



세션 로딩중..



**Association Analysis**

# Contents



1. Introduction
2. Basic Concepts
3. Apriori
4. Sequential Pattern Mining
  - 4-1) GSP algorithm
  - 4-2) PrefixSpan algorithm

# 1. Introduction

# Introduction

## Supervised learning

---

target variable Y 존재  
ex) linear regression,  
logistic regression

VS

## Unsupervised learning

---

target variable Y 존재X  
ex) clustering,  
**association**

# Introduction

이런 상품은 어때요?



문스타 M1체어 게  
이밍의자 학생의...

**67,910원**

★★★★★ (2,405)



이동식 바퀴달린 사  
이드 보조테이블, ...

**36,900원**

★★★★★ (937)



체어포커스 화이트  
바디 메쉬 알라딘...

**62,900원**

**6% 할인**

로켓배송

★★★★★ (1,946)



무료배송  
에이픽스 높이조절  
스탠딩 모션데스...

**179,000원**

★★★★★ (108)



무료배송  
이반가구 1인~8인  
접이식테이블 다...

**68,900원**

★★★★★ (405)



무료배송  
체어스코 메쉬 사무  
학생 책상 컴퓨터...

**59,310원**

**65% 할인**

★★★★★ (650)



행복의모든:  
조절 이동식

**22,900원**

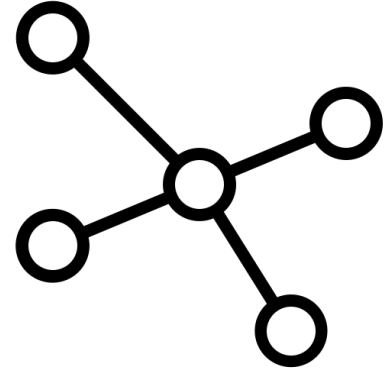
로켓배송

★★★★★ (2,...

AD

# | What is Association Analysis?

→ a technique used in data mining and machine learning to identify **patterns and relationships between variables**



# Market Basket Analysis



**Example)**





# | Goal of Association Analysis


**“What goes with what”**

**If X was purchased, then Y was also purchased**

## 2. Basic Concepts

# | Terminology

**“A -> B”**

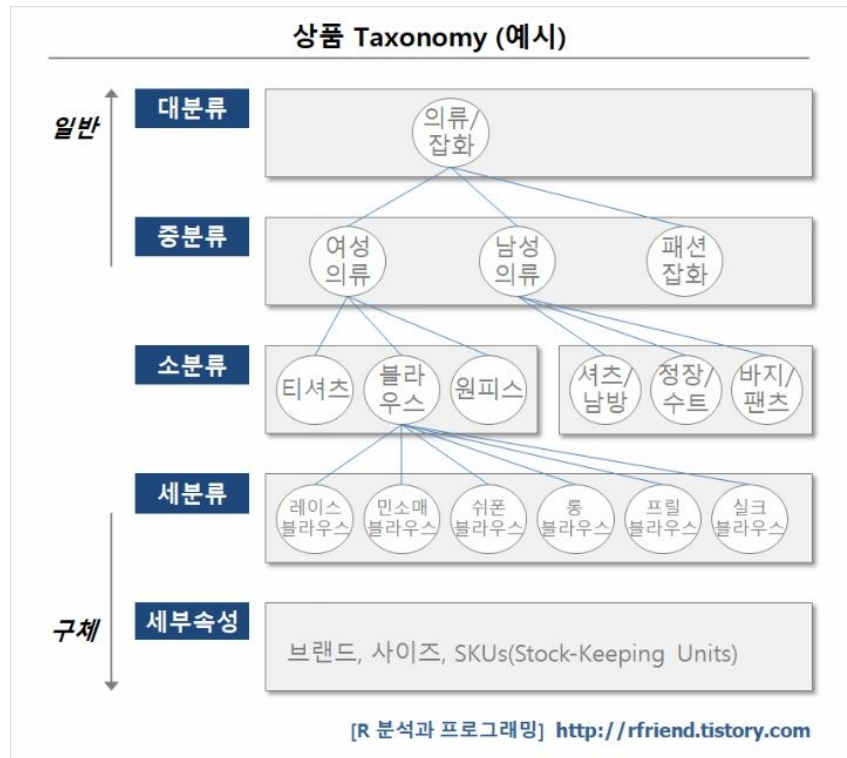
- **Antecedent: 조건절(If) “A”**
  - **Consequent: 결과절(Then) “B”**
  - **Item set: The items comprising the antecedent or consequent**
-  **Disjoint**

Ex) (라면)을 사면 (콜라)를 산다.

Ex) (라면)을 사면 (즉석밥과 콜라)를 산다.

# Taxonomy

- Item set 설정 시 적당한(?) 수준의 분류 필요
- Virtual item set의 필요성



# Type of variables

- 연관 분석에 사용되는 변수의 종류는 제한되지 않음
- Continuous variable → 구간을 나눠 dummy coding

범주형 데이터(Categorical Data) → 이항변수화(Binarization)

CUST_ID	GENDER	AGE	CHILD_PRD_YN	MOBILE_APP_USE	RE_ORDER
1	FEMALE	23	NO	YES	YES
2	MALE	28	NO	YES	NO
3	FEMALE	42	NO	NO	NO
4	FEMALE	34	YES	YES	YES
5	MALE	45	NO	NO	NO
6	FEMALE	36	YES	YES	YES

CUST_ID	GENDER = MALE	GENDER = FEMALE	AGE = 20	AGE = 30	AGE = 40	CHILD_PRD_YN = YES	CHILD_PRD_YN = NO	MOBILE_APP_USE = YES	MOBILE_APP_USE = NO	RE_ORDER = YES	RE_ORDER = NO
1	0	1	1	0	0	0	1	1	0	1	0
2	1	0	1	0	0	0	1	1	0	0	1
3	0	1	0	0	1	0	1	0	1	0	1
4	0	1	0	1	0	1	0	1	0	1	0
5	1	0	0	0	1	0	1	0	1	0	1
6	0	1	0	1	0	1	0	1	0	1	0

# Performance Measures

## 1) Support

$$\text{Support}(A) = P(A)$$

$$\text{Support}(A \rightarrow B) = P(A \cap B)$$

규칙의 범용성 평가

Ex)  $\text{Support}(\text{라면}) = ?$

Ex)  $\text{Support}(\text{라면} \rightarrow \text{계란}) = ?$

거래	라면	햇반	계란	맥주
1	O	X	O	O
2	O	O	X	X
3	O	O	O	O
4	O	X	O	X
5	X	O	X	O
6	O	X	O	O
7	X	X	X	O
8	O	O	O	X
9	O	O	X	X
10	O	O	O	X

# Performance Measures

## 2) Confidence

$$\text{Confidence}(A \rightarrow B) = P(A \cap B) / P(A)$$

Ex) Confidence(라면 → 계란) = ?

Ex) Confidence(라면 → 맥주) = ?

거래	라면	햇반	계란	맥주
1	O	X	O	O
2	O	O	X	X
3	O	O	O	O
4	O	X	O	X
5	X	O	X	O
6	O	X	O	O
7	X	X	X	O
8	O	O	O	X
9	O	O	X	X
10	O	O	O	X

# Performance Measures

## 3) Lift

$$\text{Lift}(A \rightarrow B) = P(A \cap B) / [P(A) * P(B)]$$

Lift=1: A and B are statistically independent.

Lift>1: positive relationship between A and B.

Lift<1: negative relationship between A and B.

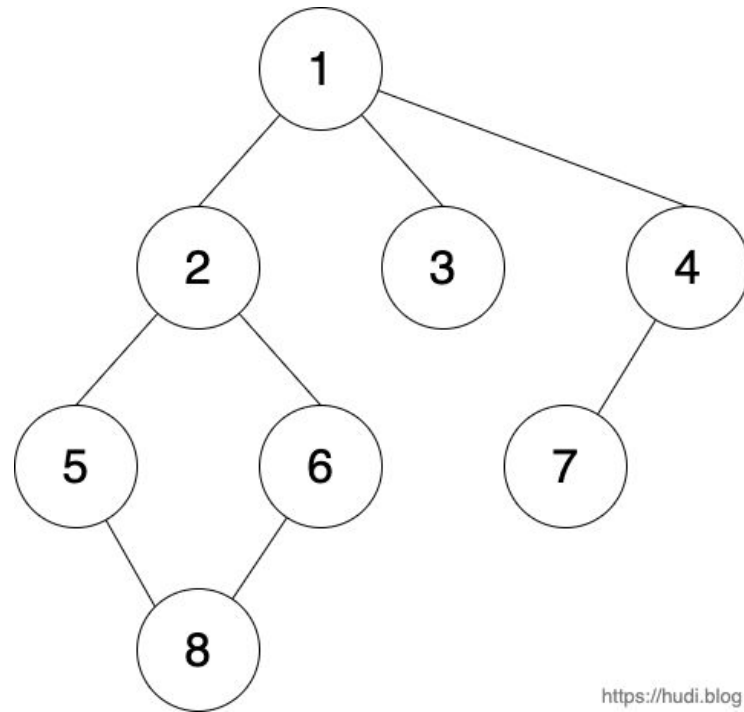
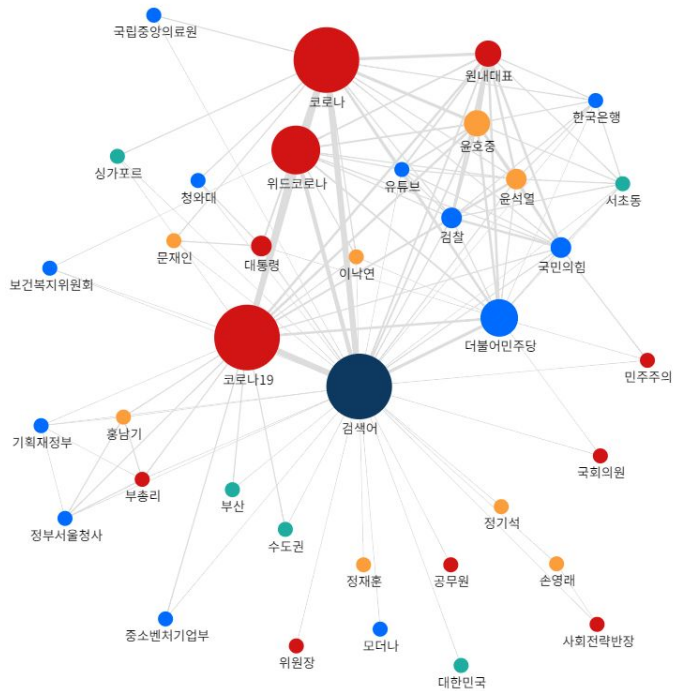
Ex) Lift(라면 -> 계란) = ?

Ex) Lift(라면 -> 맥주) = ?

거래	라면	햇반	계란	맥주
1	O	X	O	O
2	O	O	X	X
3	O	O	O	O
4	O	X	O	X
5	X	O	X	O
6	O	X	O	O
7	X	X	X	O
8	O	O	O	X
9	O	O	X	X
10	O	O	O	X



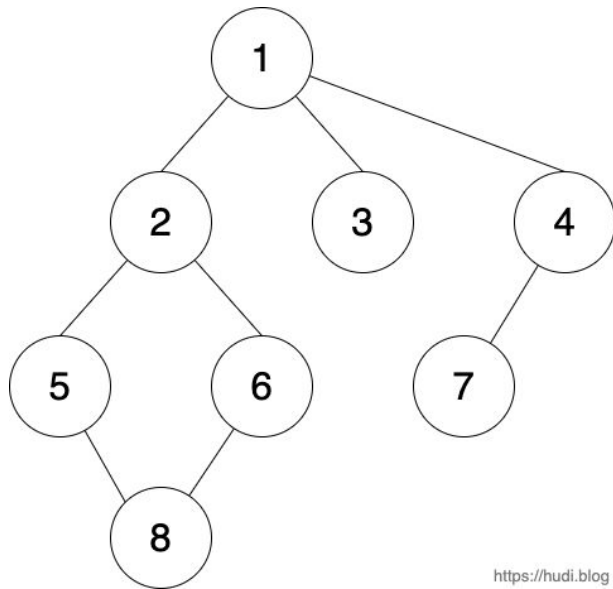
# Visualization



# Graph Search Algorithm

## 1) DFS(Depth-First Search)

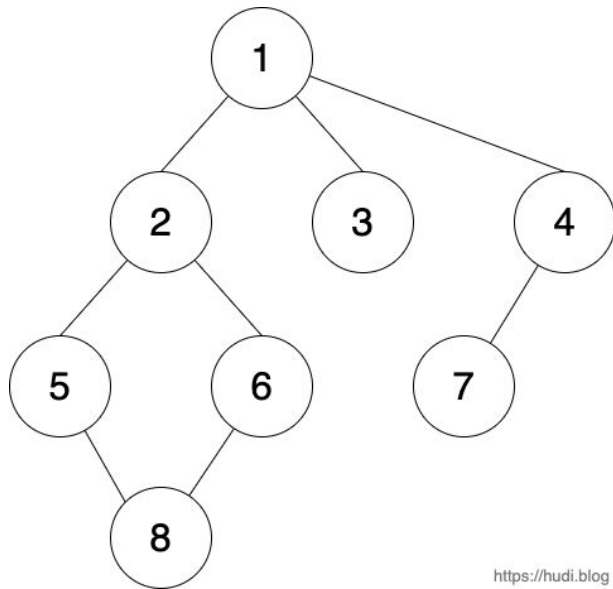
- 루트 노드에서 시작해서 다음 branch로 넘어가기 전에 해당 branch를 모두 완벽하게 탐색
- BFS에 비해 간단하지만 검색 속도가 느림
- 경로의 특징을 저장해둬야 할 때 유용



# Graph Search Algorithm

## 2) BFS(Breadth-First Search)

- 루트 노드에서 시작해서 인접한 노드를 먼저 탐색
- DFS에 비해 복잡하지만 검색 속도가 빠름
- 최단 경로를 찾고자 할 때 유용



# Case 1

## 도로교통사고 분석

- 인적/환경요인, 차량/행동유형과 피해 심각도, 피해부위 및 상태를 나타내는 변수들 간의 연관규칙

721: 면허소지, 273: 고졸, 233: 30대

901: 차량내부충격, 881: 골절, 861: 머리, 641: 사망

표 1. 머리골절사망사고의 인적요인 연관규칙

신뢰도	지지도	연 관 규 칙
83.33	8.89	721, 452, 411, 273, 251 → 901, 881, 861, 641
100.0	8.89	721, 273, 233 → 901, 881, 861, 641

표 9. 머리골절사망사고의 행동유형 연관규칙

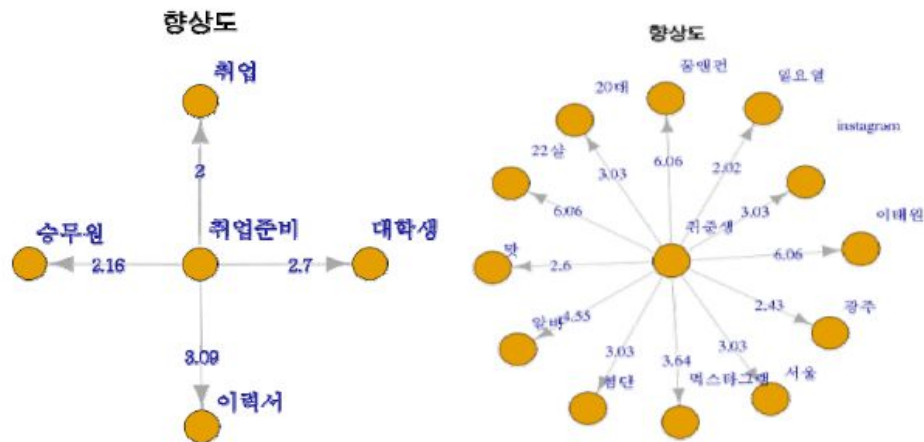
신뢰도	지지도	연 관 규 칙
71.43	8.89	998, 802, 584, 562, 493 → 901, 881, 861, 641

▲ 출처: 손소영, 오기열, 신형원 (2002). 다수의 결과를 고려한 한국의 도로교통사고 연관규칙분석. 산업공학, 15(4), 426-431.

# Case 2

## 해시태그 분석

- 인스타그램 해시태그를 이용한 연관 규칙 분석
- 실시간 트렌드 분석을 용이하게 하여 고객들이 니즈 파악



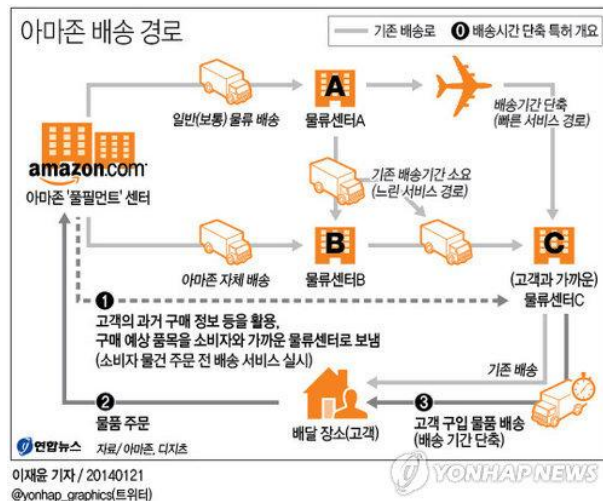
▲ 출처: 이종화, 이현규 (2017). 해시태그를 이용한 실시간 연관 규칙 분석 연구. 인터넷전자상거래연구, 17(4), 105-117.

# Case 3

## Anticipatory Shipping

- 물건 구매 전, 구매 예상 물품을 미리 가까운 물류 창고에 배송, 고객이 주문하면 바로 배송
- 고객의 쇼핑 습관, 취향, 소비 패턴 등을 분석
- 물건 배송 소요 시간과 배송 과정의 비용을 동시에 줄임

[https://www.youtube.com/watch?v=JhAIW\\_DCYxk](https://www.youtube.com/watch?v=JhAIW_DCYxk)



# 3. Apriori

# Basics: 데이터 구조

- Some Representations...

**Table 5.1.** An example of market basket transactions.

<i>TID</i>	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

**Table 5.2.** A binary 0/1 representation of market basket data.

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

**\*\*** 기본단위는 ‘Transaction’이다.

**\*\*** 주의사항: 여러 번 샀더라도 한 번만 기록한다.



# Basics: Item set & Support count

- Item set: 아이템 집합을 의미함.

$$I = \{i_1, i_2, \dots, i_d\}$$
$$\{\text{Beer, Diapers, Milk}\}$$

- 1) Support count (몇 번 나오는가?):

$$\sigma(X) = |\{t_i | X \subseteq t_i, t_i \in T\}|.$$

- 2) Support (나오는 빈도):

$$s(X) = \sigma(X)/N.$$

# Our goal: What goes with what?

- 1) Find Association Rule: What goes with What?

$$X \rightarrow Y, X \cap Y = \emptyset.$$

$$\{\text{Milk, Diapers}\} \rightarrow \{\text{Beer}\}$$

- 2) Common steps:

Step 1: Frequent Itemset Generation (Support)

Step 2: Rule generation (Confidence)

Step 3: Evaluation (Lift)

# Our goal: What goes with what?

- Common steps:

Step 1: Frequent Itemset Generation (Min Support보다 큰 item set을 찾는다.)

`{Beer, Diapers, Milk}`

Step 2: Rule generation (Confidence가 큰 규칙을 찾는다.)

`{Milk, Diapers} → {Beer}`

Step 3: Evaluation (다양한 객관/주관적 평가지표로 의미있는 규칙을 발견한다.)

보통 minsup, minconf → Lift 내림차순 정리 의 과정을 거친다.

# STEP1: Frequent item set generation

- 후보가 될 수 있는 item set의 개수를 생각해보자.. 너무 많다!

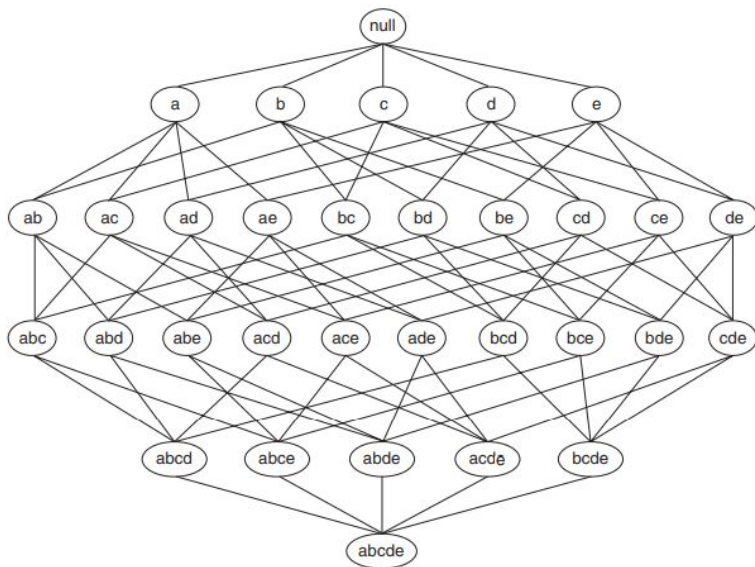


Figure 5.1. An itemset lattice.

# STEP1: Frequent item set generation

- 그렇다면 빠른 계산을 위해서는,  $N$ 을 줄이거나  $M$ 을 줄이면 된다.

(여기서는  $M$ 을 줄이는 방법에 대해 알아보자.)

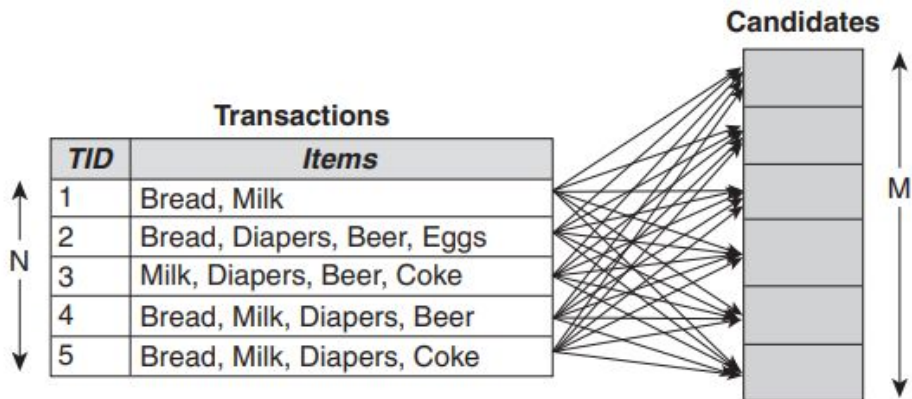
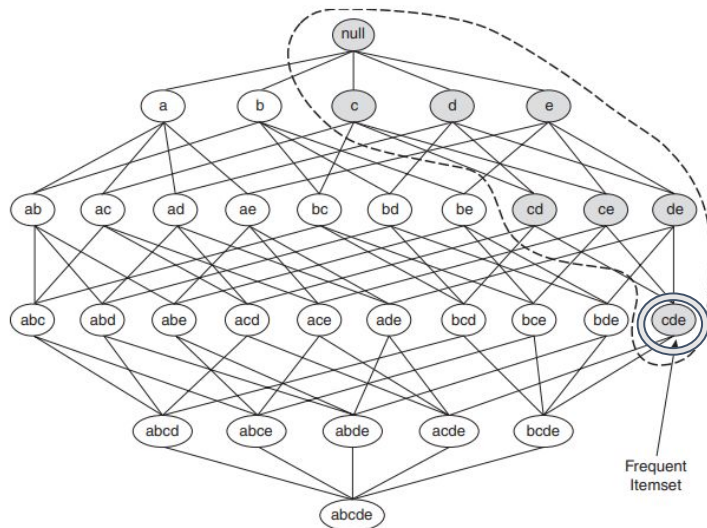


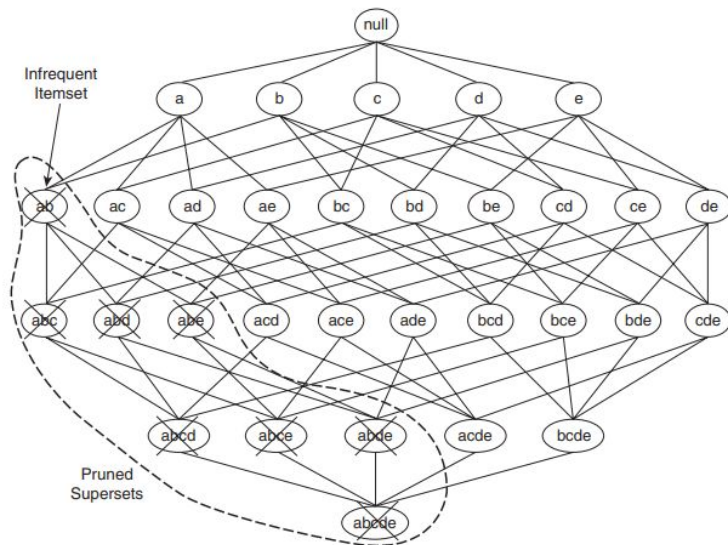
Figure 5.2. Counting the support of candidate itemsets.

# “The Apriori Principle”

**Theorem 5.1** (*Apriori Principle*). *If an itemset is frequent, then all of its subsets must also be frequent.*



**Figure 5.3.** An illustration of the *Apriori* principle. If  $\{c, d, e\}$  is frequent, then all subsets of this itemset are frequent.



**Figure 5.4.** An illustration of support-based pruning. If  $\{a, b\}$  is infrequent, then all supersets of  $\{a, b\}$  are infrequent.

# “The Apriori Principle”

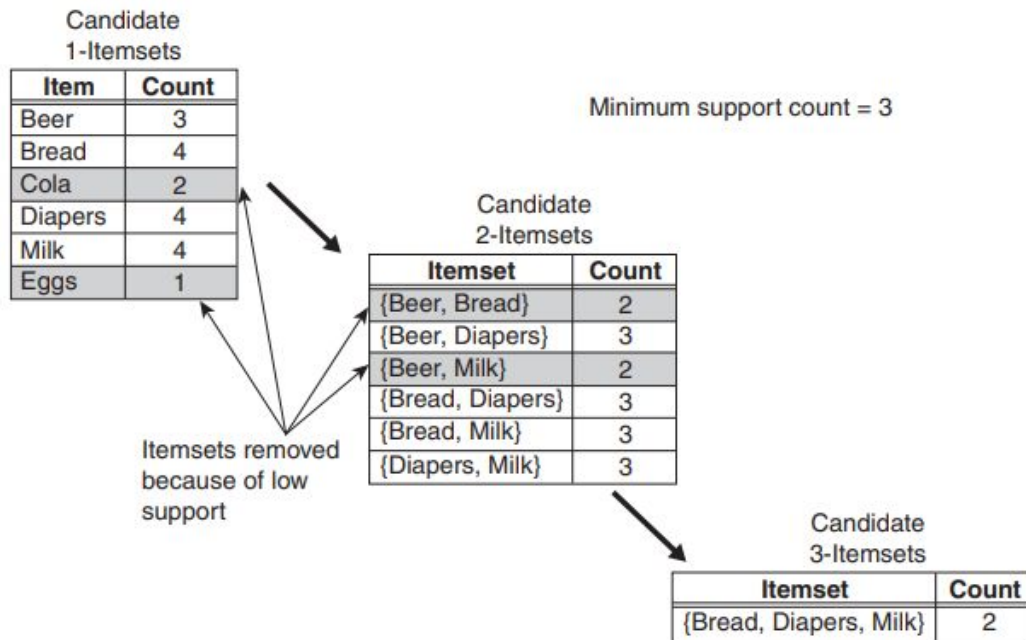
**Table 5.1.** An example of market basket transactions.

TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

BFS를 통해 하나씩 확인하는 과정.

의문점1 : Candidate의 기준은??

의문점2: Support count하는 법?



**Figure 5.5.** Illustration of frequent itemset generation using the *Apriori* algorithm.

# STEP1-1: Candidate

- 의문점 1 : Candidate?

$$F_{k-1} \Rightarrow C_k \Rightarrow F_k$$

“Candidate Generation” & “Candidate Pruning (부분집합 Frequent?)”

: Brute-force method

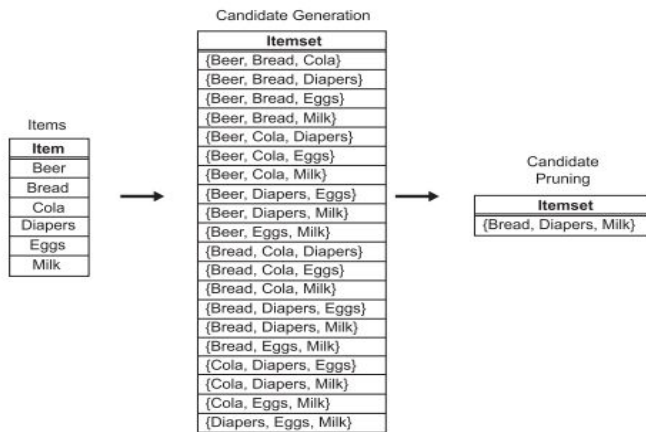


Figure 5.6. A brute-force method for generating candidate 3-itemsets.

Candidate Generation:

20가지.

Candidate Pruning:

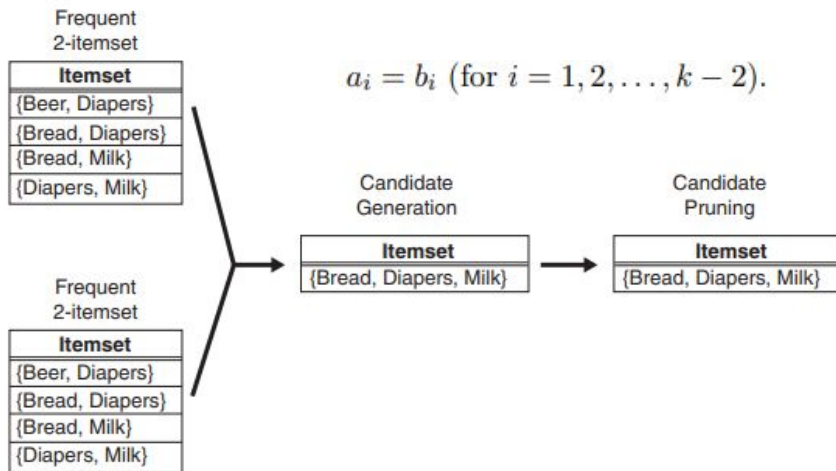
부분집합 3개씩 고려.



# STEP1-1: Candidate

- 의문점 1 : Candidate?  $F_{k-1} \Rightarrow C_k \Rightarrow F_k$

이전의 정보를 쓰는 것이 더 합리적으로 보임.  $F_{k-1} \times F_{k-1}$  Method



Candidate Generation:

1가지.

Candidate Pruning:

1번만 확인함!

Figure 5.8. Generating and pruning candidate  $k$ -itemsets by merging pairs of frequent  $(k-1)$ -itemsets.

# STEP1-2: Support count

- 의문점 2: Support count?

	noodle	egg	cola	rice	tuna
noodle		40%	40%	20%	20%
egg			30%	0%	20%
cola				0%	10%
rice					0%
tuna					

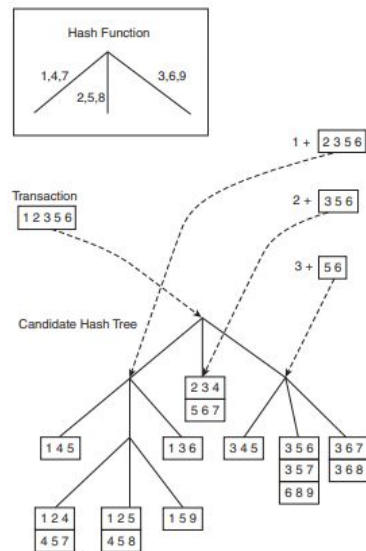
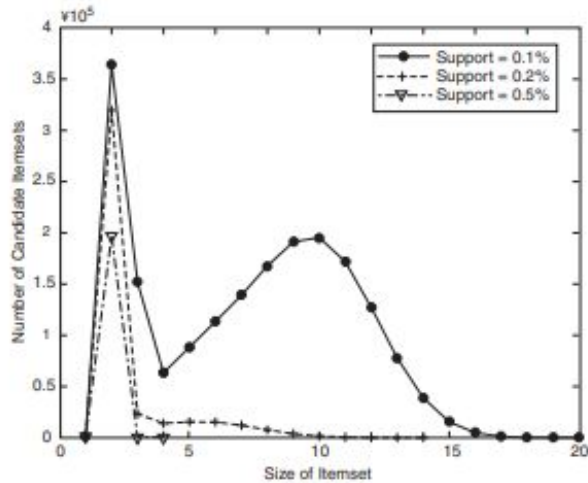
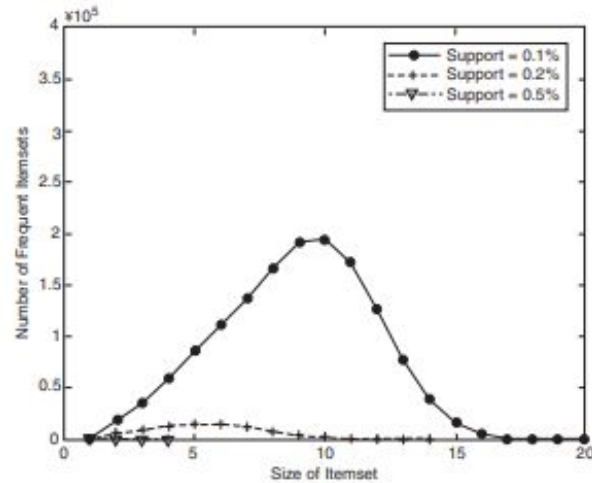


Figure 5.11. Hashing a transaction at the root node of a hash tree.

# STEP1: Frequent item set generation



(a) Number of candidate itemsets.



(b) Number of frequent itemsets.

**Figure 5.13.** Effect of support threshold on the number of candidate and frequent itemsets obtained from a benchmark data set.

# STEP2: Rule Generation

Use confidence!!!

**Theorem 5.2.** *Let  $Y$  be an itemset and  $X$  is a subset of  $Y$ . If a rule  $X \rightarrow Y - X$  does not satisfy the confidence threshold, then any rule  $\tilde{X} \rightarrow Y - \tilde{X}$ , where  $\tilde{X}$  is a subset of  $X$ , must not satisfy the confidence threshold as well.*

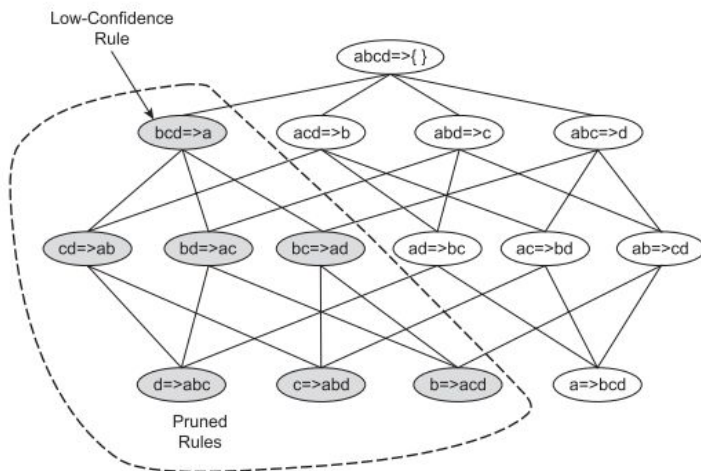


Figure 5.15. Pruning of association rules using the confidence measure.

# Maximal Frequent item set

너무 Frequent item set이 많으니까.. 대표만 가져오자.

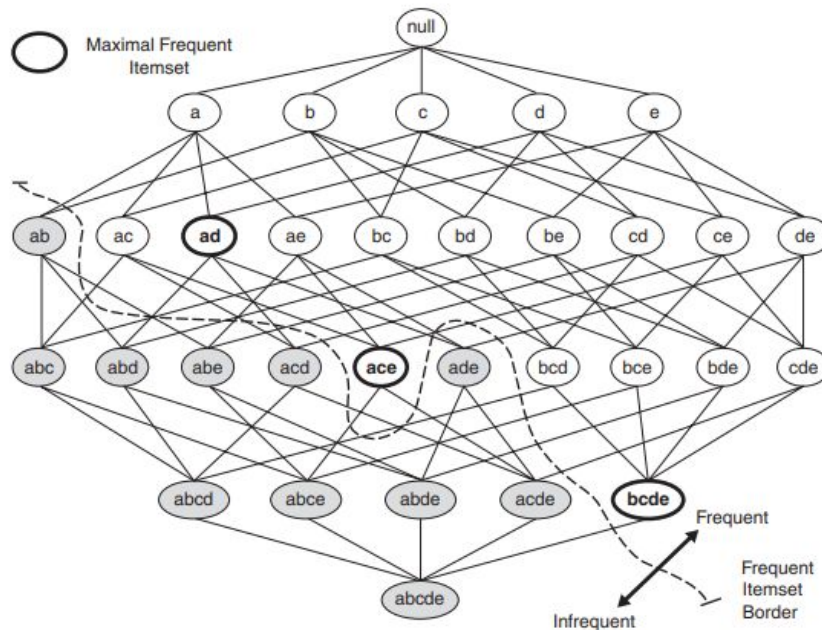
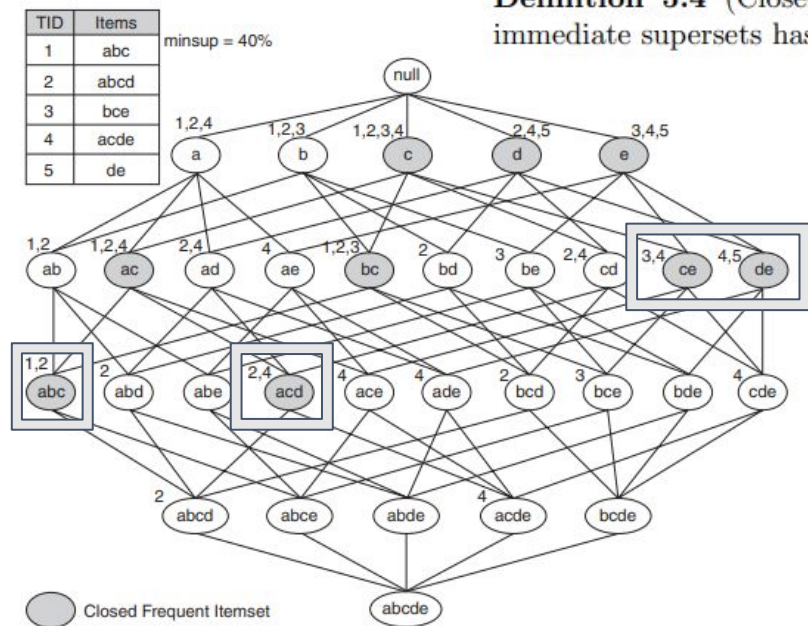


Figure 5.16. Maximal frequent itemset.

# Maximal Frequent item set

너무 Frequent item set이 많으니까.. 대표만 가져오자.



**Definition 5.4 (Closed Itemset).** An itemset  $X$  is closed if none of its immediate supersets has exactly the same support count as  $X$ .

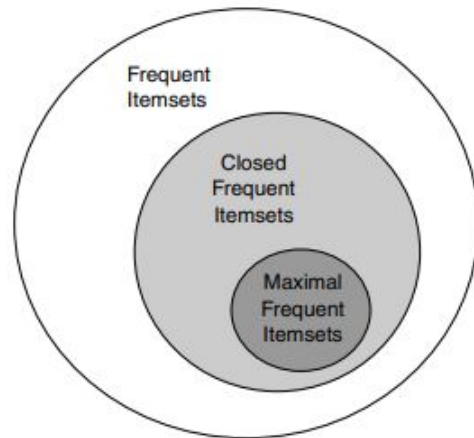
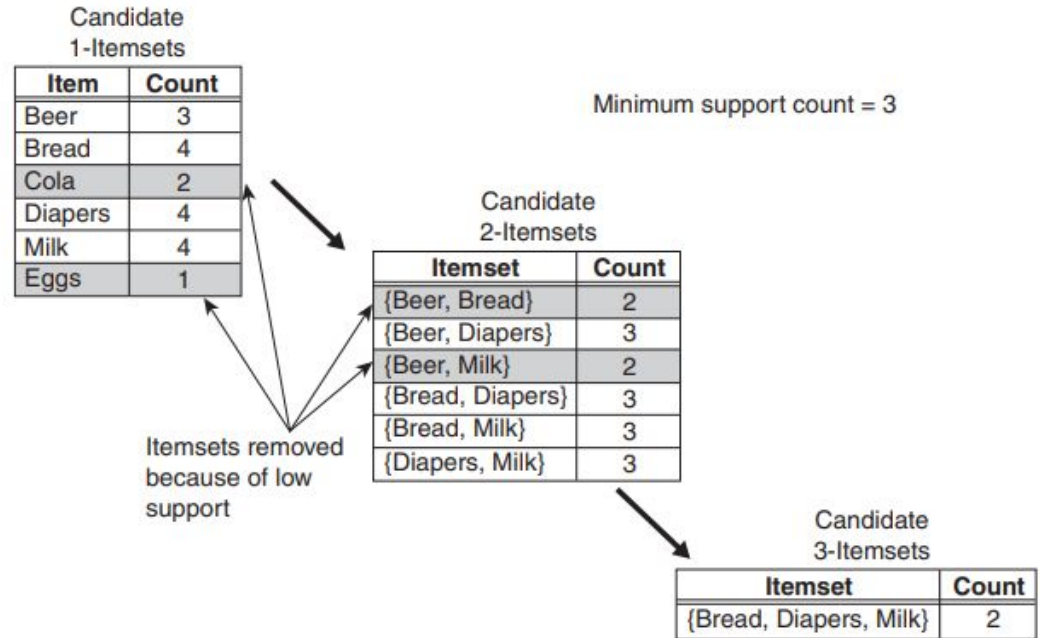


Figure 5.17. An example of the closed frequent itemsets (with minimum support equal to 40%).

# Apriori Algorithm

**Table 5.1.** An example of market basket transactions.

<i>TID</i>	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}



**Figure 5.5.** Illustration of frequent itemset generation using the *Apriori* algorithm.

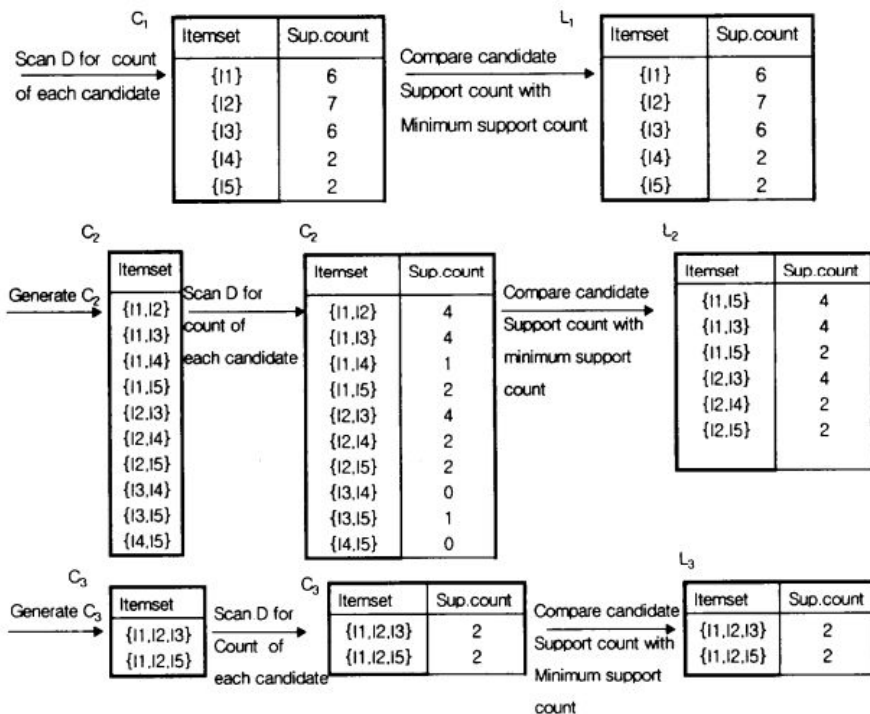
# 예제

TID	List of item IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

1. min sup count = 2로 하고 Apriori algorithm을 수행해주세요.
2. Maximal Frequent Item set을 찾아주세요.
3. item set {I1, I2, I5}에서 최소 confidence = 0.5로 하고 Rule mining을 해주세요.



# 예제



i)  $L_3 = \{I_1, I_2, I_5\}$

R:  $I_1 \wedge I_2 \Rightarrow I_5$ ,  $\text{Conf}(I_1 \wedge I_2, I_5) = 2/4 = 50\%$

R:  $I_1 \wedge I_5 \Rightarrow I_2$ ,  $\text{Conf}(I_1 \wedge I_5, I_2) = 2/2 = 100\%$

R:  $I_2 \wedge I_5 \Rightarrow I_1$ ,  $\text{Conf}(I_2 \wedge I_5, I_1) = 2/2 = 100\%$

R:  $I_1 \Rightarrow I_2 \wedge I_5$ ,  $\text{Conf}(I_1, I_2 \wedge I_5) = 2/6 = 33\%$

R:  $I_2 \Rightarrow I_1 \wedge I_5$ ,  $\text{Conf}(I_2, I_1 \wedge I_5) = 2/7 = 29\%$

R:  $I_5 \Rightarrow I_1 \wedge I_2$ ,  $\text{Conf}(I_5, I_1 \wedge I_2) = 2/2 = 100\%$

## STEP3: Evaluation (Objective)

- Support 와 Confidence의 한계:

Table 5.7. Beverage preferences among a group of 1000 people.

	<i>Coffee</i>	$\overline{Coffee}$	
<i>Tea</i>	150	50	200
$\overline{Tea}$	650	150	800
	800	200	1000

- 단순히 Coffee를 샀을 확률= 0.8
- Tea  $\rightarrow$  Coffee의 confidence = 0.75.

Tea를 샀다는 정보를 알게 되는 것은 손해..?

$$\text{confidence}(A \rightarrow B) = \frac{P(A, B)}{P(A)}$$

# STEP3: Evaluation (Objective)

- Support 와 Confidence의 한계: 특정 물품 하나를 산 사람이 많다면?

Table 5.7. Beverage preferences among a group of 1000 people.

	<i>Coffee</i>	$\overline{Coffee}$	
<i>Tea</i>	150	50	200
$\overline{Tea}$	650	150	800
	800	200	1000

Lift를 쓰자!

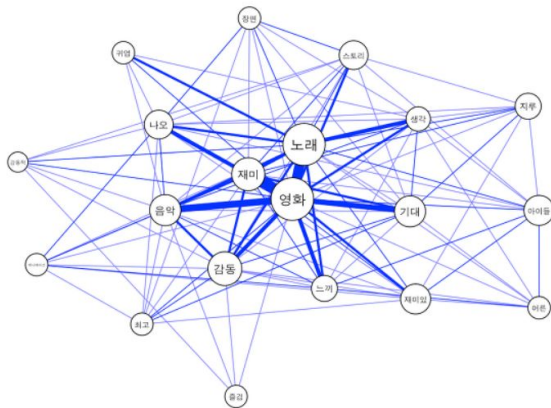
$$\text{Lift}(\text{Tea} \rightarrow \text{Coffee}) = 0.9375$$

$$\text{lift}(A \rightarrow B) = \frac{P(A, B)}{P(A) \cdot P(B)}$$

$$I(A, B) \begin{cases} = 1, & \text{if } A \text{ and } B \text{ are independent;} \\ > 1, & \text{if } A \text{ and } B \text{ are positively related;} \\ < 1, & \text{if } A \text{ and } B \text{ are negatively related.} \end{cases}$$

# STEP3: Evaluation (Subjective)

## 1. Visualization!!



Degree : Node와 연결된 Edge의 개수를 의미. 들어오는 화살표 = in degree, 나가는 화살표 = out degree

원의 크기 = Support

# STEP3: Evaluation (Subjective)

## 2. Template-based approach




- 특정 아이템을 포함하고 규칙을 찾는 방법
- LHS(left hand side), RHS(right hand side)의 값을 조정함.

```
> # subset with left-hand side item : subset(lhs %in% "item")
> rule_interest_lhs <- subset(Epub_rule_2, lhs %in% c("doc_72f", "doc_4ac"))
> inspect(rule_interest_lhs)
```

	lhs	rhs	support	confidence	lift
45	{doc_4ac}	=> {doc_16e}	0.002797381	0.4313725	53.42566
50	{doc_72f}	=> {doc_813}	0.004068917	0.3516484	16.81178

# STEP3: Evaluation (Subjective)

## 3. Subjective interestingness measure

Types of Association Rules		Examples	
	Explainable	Actionable	
Useful Rules	O	O	 <ul style="list-style-type: none"><li>▪ { 남성, 금요일, 기저귀 } → { 맥주 }</li><li>→ 상품 배치, 상품 패키징/번들링, 쿠폰(for 남성) 등</li></ul>
Trivial Rules	O (common sense)	X (result of previous marketing action)	 <ul style="list-style-type: none"><li>▪ 상식: { 노트북컴퓨터 } → { 프린터 }</li><li>▪ 마케팅 결과: '15.5월 '맥주 &amp; 아이스박스' 번들 특판 { 맥주 } → { 아이스박스 }</li></ul>
Inexplicable Rules	X	X	 <ul style="list-style-type: none"><li>▪ 해석 안 되는 경우: { 기초화장품 } → { 자동차와이어 }</li><li>▪ 실행 불가능한 경우: { 강남점, 케익 } → { 보석 } * 강남점 외 타 매장에서는 규칙 안 맞음</li></ul>

## 4. Sequential Pattern Mining

# Sequential Pattern Mining

## 순차 패턴 분석

- 연관성 분석에 **시간의 개념이 포함된** 분석 기법 -> 연관 분석의 응용형
- 구매의 시간적 흐름에 따라 연결될 수 있는 상품들의 선후행적 관계를 도출할 때 사용됨
- “A를 구매하면 **추후에** B도 구매한다” 라는 순서를 반영한 조건 규칙을 생성하는 것이 최종 목표



# Sequential Pattern Mining VS Association Analysis

	순차 패턴 분석	연관 분석
데이터 형태	Sequence Data	List Data
기본 단위	Time	Transactions
분석 지표	Support	Support, Confidence, Lift

# Sequence Data

## Categorical Sequences

- 변수가 순서 의미를 갖는 데이터
- 관측치 형태: event, item. element

Transaction

트랜잭션	구매 품목			
고객 1	A	B	C	D
고객 2	A	B	E	F
고객 3	B	A	G	H
고객 4	B	A	I	J
고객 5	K	L	A	B
고객 6	M	N	A	B
고객 7	A	O	B	P
고객 8	Q	R	S	T
고객 9	U	V	B	W
고객 10	X	Y	B	Z

Sequence

트랜잭션	시점 1	시점 2	시점 3	시점 4
고객 1	A	B	C	D
고객 2	A	B	E	F
고객 3	B	A	G	H
고객 4	B	A	I	J
고객 5	K	L	A	B
고객 6	M	N	A	B
고객 7	A	O	B	P
고객 8	Q	R	S	T
고객 9	U	V	B	W
고객 10	X	Y	B	Z

# Example #1

예시

- 가전 제품의 구매: '냉장고 -> 김치냉장고', '세탁기 -> 건조기' 등의 패턴
- 여행 관련: '항공편 예약 -> 호텔 -> 렌터카', '싱가폴 -> 홍콩' 등의 패턴
- 비디오 대여: 'Star Wars -> Return of the Jedi'

# Example #2

## 멀티플렉스 L사 실제 관람 패턴에 따른 고객 선호 장르 판단

- 멀티플렉스 영화관 L사는 회원가입 시 기재하는 '좋아하는 영화장르(공포, 액션, 멜로 등)' 정보를 바탕으로 동영상 클립이 포함된 영화홍보 이메일을 발송해 왔음.
- 그러나, 그 효과가 미비하다고 판단하여 '좋아하는 영화 장르'와 실제 관람하는 영화장르 패턴이 완전히 일치하지는 않을 것이라는 판단 하에 새로운 이메일 마케팅 전략을 수립하고자 했음.



- 최근 1년간 4회 이상 영화를 관람한 회원 65만 명의 구매이력 정보
- 영화 장르 8종에 대한 시계열적 구매여부(1,0) 데이터



- 순차 패턴 분석
- 시계열 교차판매 30개 규칙 도출 (지지도 20%)
- 4개 시퀀스 내에 실제 좋아하는 장르가 2회 연속되는 경우가 거의 없음

단순히 회원정보의 '좋아하는 영화장르'에 기반한 이메일보다는 도출된 순차연관성 규칙에 기반한 이메일 발송 규칙을 수용하는 것이 효과적

# Sequence

item set을 순차적으로 나열한 리스트를 시퀀스(Sequence)라고 하며, 통상 시간 순서를 사용한다

$A_j$  ( $j=1,2,\dots,n$ )을  $j$ 번째의 item set이라 할 때, 다음과 같이 표기한다.

$$S = \langle A_1, A_2, \dots, A_n \rangle$$

시퀀스에 포함된 항목집합의 수를 **시퀀스의 길이**라 하며, **k-시퀀스**라 한다.

# Sequence

- 두 시퀀스  $s_1 = \langle A_1, A_2, \dots, A_n \rangle$  과  $s_2 = \langle B_1, B_2, \dots, B_m \rangle$  에 대하여,  
 $A_1 \subseteq B_{i_1}, A_2 \subseteq B_{i_2}, \dots, A_n \subseteq B_{i_n}$  이 성립하는  $i_1 < i_2 < \dots < i_n$  이 존재할 때,  $s_1$  은  $s_2$  에 포함되었다고 하며,  $s_1$  을  $s_2$  의 **부분 시퀀스 (subsequence)**라 한다.
- 시퀀스  $s$ 가 어떤 다른 시퀀스에 포함되지 않을 경우, **최대 시퀀스 (maximal sequence)**라 한다.
- 시퀀스  $s$ 에 대한 지지도는  $\text{supp}(s)$ ='시퀀스  $s$ 를 포함하는 고객의 비율'로 정의한다.
- 정해진 최소 지지도 이상을 갖는 시퀀스를 **빈발 시퀀스 (large sequence)**라 하며, 순차패턴 분석은 빈발 시퀀스 중 최대 시퀀스를 찾는 과정이다.

# Example #1 \_ Sequence

다음 시퀀스에서 최소지지도=0.4일 때, 최대 시퀀스는?

고객	고객 시퀀스
1	$\langle \{a\}, \{b\} \rangle$
2	$\langle \{c, d\}, \{a\}, [e, f, g] \rangle$
3	$\langle \{a, h, g\} \rangle$
4	$\langle \{a\}, \{e, g\}, \{b\} \rangle$
5	$\langle \{b\} \rangle$

$s1 = \langle \{a\}, \{b\} \rangle$

$s2 = \langle \{a\}, \{e, g\} \rangle$

\*\*  $\langle \{a\}, \{e\} \rangle$  는 최소지지도를 만족하나,  
s2에 포함되므로 최대 시퀀스는 아니다.

## 4-1. GSP algorithm



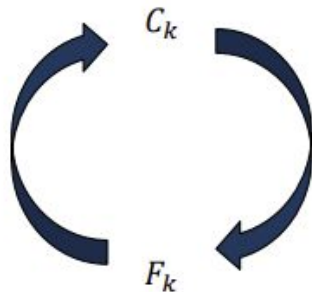
# Generalized Sequential Pattern(GSP)

## GSP algorithm: Apriori 기반

- 크기가 1인 후보집합부터 시작해 빈번한 집합을 찾아가는 방식
- 지지도를 기반으로 길이가 가장 긴 빈번한 순차 패턴 탐색
  - $F_k$ : 길이가 k인 빈번한 패턴(frequent pattern)
  - $C_k$ : 길이가 k인 후보 패턴(candidate pattern)
    - $F_k$ 가 될 가능성이 큰 패턴

Sequence data

트랜잭션	시점 1	시점 2	시점 3	시점 4
고객 1	A	B	C	D
고객 2	A	B	E	F
고객 3	B	A	G	H
고객 4	B	A	I	J
고객 5	K	L	A	B
고객 6	M	N	A	B
고객 7	A	O	B	P
고객 8	Q	R	S	T
고객 9	U	V	B	W
고객 10	X	Y	B	Z



# Generalized Sequential Pattern(GSP)

**Step 1:** Make the first pass over the sequence database  $D$  to yield all the 1-element frequent sequences

**Step 2:** Repeat until no new frequent sequences are found

- **Candidate Generation:** Merge pairs of frequent subsequences found in the  $(k-1)$ th pass to generate candidate sequences that contain  $k$  items
- **Candidate Pruning:** Prune candidate  $k$ -sequences that contain infrequent  $(k-1)$ -subsequences
- **Support Counting:** Make a new pass over the sequence database  $D$  to find the support for these candidate sequences
- **Candidate Elimination:** Eliminate candidate  $k$ -sequences whose actual support is less than minsup (최소지지도)

# Generalized Sequential Pattern(GSP)

- 지지도: 각 시퀀스에 해당 항목이 있는지 없는지 여부
- 지지도 100%는 시퀀스 개수와 동일
- $c_1$ : 데이터 내 모든 아이템

최소 지지도 = 100%(4)

고객 ID	구매 기록(sequence)
1	< (맥주, 땅콩) → (맥주, 오징어) → (신문) >
2	< (신문) → (오징어) → (소주) → (맥주) → (신문, 땅콩) >
3	< (맥주, 신문) → (오징어) → (오징어, 땅콩) → (신문) >
4	< (맥주, 오징어) → (신문, 양주) → (신문, 오징어) >

$c_1$	지지도
맥주	4
오징어	4
신문	4
땅콩	3
소주	1
양주	1

# Generalized Sequential Pattern(GSP)

- $C_{k+1}$  은  $F_k$ 로부터 생성
- $F_k$ 로 생성된 항목의 모든 조합
  - 다른 시점(Temporal join)
  - 한 시점(Non-temporal join)

$F_1$
맥주
오징어
신문

$F_1$	맥주	오징어	신문
맥주	*	*	*
오징어	*	*	*
신문	*	*	*

$F_1$	맥주	오징어	신문
맥주		*	*
오징어			*
신문			

$C_2$	
다른 시점	한 시점
맥주 → 맥주	(맥주, 오징어)
맥주 → 오징어	(맥주, 신문)
맥주 → 신문	(오징어, 신문)
오징어 → 오징어	
오징어 → 맥주	
오징어 → 신문	
신문 → 신문	
신문 → 오징어	
신문 → 맥주	

# Generalized Sequential Pattern(GSP)

- $C_{k+1}$  은  $F_k$  로부터 생성
- $F_k$  로 생성된 항목의 모든 조합
  - 다른 시점(Temporal join)
  - 한 시점(Non-temporal join)

$C_2$	
다른 시점	한 시점
맥주 → 맥주	맥주, 오징어
맥주 → 오징어	맥주, 신문
맥주 → 신문	오징어, 신문
오징어 → 오징어	
오징어 → 맥주	
오징어 → 신문	
신문 → 신문	
신문 → 오징어	
신문 → 맥주	

$C_2$	지지도
맥주 → 맥주	1
맥주 → 오징어	3
맥주 → 신문	4
오징어 → 오징어	2
오징어 → 맥주	0
오징어 → 신문	4
신문 → 신문	3
신문 → 오징어	3
신문 → 맥주	2
(맥주, 오징어)	2
(맥주, 신문)	1
(신문, 오징어)	1

$F_2$
맥주 → 신문
오징어 → 신문

데이터 내 가장 빈번한 순차 패턴

# Generalized Sequential Pattern(GSP)

- $C_{k+1}$  을 더 이상 만들 수 없을 때 종료
- 후보 패턴을 만드는데 시간 걸림

패턴 길이 ( $k$ )	후보 패턴 ( $C_k$ )	빈번 패턴 ( $F_k$ )
1	<(맥주)>, <(신문)>, <(오징어)>, <(땅콩)>, <(소주)>, <(양주)>	<(맥주)>, <(신문)>, <(오징어)>
2	<(맥주), (맥주)>, <(맥주, 신문)>, <(맥주), (신문)>, <(맥주, 오징어)>, <(맥주), (오징어)>, <(신문), (맥주)>, <(신문), (신문)>, <(신문, 오징어)>, <(오징어), (맥주)>, <(오징어), (신문)>, <(오징어), (오징어)>	<(맥주), (신문)>, <(오징어), (신문)>
3	없음	없음

## 4-2. PrefixSpan algorithm

# PrefixSpan

- 후보 패턴을 만들지 않으면서 빈번한 패턴을 찾는 방법  
=> **only consider patterns that exist in the database**
- database projection
- depth-first search
- GSP보다 대용량 데이터에 적합



# Example

This is the input:

Sequence database

S1	< (참이슬, 새우깡) → (메로나) → (참이슬) >
S2	< (참이슬, 새우깡) → (새우깡) → (메로나) >
S3	< (새우깡) → (메로나) → (레드불) >
S4	< (새우깡) → (참이슬, 새우깡) → (메로나) >

최소 지지도: 3

# Example: Step 1

count the support of each item by scanning the database:

Sequence database

S1	< (참이슬, 새우깡) → (메로나) → (참이슬) >
S2	< (참이슬, 새우깡) → (새우깡) → (메로나) >
S3	< (새우깡) → (메로나) → (레드불) >
S4	< (새우깡) → (참이슬, 새우깡) → (메로나) >

최소 지지도: 3

Result:

항목	지지도
참이슬	3
새우깡	4
메로나	4
레드불	1

# Example: Step 2

eliminate infrequent items:

Sequence database

S1	< (참이슬, 새우깡) → (메로나) → (참이슬) >
S2	< (참이슬, 새우깡) → (새우깡) → (메로나) >
S3	< (새우깡) → (메로나) → (레드불) >
S4	< (새우깡) → (참이슬, 새우깡) → (메로나) >

최소 지지도: 3

Result:

항목	지지도
참이슬	3
새우깡	4
메로나	4
<del>레드불</del>	<del>1</del>

Those are the sequential patterns containing one item.

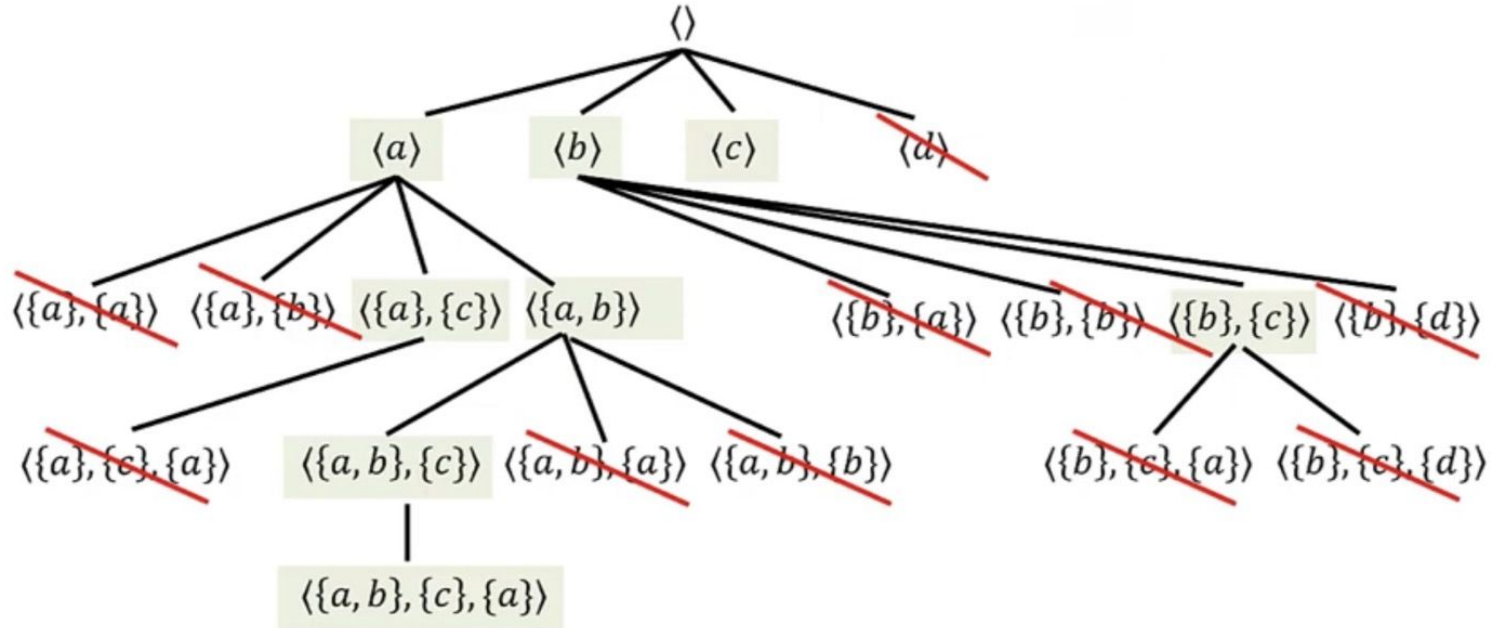
PrefixSpan then extends each item recursively.

레드불은 최소 지지도를 넘지 못했으므로 더 이상 고려하지 않는다.

Let's start with < (참이슬) > →

# Example: PrefixSpan Tree

PrefixSpan performs a depth-first search :



# Example: Step 3

find patterns starting with  $\langle a \rangle$ .

PrefixSpan does a **database projection** with  $\langle a \rangle$  :

- **Database Projection**

to keep only the sequences **containing  $\langle a \rangle$**  .

for these sequences, we **delete the first occurrence of  $\langle a \rangle$  and everything that appears before.**

# Example: Step 3

find patterns starting with < (참이슬) >.

Prefix

PrefixSpan does a database projection with < (참이슬) > :

Sequence database

S1	< (참이슬, 새우깡) → (메로나) → (참이슬) >
S2	< (참이슬, 새우깡) → (새우깡) → (메로나) >
S3	< (새우깡) → (메로나) → (레드불) >
S4	< (새우깡) → (참이슬, 새우깡) → (메로나) >



Projected database of < (참이슬) >

Suffix

S1	< <del>(참이슬)</del> , 새우깡) → (메로나) → (참이슬) >
S2	< <del>(참이슬)</del> , 새우깡) → (새우깡) → (메로나) >
S3	< <del>(새우깡) → (메로나) → (레드불)</del> >
S4	< <del>(새우깡)</del> → <del>(참이슬)</del> , 새우깡) → (메로나) >

최소 지지도: 3

# Example: Step 3

Then, PrefixSpan counts the support of each sequential pattern starting with < (참이슬) > that has one more item :

Projected database of < (참이슬) >

S1	< ( , 새우깡) → (메로나) → (참이슬) >
S2	< ( , 새우깡) → (새우깡) → (메로나) >
S4	< ( , 새우깡) → (메로나) >

최소 지지도: 3

Result:

항목	지지도
< (참이슬) → (참이슬) >	1
< (참이슬) → (새우깡) >	1
< (참이슬) → (메로나) >	3
< (참이슬, 새우깡) >	3

# Example: Step 3

Then, infrequent patterns are removed :

Projected database of < (참이슬) >

S1	< ( , 새우깡) → (메로나) → (참이슬) >
S2	< ( , 새우깡) → (새우깡) → (메로나) >
S4	< ( , 새우깡) → (메로나) >

최소 지지도: 3

PrefixSpan then extends each pattern recursively.

Result:

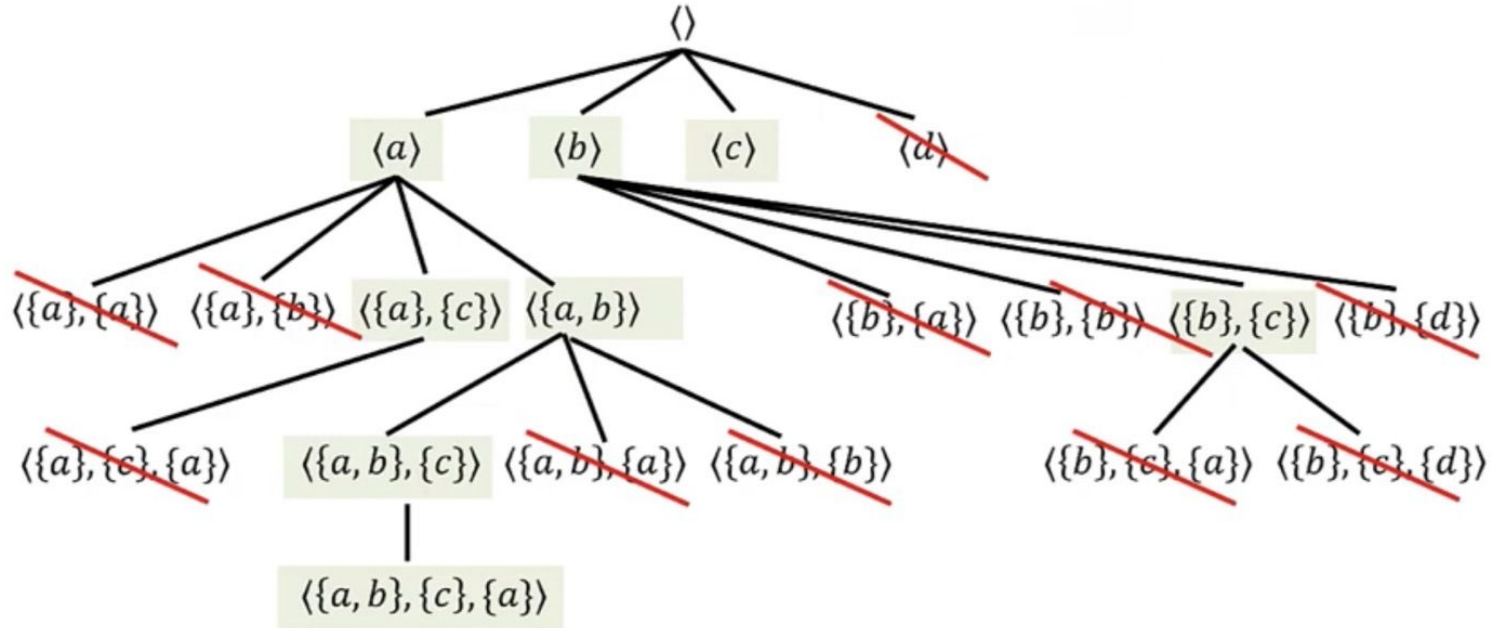
항목	지지도
<del>&lt; (참이슬) → (참이슬) &gt;</del>	<del>1</del>
<del>&lt; (참이슬) → (새우깡) &gt;</del>	<del>1</del>
< (참이슬) → (메로나) >	3
< (참이슬, 새우깡) >	3

Let's start with < (참이슬) → (메로나) > →



# Example: PrefixSpan Tree

PrefixSpan performs a depth-first search :



# Example: Step 4

PrefixSpan does a database projection with  $\langle (\text{참이슬}) \rightarrow (\text{메로나}) \rangle$  :

Projected database of  $\langle (\text{참이슬}) \rangle$

S1	$\langle (\_, \text{새우깡}) \rightarrow (\text{메로나}) \rightarrow (\text{참이슬}) \rangle$
S2	$\langle (\_, \text{새우깡}) \rightarrow (\text{새우깡}) \rightarrow (\text{메로나}) \rangle$
S4	$\langle (\_, \text{새우깡}) \rightarrow (\text{메로나}) \rangle$



Projected database of  $\langle (\text{참이슬}) \rightarrow (\text{메로나}) \rangle$

S1	<del><math>\langle (\_, \text{새우깡}) \rightarrow (\text{메로나}) \rightarrow (\text{참이슬}) \rangle</math></del>
S2	<del><math>\langle (\_, \text{새우깡}) \rightarrow (\text{새우깡}) \rightarrow (\text{메로나}) \rangle</math></del>
S4	<del><math>\langle (\_, \text{새우깡}) \rightarrow (\text{메로나}) \rangle</math></del>



S1	$\langle (\text{참이슬}) \rangle$
----	--------------------------------

## Example: Step 4

Then, PrefixSpan counts the support of each sequential pattern starting with  $\langle \text{(참이슬)} \rightarrow \text{(메로나)} \rangle$  that has one more item:

Projected database of  $\langle \text{(참이슬)} \rightarrow \text{(메로나)} \rangle$

S1	$\langle \text{(참이슬)} \rangle$
----	--------------------------------

Result:

항목	지지도
<del><math>\langle \text{(참이슬)} \rightarrow \text{(메로나)} \rightarrow \text{(참이슬)} \rangle</math></del>	<del>1</del>

This pattern is infrequent.

최소 지지도를 만족하는 패턴이 더 이상 없으면 Stop!

Then PrefixSpan try to find patterns starting with  $\langle \text{(참이슬, 새우깡)} \rangle \rightarrow$

# Example: Step 5

PrefixSpan does a database projection with  $\langle \text{(참이슬, 새우깡)} \rangle$  :

Projected database of  $\langle \text{(참이슬)} \rangle$

S1	$\langle \_, \text{새우깡} \rangle \rightarrow (\text{메로나}) \rightarrow (\text{참이슬})$
S2	$\langle \_, \text{새우깡} \rangle \rightarrow (\text{새우깡}) \rightarrow (\text{메로나})$
S4	$\langle \_, \text{새우깡} \rangle \rightarrow (\text{메로나})$



Projected database of  $\langle \text{(참이슬, 새우깡)} \rangle$

S1	$\langle \_, \text{새우깡} \rangle \rightarrow (\text{메로나}) \rightarrow (\text{참이슬})$
S2	$\langle \_, \text{새우깡} \rangle \rightarrow (\text{새우깡}) \rightarrow (\text{메로나})$
S4	$\langle \_, \text{새우깡} \rangle \rightarrow (\text{메로나})$



S1	$\langle (\text{메로나}) \rightarrow (\text{참이슬}) \rangle$
S2	$\langle (\text{새우깡}) \rightarrow (\text{메로나}) \rangle$
S4	$\langle (\text{메로나}) \rangle$

# Example: Step 5

Then, PrefixSpan counts the support of each sequential pattern starting with  $\langle (\text{참이슬}, \text{새우깡}) \rangle$  that has one more item:

Projected database of  $\langle (\text{참이슬}, \text{새우깡}) \rangle$

S1	$\langle (\text{메로나}) \rightarrow (\text{참이슬}) \rangle$
S2	$\langle (\text{새우깡}) \rightarrow (\text{메로나}) \rangle$
S4	$\langle (\text{메로나}) \rangle$

Result:

항목	지지도
$\langle (\text{참이슬}, \text{새우깡}) \rightarrow (\text{메로나}) \rangle$	3
<del><math>\langle (\text{참이슬}, \text{새우깡}) \rightarrow (\text{새우깡}) \rangle</math></del>	<del>1</del>
<del><math>\langle (\text{참이슬}, \text{새우깡}) \rightarrow (\text{참이슬}) \rangle</math></del>	<del>1</del>

Then, PrefixSpan try to find patterns starting with  $\langle (\text{참이슬}, \text{새우깡}) \rightarrow (\text{메로나}) \rangle \rightarrow$

# Example: Step 6

PrefixSpan does a database projection with  $\langle (\text{참이슬}, \text{새우깡}) \rightarrow (\text{메로나}) \rangle$  :

Projected database of  $\langle (\text{참이슬}, \text{새우깡}) \rangle$

S1	$\langle (\text{메로나}) \rightarrow (\text{참이슬}) \rangle$
S2	$\langle (\text{새우깡}) \rightarrow (\text{메로나}) \rangle$
S4	$\langle (\text{메로나}) \rangle$



Projected database of  $\langle (\text{참이슬}, \text{새우깡}) \rightarrow (\text{메로나}) \rangle$

S1	$\langle (\text{참이슬}) \rangle$
----	--------------------------------

## Example: Step 6

Then, PrefixSpan counts the support of each sequential pattern starting with  $\langle (\text{참이슬}, \text{새우깡}) \rightarrow (\text{메로나}) \rangle$  that has one more item:

Projected database of  $\langle (\text{참이슬}, \text{새우깡}) \rightarrow (\text{메로나}) \rangle$

>	S1	$\langle (\text{참이슬}) \rangle$
---	----	--------------------------------

Result:

항목	지지도
<del><math>\langle (\text{참이슬}, \text{새우깡}) \rightarrow (\text{메로나}) \rightarrow (\text{참이슬}) \rangle</math></del>	<del>1</del>

This pattern is infrequent!

Then, PrefixSpan tries to find patterns starting with  $\langle (\text{새우깡}) \rangle \rightarrow$

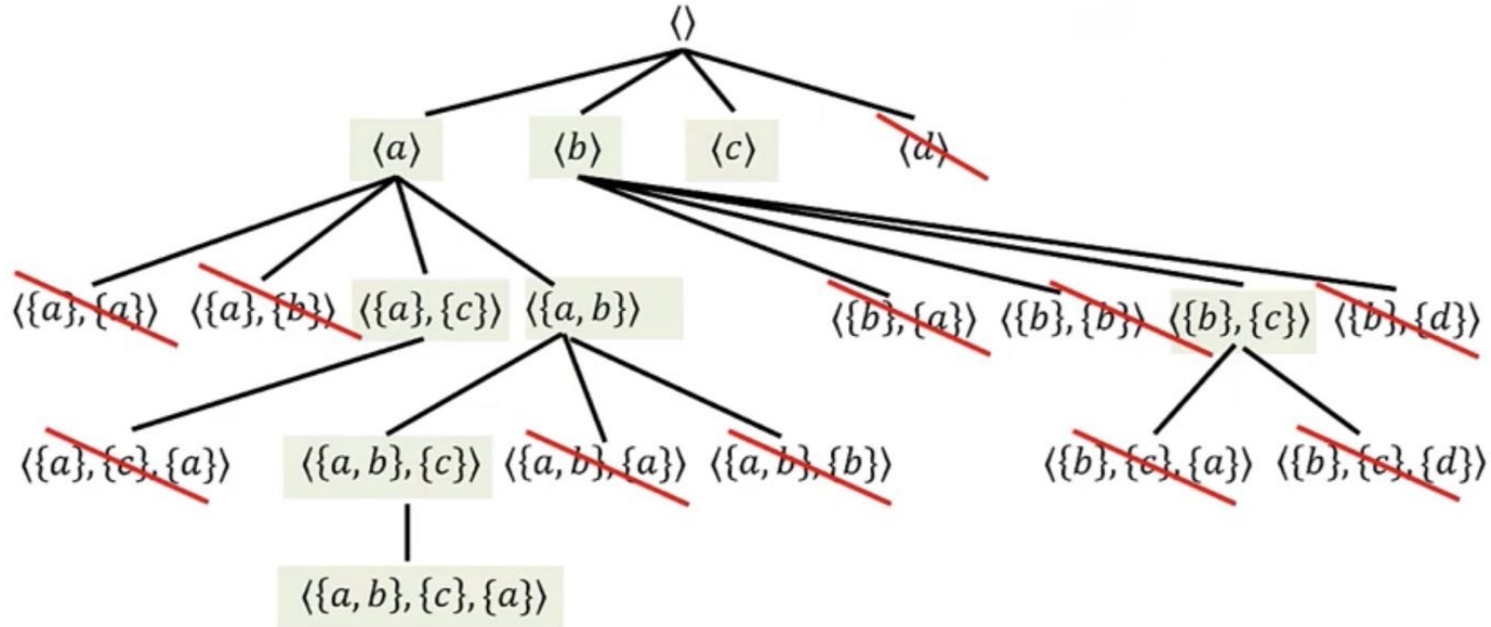
## Example: Step 7.....

지지도가 최소지지도 이상이었던 < (새우깡) >과 < (메로나) >에 대해서도 동일한 과정을 반복한다.



# Example: PrefixSpan Tree

PrefixSpan performs a depth-first search :



End of the session!