Association Rule Mining

Jeonghun Yoon

Motivation

- Association rule mining은 데이터들 간의 연결(Link)를 찾는 것이다.
- Input
  - 고객들 장바구니(transaction)의 상품들
    1. 고객 1의 장바구니 $T_1 = (t_{11}, \dots, t_{1n})$
    2. 고객 2의 장바구니 $T_2 = (t_{21}, \dots, t_{2m})$
    3. .....
    4. 고객 J의 장바구니 $T_j = (t_{j1}, \dots, t_{jk})$
- Output
  - 상품들 사이의 관계 (or 법칙 or rule)
  - $\{X_1, \dots, X_n\} \rightarrow Y$

Confidence

$\{x_1, x_2, \dots, x_n, Y, \dots\}$

$\{x_1, x_2, \dots, x_n, Y, \dots\}$

$\{x_1, x_2, \dots, x_n, Y, \dots\}$

$\{x_1, x_2, \dots, x_n, \dots\dots\}$

$\{x_1, x_2, \dots, x_n, Y, \dots\}$

$\{x_1, x_2, \dots, x_n, Y, \dots\}$

$\{x_1, x_2, \dots, x_n, Y, \dots\}$

$\{x_1, x_2, \dots, x_n, Y, \dots\}$

$\{x_1, x_2, \dots, x_n, \dots\dots\}$

$\{x_1, x_2, \dots, x_n, Y, \dots\}$

10개의 아이템의 집합중에서, Y를 포함하는 집합은 8개이다.

$x_1, \dots, x_n$을 포함하는 집합중에서 Y가 발생할(보일) 확률은 $\frac{8}{10}$이다.

$\frac{8}{10}$은 충분히 큰 확률이다.

$x_1, \dots, x_n$과 Y는 association rule이 존재한다고 생각할 수 있다.

association rule이 존재한다고 받아 드리기 위한 Y의 발생(빈도) 확률을 confidence 라고 한다.

Support

$\{x_1, x_2, \dots, x_n, Y, \dots\}$

$\{x_1, x_8, \dots, x_n, Y, \dots\}$

$\{x_1, x_2, \dots\dots\dots\dots\}$

$\{x_1, x_2, \dots, x_n, \dots\dots\}$

$\{x_1, x_3, \dots, x_n, Y, \dots\}$

$\{x_1, x_4, \dots, x_n, Y, \dots\}$

$\{x_1, x_2, \dots, x_n, Y, \dots\}$

$\{x_1, x_2, \dots\dots\dots\dots\}$

$\{x_1, x_2, \dots\dots\dots\dots\}$

$\{x_1, x_3, \dots, x_n, Y, \dots\}$

10개의 아이템의 집합중에서, $x_1, \dots, x_n, Y$를 모두 포함하는 집합은 2개이다.

전체의 집합중에서 $x_1, \dots, x_n, Y$가 동시에 존재 할 확률은 $\frac{2}{10}$이다.

$\frac{2}{10}$은 큰 확률은 아니다.

$x_1, \dots, x_n$과 Y는 association rule이 존재한다고 판단하기 어려울 수 있다.

association rule이 있다고 받아 드리기 위한 $x_1, \dots, x_n, Y$의 발생(빈도) 확률을 support 라고 한다.

Confidence & Support

- Associaion Rule의 유효성을 측정할 수 있는 방법
  - Confidence
    - ① IF X and Y THEN Z with confidence c.
    - ② X와 Y가 같은 시장 바구니에 있을 때, Z가 그 시장 바구니에 들어 있을 확률이 c%이다.
  - Support
    - ① IF X and Y THEN Z with support s.
    - ② Association Rule은 시장 바구니 전체에서 s%에서 성립한다.

Confidence & Support

Example)

Transaction	Items
12345	A B C
12346	A C
12347	A D
12348	B E F

$$(A \rightarrow C) : c = 66.6\% , s = 50\%$$

$$(C \rightarrow A) : c = 100\% , s = 50\%$$

Algorithm

- Association Rule Mining 알고리즘의 목표
  - minimum support와 minimum confidence보다 큰 값을 가지는 rules를 찾아라.

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

minimum support = 60%

minimum cofidence = 50%

Rule ??

Algorithm

- Association Rule Mining 알고리즘의 framework

Part 1)

○ 데이터 집합에서 minimum support 보다 큰 support 값을 가지는, 모든 frequent patterns p를 찾아라.

※ Frequent pattern : 데이터 집합에서 자주 발생하는 pattern
minimum support 보다 큰 support를 가지는 패턴

Part 2)

○ 각각의 pattern p에 대하여 공집합이 아닌 p의 부분집합 s를 모두 생성한다.

○ p의 모든 부분집합 s에 대하여 $\frac{\sup(p)}{\sup(s)}$가 minimum confidence값 보다 크면, association rule $s \rightarrow (p - s)$를 반환한다.

Frequent pattern을 찾는 문제가 훨씬 더 어렵다.

Basic concept for algorithm

Transaction	Items
12345	A B C
12346	A C
12347	A D
12348	B E F

Itemset	Support
A	75%
B	50%
C	50%
A, C	50%

rule A → C를 계산하는 방법은?

$$s(A) = 75\%$$

$$s(A, C) = 50\%$$

$$s = 50\%$$

$$c = \frac{s(A,C)}{S(A)} = 66.6\%$$

Part 2보다 Part 1이 구하기가 훨씬 어렵다.

어떻게 Frequent Pattern을 구할 수 있을까?

Frequent Pattern

- Frequent pattern itemset의 성질
  - frequent pattern의 부분집합은 모두 frequent pattern이다.
    - ① $\{beer, chips, nuts\}$가 frequent하면 $\{beer, chips\}$ 또한 frequent하다.
  - infrequent pattern의 superset(그 자신을 부분집합으로 포함하는 상위집합)은 역시 infrequent하다. infrequent하면, 그것들의 superset은 frequent pattern에서 제외하면 된다.
    - ① $\{beer, chips\}$가 infrequent하면 $\{beear, chips, nuts\}$가 infrequent하다.

Apriori algorithm

- DB를 1회 scan하면서 1-frequent item을 찾는다.
- 각각 step k에 대하여,
  - k-fequent items으로 부터 $(k+1)$개의 item을 가지는 후보군을 생성한다.
  - DB를 1회 scan하면서 infrequent item을 제거(pruning)한다.
- 후보군 집합이 생성되지 않을 때까지 반복 후 종료한다.

- 후보군 생성 : k개의 item에서, $k+1$개의 item을 생성하는 중이라고 가정
  - Step 1 (self joining) : $(k-1)$ prefix가 동일한, 2개의 k-frequent pattern을 합친다.
  - step 2 (pruning) : 합쳐진 후보군에서, infrequent한 k-pattern을 가지고 있는 후보를 지운다.

ex) $L_3 = \{abc\ ,\ abd\ ,\ acd\ ,\ ace\ ,\ bcd\}$

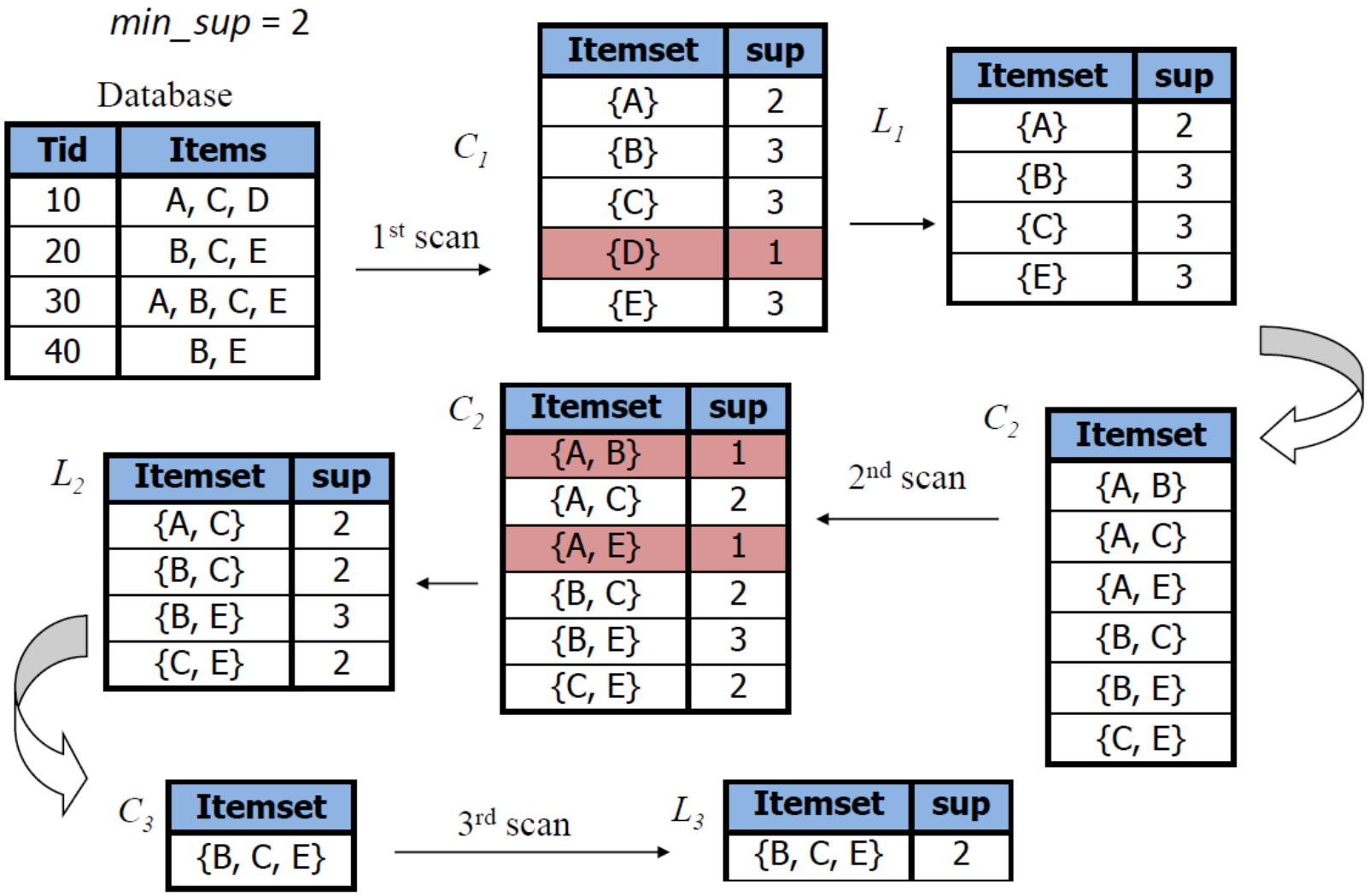
① self joinging : $L_3 * L_3$

- abc and $abd \rightarrow abcd$
- acd and $ace \rightarrow acde$

② pruning :

- $acde$는 지워진다. 왜냐하면 ade가 L_3(frequent pattern)에 속하지 않기 때문이다.

③ $C_4 = \{abcd\}$


min_sup = 2
Database
Tid | Items
10 | A, C, D
20 | B, C, E
30 | A, B, C, E
40 | B, E
1st scan
C1
Itemset | sup
{A} | 2
{B} | 3
{C} | 3
{D} | 1
{E} | 3
L1
Itemset | sup
{A} | 2
{B} | 3
{C} | 3
{E} | 3
C2
Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}
2nd scan
C2
Itemset | sup
{A, B} | 1
{A, C} | 2
{A, E} | 1
{B, C} | 2
{B, E} | 3
{C, E} | 2
L2
Itemset | sup
{A, C} | 2
{B, C} | 2
{B, E} | 3
{C, E} | 2
C3
Itemset
{B, C, E}
3rd scan
L3
Itemset | sup
{B, C, E} | 2

- Apriori algorithm의 문제점
  - Huge candidates
  - Mutiple scans : $(n+1)$ scans, n : 가장 긴 pattern의 길이

그럼 다른 방법은 없을까? 있다! FP-Tree를 이용하자!

FP-Tree algorithm

Example)

Transaction Data Set

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

Step 1 : Frequent Pattern tree construction

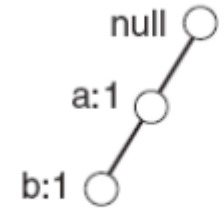
PASS 1

1. Data set을 1번 scan한다.
2. Scan과 동시에 각 single data item에 대한 support를 구해준다.
3. minimum support를 넘지 못하는 item은 무시한다.
4. item을 support의 크기 순에 따라 내림차순(descending order)로 정렬한다.

   → (FT-tree 생성시 사용한다. 공통된 prefix가 공유 될 수 있다.)

PASS 2

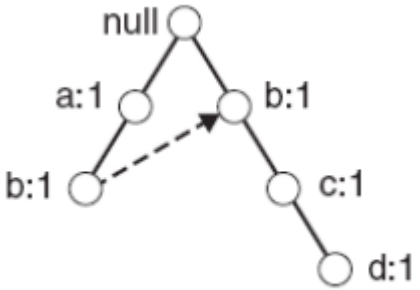
transaction 1: $\{a, b\}$



(i) After reading TID=1

a, b 2개의 노드와 path ($null$-a-b)를 생성한다.
a와 b의 count를 각각 1이라고 한다.

transaction 2: $\{b, c, d\}$



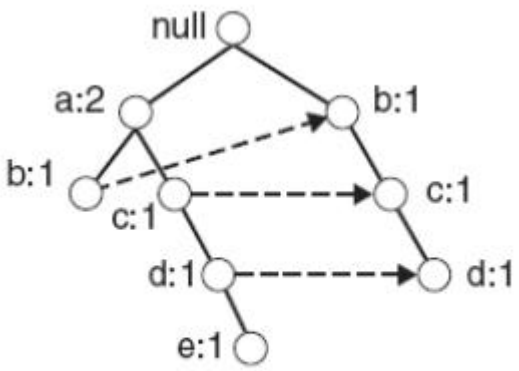
(ii) After reading TID=2

b, c, d 3개의 노드와 path ($null$-b-c-d)를 생성한다.
b와 c와 d의 count를 각각 1이라고 한다.

transaction 1과 2가 각각 b를 포함하지만,
서로 같은 prefix를 공유하지 않기 때문에,
path는 같지 않다.

같은 b끼리는 link로 연결한다. (single linked list)

transaction 3: $\{a, c, d, e\}$



(iii) After reading TID=3

transaction 1과 같은 prefix item a를 공유한다.
따라서 transaction 1과 3는 노드 a에서 겹쳐지고,
노드 a의 frequency는 2가 된다.

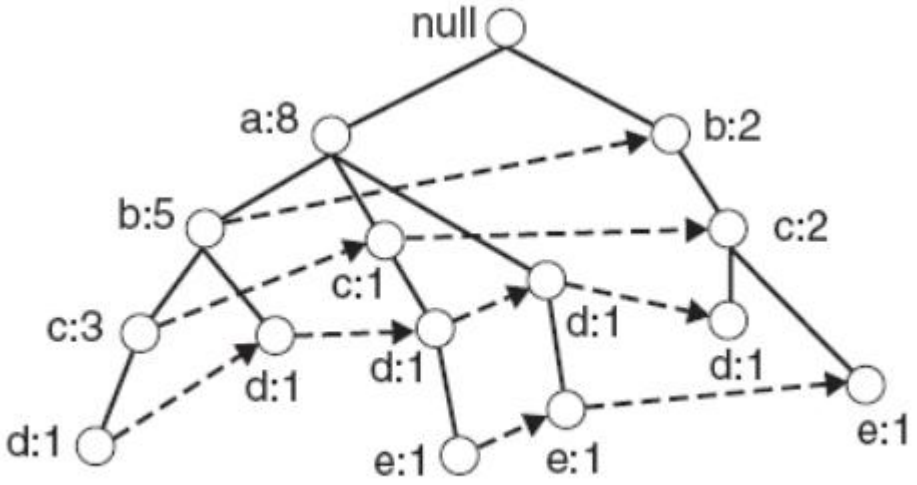
노드 c, d, e를 생성하고 count를 각각 1로 한다.

노드 c와 d는 각각 링크로 연결한다.
(single linked list)

FP-Tree for data

Transaction Data Set

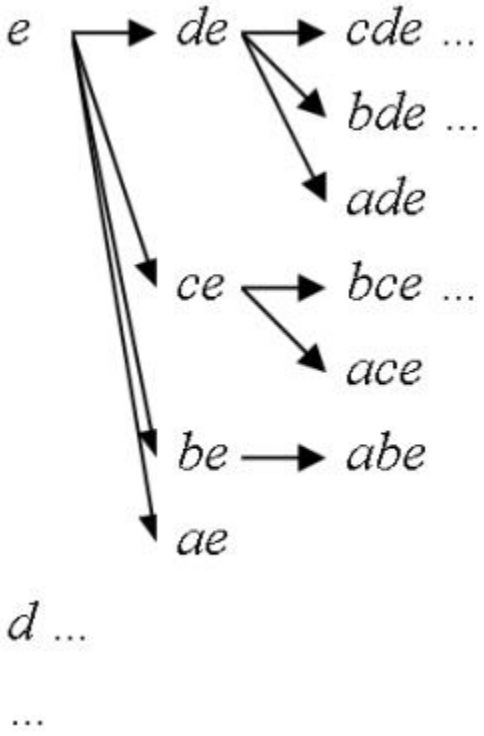
TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



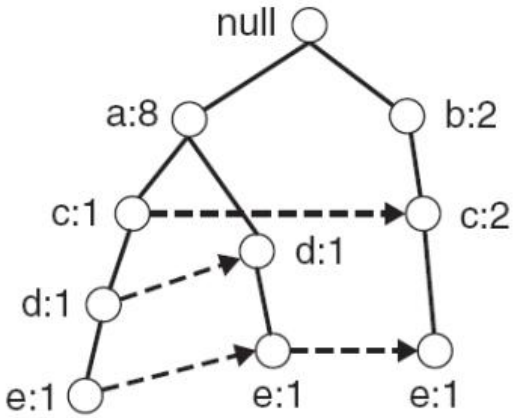
(iv) After reading TID=10

Step 2 : Frequent Itemset Generation

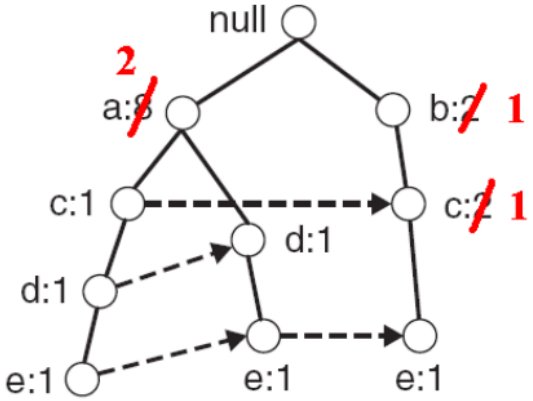
- Bottom-up algorithm



- min_sup = 2일때 *e*로 끝나는 모든 frequent item을 찾자.
  - 먼저, *e*로 끝나는 prefix path sub-tree를 구하자.

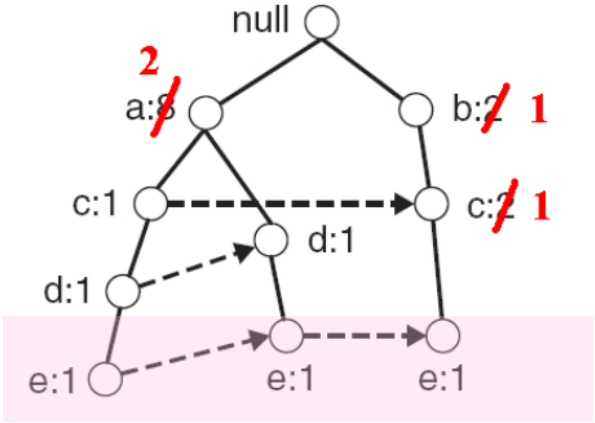


  - tree의 node count를 update 하자.

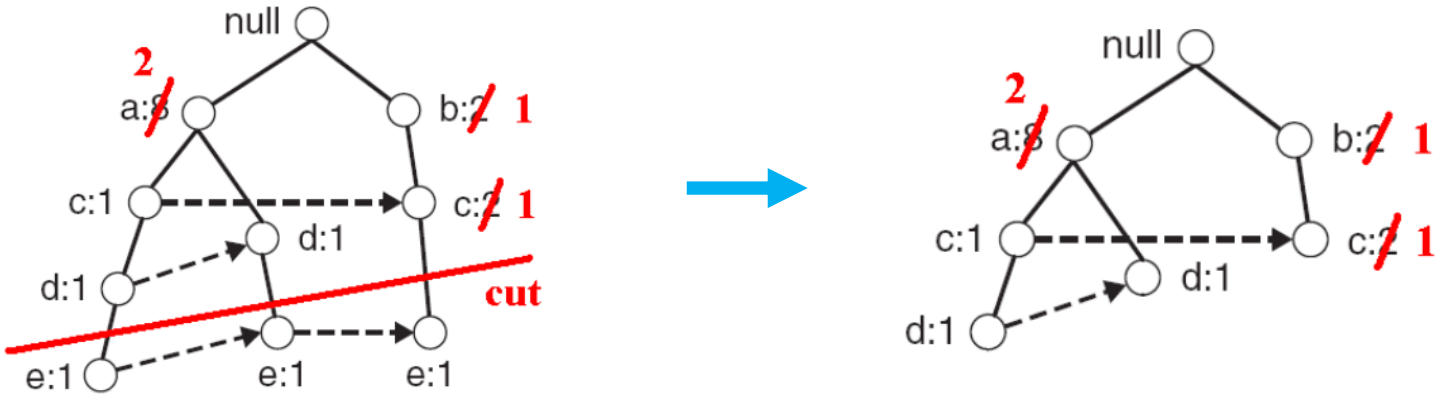


○ bottom line의 linked list를 따라가면서 e에 대한 count를 더한다. 만약 count의 합이 minimum support보다 크면, e는 frequent item이다.

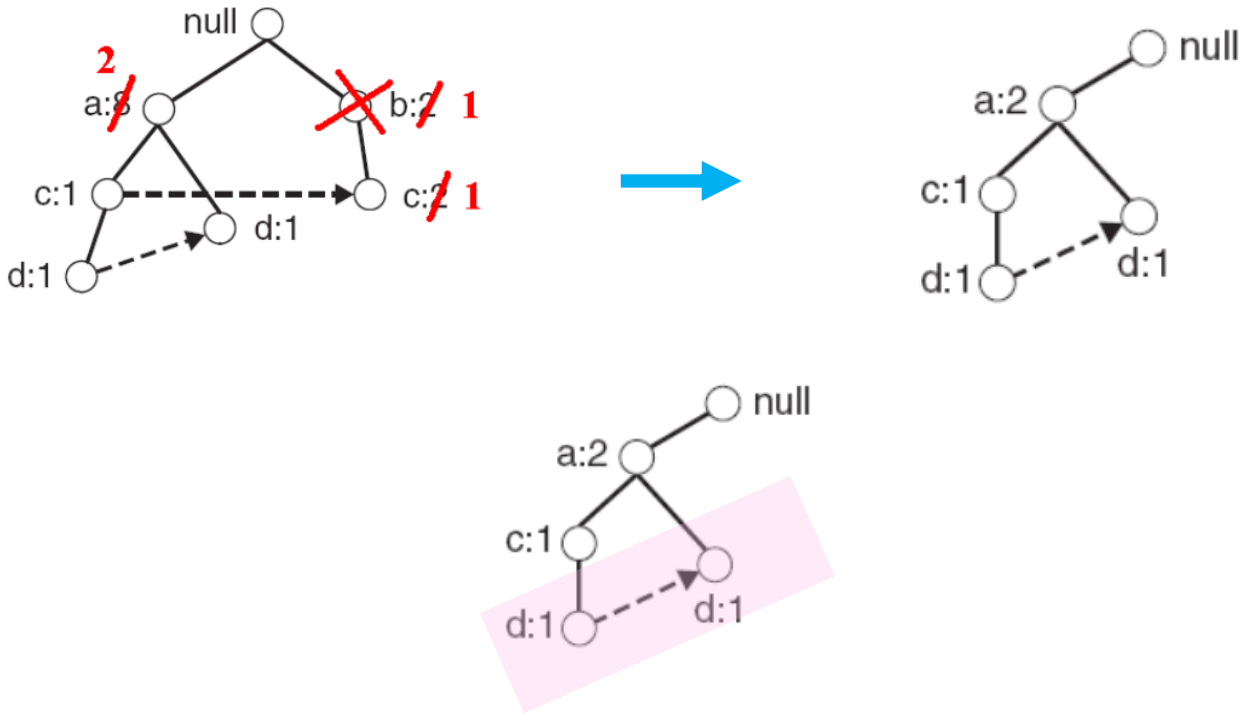
① 합이 3이므로 e는 frequent item이다.



○ e가 freqeunt item이므로 e로 끝나는 frequent itemset을 찾자. 여기서의 후보군은 de, ce, be, ae이다. 이것을 위해서, e를 포함한 노드들을 모두 지운다.



○ d로 끝나는 sub-tree를 구하자.

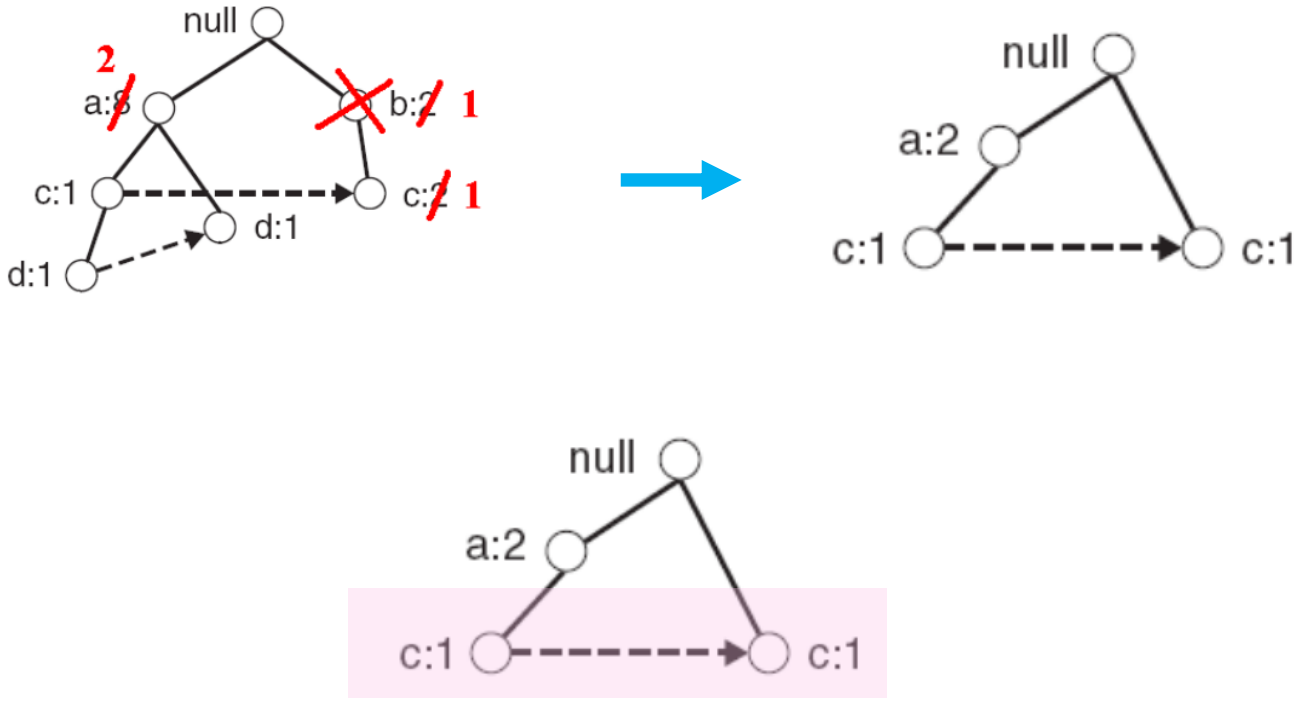


① 합이 2이므로 ed는 frequent item이다.

② 재귀적으로 c로 끝나는 sub-tree, a로 끝나는 sub-tree를 호출하여 확인한다.

○ $\{d, e\}, \{a, d, e\}$가 frequent itemset

○ c로 끝나는 sub-tree를 구하자.

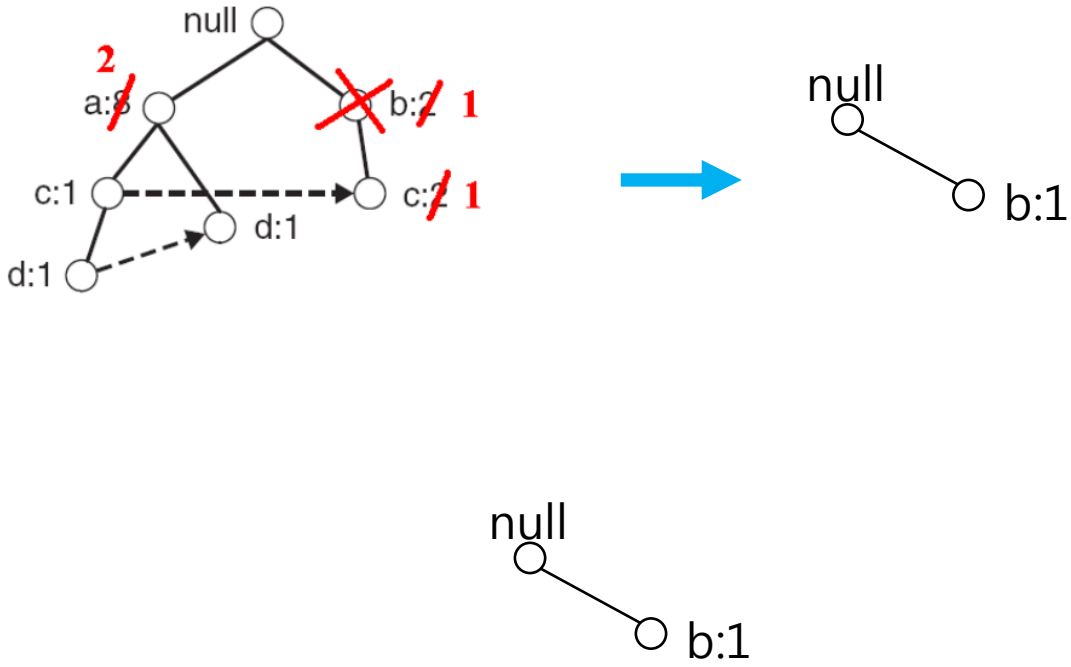


① 합이 2이므로 ce는 frequent item이다.

② 재귀적으로 a로 끝나는 sub-tree를 호출하여 확인한다.

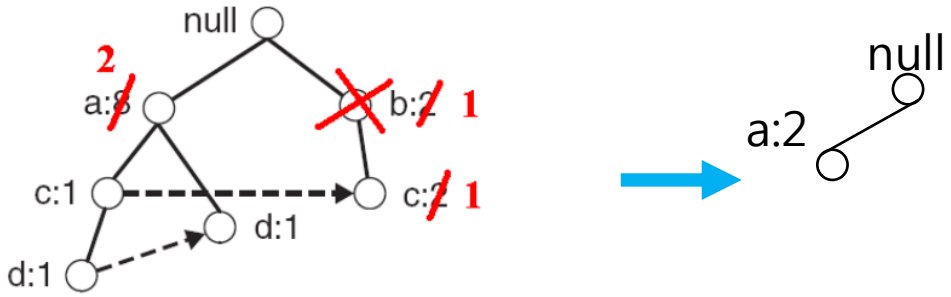
○ $\{c, e\}$가 frequent itemset

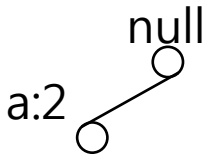
○ b로 끝나는 sub-tree를 구하자.



① 합이 1이므로 be는 frequent item이 아니다.
② 더 이상 재귀과정을 호출 할 필요 없이, prune한다.

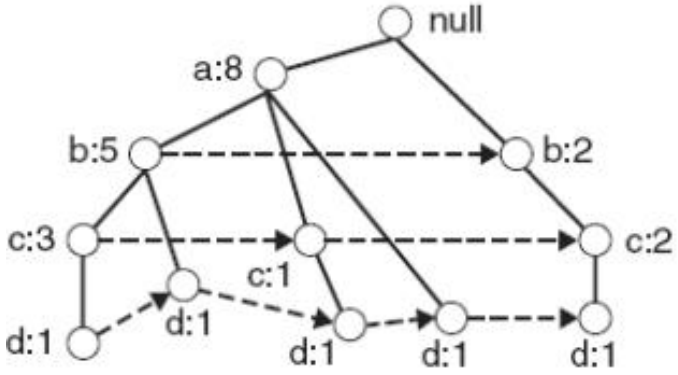
○ a로 끝나는 sub-tree를 구하자.





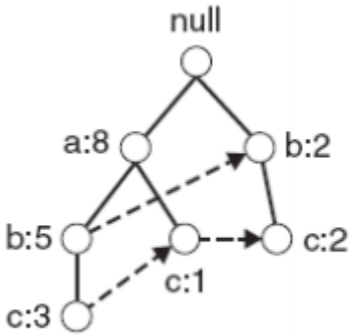
① 합이 2이므로 ae는 frequent item이다.
② 더 이상 재귀과정을 호출 할 수 없으니 멈춘다.
③ $\{a, e\}$가 frequent item임을 알 수 있다.

○ e는 이제 끝났고, d로 끝나는 frequent itemset을 찾고 싶다. e의 경우와 동일하게 찾으면 된다.

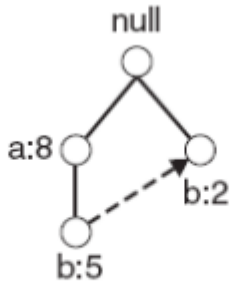


(b) Paths containing node d

○ 그 다음은 c, b, a로 끝나는 itemset을 각각 찾아주면 된다.



(c) Paths containing node c



(d) Paths containing node b

(e) Paths containing node a

Result)

Suffix	Frequent Itemsets
e	{e}, {d,e}, {a,d,e}, {c,e},{a,e}
d	{d}, {c,d}, {b,c,d}, {a,c,d}, {b,d}, {a,b,d}, {a,d}
c	{c}, {b,c}, {a,b,c}, {a,c}
b	{b}, {a,b}
a	{a}

- 장점
  - 전체 데이터 집합에 대해 2번의 scan이면 충분하다.
  - 데이터 집합이 압축(compresses) 된다.
  - frequent itemset의 후보(candidate) 집합이 필요 없다.
  - Apriori 알고리즘보다 훨씬 빠르다.

- 단점
  - FP-Tree의 크기가 memory보다 클 가능성이 있다.
  - FP-Tree를 생성하는 cost가(특히 시간) 비싸다.

References

- http://www.almaden.ibm.com/cs/quest/papers/sigmod93.pdf
- http://maths-people.anu.edu.au/~steve/pdcn.pdf
- http://aimotion.blogspot.kr/2013/01/machine-learning-and-data-mining.html
  (예제 코드)
- https://www.google.co.kr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CDwQFjAB&url=http%3A%2F%2Fwww.cs.uic.edu%2F~liub%2Fteach%2Fcs583-fall-05%2FCS583-association-rules.ppt&ei=prZ4U4vCHZfc8AXmjIL4CQ&usg=AFQjCNG3SnZEvRlC8wC46Luy13fw4DTtUQ&sig2=S-HrnSICYlW51-PvwMpwCQ&bvm=bv.66917471,d.dGc&cad=것
- http://www.florian.verhein.com/teaching/2008-01-09/fp-growth-presentation_v1%20(handout).pdf