y, d, Z)	(0, 4, Y, d, Z)	**(2, 4, Y, d, Z)**

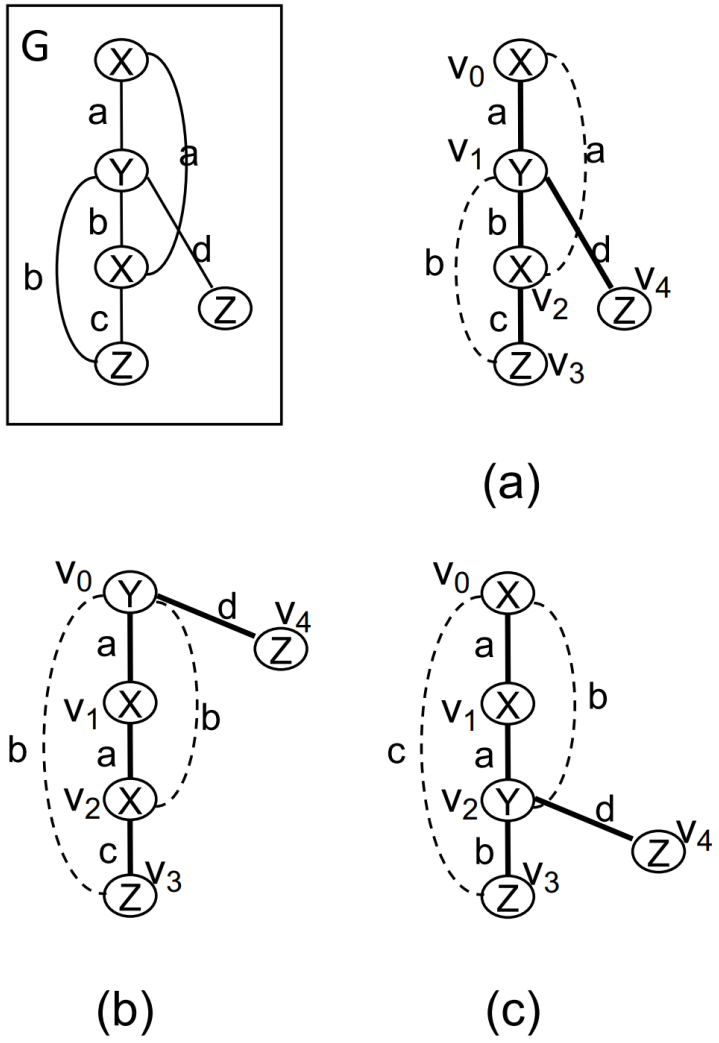


Image source: https://hpi.de/fileadmin/user_upload/fachgebiete/mueller/courses/graphmining/GraphMining-04-FrequentSubgraph.pdf

How to Determine Graph Isomorphism?

Theorem

Graphs G1 and G2 are isomorphic iff

$$\text{min-DFS}(G1) = \text{min-DFS}(G2)$$

Image source: https://hpi.de/fileadmin/user_upload/fachgebiete/mueller/courses/graphmining/GraphMining-04-FrequentSubgraph.pdf

Agenda

- Closed and Maximal Pattern Mining
- Frequent Subsequence Mining
- Frequent Subgraph Mining
- Summary