АСОЦІАТИВНІ ПРАВИЛА DATA MINING.

АЛГОРИТМ FP-РОСТУ

Надія І. Недашківська n.nedashkivska@gmail.com

Переваги і недоліки алгоритму Apriori

Переваги Apriori :

- простота
- швидке зменшення кількості згенерованих кандидатів при встановленні високого значення порогу мінімальної підтримки

Недоліки Apriori :

- багаторазове сканування БД транзакцій
- велика кількість згенерованих кандидатів при великому розмірі БД або при низькому значенні порогу мінімальної підтримки

АЛГОРИТМ FP-РОСТУ

Надія І. Недашківська n.nedashkivska@gmail.com

Алгоритм FP-росту

Frequent pattern growth =“вирощування частих наборів”
Нам потрібно ефективно зберігати в оперативній пам’яті множину підмножин – часті набори.

- БД транзакцій представляється у вигляді компактного дерева - FP-дерева.
- FP-дерево містить повну інформацію про всі часті набори БД транзакцій.
- FP-дерево забезпечує ефективне отримання частих наборів, на відміну від витратної процедури в алгоритмі Apriori.

Етап 1: Побудова FP-дерева на основі БД транзакцій

Етап 2: Рекурсивний пошук частих наборів в FP-дереві.

Поняття префіксного дерева (FP-дерево, frequent pattern)

FP-дерево G:

- Вершини – об'єкти $i \in I$

Різні вершини дерева можуть містити одні й ті самі об'єкти.

- Шлях від кореня g_0 до вершини g – набір об'єктів $F \subseteq I$

$G(i) = \{g \in G : g = i\}$ - множина вершин для об'єкта i.

$Supp(i) = \sum_{g \in G(i)} Supp(g)$ - підтримка об'єкта i.

- Рівні дерева відповідають об'єктам за спаданням $Supp(i)$,

$Supp(i) \geq Supp_{\min}$. Маємо порядок на множині об'єктів.

Приклад FP-дерева

$$I = \{a, b, c, d, e, f\}$$

(f, 8), (e, 7), (d, 6), (a, 5), (b, 5), (c, 4)

БД транзакцій D	Відсортована БД D
b d f	f d b
a b c d e f	f e d a b c
c e	e c
b d e f	f e d b
a b d	d a b
c e f	f e c
a b d e f	f e d a b
a f	f a
e f	f e
a c d e f	f e d a c

Перший стовпчик – це БД транзакцій. Перший раз проходимо цю БД і підраховуємо підтримку кожного об'єкту. Сортуємо об'єкти за спаданням величини підтримки. Сортуємо БД транзакцій.

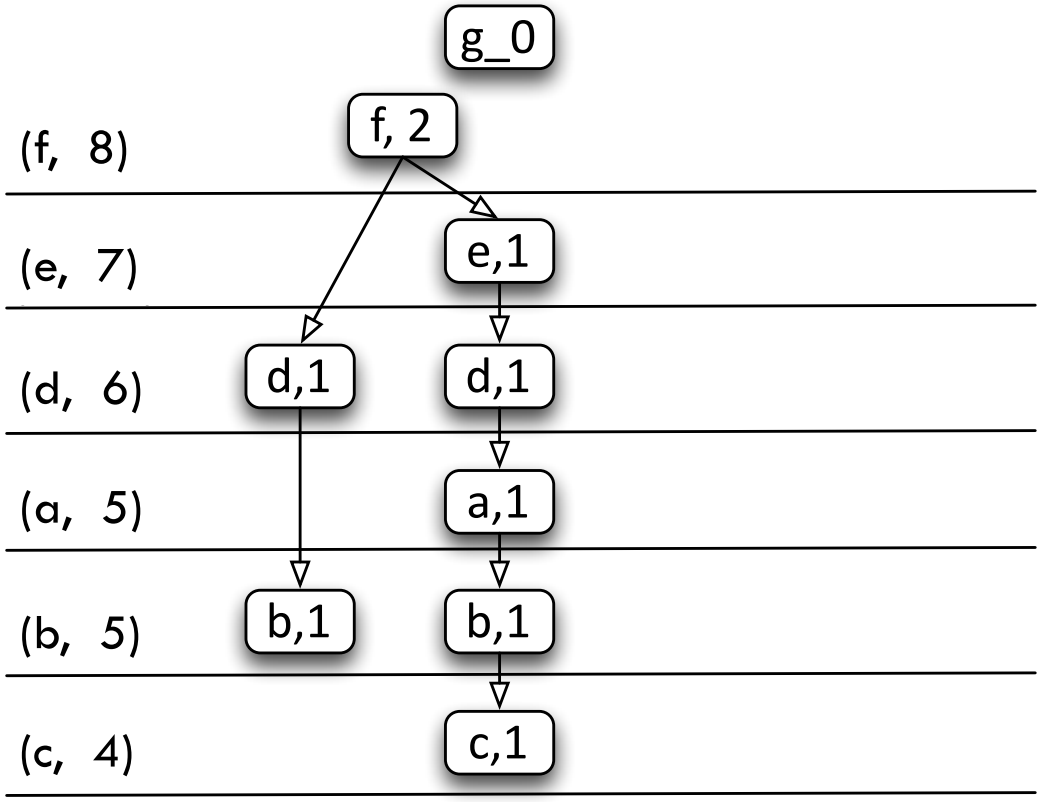
Елементи відсортованої БД можуть розглядатися як слова або словник. Тоді можна використати відомі ефективні методи збереження словників і пошуку в них.

Приклад FP-дерева

$$I = \{a, b, c, d, e, f\}$$

БД транзакцій D	Відсортована БД D
b d f	f d b
a b c d e f	f e d a b c
c e	e c
b d e f	f e d b
a b d	d a b
c e f	f e c
a b d e f	f e d a b
a f	f a
e f	f e
a c d e f	f e d a c

Побудова дерева: розглядається кожне слово – транзакція у відсортованій БД. Обробили перші два слова (транзакції), отримали таке дерево:

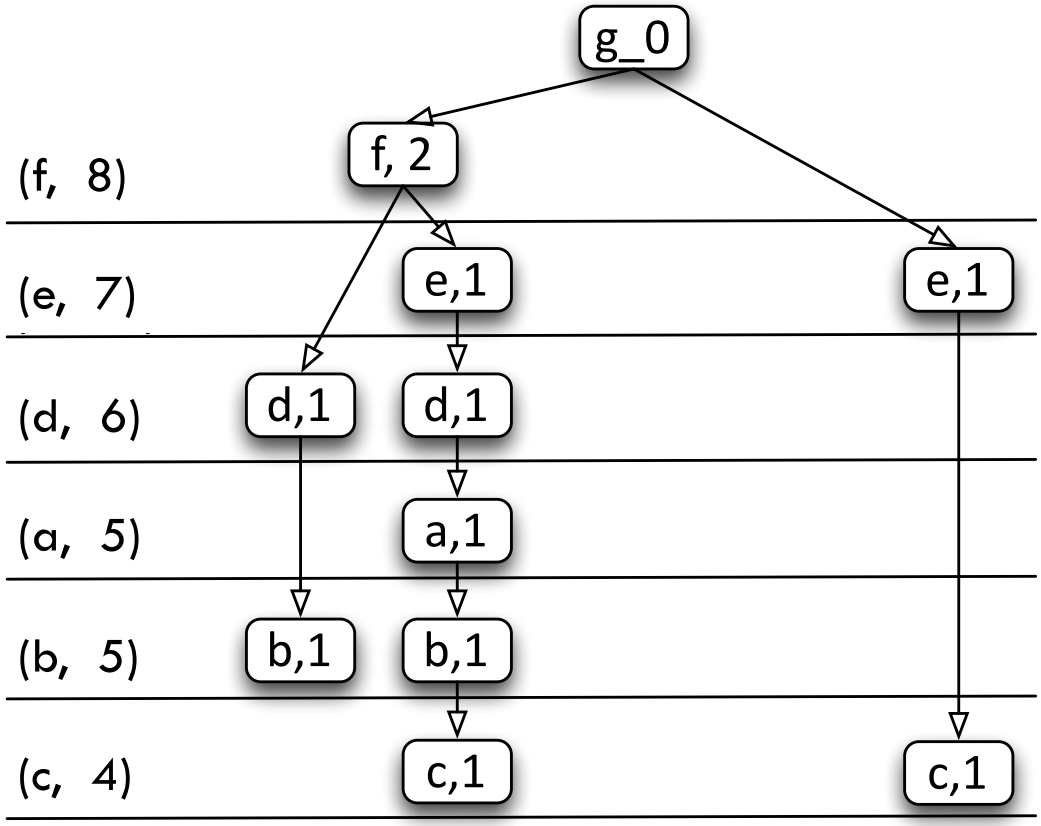


Приклад FP-дерева

$$I = \{a, b, c, d, e, f\}$$

БД D	Відсортована БД D
b d f	f d b
a b c d e f	f e d a b c
c e	e c
b d e f	f e d b
a b d	d a b
c e f	f e c
a b d e f	f e d a b
a f	f a
e f	f e
a c d e f	f e d a c

Обробили перші три слова (транзакції), отримали таке дерево:

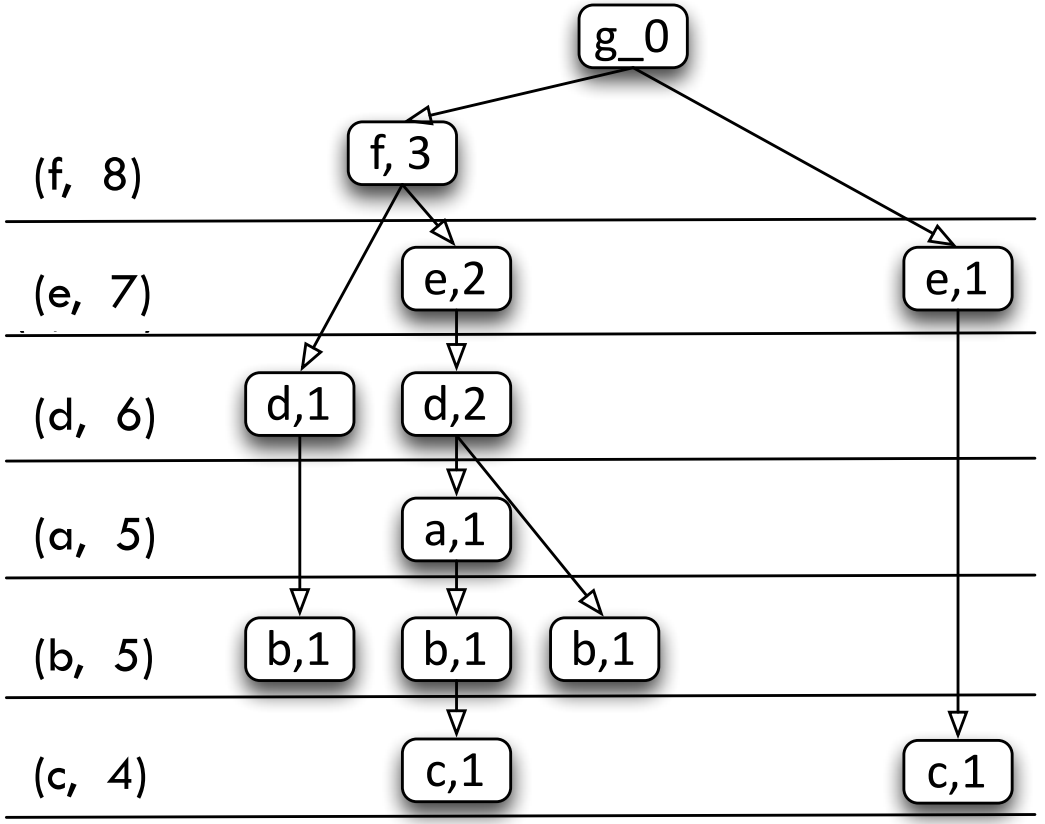


Приклад FP-дерева

$$I = \{a, b, c, d, e, f\}$$

БД D	Відсортована БД D
b d f	f d b
a b c d e f	f e d a b c
c e	e c
b d e f	f e d b
a b d	d a b
c e f	f e c
a b d e f	f e d a b
a f	f a
e f	f e
a c d e f	f e d a c

Обробили перші чотири слова (транзакції), отримали таке дерево:

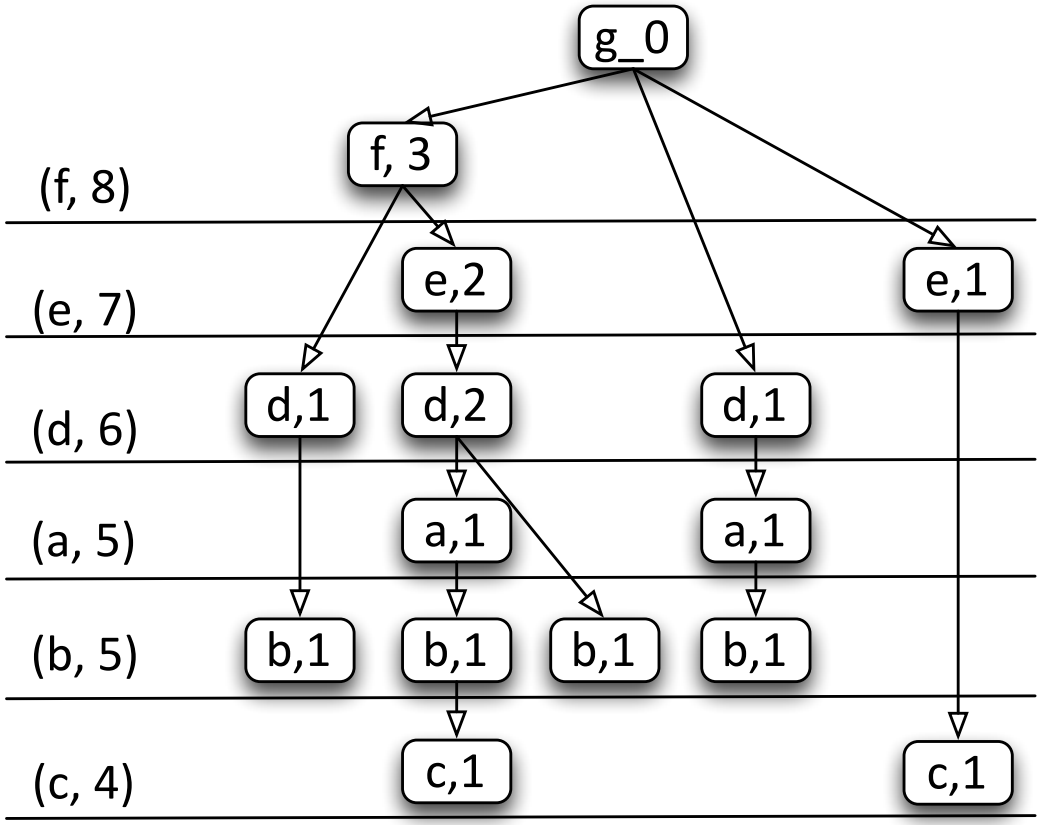


Приклад FP-дерева

$$I = \{a, b, c, d, e, f\}$$

БД D	Відсортована БД D
b d f	f d b
a b c d e f	f e d a b c
c e	e c
b d e f	f e d b
a b d	d a b
c e f	f e c
a b d e f	f e d a b
a f	f a
e f	f e
a c d e f	f e d a c

Обробили перші п’ять слів (транзакцій), отримали таке дерево:

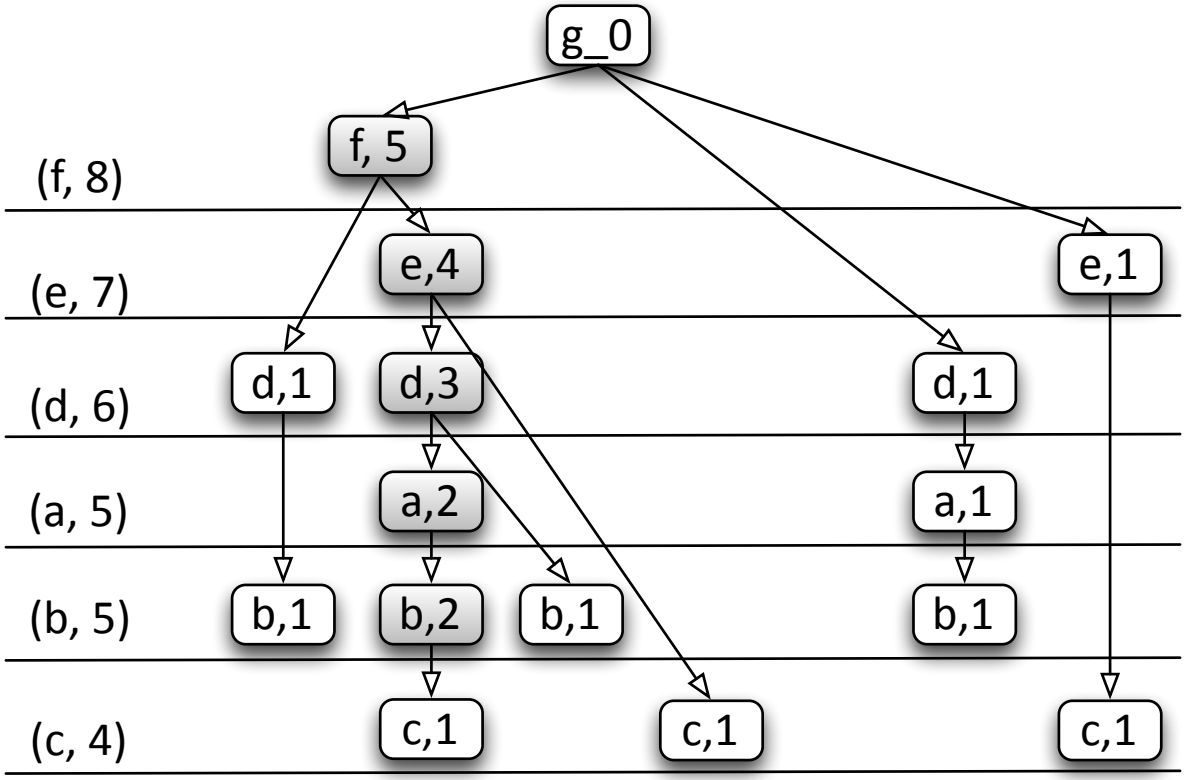


Приклад FP-дерева

$$I = \{a, b, c, d, e, f\}$$

БД D	Відсортована БД D
b d f	f d b
a b c d e f	f e d a b c
c e	e c
b d e f	f e d b
c e f	f e c
a b d	d a b
a b d e f	f e d a b
a f	f a
e f	f e
a c d e f	f e d a c

Обробили перші сім транзакцій:



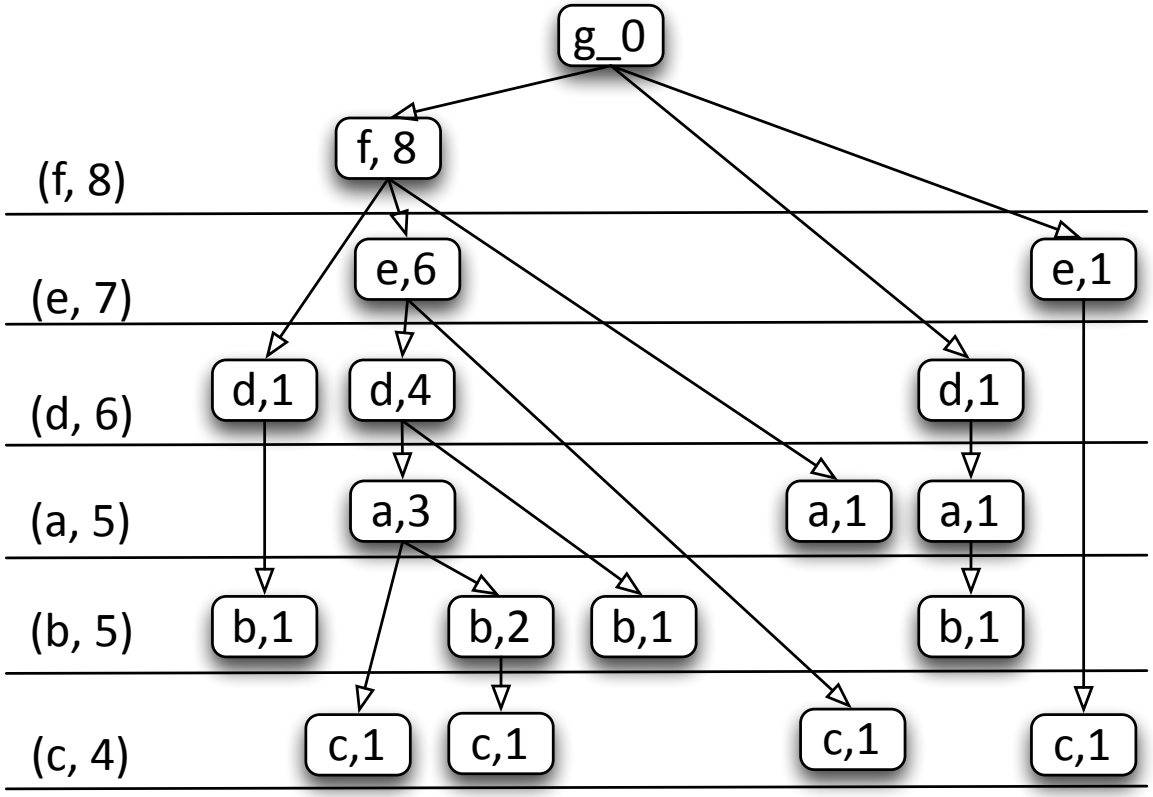
Нове слово – сьоме – розмістилося на існуючому шляху, заради цього і будуються FP-дерева. Тобто, ми будемо читати нові транзакції (слова) з БД, а структура дерева змінюватися не буде.

Приклад FP-дерева

$$I = \{a, b, c, d, e, f\}$$

БД D	Відсортована БД D
b d f	f d b
a b c d e f	f e d a b c
c e	e c
b d e f	f e d b
c e f	f e c
a b d	d a b
a b d e f	f e d a b
a f	f a
e f	f e
a c d e f	f e d a c

Обробили перші вісім транзакцій:



Якщо в початковій БД об’єкт повторюється багато-разово, то в FP-дереві він представляється у вигляді вузла, а його підтримка вказує на те, скільки разів даний об’єкт з’являється.

Алгоритм FP-росту: **етап 1** - побудова FP-дерева

Дано: $D = \{T_1, T_2, \ldots, T_m\}$ – множина транзакцій – навчальна вибірка

Знайти: **FP-дерево** G

$$G = \{g \mid g = (Name(g), Supp(g), Child(g))\}$$

Вузол FP-дерева – це структура, яка зберігає значення вузла Name, значення його підтримки Supp, а також посилання на всі його дочірні елементи Child.

Для кожного елемента кожної відсортованої транзакції з вхідного набору будуються вузли за таким правилом:

- якщо для чергового елемента в поточному вузлі є нащадок, що містить цей елемент, то новий вузол не створюється, а підтримка цього нащадка збільшується на 1,
- в іншому випадку створюється новий вузол-нащадок з підтримкою 1. Поточним вузлом при цьому стає знайдений або побудований вузол.

Алгоритм FP-росту: **етап 1** - побудова FP-дерева

Дано: $D = \{T_1, T_2, \ldots, T_m\}$ – множина транзакцій – навчальна вибірка

Знайти: **FP-дерево** G

$G = \{g \mid g = (Name(g), Supp(g), Child(g))\}$

1. В кожній транзакції впорядкувати об'єкти $i \in I$ за спаданням $Supp(i)$. Фільтрація – вилучення об'єктів з $Supp < Supp_{\min}$

2. Для всіх $T_k \in D$

   2.1. $g := g_0$

   2.2. Для всіх $i \in I : i \in T_k$

      2.2.1. Якщо у g немає дочірньої вершини, рівної i, то

      Створити дочірню вершину x, рівну i,

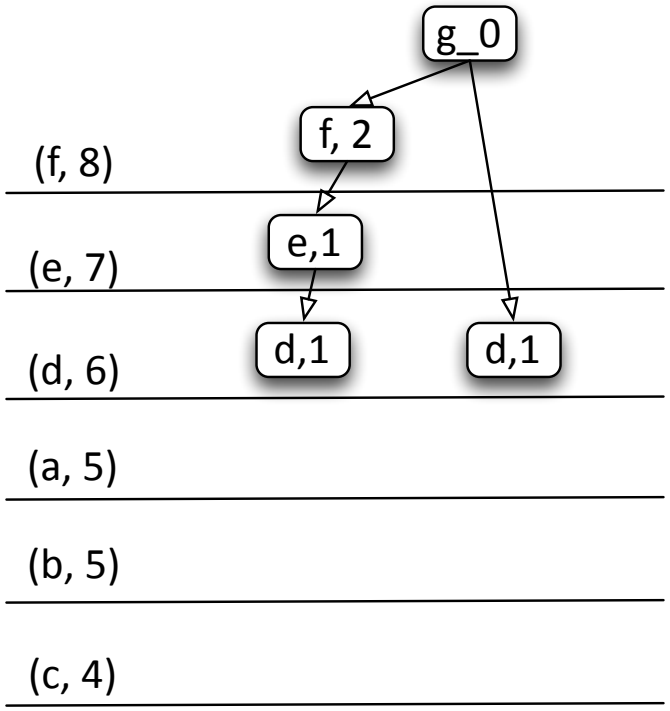
      $Name(x) := i \qquad Supp(x) := 0$

      2.2.2. $Supp(x) := Supp(x) + 1 \qquad g := x$

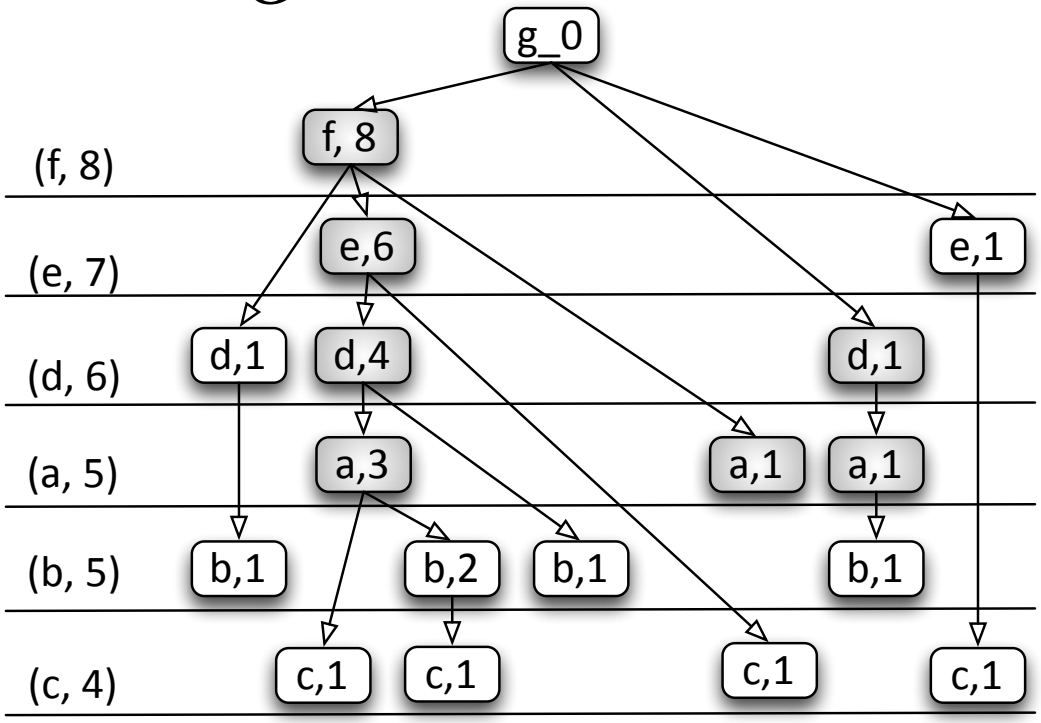
Пошук частих наборів в FP-дереві. Умовне FP-дерево

Умовне FP-дерево $G' := G \mid i$ – це FP-дерево за підмножиною транзакцій $D_i = \{T_k \mid i \in T_k\}$, які містять заданий об'єкт $i \in I$, з якого вилучені вершини $g \in G(i)$ та всі їх потомки.

$G' := G \mid "a"$



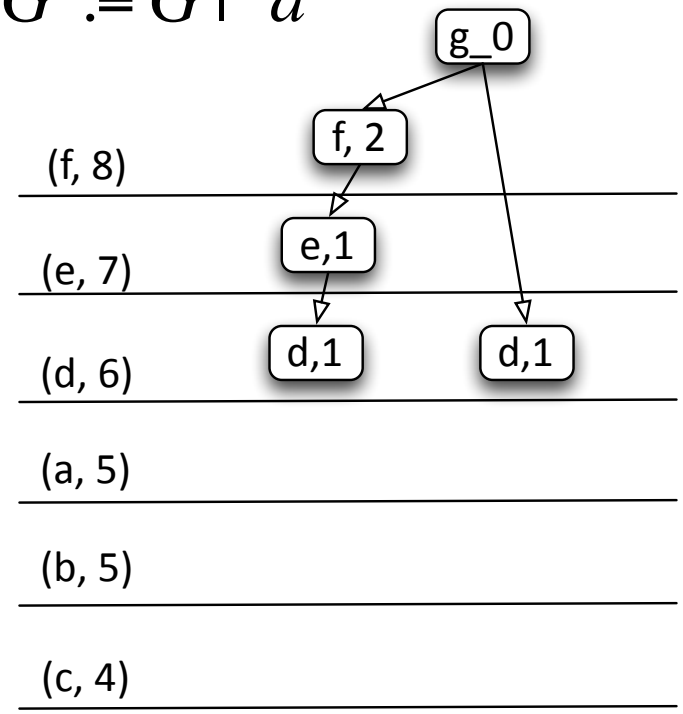
G



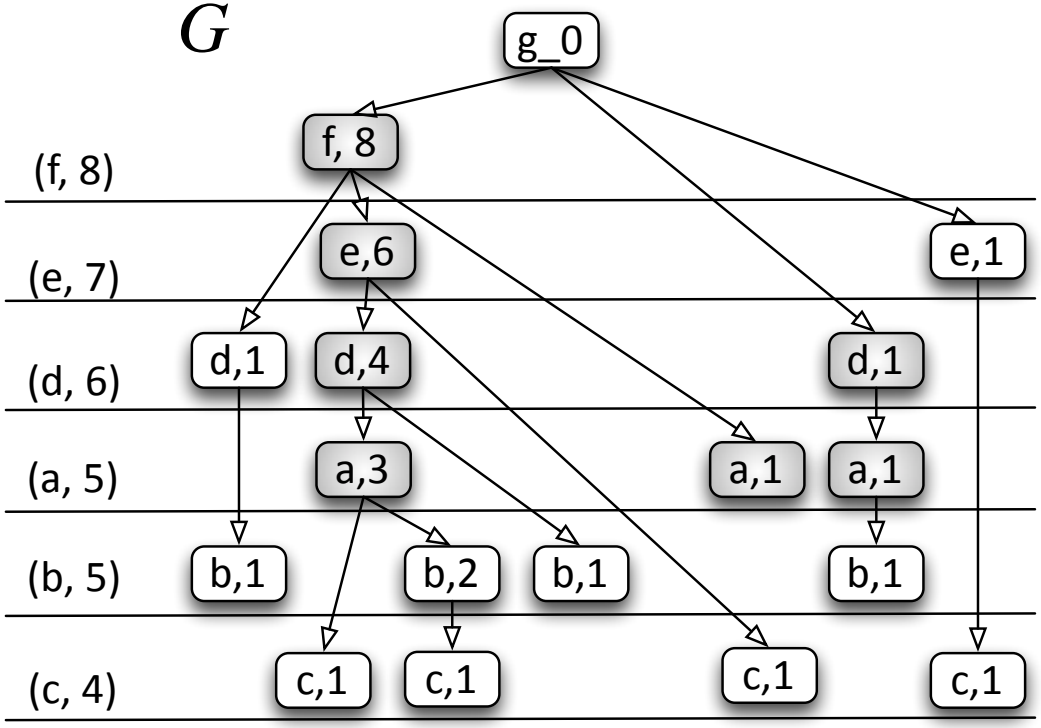
Пошук частих наборів в FP-дереві. Умовне FP-дерево

Умовне FP-дерево $G' := G \mid i$ – це FP-дерево за підмножиною транзакцій $D_i = \{T_k \mid i \in T_k\}$. БД D_i має набагато менший об'єм в порівнянні з початковою БД транзакцій D. Умовне дерево G' буде, як правило, набагато меншим порівняно з початковим FP-деревом G. Алгоритм, побудований на FP-деревах, дуже економний.

$G' := G \mid "a"$



G



Алгоритм побудови умовного FP-дерева

Дано: FP-дерево G , об'єкт $i \in I$

Знайти: **умовне FP-дерево** $G' := G \mid i$ для об'єкта i

1. В дереві G вилучаємо шляхи, які не містять об'єкт i .

2. В дереві G вилучаємо потомки вершин, які відповідають об'єкту i .

3. Перераховуємо підтримку вузлів, що залишилися в G :

$$Supp(g) := \sum_{x \in Child(g)} Supp(x) \quad \text{для всіх} \quad g \in G$$

4. В дереві G вилучаємо вершини, які відповідають об'єкту i .

5. Результуюче дерево – шукане умовне FP-дерево.

Алгоритм FP-росту: **етап 2 - пошук частих наборів в FP-дереві**

Дано: FP-дерево G , набір об'єктів F

Знайти: **часті набори для** F

$F := \varnothing$

Процедура $\mathrm{FP}(G, F)$

Для всіх $i \in I : G(i) \neq \varnothing$ по рівням **знизу вверх**

Якщо $Supp(i) \geq Supp_{\min}$, то

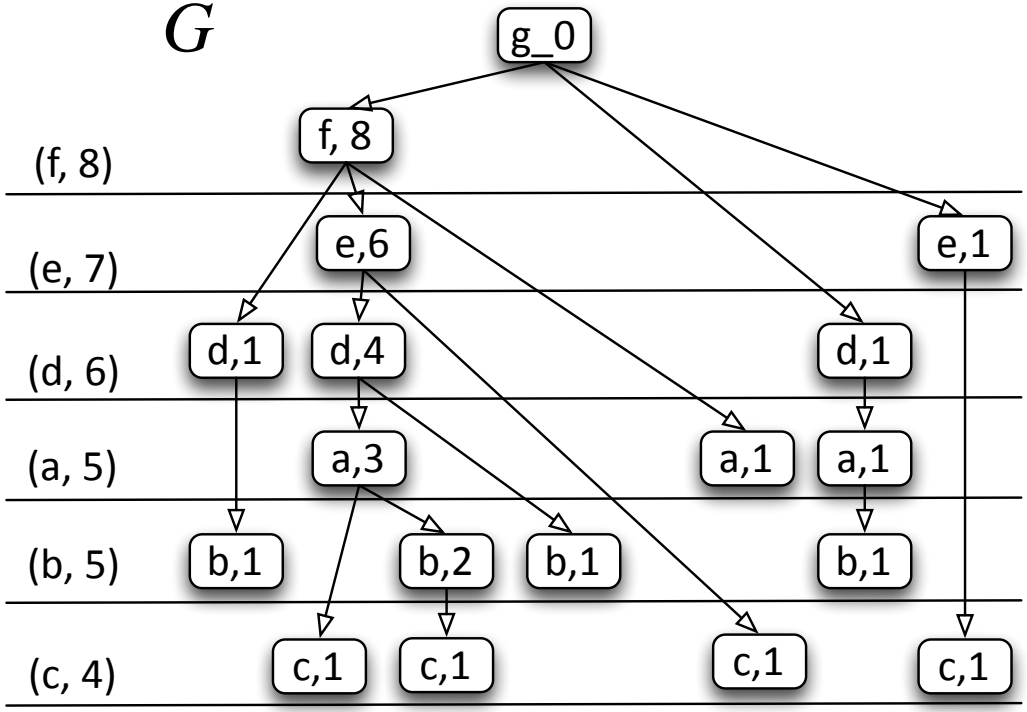
1. $F' := F \cup \{i\}$ - частий набір,
2. побудувати **умовне FP-дерево** G' за об'єктом i ,
3. $\mathrm{FP}(G', F')$ - знайти часті набори за деревом G' для частого набору F', в якому є об'єкт i .

Приклад пошуку частих наборів за алгоритмом FP-росту

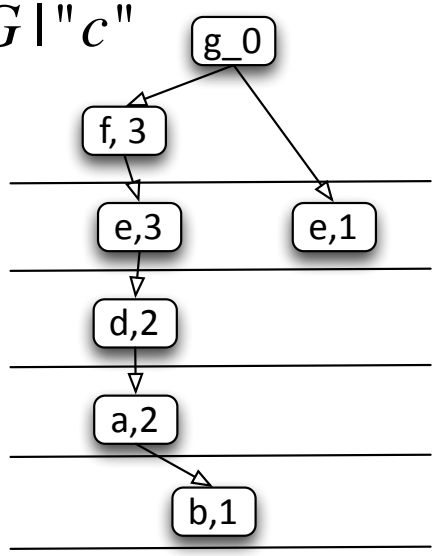
$F := \varnothing$ $\quad Supp_{\min} := 3$ $\quad i := "c"$ $\quad F' := \{c\}$

$Supp(F') = Supp(\{c\}) = 4$

G

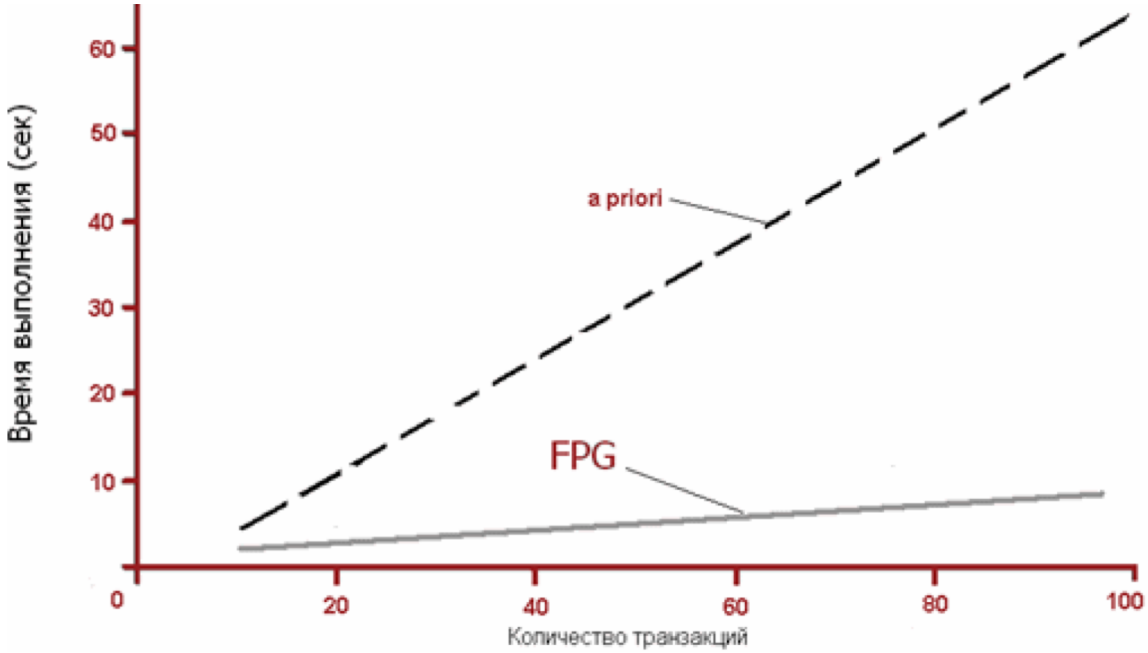


$G' := G \mid "c"$



Порівняння алгоритмів Apriori та FP-росту

Зі збільшенням об’єму БД транзакцій часові витрати на пошук частих наборів ростуть набагато повільніше при використанні алгоритму FP-росту (FPG) в порівнянні з Apriori.



https://basegroup.ru/

Переваги і недоліки алгоритму FP-росту

Переваги:

- дозволяє уникнути затратної процедури генерації кандидатів, характерної для Apriori і Eclat
- стиснення БД в компактну структуру для швидкого отримання частих наборів
- число сканування БД зменшено до двох разів
- розмір дерева зазвичай є меншим за розмір вхідного набору даних

Недоліки:

- побудова дерева – витратна за часом операція