03603452 Data Mining

Association Rule Mining

Asst.Prof.Dr.Kulwadee Somboonviwat

Department of Computer Engineering

Faculty of Engineering at Sriracha

Kasetsart University Sriracha Campus

Semester 2/2565

หัวข้อหลัก

- แนวคิดพื้นฐาน
- อัลกอริทึม Apriori
- อัลกอริทึม FP-Growth
- ประสิทธิภาพของอัลกอริทึม Apriori และ FP-Growth
- Association rule mining ด้วย Apache Spark

Association rule mining

- บุกเบิกโดยทีมวิจัยจาก IBM: Agrawal et al. (1993)
- เป็นปัญหาพื้นฐานของการทำเหมืองข้อมูล (fundamental data mining problem) และได้ถูกนำไปประยุกต์ใช้แก้ปัญหาในหลากหลายสาขา เช่น การค้าปลีก, ชีววิทยา ฯ
- ได้รับการศึกษากันอย่างแพร่หลายในวงการนักวิจัยสาขา database, data mining (แต่ไม่ค่อยแพร่หลายในสาขา machine learning)
- เริ่มแรกถูกนำไปใช้ในการวิเคราะห์ตะกร้าสินค้า (**Market Basket Analysis**) เพื่อค้นหารูปแบบพฤติกรรมการซื้อสินค้า ในรูปของกฎความสัมพันธ์ เช่น

  Chicken -> Milk, Clothes [sup = 43%, conf = 60%]

แบบจำลอง: ข้อมูล

- $I = \{i_1, i_2, ..., i_m\}$ คือเซตของ items ทั้งหมดที่มีในระบบ
- *Transaction* $t \subseteq I$ คือซับเซตของ items
- *Transaction Database* $T = \{t_1, t_2, ..., t_n\}$ คือ เซตของ transactions ทั้งหมดในระบบ

ตัวอย่าง 1: ข้อมูลทรานแซกชันของห้างสรรพสินค้า

- *I = { beef, cheese, chicken, clothes, milk, ... }* คือ เซตของสินค้าทุกรายการที่มีขายในร้าน
- *Transasction t* คือ รายการสินค้าที่ลูกค้าซื้อในแต่ละบิล

  t1 : { beef, cheese, milk }

  t2 : { clothes, milk, chicken }

  .....

  tn : { eggs, yogurt, milk, bread }

- *Transaction database T* คือ เซตของทรานแซกชันทั้งหมดในระบบ

ตัวอย่าง 2: ชุดข้อมูลเอกสาร

- ชุดข้อมูลที่ประกอบด้วยไฟล์เอกสาร (text document data set)
- แต่ละเอกสาร ประกอบด้วย ถุงคำสำคัญ (bag of keywords)
  doc1: นักศึกษา, มหาวิทยาลัย, การสอบ, อาจารย์
  doc2: โรงเรียน, นักเรียน
  doc3: โรงเรียน, จังหวัด, กีฬา
  doc4: การแข่งขัน, กีฬา, ฟุตบอล, ทีมชาติ
  doc5: นักเตะ, ดาวรุ่ง, ทีมชาติ, สโมสร
  doc6: TCAS, นักเรียน, มัธยมปลาย, มหาวิทยาลัย
  doc7: ผู้ฝึกสอน, กีฬา, ตะกร้อ, ทีมชาติ
- ***items*** ได้แก่ คำสำคัญแต่ละคำ
- *I = { นักศึกษา, มหาวิทยาลัย, การสอบ, อาจารย์, โรงเรียน, นักเรียน, จังหวัด, กีฬา, การแข่งขัน, ฟุตบอล, ทีมชาติ, นักเตะ, ดาวรุ่ง, สโมสร, TCAS, มัธยมปลาย, มหาวิทยาลัย, ผู้ฝึกสอน, ตะกร้อ }*
- ***Transaction t** : doc1, doc2, doc3, doc4, doc5, doc6, doc7*
- ***Transaction Database T** : { doc1, doc2, doc3, doc4, doc5, doc6, doc7 }*

แบบจำลอง: กฎ

- เซตของ item เรียกว่า itemset
- เซตของ item ที่มีขนาดเท่ากับ *k* เรียกว่า ***k-itemset*** เช่น
  X = { milk, boots, beef } คือ 3-itemset
  Y = { chocolate, cheese, milk, salt } คือ 4-itemset

- ถ้า *X* เป็น k-itemset แล้ว,
  Transaction t* ประกอบด้วย *X (t contains X) , เมื่อ $X \subseteq t$

- กฎความสัมพันธ์ (association rule) แสดงความสัมพันธ์ระหว่าง itemset *X* และ *Y*, มีรูปแบบคือ

$$X \rightarrow Y, \text{ เมื่อ } Y \subset I, \text{ และ } X \cap Y = \varnothing$$

<u>ความหมายของ X → Y:</u> หากในทรานแซกชันมี itemset *X* แล้วก็มีโอกาสที่จะมี itemset *Y* อยู่เช่นกัน

การวัดความน่าเชื่อถือของกฎ

- Support (**สัดส่วนของทรานแซกชันที่สนับสนุนกฎ**) กฎความสัมพันธ์ $X \rightarrow Y$ เป็นจริงสำหรับฐานข้อมูลทรานแซกชัน *T* ด้วย**ค่าสนับสนุน** *S*
  ถ้า ***S %*** ของทรานแซกชันใน *T* ประกอบด้วย $X \cup Y$ *(นั่นคือ T* contains *X* ∪ *Y)*

- Confidence (**ระดับความเชื่อมั่น**) กฎความสัมพันธ์ $X \rightarrow Y$ เป็นจริงสำหรับฐานข้อมูลทรานแซกชัน *T* ด้วย**ระดับความเชื่อมั่น** *C* ถ้า ***C %*** ของทรานแซกชันใน *T* ที่ประกอบด้วย *X* ก็ประกอบด้วย *Y*

การคำนวณ Support และ Confidence

- กำหนดให้ ***Support count ของ itemset X (X.count) ในฐานข้อมูลทรานแซกชัน*** *T* หมายถึง จำนวนทรานแซกชันใน *T* ที่ประกอบด้วย *X*

- ถ้า *T* ประกอบด้วยทรานแซกชันทั้งหมดจำนวน *n* ทรานแซกชัน แล้ว

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

Frequent Itemset Mining

- กำหนดเซตของ item *I*, ฐานข้อมูลทรานแซกชัน *T,* และค่า ***minsup***

  ค้นหา itemsets **<u>ทั้งหมด</u>**ของฐานข้อมูลทรานแซกชัน *T* ที่มีค่า *support s* มากกว่า *minsup*

Association rules mining

- กำหนดเซตของ item *I*, ฐานข้อมูลทรานแซกชัน *T,* ค่า *minsup*, และค่า *minconf*

  ค้นหา กฎความสัมพันธ์ระหว่าง itemsets **ทุกกฎ**ของฐานข้อมูลทรานแซกชัน *T* ที่มีค่า support และค่า confidence มากกว่าค่า minimum support (*minsup*) และ minimum confidence (*minconf*) ตามลำดับ

ตัวอย่าง: Frequent itemset และ Association Rule Mining

กำหนดเซตของ item *I*, ฐานข้อมูลทรานแซกชัน *T* :

I = { antivirus, monitor, pc, printer, usb_stick }

Transaction Database T

Transaction	Items appearing in the transaction
T1	{ antivirus, pc, printer }
T2	{ antivirus, monitor }
T3	{ antivirus, usb_stick }
T4	{ antivirus, monitor, printer}
T5	{ printer, usb_stick }
T6	{ antivirus, usb_stick }
T7	{ printer, usb_stick }
T8	{ antivirus, pc, printer, usb_stick }
T9	{ antivirus, printer, usb_stick }

- ถ้า minsup = 20% และ minconf = 30%
- ตัวอย่าง frequent itemsets:

  {printer, antivirus, usb_stick} [sup = 2/9]
- ตัวอย่าง Association rules:

  {printer} -> {antivirus, usb_stick} [sup =1/3]

  {antivirus} -> {printer, usb_stick} [sup =1/3]

  .....

ตัวอย่าง: การแจกแจง itemsets ทั้งหมดที่เป็นไปได้

- กำหนดให้เซต I = { printer, antivirus, usb_stick, monitor, pc } คือ**เซตของ items ทั้งหมด**
- จำนวน subsets ที่เป็นไปได้ทั้งหมดของเซต $I = 2^{|I|} - 1 = 2^5 - 1 = 31$

{ printer } { antivirus } { usb_stick } { monitor } { pc }

{printer, antivirus} {printer, usb_stick} {printer, monitor} {printer, pc} {antivirus, usb_stick} {antivirus, monitor} {antivirus, pc} {usb_stick, monitor} {usb_stick, pc} {monitor, pc}

{printer, antivirus, usb_stick} {printer, antivirus, monitor} {printer, antivirus, pc}

{printer, usb_stick, monitor} {printer, usb_stick, pc} {printer, monitor, pc}

{antivirus, usb_stick, monitor} {antivirus, usb_stick, pc} {antivirus, monitor, pc}

{usb_stick, monitor, pc}

{printer, antivirus, usb_stick, monitor} {printer, antivirous, usb_stick, pc}

{printer, antivirus, monitor, pc} {printer, usb_stick, monitor, pc}

{antivirus, usb_stick, monitor, pc} {printer, antivirus, usb_stick, monitor, pc}

ตัวอย่าง: การแจกแจง k-itemsets

- **k-Itemsets** คือ itemsets ที่มีขนาดเท่ากับ **k**

1-itemsets:

{ printer } { antivirus } { usb_stick } { monitor } { pc }

2-itemsets:

{printer, antivirus} {printer, usb_stick} {printer, monitor} {printer, pc} {antivirus, usb_stick} {antivirus, monitor}

{antivirus, pc} {usb_stick, monitor} {usb_stick, pc} {monitor, pc}

3-itemsets:

{printer, antivirus, usb_stick} {printer, antivirus, monitor} {printer, antivirus, pc} {printer, usb_stick, monitor}

{printer, usb_stick, pc} {printer, monitor, pc} {antivirus, usb_stick, monitor} {antivirus, usb_stick, pc}

{antivirus, monitor, pc} {usb_stick, monitor, pc}

4-itemsets:

{printer, antivirus, usb_stick, monitor} {printer, antivirous, usb_stick, pc} {printer, antivirus, monitor, pc}

{printer, usb_stick, monitor, pc} {antivirus, usb_stick, monitor, pc}

5-itemsets:

{printer, antivirus, usb_stick, monitor, pc}

ตัวอย่าง: การคำนวณค่า support

- ค่าซัพพอร์ตของ itemset X คือจำนวนทรานแซคชั่นที่มี *X* รวมอยู่

$$sup(X) = \frac{|\{ t \mid (X \text{ เป็นซับเซตของ } t) \text{ และ } t \text{ คือทรานแซกชั่นในฐานข้อมูลทรานแซกชัน } T \}|}{|T|}$$

- ตัวอย่าง

Transaction	Items appearing in the transaction
T1	{ printer, antivirus, pc }
T2	{ antivirus, monitor }
T3	{ antivirus, usb_stick }
T4	{ printer, antivirus, monitor }
T5	{ printer, usb_stick }
T6	{ antivirus, usb_stick }
T7	{ printer, usb_stick }
T8	{ printer, antivirus, usb_stick, pc }
T9	{ printer, antivirus, usb_stick }

sup({printer})
= 6 / 9
(67%)

sup({antivirus, pc})
= 2 / 9
(22%)

Frequent Itemset Mining: Naïve Approach

- ถ้าทรานแซคชั่นดาต้าเบสมี items ทั้งหมด n ชนิด จำนวน itemsets ทั้งหมดจะเท่ากับ $2^n - 1$
- Naïve Approach for Frequent Itemset Mining:
  - นับค่าซัพพอร์ตของ itemsets ทั้งหมดของทรานแซคชั่นดาต้าเบส
  - การนับค่าซัพพอร์ตของ itemset หนึ่งเซต ต้องสแกนอ่านข้อมูลทั้งฐานข้อมูล 1 ครั้ง ดังนั้น วิธีการนี้จึงต้องสแกนอ่านฐานข้อมูลเป็นจำนวน $2^n - 1$ ครั้ง!!!
  - วิธีการนี้ จึงไม่เหมาะกับการนำไปใช้งานจริง

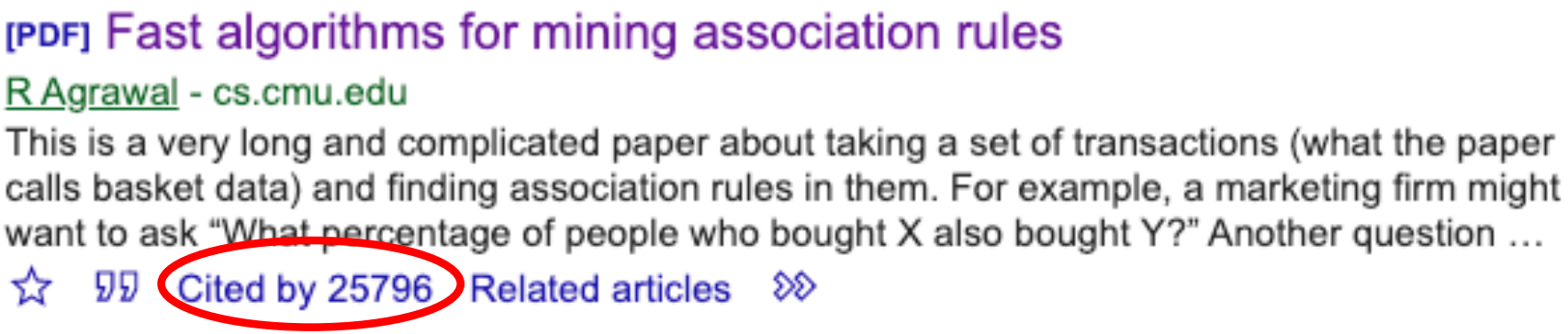
เราจะค้นหา itemsets ได้อย่างมีประสิทธิภาพมากกว่านี้ได้อย่างไร?

- ความท้าทาย 2 ประการ
  - จะนับจำนวน support count ได้อย่างมีประสิทธิภาพได้อย่างไร
  - จะลดขนาดของ search space ได้อย่างไร

หัวข้อหลัก

- แนวคิดพื้นฐาน
- อัลกอริทึม Apriori
- อัลกอริทึม FP-Growth
- ประสิทธิภาพของอัลกอริทึม Apriori และ FP-Growth
- Association rule mining ด้วย Apache Spark

Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast algorithms for mining association rules." *Proc. 20th int. conf. very large data bases, VLDB*. Vol. 1215. 1994.



Apriori Algorithm

- Apriori Property (downward closure property):

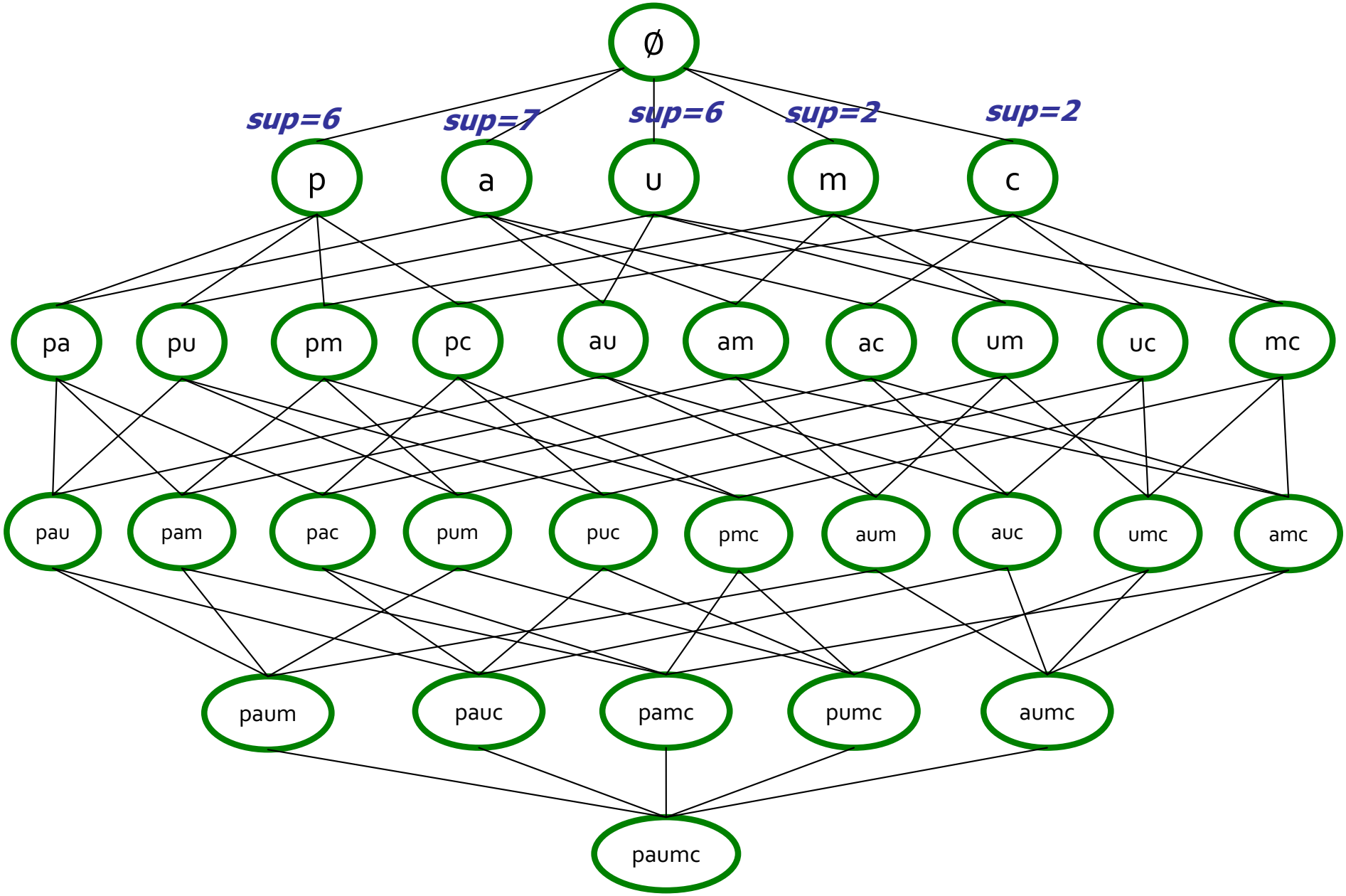
  กำหนด itemset สองเซต X และ Y ถ้า $X \subset Y$ แล้ว

  ค่าซัพพอร์ตของ Y จะน้อยกว่าหรือเท่ากับค่าซัพพอร์ตของ X

  - ตัวอย่าง
    - sup({printer}) = 6
    - sup({printer, antivirus}) = 4
    - sup({printer, antivirus, pc}) = 2

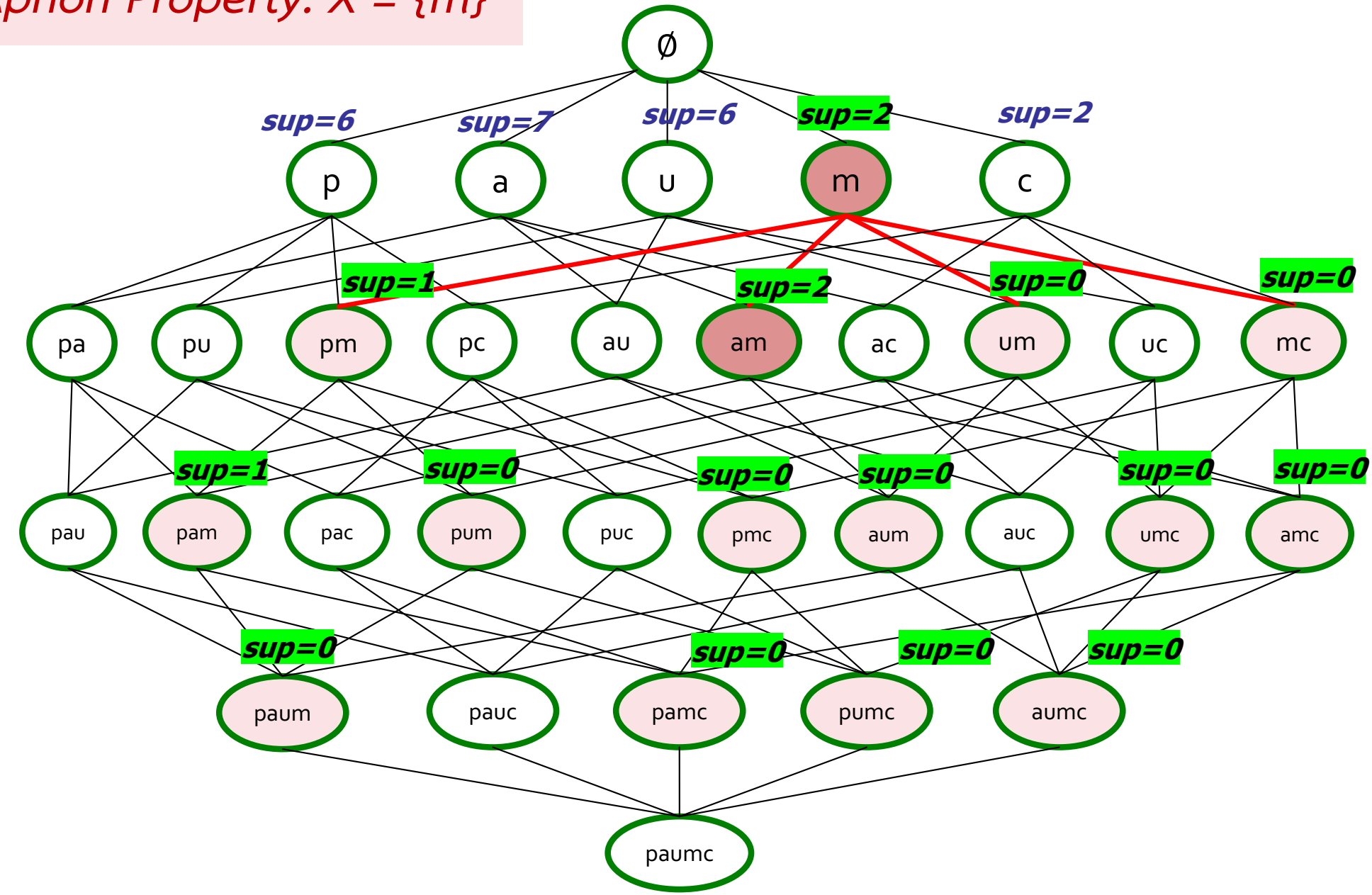
Transaction	Items appearing in the transaction
T1	{ printer, antivirus, pc }
T2	{ antivirus, monitor }
T3	{ antivirus, usb_stick }
T4	{ printer, antivirus, monitor }
T5	{ printer, usb_stick }
T6	{ antivirus, usb_stick }
T7	{ printer, usb_stick }
T8	{ printer, antivirus, usb_stick, pc }
T9	{ printer, antivirus, usb_stick }

Search Space (Hasse Diagram) : printer=p, antivirus=a, usb_stick=u, monitor=m, pc=c



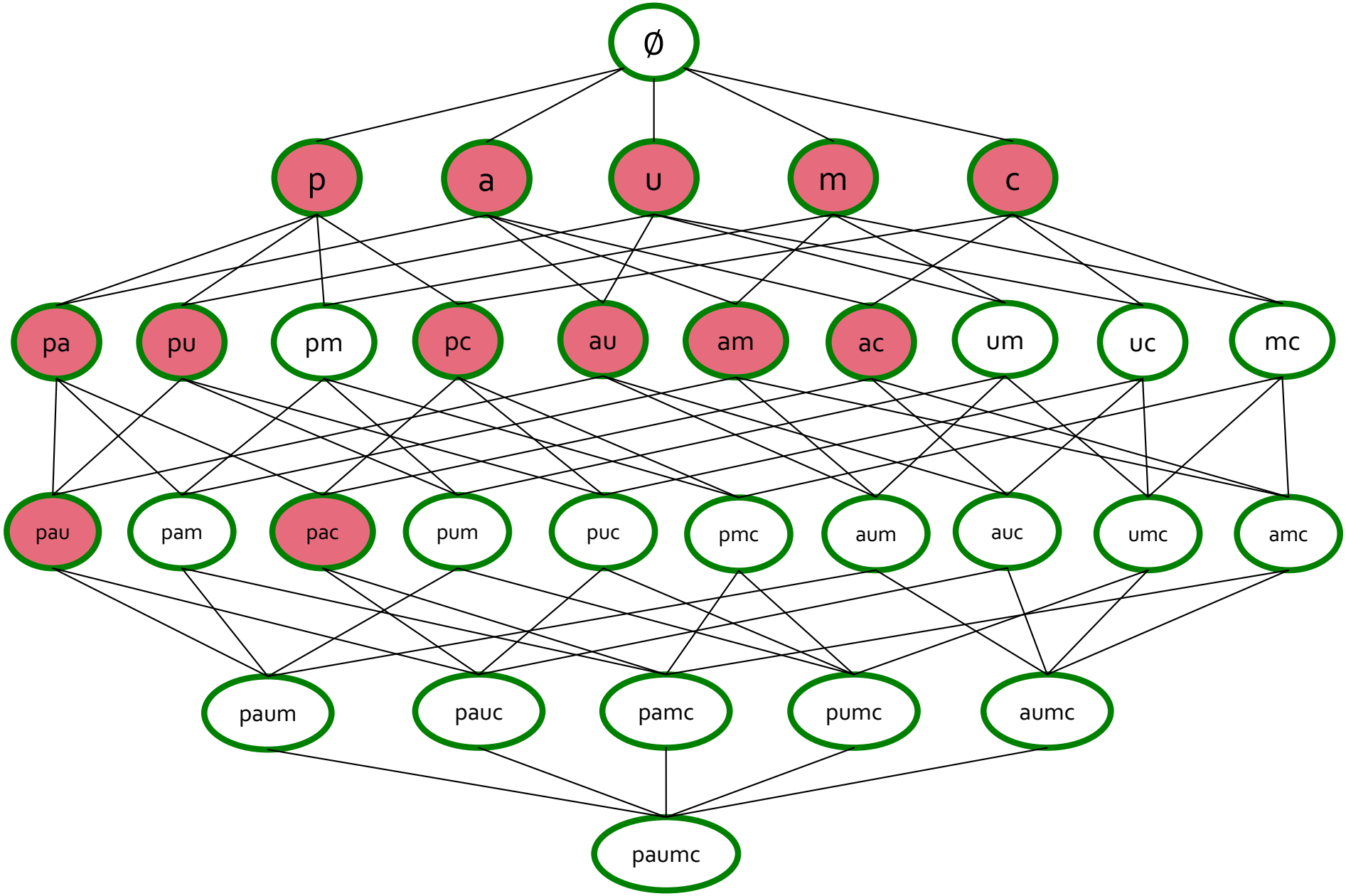
Search Space (Hasse Diagram) : printer=p, antivirus=a, usb_stick=u, monitor=m, pc=c

Apriori Property: X = {m}



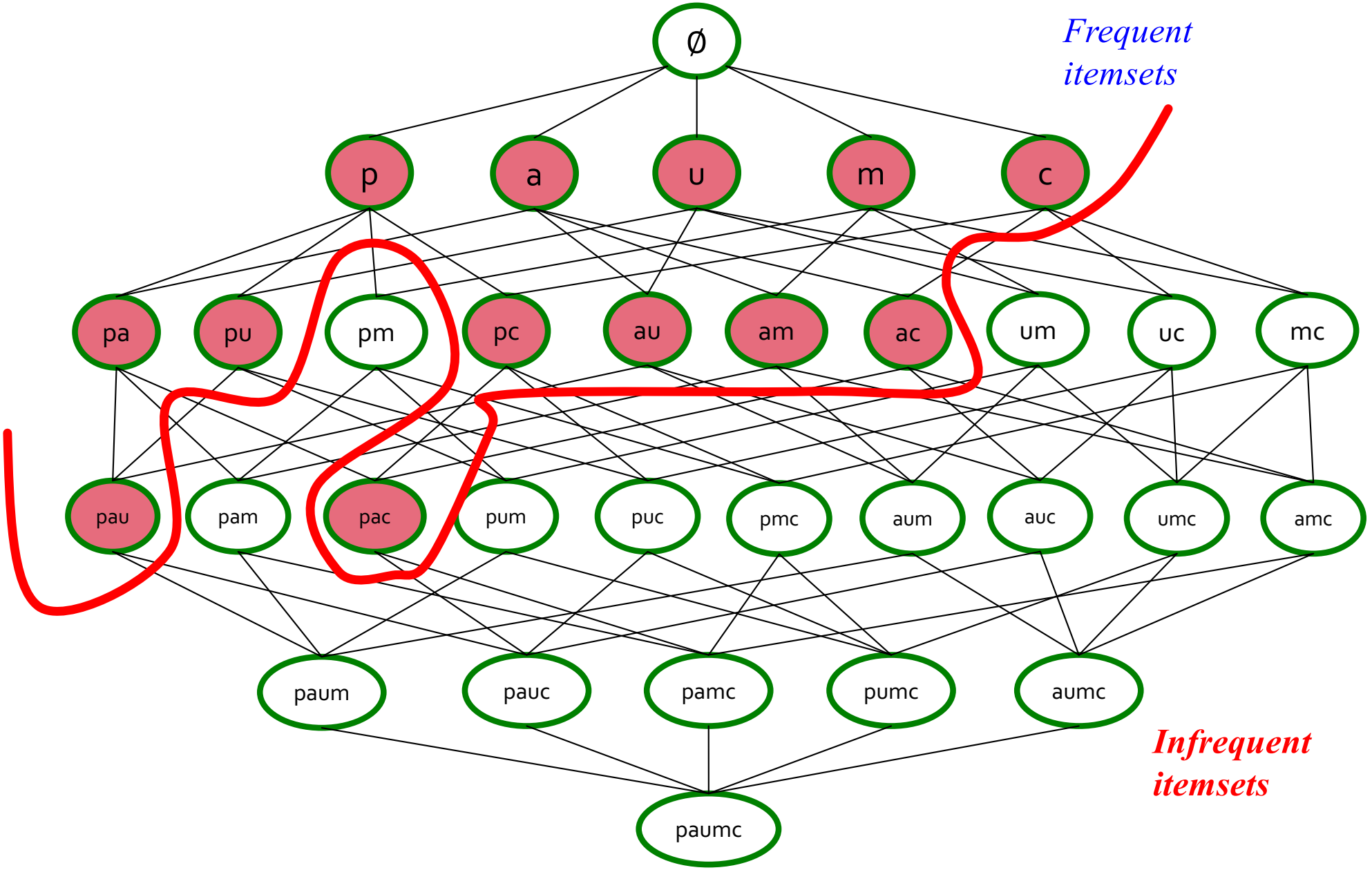
Search Space (Hasse Diagram) : printer=p, antivirus=a, usb_stick=u, monitor=m, pc=c

ถ้า minsup=2, frequent items คือโหนดที่ระบายด้วยสีชมพู



Search Space (Hasse Diagram) : printer=p, antivirus=a, usb_stick=u, monitor=m, pc=c

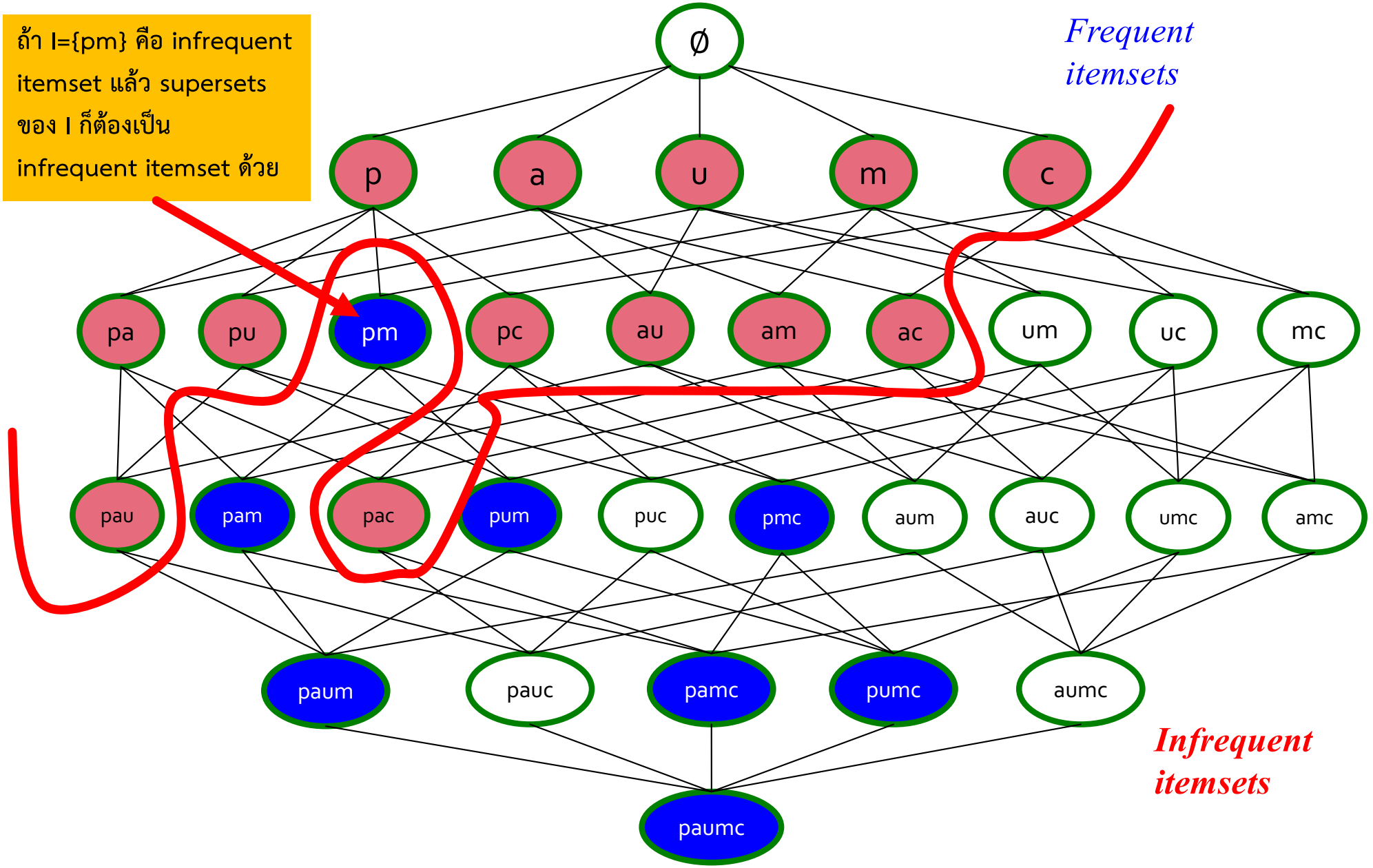
ถ้า minsup=2, frequent items คือโหนดที่ระบายด้วยสีชมพู



Search Space (Hasse Diagram) : printer=p, antivirus=a, usb_stick=u, monitor=m, pc=c

ถ้า minsup=2, frequent items คือโหนดที่ระบายด้วยสีชมพู

ถ้า I={pm} คือ infrequent itemset แล้ว supersets ของ I ก็ต้องเป็น infrequent itemset ด้วย



Apriori Algorithm

- จาก Apriori Property, เราสรุปได้ว่า

  ถ้า itemsets X เป็น infrequent itemset และ $X \subset Y$ แล้ว, Y เป็น infrequent itemset

- ตัวอย่าง
  - พิจารณา itemset {printer, antivirus, monitor}
  - ถ้าเราทราบว่า itemset {printer, monitor} คือ infrequent itemset , เราจะสามารถสรุปได้ว่า {printer, antivirus, monitor} ก็ต้องเป็น infrequent itemset ด้วย

Transaction	Items appearing in the transaction
T1	{ printer, antivirus, pc }
T2	{ antivirus, monitor }
T3	{ antivirus, usb_stick }
T4	{ printer, antivirus, monitor }
T5	{ printer, usb_stick }
T6	{ antivirus, usb_stick }
T7	{ printer, usb_stick }
T8	{ printer, antivirus, usb_stick, pc }
T9	{ printer, antivirus, usb_stick }

Apriori Algorithm

- ทำงานแบบวนซ้ำ (iterative algorithm) โดยการค้นหา frequent itemsets ทีละระดับ (level-wise search)
  - หา frequent 1-itemsets : F_1
  - *from k=2*
    - *Ck = เซตของ itemsets ที่มีขนาดเท่ากับ k ที่**อาจจะเป็น** frequent itemsets*
    - *Fk = เซตของ itemsets ที่เป็นสมาชิกของ Ck และ**เป็น** frequent itemsets*

ตัวอย่าง : Apriori Algorithm

- **อินพุท**
  - minsup = 2
  - a transactional database *T*
- **เอาท์พุท**
  - All the frequent itemsets

Transaction	Items appearing in the transaction
T1	{ printer, antivirus, pc }
T2	{ antivirus, monitor }
T3	{ antivirus, usb_stick }
T4	{ printer, antivirus, monitor }
T5	{ printer, usb_stick }
T6	{ antivirus, usb_stick }
T7	{ printer, usb_stick }
T8	{ printer, antivirus, usb_stick, pc }
T9	{ printer, antivirus, usb_stick }

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 1:** สแกนฐานข้อมูลเพื่อคำนวณ support count ของ 1-itemset ทุกเซต

sup({printer}) = 6

sup({antivirus}) = 7

sup({usb_stick}) = 6

sup({monitor}) = 2

sup({pc}) = 2

Transaction	Items appearing in the transaction
T1	{ printer, antivirus, pc }
T2	{ antivirus, monitor }
T3	{ antivirus, usb_stick }
T4	{ printer, antivirus, monitor }
T5	{ printer, usb_stick }
T6	{ antivirus, usb_stick }
T7	{ printer, usb_stick }
T8	{ printer, antivirus, usb_stick, pc }
T9	{ printer, antivirus, usb_stick }

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 2:** ตัด infrequent itemsets (support < minsup) ออก

sup({printer}) = 6

sup({antivirus}) = 7

sup({usb_stick}) = 6

sup({monitor}) = 2

sup({pc}) = 2

Transaction	Items appearing in the transaction
T1	{ printer, antivirus, pc }
T2	{ antivirus, monitor }
T3	{ antivirus, usb_stick }
T4	{ printer, antivirus, monitor }
T5	{ printer, usb_stick }
T6	{ antivirus, usb_stick }
T7	{ printer, usb_stick }
T8	{ printer, antivirus, usb_stick, pc }
T9	{ printer, antivirus, usb_stick }

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 3:** สร้าง candidate 2-itemsets โดยการรวม frequent 1-itemsets เข้าด้วยกัน

Frequent 1-itemsets

{printer}

{antivirus}

{usb_stick}

{monitor}

{pc}

Candidate 2-itemsets

{printer, antivirus}

{printer, usb_stick}

{printer, monitor}

{printer, pc}

{antivirus, usb_stick}

{antivirus, monitor}

{antivirus, pc}

{usb_stick, monitor}

{usb_stick, pc}

{monitor, pc}

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 4:** ตัด candidate 2-itemsets ที่***ขัดกับ downward closure*** ออก

  (***มี infrequent itemsets เป็นสมาชิก***)

Frequent 1-itemsets

{printer}

{antivirus}

{usb_stick}

{monitor}

{pc}

Candidate 2-itemsets

{printer, antivirus}

{printer, usb_stick}

{printer, monitor}

{printer, pc}

{antivirus, usb_stick}

{antivirus, monitor}

{antivirus, pc}

{usb_stick, monitor}

{usb_stick, pc}

{monitor, pc}

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 5:** สแกนฐานข้อมูล เพื่อคำนวณค่าซัพพอร์ตของ candidate itemsets ที่เหลืออยู่

Candidate 2-itemsets	Support
{printer, antivirus}	4
{printer, usb_stick}	4
{printer, monitor}	1
{printer, pc}	2
{antivirus, usb_stick}	4
{antivirus, monitor}	2
{antivirus, pc}	2
{usb_stick, monitor}	0
{usb_stick, pc}	1
{monitor, pc}	0

Transaction	Items appearing in the transaction
T1	{ printer, antivirus, pc }
T2	{ antivirus, monitor }
T3	{ antivirus, usb_stick }
T4	{ printer, antivirus, monitor }
T5	{ printer, usb_stick }
T6	{ antivirus, usb_stick }
T7	{ printer, usb_stick }
T8	{ printer, antivirus, usb_stick, pc }
T9	{ printer, antivirus, usb_stick }

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 6:** ตัด infrequent itemsets ออก (คือ itemset ที่มีค่าซัพพอร์ตน้อยกว่า minsup)

Candidate 2-itemsets	Support
{printer, antivirus}	4
{printer, usb_stick}	4
~~{printer, monitor}~~	~~1~~
{printer, pc}	2
{antivirus, usb_stick}	4
{antivirus, monitor}	2
{antivirus, pc}	2
~~{usb_stick, monitor}~~	~~0~~
~~{usb_stick, pc}~~	~~1~~
~~{monitor, pc}~~	~~0~~

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 7:** สร้าง candidate 3-itemsets โดยการรวม frequent 2-itemsets เข้าด้วยกัน

Frequent 2-itemsets

{printer, antivirus}

{printer, usb_stick}

{printer, pc}

{antivirus, usb_stick}

{antivirus, monitor}

{antivirus, pc}

Candidate 3-itemsets

{printer, antivirus, usb_stick}

{printer, antivirus, pc}

{printer, usb_stick, pc}

{antivirus, usb_stick, monitor}

{antivirus, usb_stick, pc}

{antivirus, monitor, pc}

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 8:** ตัด candidate 3-itemsets ที่***ขัดกับ downward closure*** ออก

  (***มี infrequent itemsets เป็นสมาชิก***)

Frequent 2-itemsets

{printer, antivirus}

{printer, usb_stick}

{printer, pc}

{antivirus, usb_stick}

{antivirus, monitor}

{antivirus, pc}

Candidate 3-itemsets

{printer, antivirus, usb_stick}

{printer, antivirus, pc}

~~{printer, usb_stick, pc}~~

~~{antivirus, usb_stick, monitor}~~

~~{antivirus, usb_stick, pc}~~

~~{antivirus, monitor, pc}~~

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 9:** สแกนฐานข้อมูล เพื่อคำนวณค่าซัพพอร์ตของ candidate itemsets ที่เหลืออยู่

Candidate 3-itemsets	Support
{printer, antivirus, usb_stick}	2
{printer, antivirus, pc}	2

Transaction	Items appearing in the transaction
T1	{ printer, antivirus, pc }
T2	{ antivirus, monitor }
T3	{ antivirus, usb_stick }
T4	{ printer, antivirus, monitor }
T5	{ printer, usb_stick }
T6	{ antivirus, usb_stick }
T7	{ printer, usb_stick }
T8	{ printer, antivirus, usb_stick, pc }
T9	{ printer, antivirus, usb_stick }

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 10:** ตัด infrequent itemsets ออกจาก candidate 3-itemsets

Frequent 3-itemsets	Support
{printer, antivirus, usb_stick}	2
{printer, antivirus, pc}	2

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 11:** สร้าง candidate 4-itemsets โดยการรวม frequent 3-itemsets เข้าด้วยกัน

Frequent 3-itemsets	→	Candidate 4-itemsets
{printer, antivirus, usb_stick}		{printer, antivirus, usb_stick, pc}
{printer, antivirus, pc}		

ตัวอย่าง : Apriori Algorithm

- **ขั้นที่ 12:** ตัด candidate 4-itemsets ที่มี infrequent itemsets เป็นสมาชิกออก

Frequent 3-itemsets	→	Candidate 4-itemsets
{printer, antivirus, usb_stick}		~~{printer, antivirus, usb_stick, pc}~~
{printer, antivirus, pc}		

- ไม่มี candidate itemset เหลือ จบการทำงาน

ตัวอย่าง : Apriori Algorithm

- **ผลลัพธ์:** Frequent Itemsets ของทรานแซคชั่นดาต้าเบส D เมื่อกำหนด minsup=2 ได้แก่

{printer}	6
{antivirus}	7
{usb_stick}	6
{monitor}	2
{pc}	2

{printer, antivirus}	4
{printer, usb_stick}	4
{printer, pc}	2
{antivirus, usb_stick}	4
{antivirus, monitor}	2
{antivirus, pc}	2

{printer, antivirus, usb_stick}	2
{printer, antivirus, pc}	2

Pseudo-code ของอัลกอริทึม Apriori

Algorithm *Apriori*(Transactions: $\mathcal{T}$, Minimum Support: *minsup*)
begin
  $k = 1$;
  $\mathcal{F}_1 = \{$ All Frequent 1-itemsets $\}$;
  while $\mathcal{F}_k$ is not empty **do begin**
    Generate $\mathcal{C}_{k+1}$ by joining itemset-pairs in $\mathcal{F}_k$;
    Prune itemsets from $\mathcal{C}_{k+1}$ that violate downward closure;
    Determine $\mathcal{F}_{k+1}$ by support counting on $(\mathcal{C}_{k+1}, \mathcal{T})$ and retaining
      itemsets from $\mathcal{C}_{k+1}$ with support at least *minsup*;
    $k = k + 1$;
  end;
  return$(\cup_{i=1}^{k} \mathcal{F}_i)$;
end

Ref: Charu C. Agarwal. Data Mining: The Textbook, 2015 (Figure 4.2)

การสร้างกฎจาก frequent itemsets

- สำหรับแต่ละ frequent itemset *L*, แจกแจงซับเซต l ⊂ L และสร้าง rule: l => (L-*l*) ที่ผ่านเกณฑ์ minimum confidence:
  - ตัวอย่าง L = {printer, antivirus, pc}:
    - R1: printer => antivirus, pc
      - conf(R1) = support(LHF U RHF)/support(LHF)
        = support({printer, antivirus, pc})/support({printer})
        = 2/6 = 33%
    - R2: pc => printer, antivirus
      - conf(R2) = support({printer, antivirus, pc})/support({pc})
        = 2 / 2 = 100%
    - ถ้ากำหนด minimum confidence = 70%, association rule ที่ได้คือ R2 เนื่องจาก R1 มีค่า confidence น้อยกว่า 70%

สรุป : Apriori Algorithm

- Level-wise search, iterative algorithm
- อัลกอริธึม Apriori สามารถลดขนาด search space ลงเหลือเพียง 17 candidates ในขณะที่ Naïve algorithm มีขนาด search space เท่ากับ 31
- ขั้นตอนการสร้างกฎ ใช้เวลาน้อยเมื่อเทียบกับขั้นตอนการค้นหา frequent itemsets
- ต้องสแกนฐานข้อมูลทั้งหมด K ครั้ง เมื่อ K คือขนาดของ itemset ที่ใหญ่ที่สุด (ในทางปฏิบัติ K <= 10)
- สามารถค้นหากฎความสัมพันธ์ได้เร็วมากในบางกรณี (linear time)
- ข้อด้อยของอัลกอริธึม Apriori:
  - ขั้นตอนการแจกแจง Candidate itemsets ใช้ทรัพยากรมาก (ทั้งหน่วยความจำ และ เวลา)
  - การคำนวณค่าซัพพอร์ตต้องมีการตรวจสอบซับเซต (computationally expensive) และสแกนข้อมูลจากฐานข้อมูล (I/O expensive)

หัวข้อหลัก

- แนวคิดพื้นฐาน
- อัลกอริทึม Apriori
- อัลกอริทึม FP-Growth
- ประสิทธิภาพของอัลกอริทึม Apriori และ FP-Growth
- Association rule mining ด้วย Apache Spark

Han, Jiawei, Jian Pei, and Yiwen Yin. "Mining frequent patterns without candidate generation." *ACM sigmod record* 29.2 (2000): 1-12.

Mining frequent patterns without candidate generation [PDF] psu.ac.th

J Han, J Pei, Y Yin - ACM sigmod record, 2000 - dl.acm.org

Mining frequent patterns in transaction databases, time-series databases, and many other kinds of databases has been studied popularly in data mining research. Most of the previous studies adopt an Apriori-like candidate set generation-and-test approach. However, candidate set generation is still costly, especially when there exist prolific patterns and/or long patterns. In this study, we propose a novel frequent pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial information about …

☆ บันทึก 99 อ้างอิง อ้างโดย9844 บทความที่เกี่ยวข้อง ทั้งหมด 73 ฉบับ

แนวคิดหลัก : อัลกอริทึม FP-Growth

- ใช้โครงสร้าง FP-tree (Frequent-Pattern tree) เพื่อย่อขนาดฐานข้อมูล
  - เก็บเฉพาะข้อมูลที่จำเป็นต่อการค้นหา frequent itemsets
  - หลีกเลี่ยงการสแกนฐานข้อมูลซึ่งเป็นโอเปอเรชันที่ใช้เวลานาน

- ค้นหา frequent itemsets จาก FP-tree ด้วยวิธีการ FP-growth
  - divide-and-conquer. ใช้วิธีค้นหา frequent itemsets บน FP-tree ที่มีขนาดเล็กลงเรื่อย ๆ
  - หลีกเลี่ยงการสร้าง candidate itemsets

การสร้าง FP-tree

1. สแกนฐานข้อมูลเพื่อหา support count ของ 1-itemsets ทุกตัว
2. เรียงลำดับ 1-itemsets จากมากไปน้อย และเก็บไว้ในลิสต์ *L*
3. สำหรับแต่ละทรานแซกชันในฐานข้อมูล,
   เรียงลำดับ frequent items ตามลำดับในลิสต์ L
4. สแกนฐานข้อมูลเพื่อสร้าง FP-tree

การสร้าง FP-tree

1. สแกนฐานข้อมูลเพื่อหา support count ของ 1-itemsets ทุกตัว

2. เรียงลำดับ 1-itemsets จากมากไปน้อย และเก็บไว้ในลิสต์ *L*

TID	*Items bought*
100	{*f, a, c, d, g, i, m, p*}
200	{*a, b, c, f, l, m, o*}
300	{*b, f, h, j, o*}
400	{*b, c, k, s, p*}
500	{*a, f, c, e, l, p, m, n*}

L

Item	*frequency*
f	4
c	4
a	3
b	3
m	3
p	3

การสร้าง FP-tree

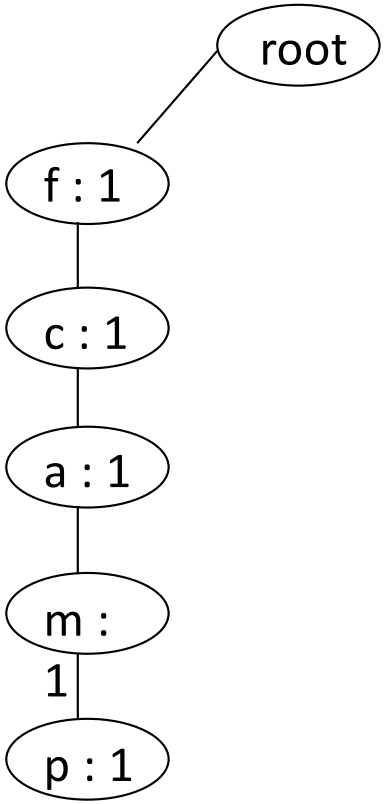
3. สำหรับแต่ละทรานแซกชันในฐานข้อมูล,
   เรียงลำดับ frequent items ตามลำดับในลิสต์ L

Transaction ID	Items Bought	(Ordered) Frequent Items
100	f, a, c, d, g, i, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

การสร้าง FP-tree

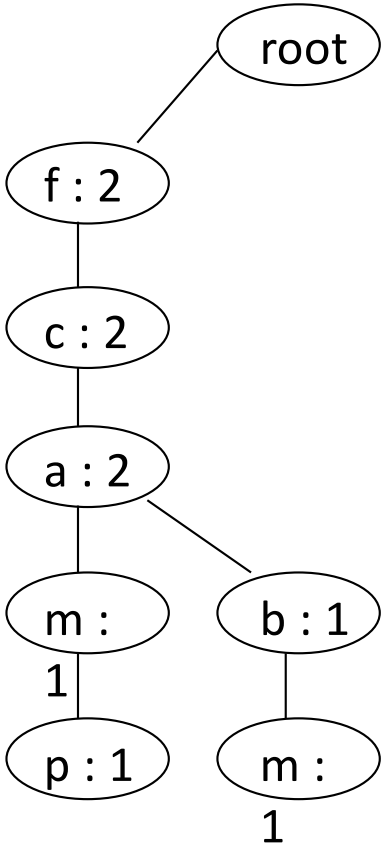
4. สร้าง FP-tree จาก ordered frequent items ของแต่ละทรานแซกชัน

สร้าง root node



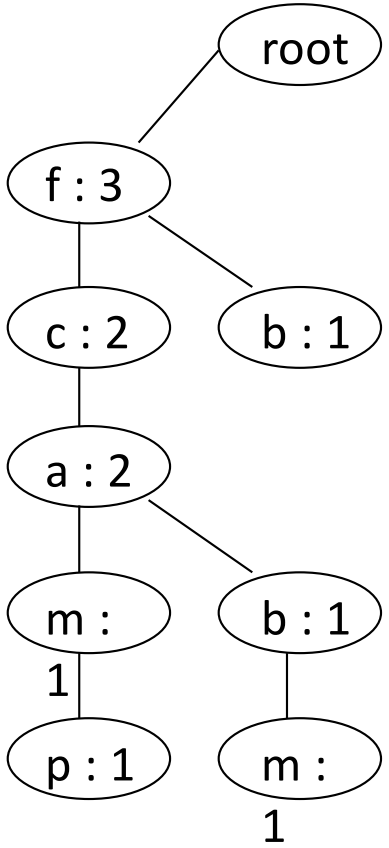
หลังจาก trans 1

(f,c,a,m,p)



หลังจาก trans

2 **(f,c,a,b,m)**

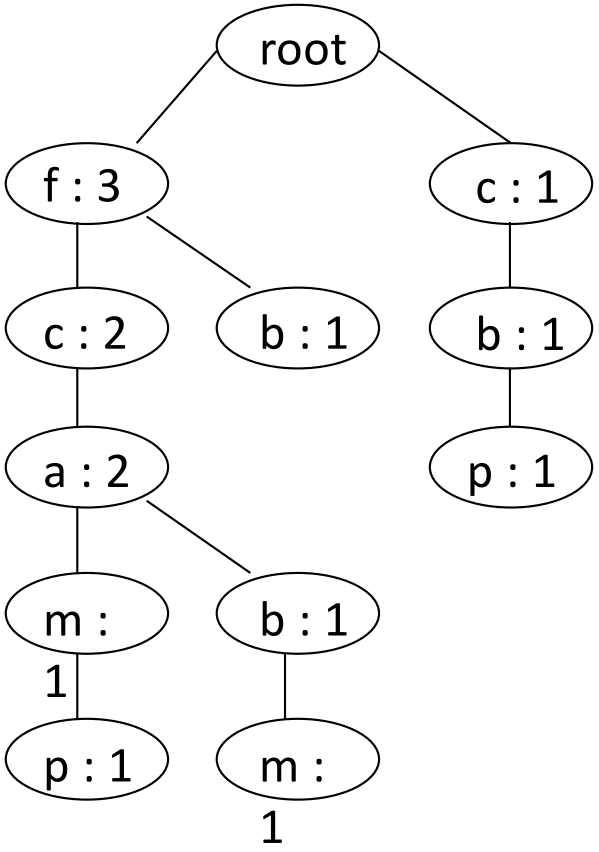


หลังจาก trans 3

(f,b)

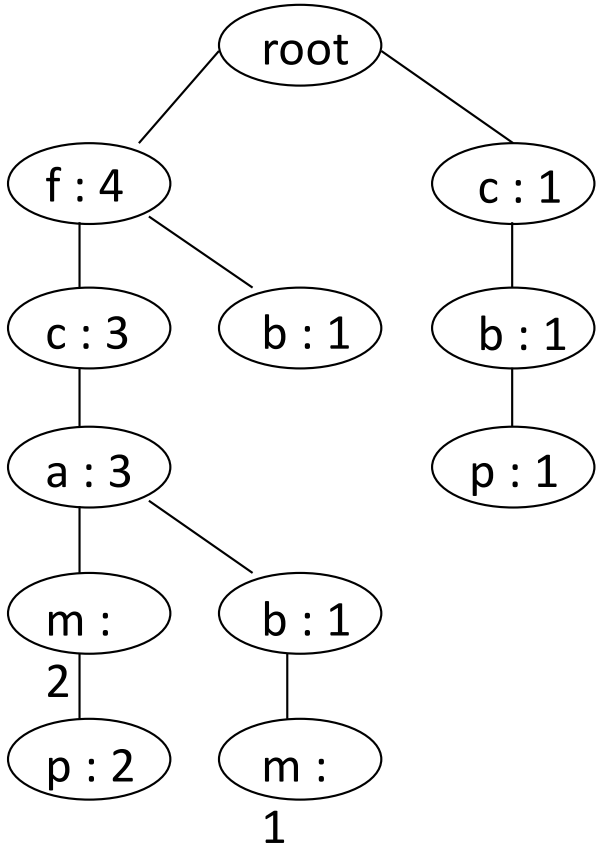
การสร้าง FP-tree

4. สร้าง FP-tree จาก ordered frequent items ของแต่ละทรานแซกชัน



หลังจาก trans 4

(c,b,p)

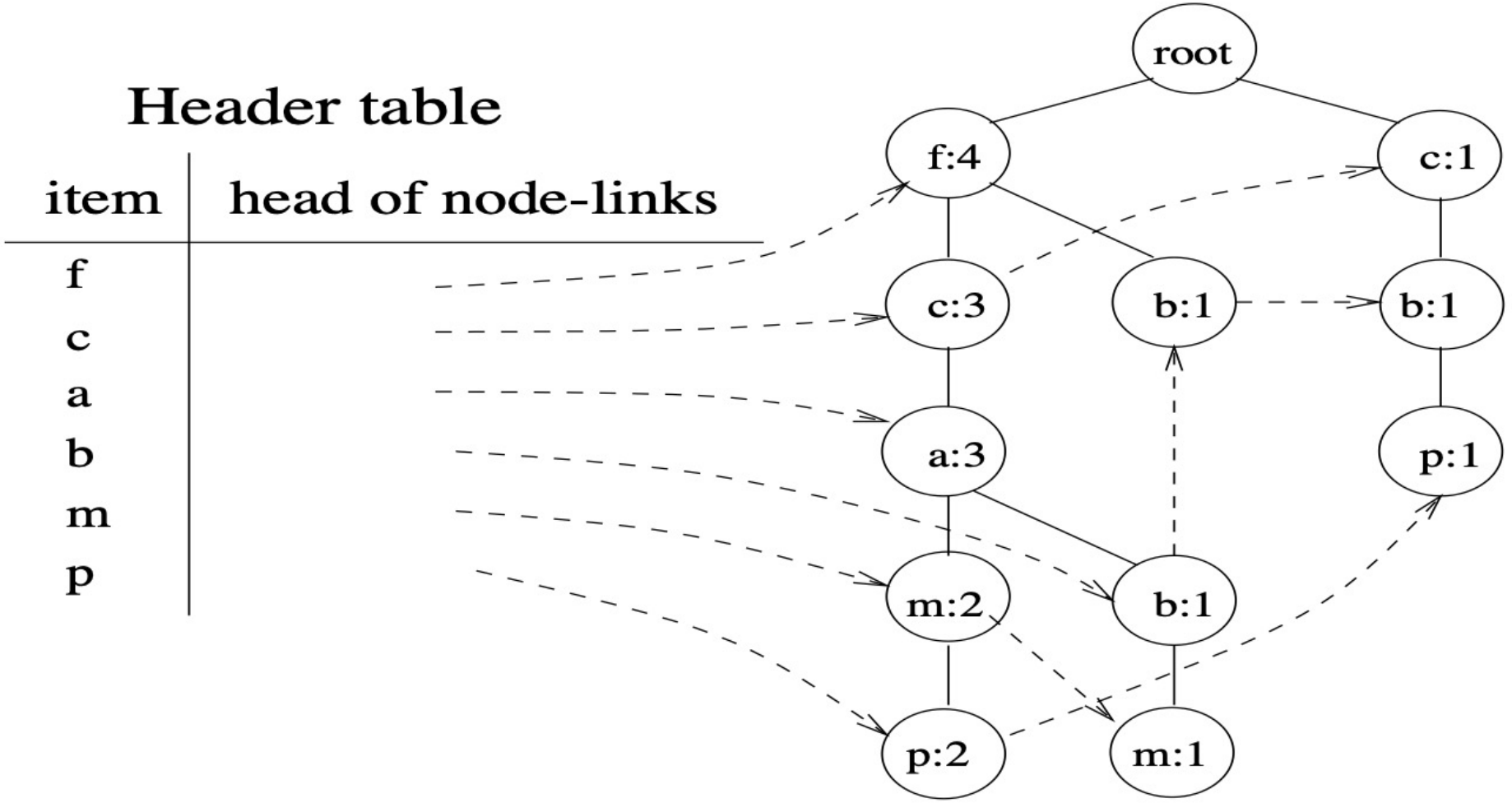


หลังจาก trans 5

(f,c,a,m,p)

การสร้าง FP-tree

4. สร้าง FP-tree จาก ordered frequent items ของแต่ละทรานแซกชัน



คุณสมบัติของ FP-tree

- สร้างโดยการสแกนฐานข้อมูลเพียง 2 ครั้ง
- Completeness

  FP-tree มีข้อมูลทั้งหมดที่จำเป็นสำหรับการค้นหา frequent itemsets ทุกตัว
- Compactness

  ความสูงของ FP-tree ถูกจำกัดโดยจำนวน item สูงสุดในทรานแซกชัน

  ขนาดของ FP-tree ถูกจำกัดโดยรูปแบบการเกิดของ frequent items

การค้นหา Frequent itemsets จาก FP-tree

- divide-and-conquer: ใช้ FP-tree เพื่อขยายขนาด (grow) ของ frequent patterns แบบ recursive
  - สร้าง conditional pattern base และ conditional FP-tree ของ frequent item แต่ละตัว
  - ทำกระบวนการข้างต้นซ้ำสำหรับ conditional FP-tree ที่สร้างขึ้นใหม่แต่ละตัว จนกระทั่ง FP-tree ที่ได้เป็นเซตว่าง หรือ มี path เพียง path เดียว

Algorithm 2 (FP-growth : **Mining frequent patterns with** FP-tree **and by pattern fragment growth)**

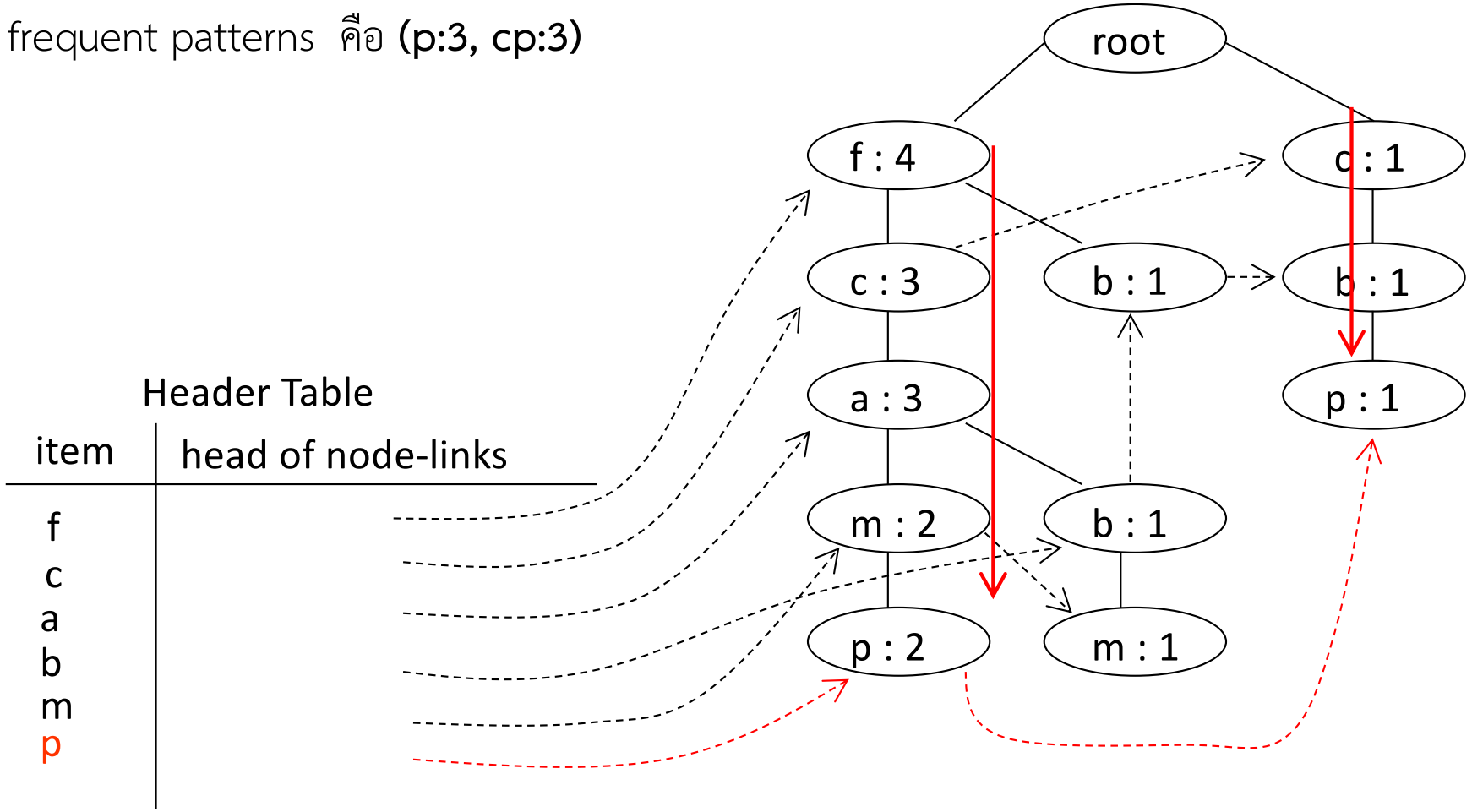
Input: FP-tree constructed based on Algorithm 1, using DB and a minimum support threshold ξ.

Output: The complete set of frequent patterns.

Method: Call FP-growth (FP-tree , $null$), which is implemented as follows.

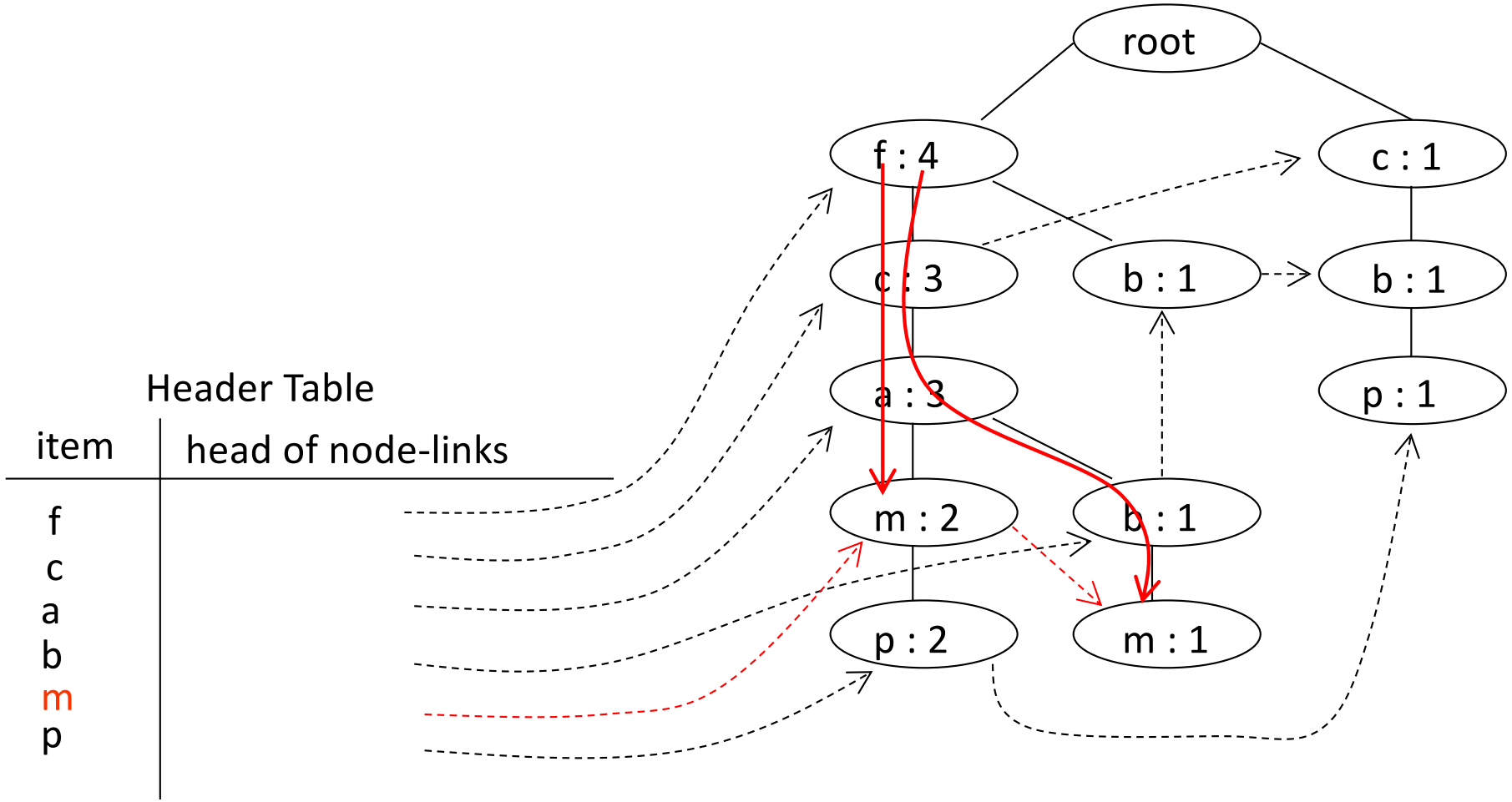
```
Procedure FP-growth (Tree, α)
{
(1)     IF      Tree contains a single path P
(2)     THEN FOR EACH combination (denoted as β) of the nodes in the path P DO
(3)             generate pattern β ∪ α with support = minimum support of nodes in β;
(4)     ELSE FOR EACH a_i in the header of Tree DO {
(5)             generate pattern β = a_i ∪ α with support = a_i.support;
(6)             Construct β's conditional pattern base and then β's conditional FP-tree Tree_β;
(7)             IF      Tree_β ≠ ∅
(8)             THEN Call FP-growth (Tree_β, β) }
}
```

- เริ่มจาก frequent item สุดท้ายใน header table : node *p*
- มี 2 paths มายัง p ดังนั้น conditional pattern base ของ p คือ
  { (f:2, c:2, a:2, m:2), (c:1, b:1) }
- conditional fp-tree คือ **(c:3)**
- frequent patterns คือ **(p:3, cp:3)**



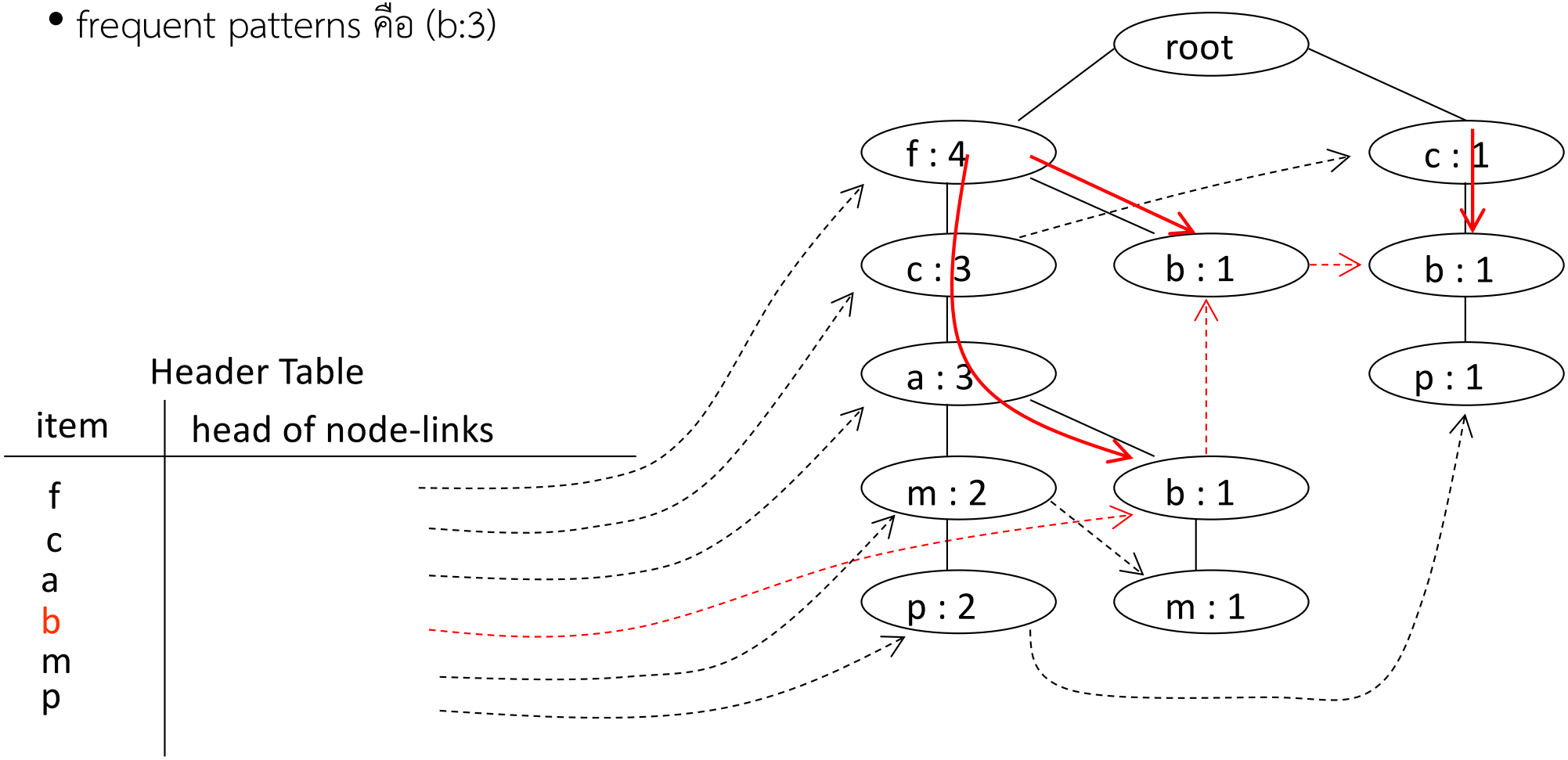
- item ถัดไปใน header table : node *m*
- มี 2 paths มายัง m ดังนั้น conditional pattern base ของ m คือ

  { (f:2, c:2, a:2), (f:1, c:1, a:1, b:1) }
- conditional fp-tree คือ **(f:3, c:3, a:3)**
- frequent patterns คือ **(m:3, am:3, cm:3, fm:3, cam:3, fam:3, fcm:3, fcam:3)**

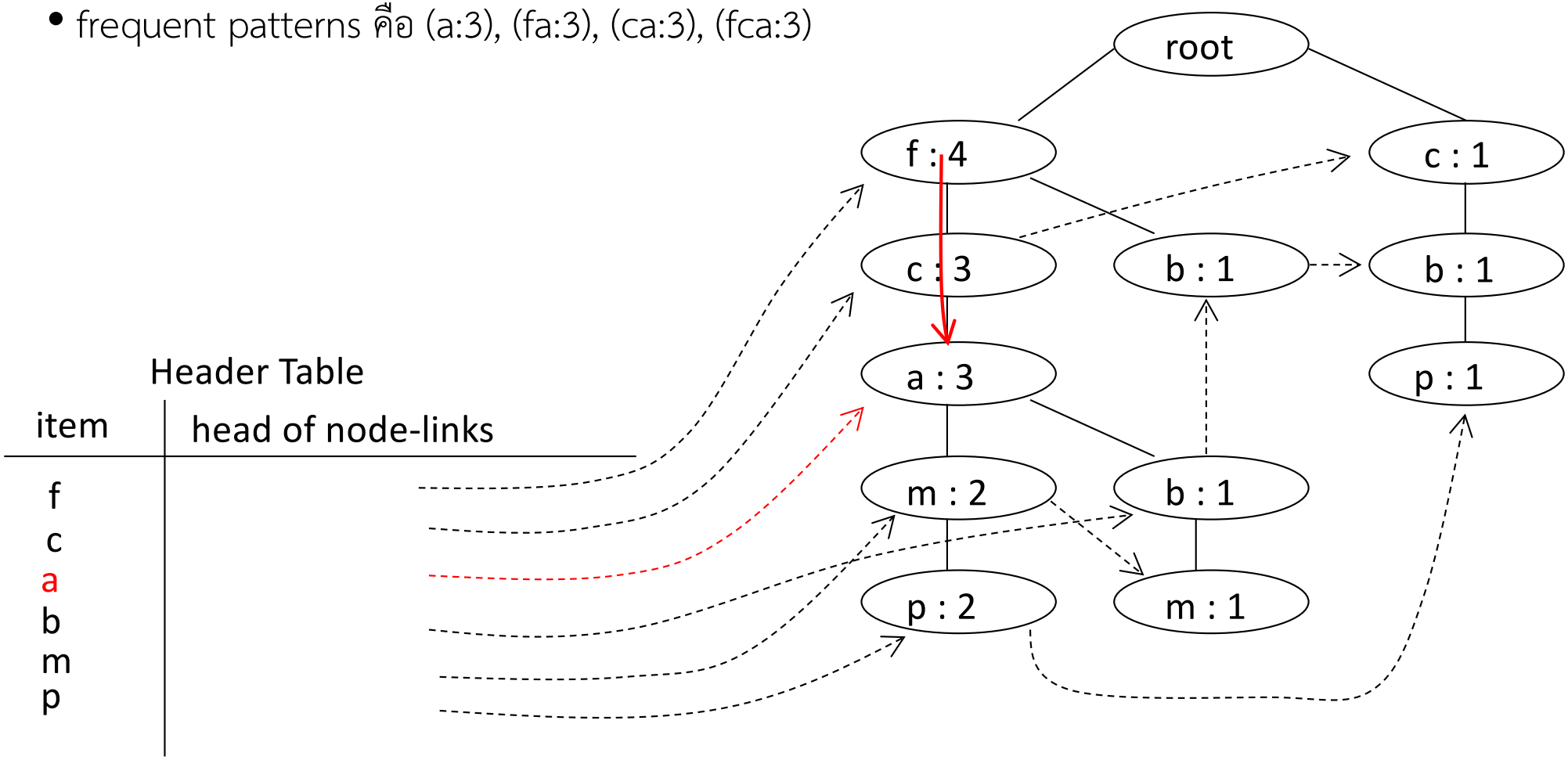


- item ถัดไปใน header table : node *b*
- มี 3 paths มายัง b ดังนั้น conditional pattern base ของ b คือ

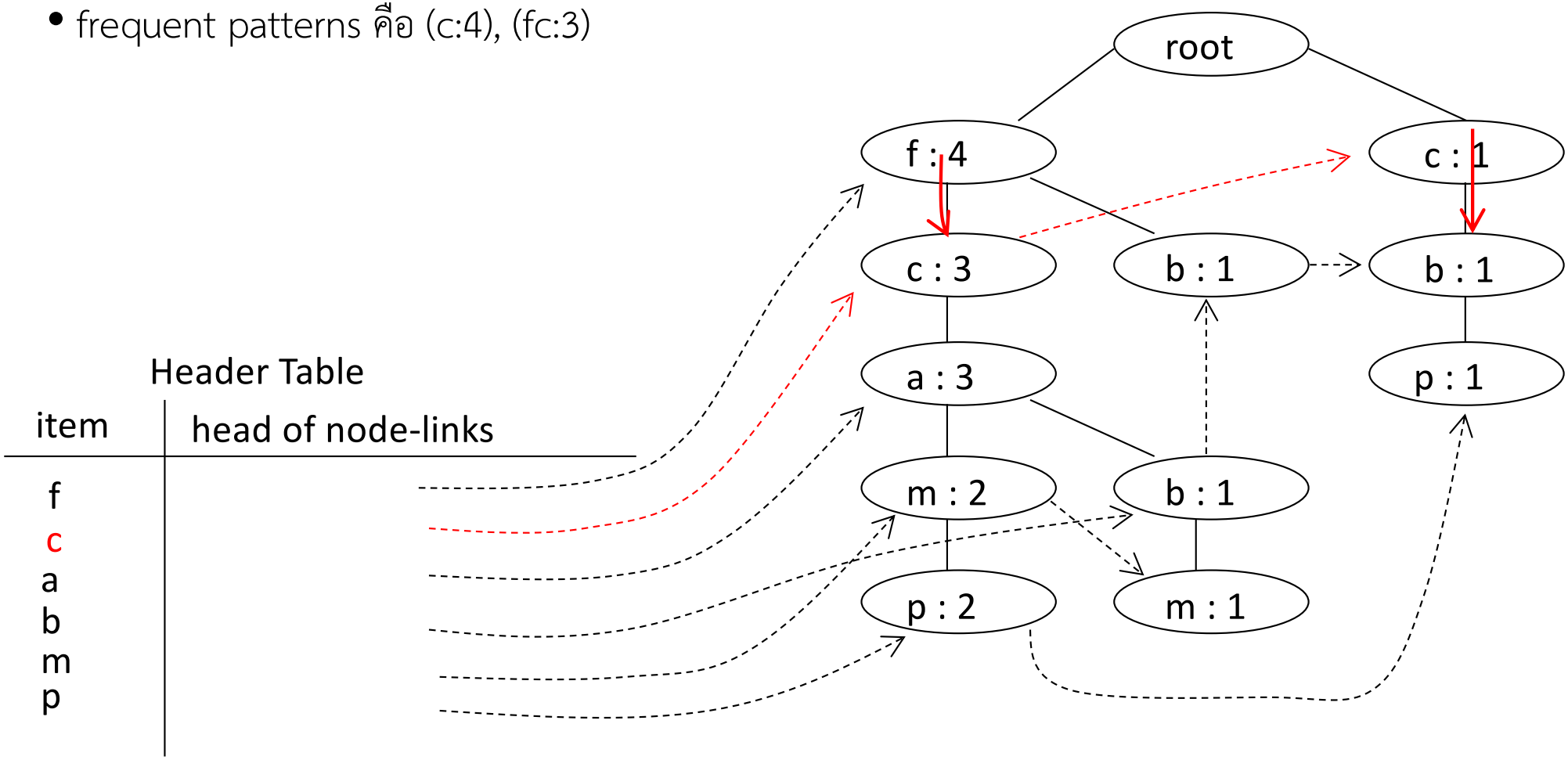
  { (f:1, c:1, a:1), (f:1), (c:1) }
- conditional fp-tree คือ เซตว่าง
- frequent patterns คือ (b:3)



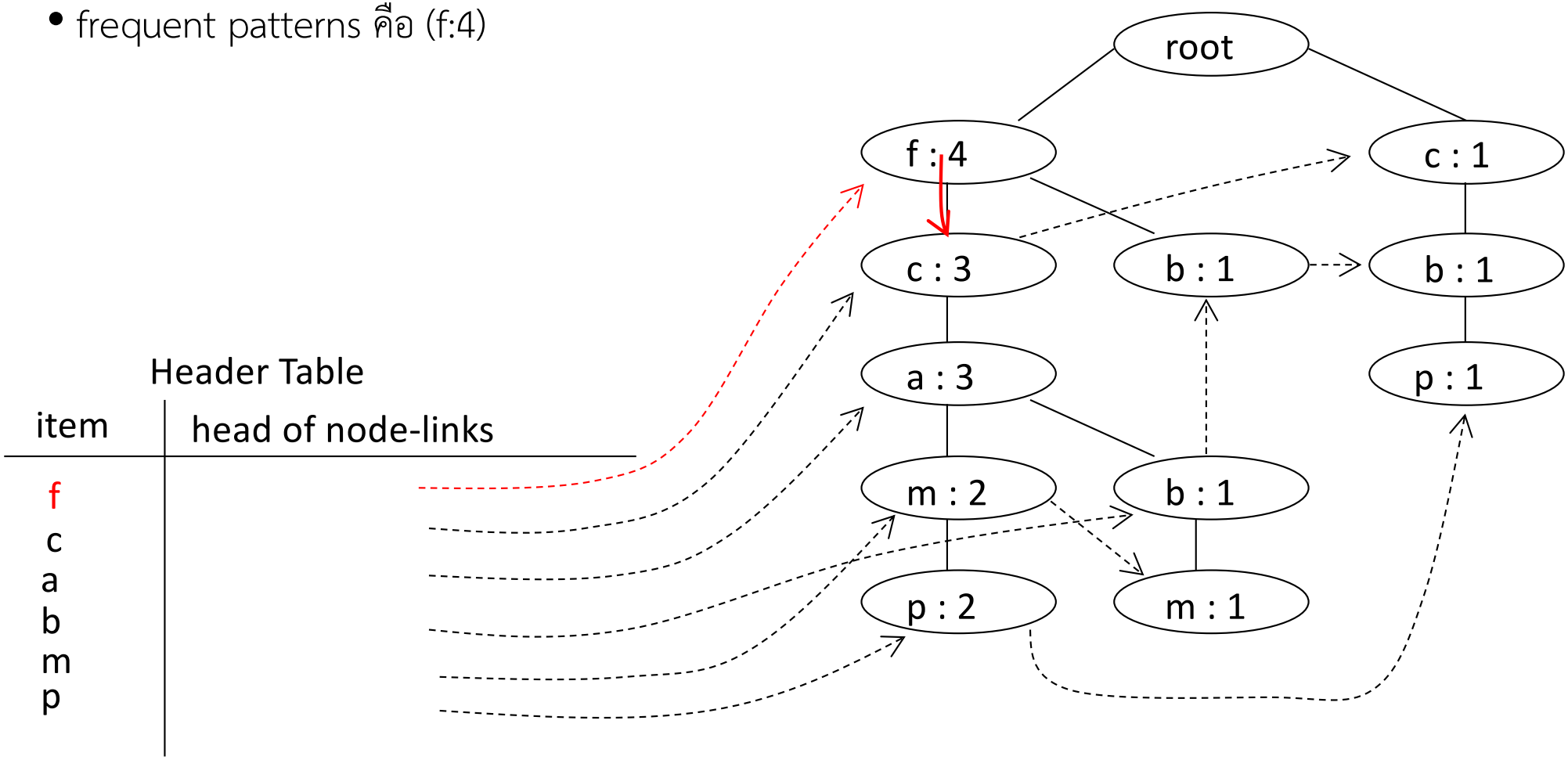
- item ถัดไปใน header table : node a
- มี 1 path มายัง a ดังนั้น conditional pattern base ของ a คือ
  { (f:3, c:3) }
- conditional fp-tree คือ (f:3, c:3)
- frequent patterns คือ (a:3), (fa:3), (ca:3), (fca:3)



- item ถัดไปใน header table : node c
- มี 2 path มายัง c , conditional pattern base ของ c คือ
  { (f:3) }
- conditional fp-tree คือ (f:3)
- frequent patterns คือ (c:4), (fc:3)



- item ถัดไปใน header table : node f
- มี 1 path มายัง f , conditional pattern base ของ f คือ { }
- conditional fp-tree คือ { }
- frequent patterns คือ (f:4)



สรุปผลลัพธ์

item	conditional pattern base	conditional FP-tree
p	$\{(f:2, c:2, a:2, m:2), (c:1, b:1)\}$	$\{(c:3)\}\|p$
m	$\{(f:4, c:3, a:3, m:2), (f:4, c:3, a:3, b:1, m:1)\}$	$\{(f:3, c:3, a:3)\}\|m$
b	$\{(f:4, c:3, a:3, b:1), (f:4, b:1), (c:1, b:1)\}$	$\emptyset$
a	$\{(f:3, c:3)\}$	$\{(f:3, c:3)\}\|a$
c	$\{(f:3)\}$	$\{(f:3)\}\|c$
f	$\emptyset$	$\emptyset$

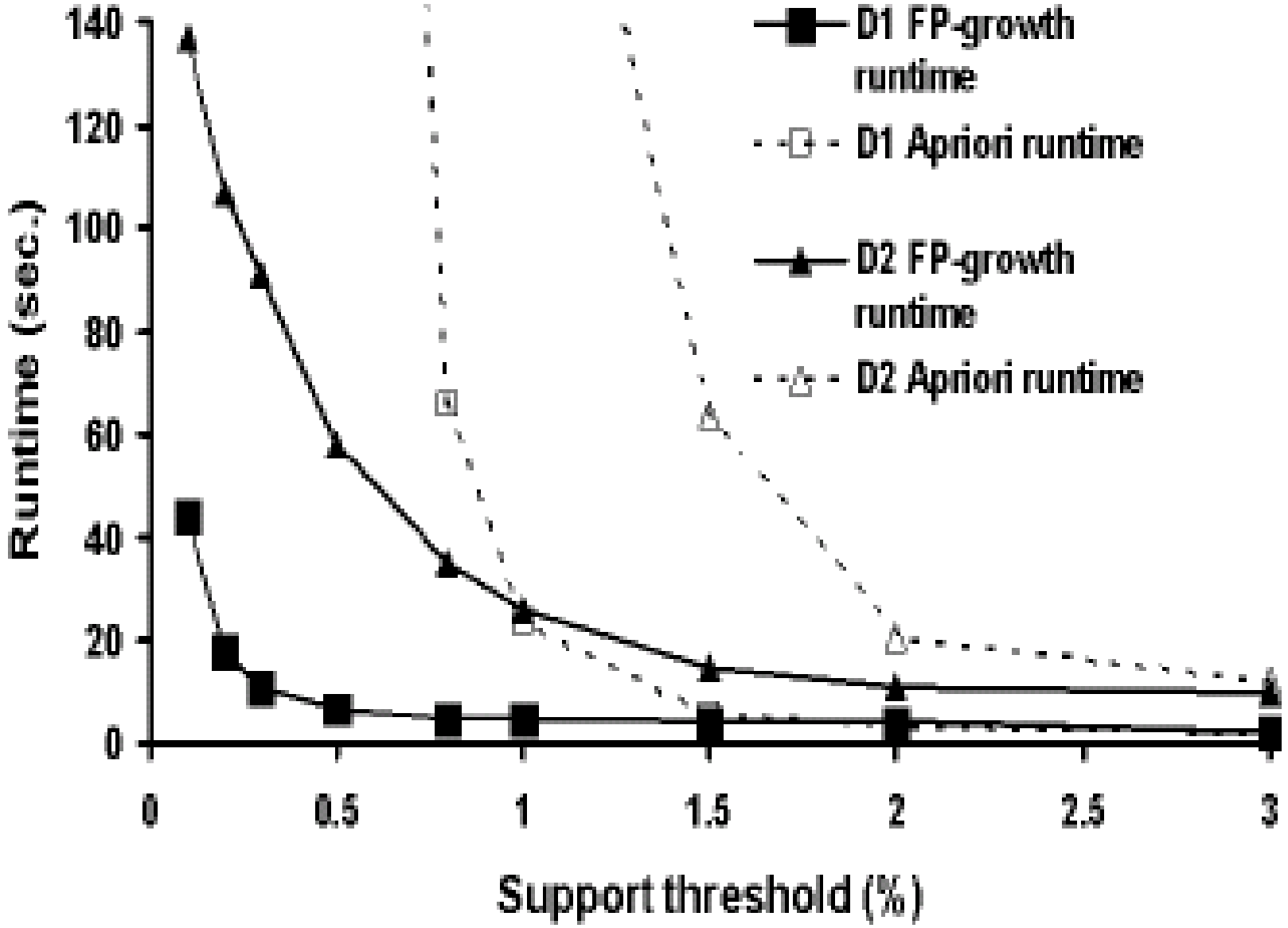
Frequent itemsets:

p:3	mc:3	a:3
pc:3	mf:3	ac:3
m:3	mfc:3	af:3
ma:3	f:4	afc:3
mac:3	fc:3	
maf:3	c:4	
macf:3	b:3	

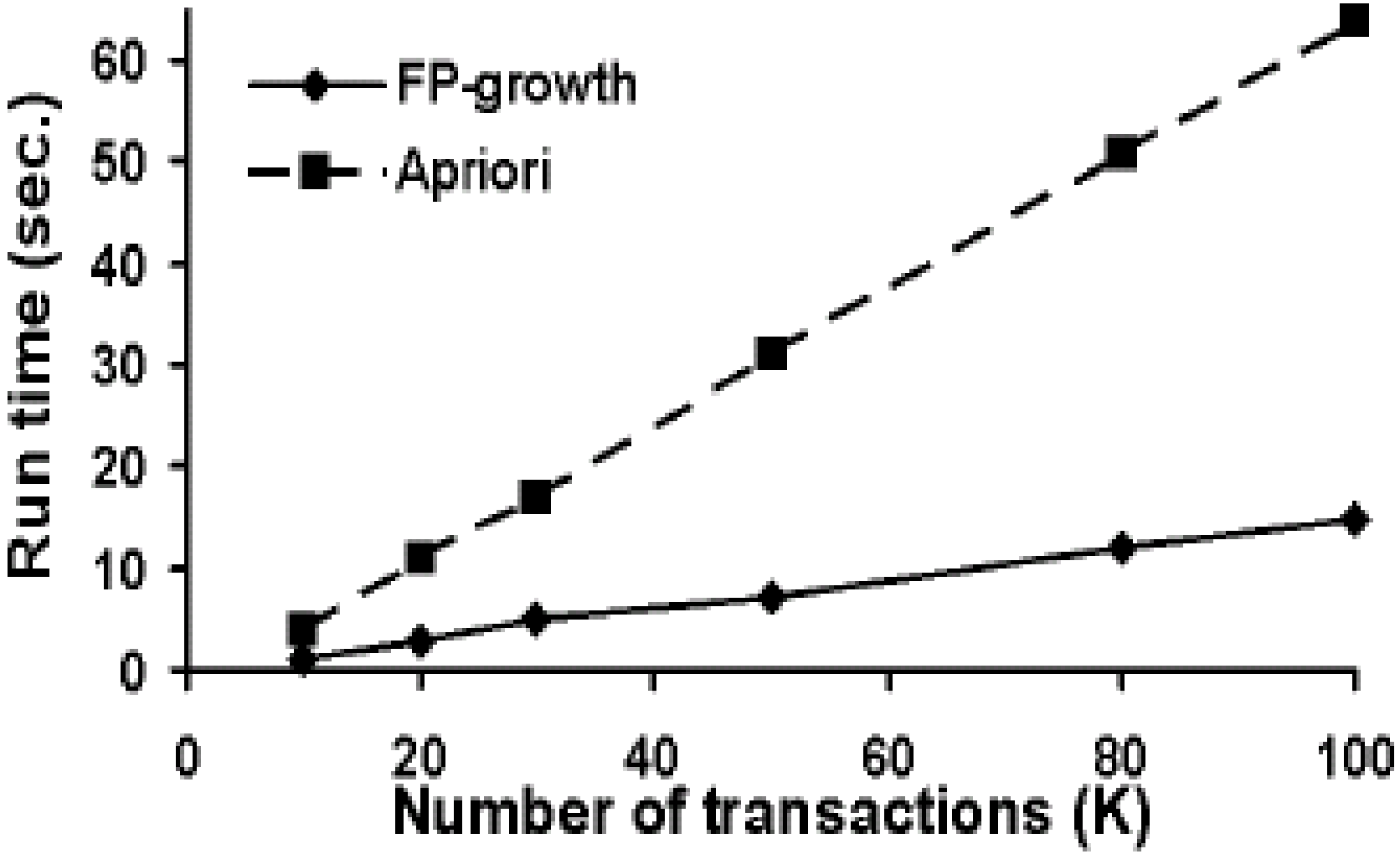
หัวข้อหลัก

- แนวคิดพื้นฐาน
- อัลกอริทึม Apriori
- อัลกอริทึม FP-Growth
- ประสิทธิภาพของอัลกอริทึม Apriori และ FP-Growth
- Association rule mining ด้วย Apache Spark

ประสิทธิภาพ: Scalability และ Support threshold



ประสิทธิภาพ: Scalability และ ขนาดของฐานข้อมูล



หัวข้อหลัก

- แนวคิดพื้นฐาน
- อัลกอริทึม Apriori
- อัลกอริทึม FP-Growth
- ประสิทธิภาพของอัลกอริทึม Apriori และ FP-Growth
- Association rule mining ด้วย Apache Spark

Spark MLlib : FP-Growth

- คลาส **FPGrowth ใน pyspark.ml.fpm** ใช้โมเดล PFP (Parallel FP-growth) ในการค้นหา frequent itemsets ค่า hyper-parameters ที่ต้องกำหนดให้กับโมเดลได้แก่
  - minSupport (มีค่าระหว่าง 0.0-1.0) คือ ค่า support ต่ำสุดของ freqeunt itemsets
  - minConfidence (มีค่าระหว่าง 0.0-1.0) คือค่า confidence ต่ำสุดสำหรับใช้ในการสร้าง association rule

- หลังจากสร้างโมเดลแล้ว frequent itemsets จะอยู่ใน DataFrame ชื่อ ***freqItemsets***, ส่วน association rules จะอยู่ใน DataFrame ชื่อ ***associationRules***

ตัวอย่างโปรแกรม Apache Spark (1/2)

```
from pyspark.ml.fpm import FPGrowth
from pyspark.sql import SparkSession

def create_df(spark, data):
  return spark.createDataFrame(data, ['id', 'items'])

def create_pcshop_dataframe(spark):
  pcshop_data = [
    (1,['antivirus', 'pc', 'printer']),
    (2,['antivirus', 'monitor']),
    (3,['antivirus', 'usb_stick']),
    (4,['antivirus', 'monitor', 'printer']),
    (5,['printer', 'usb_stick']),
    (6,['antivirus', 'usb_stick']),
    (7,['printer', 'usb_stick']),
    (8,['antivirus', 'pc', 'printer', 'usb_stick']),
    (9,['antivirus', 'printer', 'usb_stick']),
  ]
  return create_df(spark, pcshop_data)
```

ตัวอย่างโปรแกรม Apache Spark (1/2)

```
if __name__=='__main__':
  import sys

  if len(sys.argv) != 3:
     print("usage: frequent_pattern_mining.py <minsup> <minconf>")
     sys.exit(0)

  minsup  = float(sys.argv[1])
  minconf = float(sys.argv[2])

  spark = SparkSession\
     .builder\
     .appName("FP-Growth")\
     .getOrCreate()

  df = create_pcshop_dataframe(spark)
  #df = create_hans_dataframe(spark)

  fpGrowth = FPGrowth(itemsCol="items", minSupport=minsup, minConfidence=minconf)
  model = fpGrowth.fit(df)

  fsets = model.freqItemsets
  sorted_fsets = fsets.sort(fsets.items.asc())
  sorted_fsets.show(truncate=False)

  rules = model.associationRules
  rules.show(truncate=False)

  spark.stop()
```

ตัวอย่างโปรแกรม Apache Spark (1/2)

```
% bin/spark-submit python_samples/frequent_pattern_mining.py 0.22 0.6
```

```
+------------------------------+----+
|items                         |freq|
+------------------------------+----+
|[antivirus]                   |7   |
|[monitor]                     |2   |
|[monitor, antivirus]          |2   |
|[pc]                          |2   |
|[pc, antivirus]               |2   |
|[pc, printer]                 |2   |
|[pc, printer, antivirus]      |2   |
|[printer]                     |6   |
|[printer, antivirus]          |4   |
|[usb_stick]                   |6   |
|[usb_stick, antivirus]        |4   |
|[usb_stick, printer]          |4   |
|[usb_stick, printer, antivirus]|2   |
+------------------------------+----+
```

* ***lift*** is the rise in probability of having {Y} on the cart with the knowledge of {X} being present over the probability of having {Y} on the cart without any knowledge about presence of {X}.

Ref: https://towardsdatascience.com/association-rules-2-aa9a77241654

```
+--------------+----------+------------------+------------------+------------------+
|antecedent    |consequent|confidence        |lift              |support           |
+--------------+----------+------------------+------------------+------------------+
|[pc, printer] |[antivirus]|1.0               |1.2857142857142856|0.2222222222222222|
|[pc, antivirus]|[printer] |1.0               |1.5               |0.2222222222222222|
|[pc]          |[printer] |1.0               |1.5               |0.2222222222222222|
|[pc]          |[antivirus]|1.0               |1.2857142857142856|0.2222222222222222|
|[usb_stick]   |[printer] |0.6666666666666666|1.0               |0.4444444444444444|
|[usb_stick]   |[antivirus]|0.6666666666666666|0.8571428571428571|0.4444444444444444|
|[monitor]     |[antivirus]|1.0               |1.2857142857142856|0.2222222222222222|
|[printer]     |[antivirus]|0.6666666666666666|0.8571428571428571|0.4444444444444444|
|[printer]     |[usb_stick]|0.6666666666666666|1.0               |0.4444444444444444|
+--------------+----------+------------------+------------------+------------------+
```

ตัวอย่างโปรแกรม Apache Spark (2/2)

```
from pyspark.ml.fpm import FPGrowth
from pyspark.sql import SparkSession

def create_df(spark, data):
  return spark.createDataFrame(data, ['id', 'items'])

def create_hans_dataframe(spark):
  hans_data = [
    (100,['f','a','c','d','g','i','m','p']),
    (200,['a','b','c','f','l','m','o']),
    (300,['b','f','h','j','o']),
    (400,['b','c','k','s','p']),
    (500,['a','f','c','e','l','p','m','n']),
  ]
  return create_df(spark, hans_data)
```

ตัวอย่างโปรแกรม Apache Spark (2/2)

```
if __name__=='__main__':
  import sys

  if len(sys.argv) != 3:
     print("usage: frequent_pattern_mining.py <minsup> <minconf>")
     sys.exit(0)

  minsup  = float(sys.argv[1])
  minconf = float(sys.argv[2])

  spark = SparkSession\
     .builder\
     .appName("FP-Growth")\
     .getOrCreate()

  #df = create_pcshop_dataframe(spark)
  df = create_hans_dataframe(spark)

  fpGrowth = FPGrowth(itemsCol="items", minSupport=minsup, minConfidence=minconf)
  model = fpGrowth.fit(df)

  fsets = model.freqItemsets
  sorted_fsets = fsets.sort(fsets.items.asc())
  sorted_fsets.show(truncate=False)

  rules = model.associationRules
  rules.show(truncate=False)

  spark.stop()
```

ตัวอย่างโปรแกรม Apache Spark (2/2)

```
% bin/spark-submit python_samples/frequent_pattern_mining.py 0.6 0.6
```

```
+------------+----+
|items       |freq|
+------------+----+
|[a]         |3   |
|[a, c]      |3   |
|[a, f]      |3   |
|[a, f, c]   |3   |
|[b]         |3   |
|[c]         |4   |
|[f]         |4   |
|[f, c]      |3   |
|[m]         |3   |
|[m, a]      |3   |
|[m, a, c]   |3   |
|[m, a, f]   |3   |
|[m, a, f, c]|3   |
|[m, c]      |3   |
|[m, f]      |3   |
|[m, f, c]   |3   |
|[p]         |3   |
|[p, c]      |3   |
+------------+----+
```

```
+---------+---------+----------+------------------+-------+
|antecedent|consequent|confidence|lift              |support|
+---------+---------+----------+------------------+-------+
|[m, a]    |[f]      |1.0       |1.25              |0.6    |
|[m, a]    |[c]      |1.0       |1.25              |0.6    |
|[m, a, c] |[f]      |1.0       |1.25              |0.6    |
|[m, a, f] |[c]      |1.0       |1.25              |0.6    |
|[a, f]    |[c]      |1.0       |1.25              |0.6    |
|[a, f]    |[m]      |1.0       |1.6666666666666667|0.6    |
|[m, c]    |[f]      |1.0       |1.25              |0.6    |
|[m, c]    |[a]      |1.0       |1.6666666666666667|0.6    |
|[a, f, c] |[m]      |1.0       |1.6666666666666667|0.6    |
|[a]       |[f]      |1.0       |1.25              |0.6    |
|[a]       |[c]      |1.0       |1.25              |0.6    |
|[a]       |[m]      |1.0       |1.6666666666666667|0.6    |
|[a, c]    |[f]      |1.0       |1.25              |0.6    |
|[a, c]    |[m]      |1.0       |1.6666666666666667|0.6    |
|[m, f]    |[c]      |1.0       |1.25              |0.6    |
|[m, f]    |[a]      |1.0       |1.6666666666666667|0.6    |
|[m, f, c] |[a]      |1.0       |1.6666666666666667|0.6    |
|[m]       |[f]      |1.0       |1.25              |0.6    |
|[m]       |[a]      |1.0       |1.6666666666666667|0.6    |
|[m]       |[c]      |1.0       |1.25              |0.6    |
+---------+---------+----------+------------------+-------+
```