Интеллектуальный анализ данных
(Data Mining)

Шорохов С.Г.

кафедра математического моделирования и искусственного интеллекта

Лекция 2. Поиск ассоциативных правил

Рекомендательные системы

Рекомендательные системы — программы, которые, имея определенную информацию о пользователе, пытаются предсказать, какие объекты (фильмы, музыка, книги, новости, веб-сайты и т.п.) будут ему интересны. Рекомендательные системы являются одним из ключевых драйверов увеличения продаж для бизнеса.

Данные в задаче рекомендаций выглядят как таблица, по строкам которой клиенты, по столбцам — товары (фильмы, книги и т.д.), и для некоторых ячеек известна оценка товара клиентом (для большинства ячеек таблицы оценка не известна, поскольку каждый клиент, скорее всего, взаимодействовал только с небольшим количеством товаров). Значения в ячейках могут задавать явную оценку товара клиентом (например, если он нажал в интерфейсе на некоторое количество звездочек) или неявную, например, клиент зашел на страницу товара и не купил его, или пропустил в списке товаров и не зашел. Для клиентов и товаров могут быть известны некоторые признаки, как в других задачах машинного обучения.

Задача рекомендаций ставится так: для каждого клиента мы хотим найти товары, которые он захочет купить, кроме тех, что он уже купил.

Подходы к построению рекомендательных систем

- Контентный подход

Для каждого клиента и/или продукта создаются профили с признаками. Рассматриваются пары (клиент, товар), в качестве признаков — совокупность признаков товара и клиента, а в качестве целевой переменной — оценка товара клиентом. Для решения этой задачи регрессии можно использовать любые методы, например линейную регрессию или решающие деревья. Для построения рекомендаций для клиента для всех пар (этот клиент, товар) выполняется прогнозирование и выбираются для рекомендаций те товары, для которых предсказано наибольшее значение оценки. Описанный подход называется контентным (content-based).

- Коллаборативная фильтрация

Чтобы порекомендовать клиенту товары, алгоритм определяет товары, которые клиент уже купил, выделяет клиентов, купивших ранее эти же товары и находит другие товары, которые покупали эти клиенты. Этот подход называется основанным на памяти (memory-based) или коллаборативной фильтрацией. Такой подход активно использовался до соревнования Netflix Prize (2009), после чего ему на смену пришли подходы, основанные на скрытых переменных.

Подходы к построению рекомендательных систем

- Скрытые переменные

Подходы, основанные на скрытых переменных, работают так: создаются новые признаковые описания (скрытые переменные) для клиентов и для товаров и нужность товара клиенту оценивается на основе этих признаковых описаний.

- Современные подходы к построению рекомендательных систем
  - разложение (разреженной) матрицы оценок товаров на множители (начиная с соревнования Netflix Prize 2006-2009)
  - факторизационные машины (factorization machine)
  - бустинг на деревьях
  - нейронные сети-трансформеры
  - обучение с подкреплением
- Анализ ассоциативных правил

Классический подход на основе ассоциативных правил позволяет обнаружить, какие товары клиенты покупают часто вместе с другими, на основе этого предложить клиенту, что ему еще купить.

Анализ ассоциативных правил

Пусть данные имеют вид набора транзакций, где каждая транзакция — это набор товаров. Например, транзакцией может быть набор продуктов, купленных покупателем при походе в магазин, а весь набор данных будет состоять из всех походов всех покупателей в этот магазин (или все магазины сети). Важно, чтобы в транзакциях был не один или два товара, а больше.

Задача поиска ассоциативных правил формулируется так: нужно найти закономерности вида «если покупатель купил <некоторый набор товаров>, то он купит и <еще один товар>» — эта закономерность и называется ассоциативным правилом. Пример: «если покупатель купил зубную щетку, мыло и мыльницу, то он купит и чехол для зубной щетки».

Ассоциативные правила применяются для оптимизации размещения товаров на полках магазинов, для формирования персональных рекомендаций в онлайн-магазинах или для планирования акций (можно сделать скидку на один товар, зная, что вместе с ним купят другой, более дорогой товар).

Поиск ассоциативных правил обычно выполняется в два этапа. На первом этапе находятся часто покупаемые наборы товаров. На втором этапе происходит непосредственное выделение правил на основе найденных на первом этапе частых комбинаций товаров.

Задача поиска ассоциативных правил

Поиск популярных (часто встречающихся) наборов предметов и соответствующих ассоциативных правил представляет собой, с одной стороны, простой, с другой стороны, довольно часто применимый в реальной жизни метод выявления зависимостей между полями (признаками) в больших базах данных.

Пример 1

База транзакций (покупок):

ID транзакции	Предметы в транзакции
1	молоко, хлеб
2	хлеб, масло
3	пиво
4	молоко, хлеб, масло
5	хлеб, масло

Популярный набор предметов – $\{\text{хлеб}, \text{масло}\}$, ассоциативное правило – покупатель хлеба скорее всего купит и масло: $\{\text{хлеб}\} \Longrightarrow \{\text{масло}\}$.

Терминология и обозначения

Наборы предметов: Пусть $\mathcal{I} = \{x_1, x_2, ..., x_m\}$ – некоторое множество элементов, называемых предметами (items). Подмножество $X \subseteq \mathcal{I}$ называется набором предметов (itemset). Обозначим через $\mathcal{I}^{(k)}$ множество всех наборов, состоящих из k предметов. В наборах предметов будет, как правило, использоваться лексикографический порядок.

Идентификаторы транзакций: Пусть $\mathcal{T} = \{t_1, t_2, ..., t_n\}$ – это другое множество элементов, называемых идентификаторами транзакций.

Транзакцией называется кортеж вида $\langle t, X \rangle$, где $t \in \mathcal{T}$ – это уникальный идентификатор транзакции и X – это набор предметов. Для удобства будем обращаться к транзакции $\langle t, X \rangle$ по ее идентификатору t.

Бинарная база данных $\mathbf{D}$ – это бинарное отношение на множестве идентификаторов транзакций и предметов, т.е. $\mathbf{D} \subseteq \mathcal{T} \times \mathcal{I}$. Будем говорить, что транзакция с идентификатором $t \in \mathcal{T}$ содержит предмет $x \in \mathcal{I}$, если и только если $(t, x) \in \mathbf{D}$. Другими словами, $(t, x) \in \mathbf{D}$ тогда и только тогда, когда для некоторой транзакции $\langle t, X \rangle$ $x \in X$. Будем также говорить, что транзакция с идентификатором $t \in \mathcal{T}$ содержит набор предметов $X = \{x_1, x_2, ..., x_k\}$, если и только если $(t, x_i) \in \mathbf{D}$ для всех $i = \overline{1, k}$.

Пример бинарной базы данных

D	A	B	C	D	E
1	1	1	0	1	1
2	0	1	1	0	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	1
6	0	1	1	1	0

Здесь $\mathcal{I} = \{A, B, C, D, E\}$ и $\mathcal{T} = \{1, 2, 3, 4, 5, 6\}$. В бинарной базе данных ячейка на пересечении ряда t и столбца x равна 1, если и только если $(t,\, x) \in \mathbf{D}$, и равна 0 в противном случае.

Например, из бинарной базы данных видно, что транзакция 1 содержит предмет B, также она содержит набор предметов BE и т.д.

Горизонтальное представление базы данных

Для множества X обозначим через 2^X множество всех подмножеств X. Пусть $\mathbf{i} : 2^{\mathcal{T}} \to 2^{\mathcal{I}}$ – функция, определенная следующим образом:

$$\mathbf{i}(T) = \{x \mid \forall t \in T, \text{ идентификатор транзакции } t \text{ содержит предмет } x\},$$

где $T \subseteq \mathcal{T}$, и множество $\mathbf{i}(T) \subseteq \mathcal{I}$ представляет собой множество предметов, которые являются общими для всех транзакций в множестве идентификаторов транзакций T. В частности, $\mathbf{i}(t)$ – это множество предметов, содержащихся в транзакции с идентификатором $t \in \mathcal{T}$.

Иногда бывает удобно рассматривать бинарную базу данных $\mathbf{D}$ как транзакционную базу данных, состоящую из кортежей вида $\langle t, \mathbf{i}(t)\rangle$ при $t \in \mathcal{T}$. Транзакционная база данных может рассматриваться как горизонтальное представление бинарной базы данных, где мы опускаем предметы, не содержащиеся в транзакции с заданным идентификатором.

Вертикальное представление базы данных

Пусть $\mathbf{t} : 2^{\mathcal{I}} \to 2^{\mathcal{T}}$ – функция, определенная следующим образом:

$$\mathbf{t}(X) = \{t \mid t \in \mathcal{T}, \text{ идентификатор транзакции } t \text{ содержит } X\},$$

где $X \subseteq \mathcal{I}$, и множество $\mathbf{t}(X)$ представляет собой множество идентификаторов транзакций, которые содержат все предметы в наборе предметов X. В частности, $\mathbf{t}(x)$ – это множество идентификаторов транзакций, которые содержат предмет $x \in \mathcal{I}$.

Также иногда удобно рассматривать бинарную базу данных $\mathbf{D}$ как базу данных идентификаторов транзакций, содержащую коллекции кортежей вида $\langle x, \mathbf{t}(x)\rangle$ при $x \in \mathcal{I}$. База данных идентификаторов транзакций является вертикальным представлением бинарной базы данных, где мы опускаем идентификаторы транзакций, которые не содержат заданный предмет.

Пример горизонтальной и вертикальной баз данных

Транзакционная (горизонтальная) база данных для бинарной базы данных предыдущего примера имеет вид:

t	$\mathbf{i}(t)$
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

Например, первая транзакция представляет собой кортеж $\langle 1, \{A, B, D, E\}\rangle$, где мы опускаем предмет C, так как $(1, C) \notin \mathbf{D}$.

Вертикальная база данных для бинарной базы данных предыдущего примера имеет вид:

x	A	B	C	D	E
$\mathbf{t}(x)$	1	1	2	1	1
	3	2	4	3	2
	4	3	5	5	3
	5	4	6	6	4
		5			5
		6			

Например, кортеж, соответствующий объекту A и показанный в первом столбце, равен $\langle A, \{1, 3, 4, 5\}\rangle$. Здесь опущены идентификаторы 2 и 6, так как $(2, A) \notin \mathbf{D}$ и $(6, A) \notin \mathbf{D}$.

Поддержка и популярные наборы предметов

Поддержка (support) набора предметов X в базе данных $\mathbf{D}$, обозначаемая $\sup(X, \mathbf{D})$, – это число транзакций в $\mathbf{D}$, которые содержат X:

$$\sup(X, \mathbf{D}) = |\{t \mid \langle t, \mathbf{i}(t)\rangle \in \mathbf{D},\ X \subseteq \mathbf{i}(t)\}| = |\mathbf{t}(X)|$$

Относительная поддержка (relative support) X – это доля транзакций, содержащих набор X:

$$\mathrm{rsup}(X, \mathbf{D}) = \frac{\sup(X, \mathbf{D})}{|\mathbf{D}|}.$$

Показатель rsup является оценкой совместной вероятности выбора набора предметов X. Показатели поддержки характеризуют, насколько часто набор предметов обнаруживается в базе данных.

Набор предметов X называется популярным (frequent) в $\mathbf{D}$, если поддержка $\sup(X, \mathbf{D}) \geqslant minsup$, где $minsup \in \mathbb{N}$ – это определяемый пользователем минимальный порог поддержки (minimum support threshold). Если $minsup \in (0, 1)$, то подразумевается неравенство $\mathrm{rsup}(X, \mathbf{D}) \geqslant minsup$.

Множество $\mathcal{F}$ будет обозначать множество всех популярных наборов предметов, а $\mathcal{F}^{(k)}$ будет обозначать множество популярных наборов из k предметов.

Пример: популярные наборы предметов

t	$\mathbf{i}(t)$
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

Транзакционная база данных

Под-держ-ка	Наборы предметов
6	B
5	E, BE
4	A, C, D, AB, AE, BC, BD, ABE
3	AD,CE,DE,ABD,ADE,BCE,BDE,$ABDE$

Популярные наборы предметов для $minsup = 3$

Все 19 популярных наборов предметов, перечисленных в таблице выше, составлют множество $\mathcal{F}$. Множества популярных наборов из k предметов равны

$$\mathcal{F}^{(1)} = \{A, B, C, D, E\},$$
$$\mathcal{F}^{(2)} = \{AB, AD, AE, BC, BD, BE, CE, DE\},$$
$$\mathcal{F}^{(3)} = \{ABD, ABE, ADE, BCE, BDE\},$$
$$\mathcal{F}^{(4)} = \{ABDE\}.$$

Ассоциативные правила

Ассоциативное правило (или просто правило) – это выражение $X \longrightarrow Y$, где X и Y – непересекающиеся наборы предметов, т.е. $X, Y \subseteq \mathcal{I}$ и $X \cap Y = \emptyset$. Набор предметов $X \cup Y$ будет обозначаться для краткости как XY.

Поддержкой правила $X \to Y$ является количество транзакций, в которых X и Y сосуществуют как подмножества:

$$s = \sup(X \to Y) = |\mathbf{t}(XY)| = \sup(XY).$$

Относительная поддержка правила $X \to Y$ определяется как доля транзакций, в которых X и Y сосуществуют, и она является оценкой совместной вероятности выбора наборов X и Y:

$$\text{rsup}(X \to Y) = \frac{\sup(XY)}{|\mathbf{D}|} = \mathbb{P}[X \wedge Y].$$

Достоверностью (confidence) правила $X \to Y$ является условная вероятность того, что транзакция содержит Y при условии, что она содержит X:

$$c = \text{conf}(X \to Y) = \mathbb{P}[Y \mid X] = \frac{\mathbb{P}[X \wedge Y]}{\mathbb{P}[X]} = \frac{\sup(XY)}{\sup(X)}.$$

Популярные ассоциативные правила

Правило $X \to Y$ называется популярным (frequent), если набор предметов XY является популярным, т.е. $\sup(XY) \geqslant minsup$.

Правило $X \to Y$ называется сильным (strong), если достоверность этого правила $\text{conf}(X \to Y) \geqslant minconf$, где $minconf$ – это определяемый пользователем минимальный порог достоверности (minimum confidence threshold).

Пример 2

Рассмотрим ассоциативное правило $BC \to E$. Используя значения поддержки наборов предметов, вычисленные ранее, найдем поддержку и достоверность этого правила:

$$s = \sup(BC \to E) = \sup(BCE) = 3,$$

$$c = \text{conf}(BC \to E) = \frac{\sup(BCE)}{\sup(BC)} = \frac{3}{4} = 0.75$$

Майнинг наборов предметов и правил

Из определений поддержки и достоверности правила вытекает, что для того, чтобы получить популярные и сильные ассоциативные правила, нужно сперва найти все популярные наборы предметов с их значениями поддержки. Так, если дана бинарная база данных $\mathbf{D}$ и определяемый пользователем минимальный порог поддержки $minsup$, то задача майнинга популярных наборов предметов состоит в том, чтобы выявить все наборы предметов, которые являются популярными, т.е. имеют поддержку не менее $minsup$. Далее, если дано множество популярных наборов предметов $\mathcal{F}$, то задача майнинга ассоциативных правил для минимального порога поддержки $minsup$ и минимального порога достоверности $minconf$ состоит в том, чтобы найти соответствующие популярные и сильные правила. При этом нас интересуют популярные наборы, содержащие не менее двух предметов.

Пример: набор транзакций Instacart

Instacart (сайт https://www.instacart.com) – розничный продавец овощей в США и Канаде (слоган «Groceries delivered in as little as 1 hour»).

Для исследователей на сайте Instacart выложен набор данных “The Instacart Online Grocery Shopping Dataset 2017”, адрес набора

https://www.instacart.com/datasets/grocery-shopping-2017

В набор входят, в частности, следующие таблицы:

- products (>50 тыс записей):
  - product_id: product identifier
  - product_name: name of the product
  - aisle_id: foreign key
  - department_id: foreign key
- order_products (>3 млн записей):
  - order_id: foreign key
  - product_id: foreign key
  - add_to_cart_order: order in which each product was added to cart
  - reordered: 1 if this product has been ordered by this user in the past, 0 otherwise

Майнинг наборов предметов: алгоритм Brute Force

Алгоритм Brute Force («грубая сила») перебирает все возможные наборы предметов $X \subseteq \mathcal{I}$ и для каждого набора вычисляет его поддержку во входном наборе данных $\mathbf{D}$. Алгоритм состоит из двух основных шагов: 1) порождение наборов-кандидатов; 2) вычисление поддержки.

Порождение кандидатов: на этом шаге порождаются все подмножества $\mathcal{I}$, называемые кандидатами, так как каждый набор предметов может оказаться популярным. Пространство наборов-кандидатов растет экспоненциально с ростом количества предметов, так как имеется $2^{|\mathcal{I}|}$ потенциально популярных наборов предметов.

Вычисление поддержки: на этом шаге вычисляется поддержка каждого набора-кандидата X и определяется, является ли он популярным. Для каждой транзакции $\langle t, \mathbf{i}(t) \rangle$ в базе данных $\mathbf{D}$ определяем, является ли X подмножеством $\mathbf{i}(t)$. Если да, то увеличиваем на единицу значение поддержки набора-кандидата X.

Вычислительная сложность: вычисление поддержки одного набора занимает время $\mathcal{O}(|\mathcal{I}| \cdot |\mathbf{D}|)$ и, так как имеется $\mathcal{O}\left(2^{|\mathcal{I}|}\right)$ возможных кандидатов, вычислительная сложность алгоритма составляет $\mathcal{O}\left(|\mathcal{I}| \cdot |\mathbf{D}| \cdot 2^{|\mathcal{I}|}\right)$.

Алгоритм Brute Force

BruteForce ($\mathbf{D}$, $\mathcal{I}$, $minsup$):

1 $\mathcal{F} \leftarrow \emptyset$ // множество популярных наборов предметов
2 foreach $X \subseteq \mathcal{I}$ do
3 $sup(X) \leftarrow$ ComputeSupport $(X, \mathbf{D})$
4 if $sup(X) \geq minsup$ then
5 $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X, sup(X))\}$
6 return $\mathcal{F}$

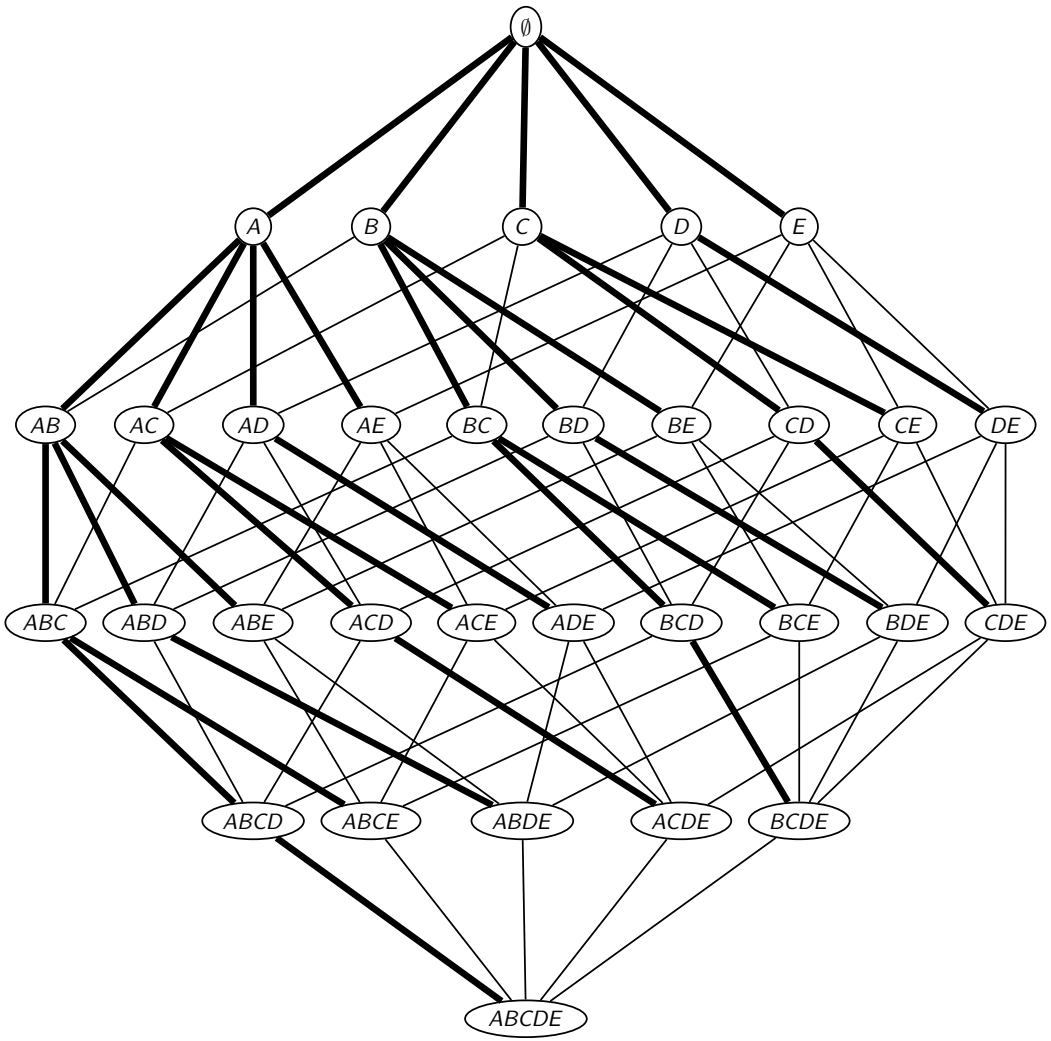
ComputeSupport $(X, \mathbf{D})$:

1 $sup(X) \leftarrow 0$
2 foreach $\langle t, \mathbf{i}(t) \rangle \in \mathbf{D}$ do
3 if $X \subseteq \mathbf{i}(t)$ then
4 $sup(X) \leftarrow sup(X) + 1$
5 return $sup(X)$

Пространство и дерево поиска наборов предметов

Пространство поиска наборов предметов представляет собой граф, в котором любые два набора X и Y соединены (серой линией), если и только если X является непосредственным подмножеством Y, то есть $X \subseteq Y$ и $|X| = |Y| - 1$. Популярные наборы предметов могут быть выявлены используя поиск в ширину (BFS) или поиск в глубину (DFS) на дереве поиска, где наборы предметов X и Y соединены (черной линией), если и только если X является непосредственным подмножеством и префиксом Y.

Это позволяет перечислить наборы предметов, начиная с пустого множества и добавляя на каждом шаге по одному предмету.



Уровневый подход: алгоритм Apriori

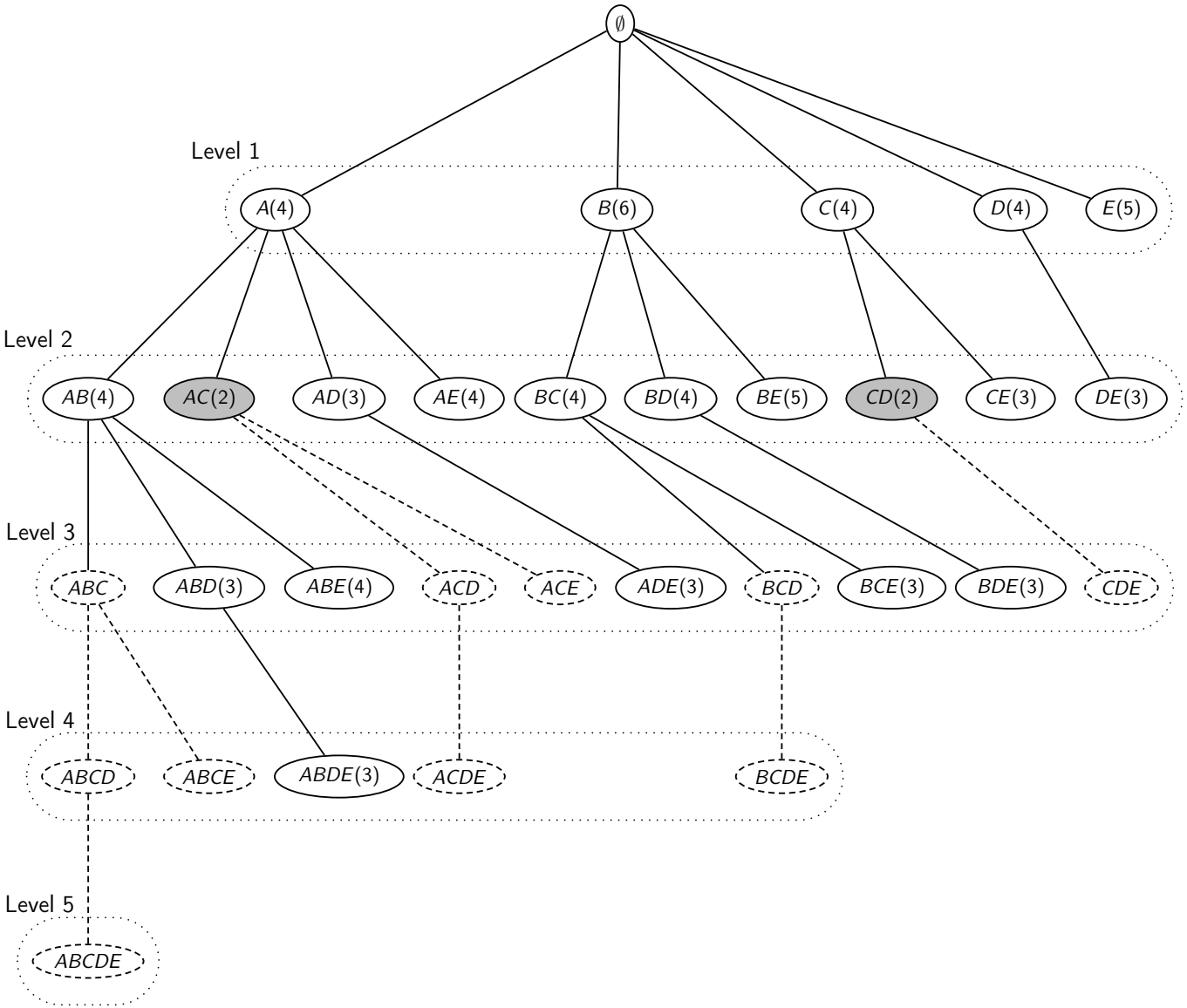
Если $X \subseteq Y$ для двух наборов предметов X, Y, то $\sup(X) \geqslant \sup(Y)$, поэтому:

1. если набор Y является популярным, то любое его подмножество $X \subseteq Y$ также является популярным
2. если набор X не является популярным, то любое надмножество $Y \supseteq X$ не является популярным

Алгоритм Apriori использует эти два свойства, чтобы существенно улучшить алгоритм Brute Force. Алгоритм использует поиск в ширину (BFS) на дереве поиска и в алгоритме Apriori удаляются все надмножества непопулярных кандидатов, поскольку никакие надмножества непопулярного набора предметов не могут быть популярными (обрезка дерева).

В дополнение к улучшению процесса создания наборов-кандидатов при помощи обрезки дерева алгоритм Apriori существенно улучшает подсчет поддержки. Вместо подсчета поддержки наборов предметов по одному алгоритм просматривает дерево поиска в соответствии с методом поиска в ширину и вычисляет поддержку всех имеющихся наборов-кандидатов размера k, которые находятся на уровне k в дереве поиска.

Обрезка дерева поиска



Детали алгоритма Apriori

При использовании Apriori используется упорядочение в лексикографическом (алфавитном) порядке.

Пусть $\mathcal{C}^{(k)}$ обозначает дерево поиска, включающее все наборы-кандидаты из k предметов и их предков.

Алгоритм Apriori начинает с включения отдельных предметов в первоначально пустое дерево поиска, чтобы получить $\mathcal{C}^{(1)}$.

Поддержка наборов-кандидатов из $\mathcal{C}^{(k)}$ вычисляется при помощи функции **ComputeSupport**, которая перебирает k-подмножества для каждой транзакции в $\mathbf{D}$ и для каждого такого подмножества увеличивает на единицу поддержку соответствующего набора-кандидата в $\mathcal{C}^{(k)}$ (если он существует). Далее из $\mathcal{C}^{(k)}$ удаляются все наборы-кандидаты, не являющиеся популярными (с поддержкой меньше $minsup$).

Оставшиеся листья дерева поиска составляют множество популярных наборов из k предметов $\mathcal{F}^{(k)}$, которые используются, чтобы получить наборы-кандидаты из $(k+1)$ предметов на следующем уровне.

Детали алгоритма Apriori (продолжение)

Функция **ExtendPrefixTree** для создания новых наборов-кандидатов использует расширение имеющихся префиксов узлов. Для двух популярных наборов из k предметов X_a и X_b с общим префиксом длиной $k-1$, то есть для двух узлов-листьев, имеющих общего родителя, создается набор-кандидат $X_{ab} = X_a \bigcup X_b$ длиной $(k+1)$. Этот набор-кандидат сохраняется в дереве поиска, только если у него отсутствуют подмножества из k предметов, являющиеся непопулярными. Наконец, если префикс набора из k предметов X_a не может быть расширен, то он удаляется из дерева поиска вместе со всеми его узлами-предками, которые не имеют других подузлов. Таким образом, в $\mathcal{C}^{(k+1)}$ все листья находятся на уровне $k+1$.

Если на текущем шаге были добавлены новые наборы-кандидаты, то процесс повторяется для следующего уровня k. Процесс добавления новых префиксов продолжается до тех пор, пока возможно добавление префиксов новых наборов-кандидатов.

Итак, популярным наборам из k предметов, входящим в $\mathcal{F}^{(k)}$, соответствуют листья на дереве поиска $\mathcal{C}^{(k)}$.

Алгоритм Apriori

Apriori $(\mathbf{D}, \mathcal{I}, minsup)$:

$\mathcal{F} \leftarrow \emptyset$

$\mathcal{C}^{(1)} \leftarrow \{\emptyset\}$ // Начальное дерево поиска

foreach $i \in \mathcal{I}$ do

  добавить i в $\mathcal{C}^{(1)}$ с поддержкой $sup(i) \leftarrow 0$

$k \leftarrow 1$ // k обозначает уровень

while $\mathcal{C}^{(k)} \neq \emptyset$ do

  ComputeSupport $(\mathcal{C}^{(k)}, \mathbf{D})$

  foreach префикс $X \in \mathcal{C}^{(k)}$ do

    if $sup(X) \geq minsup$ then $\mathcal{F} \leftarrow \mathcal{F} \cup \{(X, sup(X))\}$

    else удалить X из $\mathcal{C}^{(k)}$

  $\mathcal{C}^{(k+1)} \leftarrow$ ExtendPrefixTree $(\mathcal{C}^{(k)})$

  $k \leftarrow k + 1$

return $\mathcal{F}$

Алгоритм Apriori (продолжение)

ComputeSupport $(\mathcal{C}^{(k)}, \mathbf{D})$:

```
1 foreach ⟨t, i(t)⟩ ∈ D do
2     foreach набор из k предметов X ⊆ i(t) do
3         if X ∈ C^(k) then sup(X) ← sup(X) + 1
```

ExtendPrefixTree $(\mathcal{C}^{(k)})$:

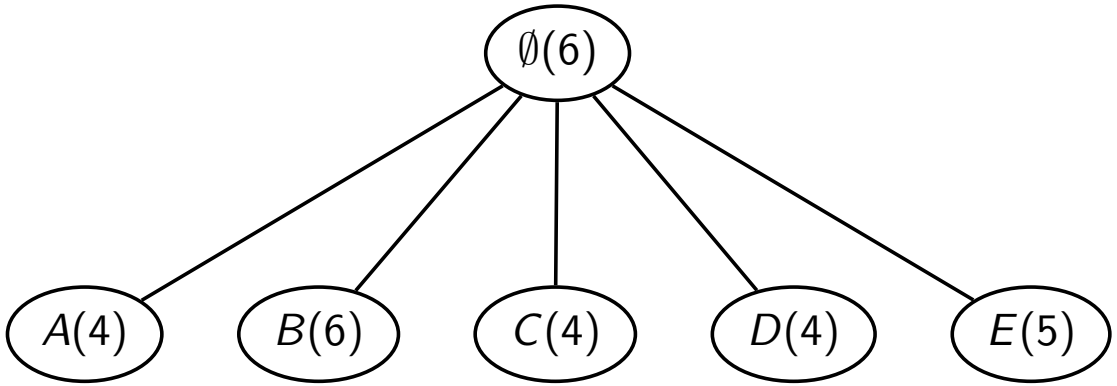
```
1 foreach префикс X_a ∈ C^(k) do
2     foreach префикс X_b из семейства узла X_a, где b > a do
3         X_ab ← X_a ∪ X_b
          // удалить кандидата с непопулярными подмножествами
4         if X_j ∈ C^(k) для всех X_j ⊂ X_ab, где |X_j| = |X_ab| − 1 then
5             добавить X_ab как ветвь X_a с поддержкой sup(X_ab) ← 0
6     if невозможно расширение префикса X_a then
7         удалить X_a и всех предков узла X_a без расширений из C^(k)
8 return C^(k)
```

Пример: построение дерева поиска ($minsup = 3$)

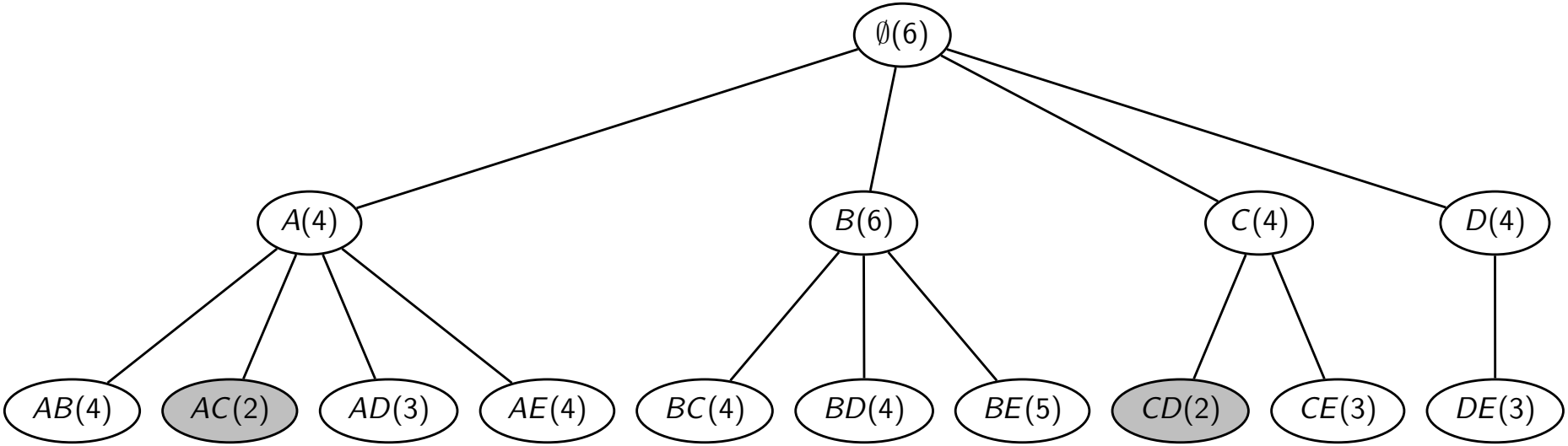
D

t	$\mathbf{i}(t)$
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

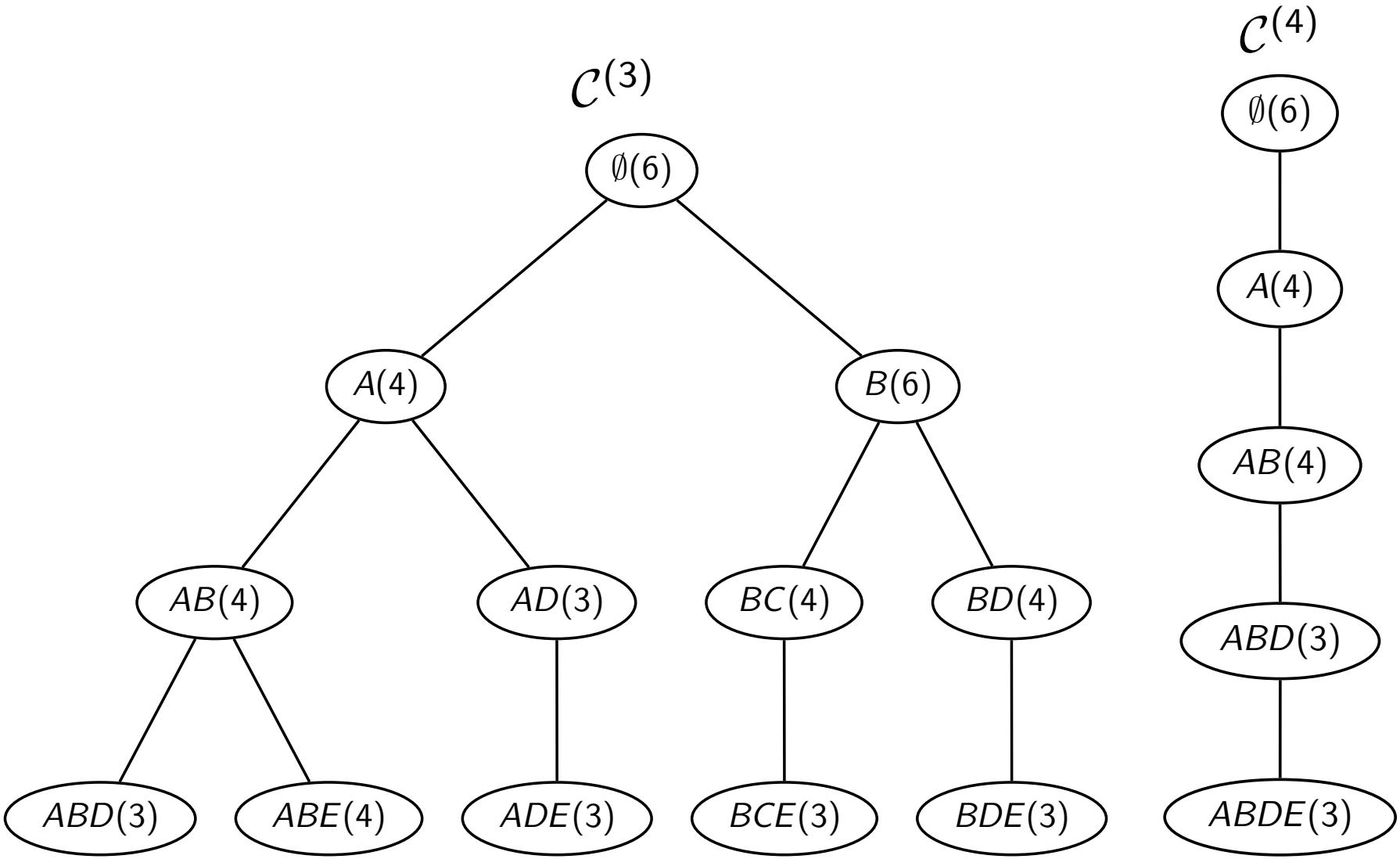
$\mathcal{C}^{(1)}$



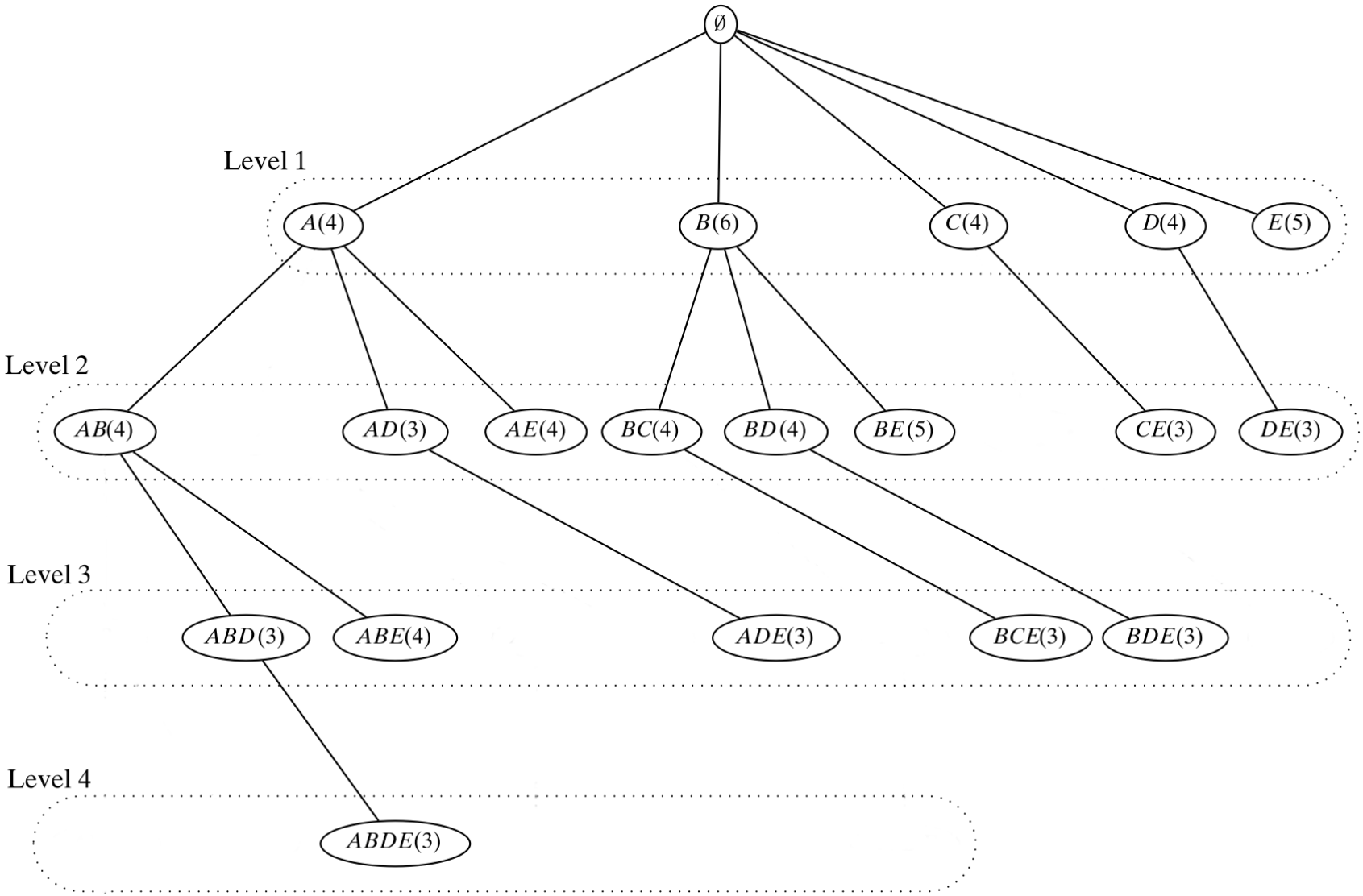
$\mathcal{C}^{(2)}$



Пример: построение дерева поиска (продолжение)



Пример: построение дерева поиска (результат)



Пересечения наборов транзакций в алгоритме Eclat

Этап вычисления поддержки может быть существенно ускорен, если проиндексировать базу данных так, чтобы она допускала быстрое вычисление поддержки.

Алгоритм Eclat использует непосредственно наборы транзакций для вычисления поддержки. Основная идея состоит в том, что поддержка набора-кандидата может быть вычислена при помощи пересечения наборов транзакций подмножеств предметов, выбранных надлежащим образом. Если даны $\mathbf{t}(X)$ и $\mathbf{t}(Y)$ для любых двух популярных наборов предметов X и Y, то

$$\mathbf{t}(X\,Y) = \mathbf{t}(X) \cap \mathbf{t}(Y)\,.$$

Поддержка набора-кандидата $X\,Y$ равна размеру множества $\mathbf{t}(X\,Y)$, т.е. $\sup(X\,Y) = |\mathbf{t}(X\,Y)|$.

Алгоритм Eclat выполняет пересечение множеств транзакций, только если популярные наборы предметов имеют общий префикс, и обходит дерево поиска, как в алгоритме поиска в глубину (DFS), обрабатывая группы наборов предметов, имеющих общий префикс.

Алгоритм Eclat

$// \ \mathcal{F} \leftarrow \emptyset, P \leftarrow \left\{ \langle i, \mathbf{t}(i) \rangle \mid i \in \mathcal{I}, |\mathbf{t}(i)| \geqslant minsup \right\}$

Eclat (P, $minsup$, $\mathcal{F}$):

1 foreach $\langle X_a, \mathbf{t}(X_a) \rangle \in P$ do

2 $\quad \mathcal{F} \leftarrow \mathcal{F} \cup \left\{ (X_a, sup(X_a)) \right\}$

3 $\quad P_a \leftarrow \emptyset$

4 $\quad$ foreach $\langle X_b, \mathbf{t}(X_b) \rangle \in P, X_b > X_a$ do

5 $\quad\quad X_{ab} \leftarrow X_a \cup X_b$

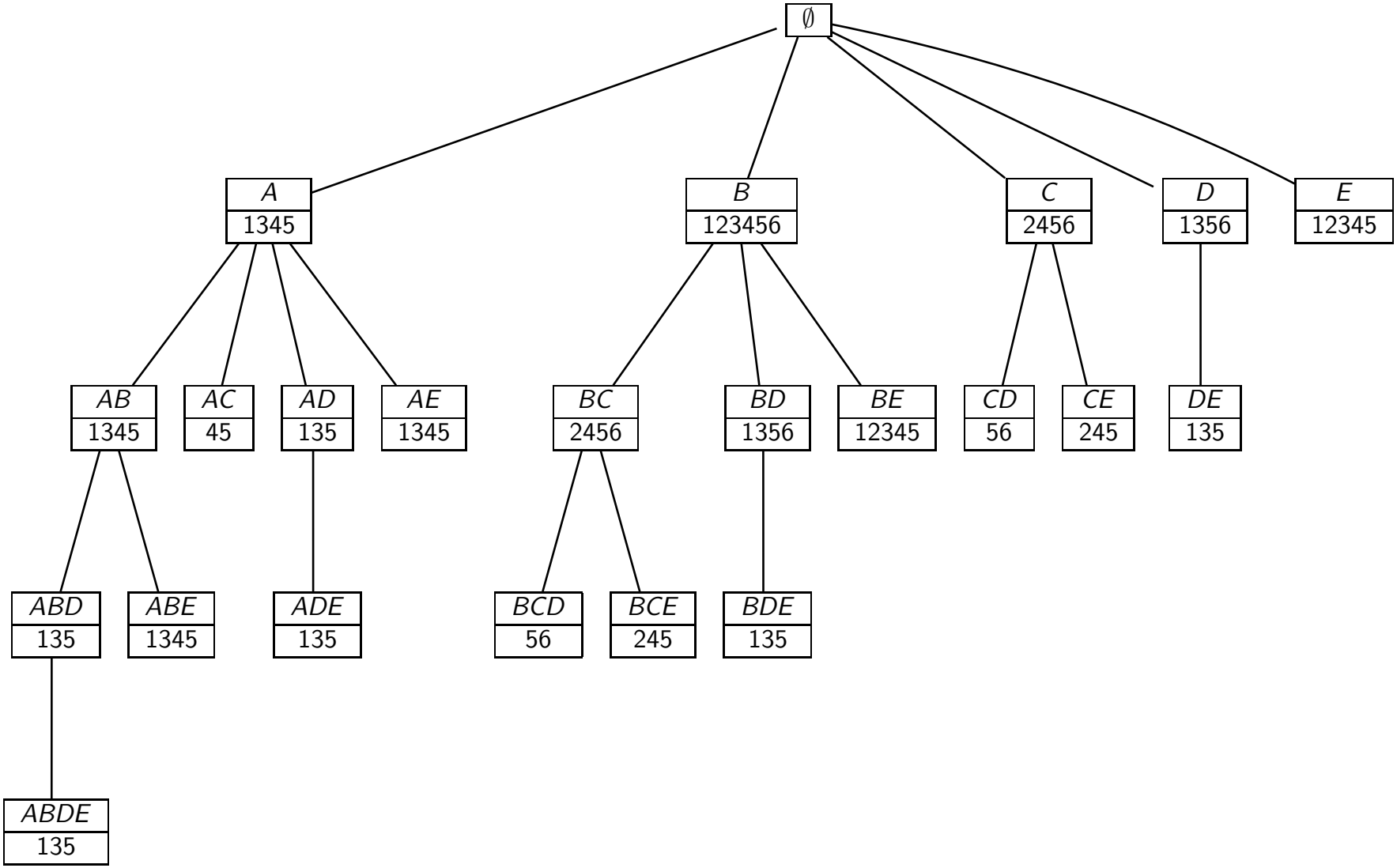
6 $\quad\quad \mathbf{t}(X_{ab}) \leftarrow \mathbf{t}(X_a) \cap \mathbf{t}(X_b)$

7 $\quad\quad$ if $sup(X_{ab}) \geqslant minsup$ then

8 $\quad\quad\quad P_a \leftarrow P_a \cup \left\{ \langle X_{ab}, \mathbf{t}(X_{ab}) \rangle \right\}$

9 $\quad$ if $P_a \neq \emptyset$ then Eclat (P_a, $minsup$, $\mathcal{F}$)

Алгоритм Eclat: пересечения наборов транзакций



Пример: популярные наборы в алгоритме Eclat

$\mathcal{F} = \emptyset,\ P = \{\langle A, 1345\rangle, \langle B, 123456\rangle, \langle C, 2456\rangle, \langle D, 1356\rangle, \langle E, 12345\rangle\}$

$X_a = A,\ \mathcal{F} = \{(A, 4)\},\ P_a = \emptyset$

$X_b = B,\ X_{ab} = AB,\ \mathbf{t}(AB) = \mathbf{t}(A) \cap \mathbf{t}(B) = 1345$

$|\mathbf{t}(AB)| = 4 > 3 \Rightarrow P_a = \{(AB, 1345)\}$

$X_b = C,\ X_{ab} = AC,\ \mathbf{t}(AC) = \mathbf{t}(A) \cap \mathbf{t}(C) = 45$

$|\mathbf{t}(AC)| = 2 < 3$

$X_b = D,\ X_{ab} = AD,\ \mathbf{t}(AD) = \mathbf{t}(A) \cap \mathbf{t}(D) = 135$

$|\mathbf{t}(AD)| = 3 \geqslant 3 \Rightarrow P_a = \{(AB, 1345), (AD, 135)\}$

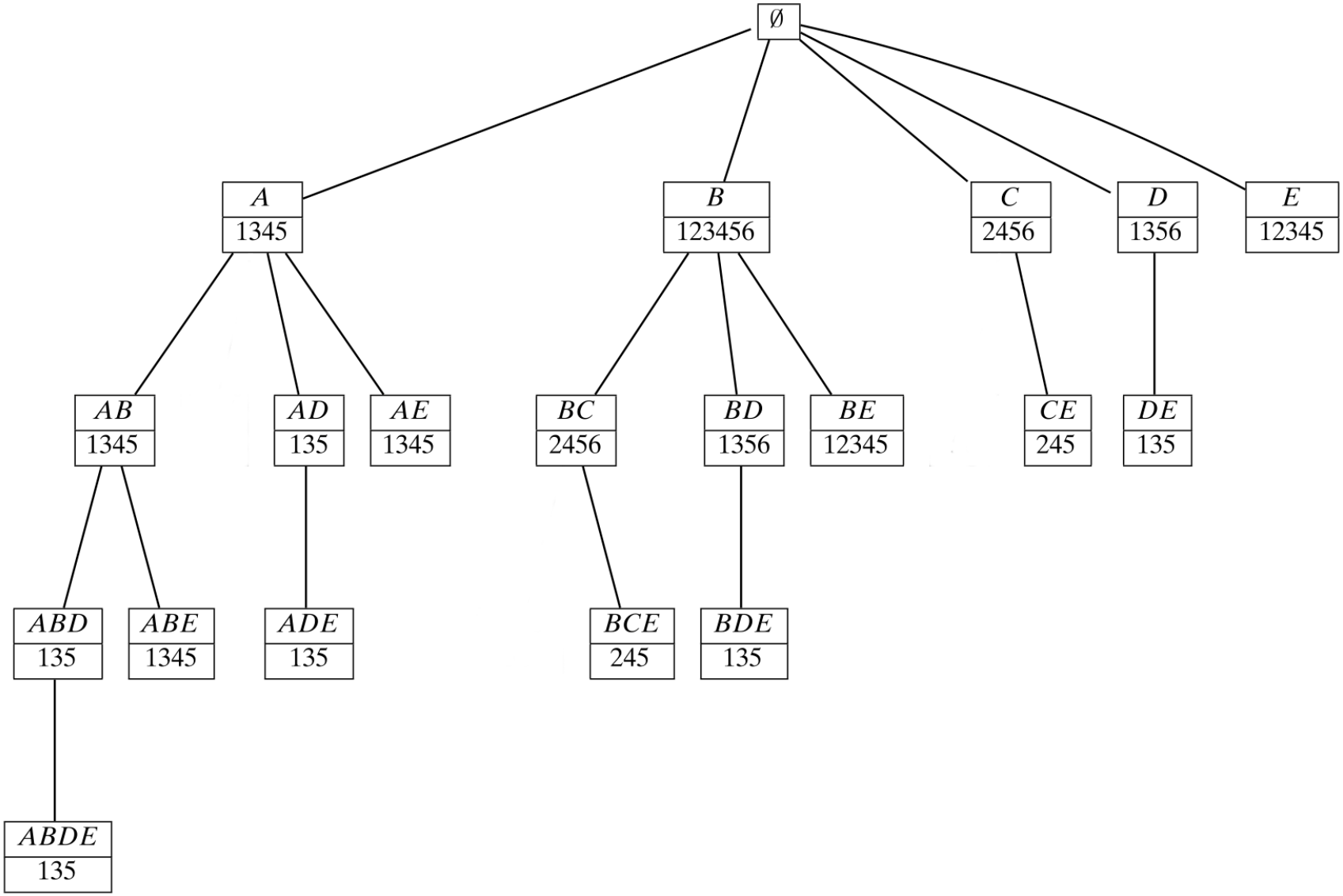
$X_b = E,\ X_{ab} = AE,\ \mathbf{t}(AE) = \mathbf{t}(A) \cap \mathbf{t}(E) = 1345$

$|\mathbf{t}(AE)| = 4 > 3 \Rightarrow P_a = \{(AB, 1345), (AD, 135), (AE, 1345)\}$

$P_a \neq \emptyset \Rightarrow Eclat(\{(AB, 1345), (AD, 135), (AE, 1345)\}, 3, \{(A, 4)\})$

...

Алгоритм Eclat: итоговое дерево поиска



Алгоритм dEclat: разности наборов транзакций

Алгоритм Eclat может быть существенно улучшен, если мы сможем уменьшить размер промежуточных наборов транзакций. Этого можно достичь, если сохранять информацию о различиях в наборах транзакций вместо полных наборов транзакций (алгоритм dEclat).

Для набора предметов $X_k = \{x_1, ..., x_{k-1}, x_k\}$ определим множество (diffset) $\mathbf{d}(X_k)$ как множество идентификаторов транзакций, которые содержат префикс $X_{k-1} = \{x_1, ..., x_{k-1}\}$, но не содержат последний предмет x_k:

$$\mathbf{d}(X_k) = \mathbf{t}(X_{k-1}) \setminus \mathbf{t}(X_k)$$

Рассмотрим наборы $X_a = \{x_1, ..., x_{k-1}, x_a\}$, $X_b = \{x_1, ..., x_{k-1}, x_b\}$ и набор

$$X_{ab} = X_a \cup X_b = \{x_1, ..., x_{k-1}, x_a, x_b\}$$

Тогда

$$\mathbf{d}(X_{ab}) = \mathbf{t}(X_a) \setminus \mathbf{t}(X_{ab}) = \mathbf{t}(X_a) \setminus \mathbf{t}(X_b)$$

Алгоритм dEclat: разности наборов транзакций 2

Но

$$\mathbf{t}(X_a) \setminus \mathbf{t}(X_b) = \mathbf{t}(X_a) \cap \overline{\mathbf{t}(X_b)},$$

поэтому можем получить выражение для $\mathbf{d}(X_{ab})$ через $\mathbf{d}(X_a)$ и $\mathbf{d}(X_b)$ следующим образом:

$$\mathbf{d}(X_{ab}) = \mathbf{d}(X_b) \setminus \mathbf{d}(X_a),$$

что означает, что все операции пересечения в Eclat могут быть выполнены (в dEclat) при помощи операции разности множеств.

Поддержка набора-кандидата может быть вычислена путем вычитания размера множества $\mathbf{d}(X_{ab})$ из поддержки префикса

$$\sup(X_{ab}) = \sup(X_a) - |\mathbf{d}(X_{ab})|$$

Для отдельных предметов $i \in \mathcal{I}$ справедливы равенства

$$\mathbf{d}(i) = \mathbf{t}(\emptyset) \setminus \mathbf{t}(i) = \mathcal{T} \setminus \mathbf{t}(i)$$

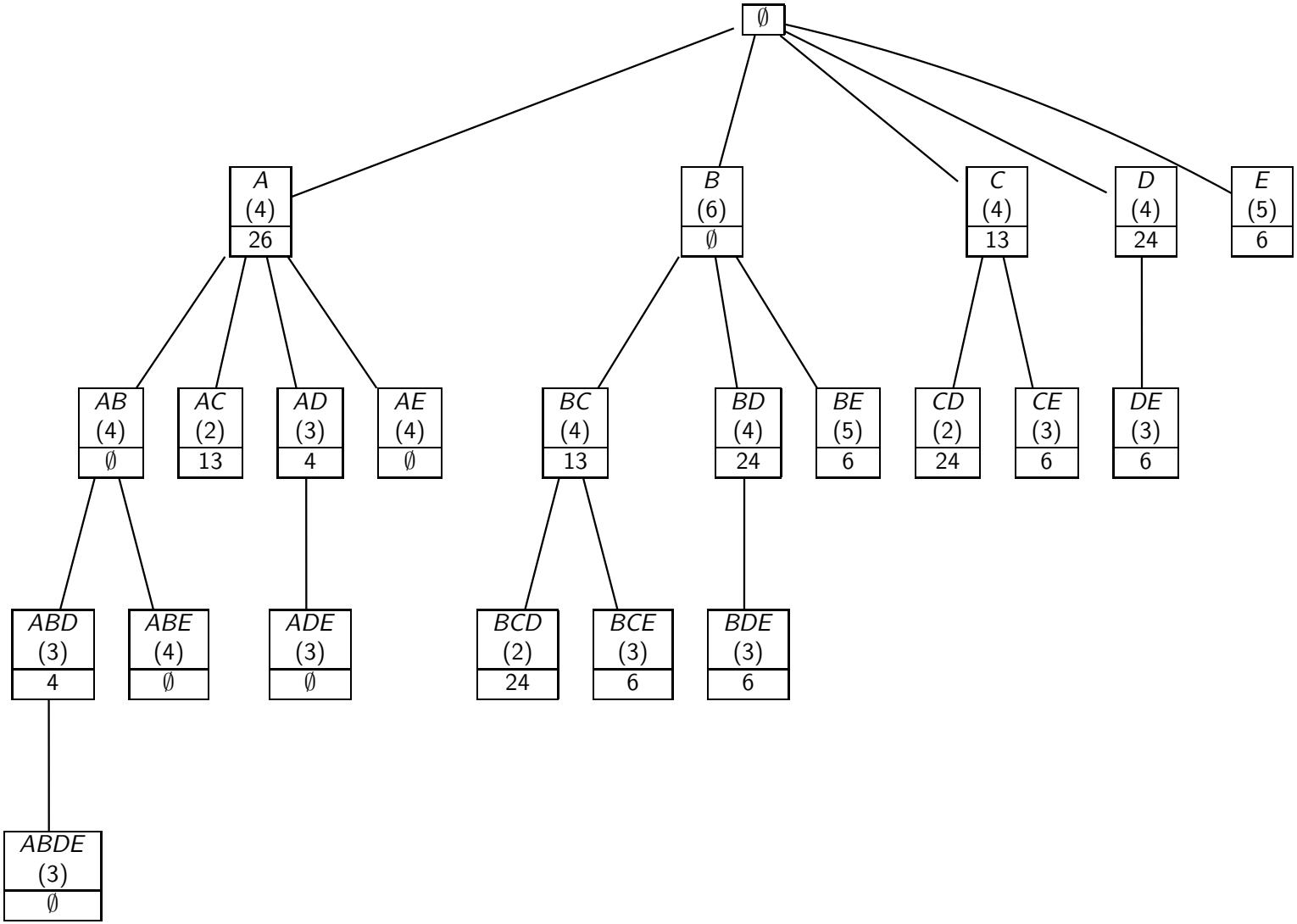
Алгоритм dEclat

// $P \leftarrow \left\{\langle i, \mathbf{d}(i), sup(i)\rangle \mid i \in \mathcal{I}, \mathbf{d}(i) = \mathcal{T} \setminus \mathbf{t}(i), sup(i) \geqslant minsup\right\}$,
$\mathcal{F} \leftarrow \emptyset$

dEclat (P, $minsup$, $\mathcal{F}$):

1 foreach $\langle X_a, \mathbf{d}(X_a), sup(X_a)\rangle \in P$ do
2 $\quad \mathcal{F} \leftarrow \mathcal{F} \cup \left\{(X_a, sup(X_a))\right\}$
3 $\quad P_a \leftarrow \emptyset$
4 $\quad$ foreach $\langle X_b, \mathbf{d}(X_b), sup(X_b)\rangle \in P$, $X_b > X_a$ do
5 $\quad\quad X_{ab} \leftarrow X_a \cup X_b$
6 $\quad\quad \mathbf{d}(X_{ab}) \leftarrow \mathbf{d}(X_b) \setminus \mathbf{d}(X_a)$
7 $\quad\quad sup(X_{ab}) \leftarrow sup(X_a) - |\mathbf{d}(X_{ab})|$
8 $\quad\quad$ if $sup(X_{ab}) \geqslant minsup$ then
9 $\quad\quad\quad P_a \leftarrow P_a \cup \left\{\langle X_{ab}, \mathbf{d}(X_{ab}), sup(X_{ab})\rangle\right\}$
10 $\quad$ if $P_a \neq \emptyset$ then dEclat (P_a, $minsup$, $\mathcal{F}$)

Алгоритм dEclat: разности наборов транзакций



Пример: популярные наборы в алгоритме dEclat

$\mathcal{F} = \emptyset,\ P = \{\langle A, 26, 4\rangle, \langle B, \emptyset, 6\rangle, \langle C, 13, 4\rangle, \langle D, 24, 4\rangle, \langle E, 6, 5\rangle\}$

$X_a = A,\ \mathcal{F} = \{(A, 4)\},\ P_a = \emptyset$

$X_b = B,\ X_{ab} = AB,\ \mathbf{d}(AB) = \mathbf{d}(B) \setminus \mathbf{d}(A) = \emptyset$

$\sup(AB) = \sup(A) - |\mathbf{d}(AB)| = 4 > 3 \Rightarrow P_a = \{(AB, \emptyset, 4)\}$

$X_b = C,\ X_{ab} = AC,\ \mathbf{d}(AC) = \mathbf{d}(C) \setminus \mathbf{d}(A) = 13$

$\sup(AC) = \sup(A) - |\mathbf{d}(AC)| = 4 - 2 = 2 < 3$

$X_b = D,\ X_{ab} = AD,\ \mathbf{d}(AD) = \mathbf{d}(D) \setminus \mathbf{d}(A) = 4$

$\sup(AD) = \sup(A) - |\mathbf{d}(AD)| = 4 - 1 = 3 \geqslant 3 \Rightarrow$

$P_a = \{(AB, \emptyset, 4), (AD, 4, 3)\}$

$X_b = E,\ X_{ab} = AE,\ \mathbf{d}(AE) = \mathbf{d}(E) \setminus \mathbf{d}(A) = \emptyset$

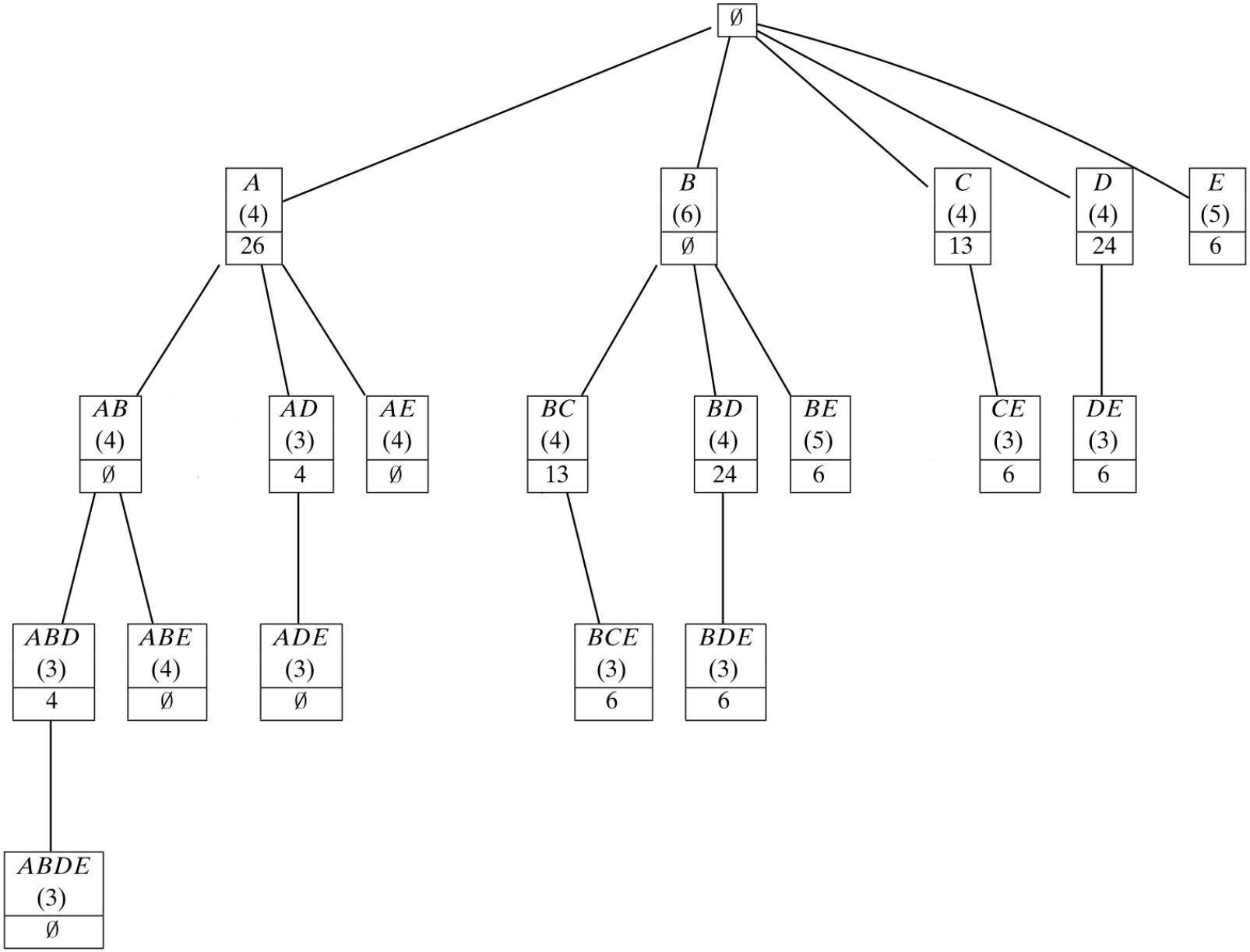
$\sup(AE) = \sup(A) - |\mathbf{d}(AE)| = 4 \geqslant 3 \Rightarrow$

$P_a = \{(AB, \emptyset, 4), (AD, 4, 3), (AE, \emptyset, 4)\}$

$P_a \neq \emptyset \Rightarrow dEclat(\{(AB, \emptyset, 4), (AD, 4, 3), (AE, \emptyset, 4)\}, 3, \{(A, 4)\})$

...

Алгоритм dEclat: итоговое дерево поиска



Алгоритм FPGrowth

Узким местом в алгоритмах Apriori, Eclat, dEclat является процесс генерации кандидатов в популярные наборы предметов. Например, если база транзакций содержит 100 предметов, то, вообще говоря, потребуется сгенерировать до $2^{100} \sim 10^{30}$ кандидатов. Таким образом, возникают значительные вычислительные и временные затраты.

Кроме этого, алгоритмы Apriori, Eclat, dEclat требуют многократного сканирования базы транзакций.

Алгоритм FPGrowth (или FPG) позволяет избежать затратной процедуры генерации кандидатов и уменьшить необходимое число проходов базы транзакций до двух.

При первом проходе алгоритм индексирует базу транзакций для быстрого вычисления поддержки при помощи расширенного префиксного дерева, называемого деревом популярных наборов предметов (frequent pattern tree) или FP-деревом.

Подход на основе FP-дерева в FPGrowth

Каждый узел в FP-дереве маркируется единственным предметом и каждый дочерний узел представляет другой предмет. Каждый узел также хранит информацию о поддержке для предметного набора, состоящего из предметов на пути из корневого узла к этому узлу.

Изначально FP-дерево состоит из корневого узла с пустым предметом $\emptyset$. Далее для каждого кортежа $\langle t, X \rangle \in \mathbf{D}$, где $X = \mathbf{i}(t)$, мы вставляем набор X в FP-дерево, увеличивая на единицу счетчики всех узлов вдоль пути, представляющего X.

Если X имеет тот же префикс, что и некоторая ранее вставленная транзакция, то X следует существующему пути с общим префиксом. Для остающихся предметов в X под общим префиксом создаются новые узлы со счетчиками, установленными в единицу. FP-дерево полностью построено, когда в него вставлены все транзакции из $\mathbf{D}$.

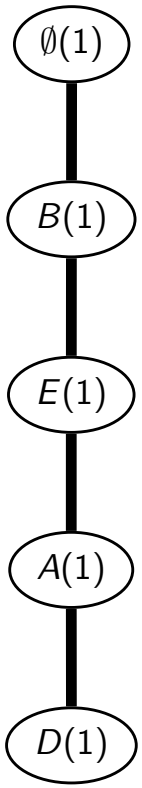
FP-дерево представляет собой сжатое префиксное представление $\mathbf{D}$. Для наибольшего сжатия предметы сортируются в порядке убывания поддержки (а не лексикографическом).

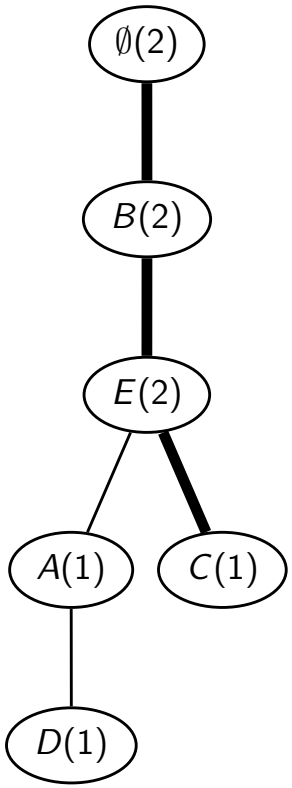
FPGrowth: построение FP-дерева (1)

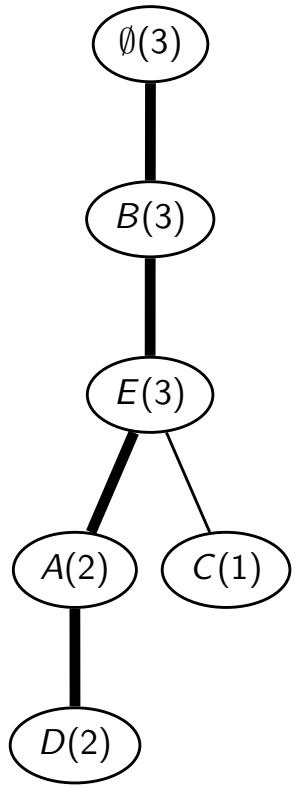
Допустим, что для рассматриваемой базы из 6 транзакций требуется построить все популярные наборы предметов с минимальной поддержкой, равной 3. Предметы в транзакциях упорядочиваются по убыванию значений их поддержек («BEACD»). Построим FP-дерево R:

D

t	$\mathbf{i}(t)$
1	$BEAD$
2	BEC
3	$BEAD$
4	$BEAC$
5	$BEACD$
6	BCD



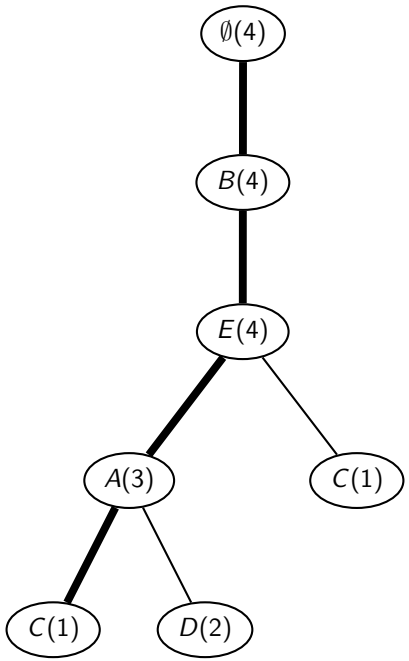


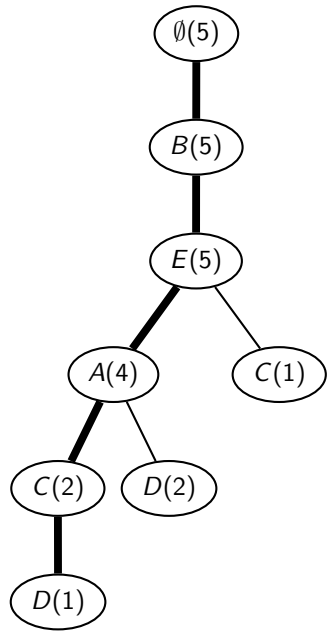


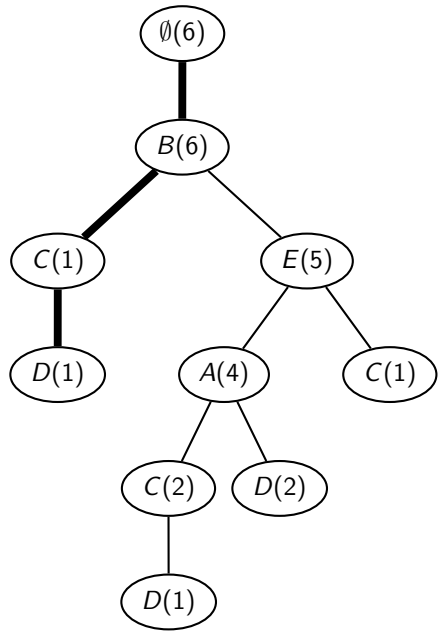
FPGrowth: построение FP-дерева (2)

D

t	$\mathbf{i}(t)$
1	$BEAD$
2	BEC
3	$BEAD$
4	$BEAC$
5	$BEACD$
6	BCD







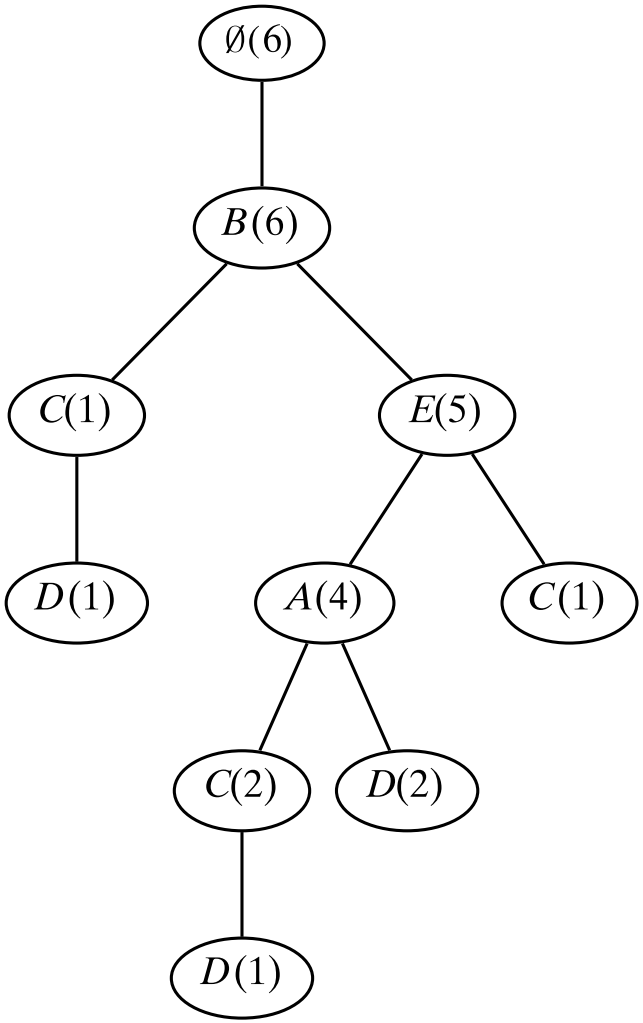
Таким образом, после первого прохода базы транзакций построено FP-дерево, которое в компактном виде представляет информацию о популярных наборах предметов и позволяет производить их эффективное извлечение, что и делается на втором сканировании.

Для каждого предмета в FP-дереве, представленного своим узлами, можно указать пути, т.е. последовательности узлов, которую надо пройти от корневого узла до узлов, связанных с этим предметом.

FPGrowth: извлечение популярных наборов

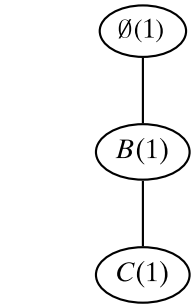
1. Выбираем предмет (например, «D») и находим в дереве R все пути, которые ведут к узлам этого предмета {BCD, BEACD,BEAD} Затем для каждого пути подсчитываем, сколько раз предмет встречается в нем, и записываем это в виде (BCD, 1), (BEACD, 1), (BEAD,2).
2. Удалим сам предмет (суффикс набора) из ведущих к нему путей, т.е. {BCD, BEACD,BEAD}, и оставим только префиксы: {BC, BEAC,BEA}.
3. Подсчитаем, сколько раз каждый предмет появляется в префиксах путей, полученных на предыдущем шаге, и упорядочим в порядке убывания этих значений, получив новый набор транзакций.
4. На его основе построим новое FP-дерево R_D, которое назовем проекцией FP-дерева R на предмет D.
5. В проекции FP-дерева R_D найдем все предметы (узлы), для которых поддержка (количество появлений в дереве) равна 3 и больше, что соответствует заданному уровню минимальной поддержки. Если узел встречается два или более раза, то его индексы, т.е. частоты появлений предмета в проекции FP-дерева, суммируются.
6. Записываем пути от корня дерева к каждому узлу, для которого поддержка больше или равна 3, добавляем предмет (суффикс), удаленный на шаге 2, и подсчитываем поддержку, полученную в результате.

Проекция FP-дерева для популярных наборов (1)

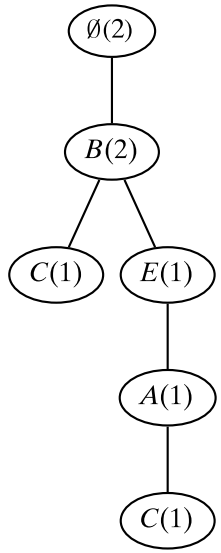


Рассмотрим применение FPGrowth на построенном ранее FP-дереве R для $minsup = 3$. Начальный префикс $P = \emptyset$ и набор набор популярных предметов i в R содержит $B\,(6)$, $E\,(5)$, $A\,(4)$, $C\,(4)$, $D\,(4)$. FPGrowth создает проекцию FP-дерева для каждого предмета, но в возрастающем порядке поддержки.

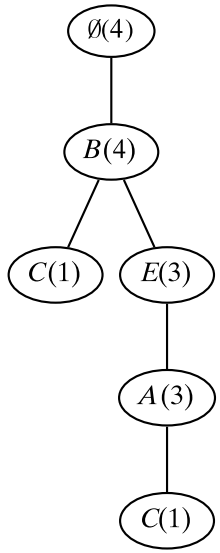
Для D имеются три пути от корня к узлам с D, поэтому новое FP-дерево R_D получается так:



(a) Add $BC, cnt = 1$



(b) Add $BEAC, cnt = 1$

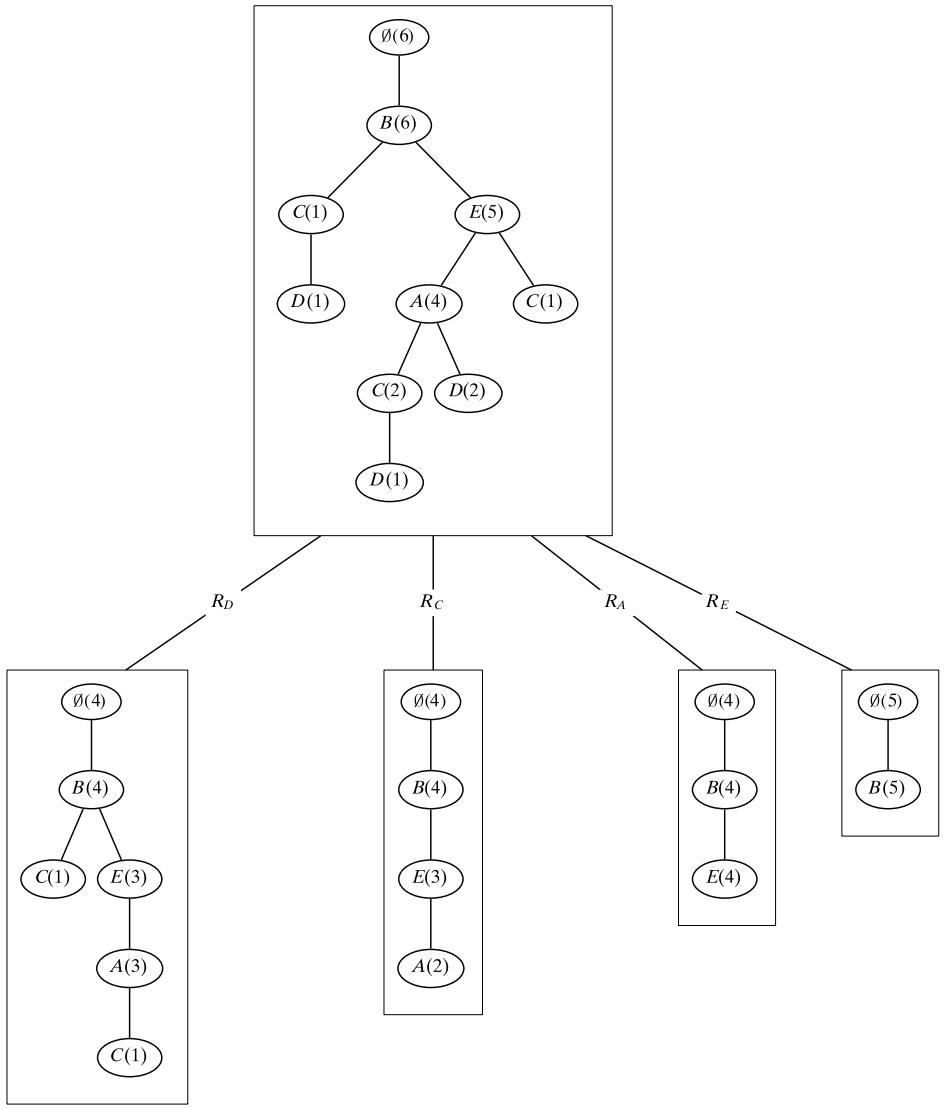


(c) Add $BEA, cnt = 2$

Проекция FP-дерева для популярных наборов (2)

Когда обрабатывается R_D, префикс $P = D$ и после удаления непопулярного предмета C (с поддержкой 2) получаем итоговое FP-дерево с единственным путем $B\,(4) - E\,(3) - A\,(3)$. Далее выбираем все подпути этого пути и добавляем префикс D, получая популярные наборы $DB\,(4)$, $DE\,(3)$, $DA\,(3)$, $DBE\,(3)$, $DBA\,(3)$, $DEA\,(3)$, $DBEA\,(3)$.

Аналогично, проекции FP-дерева для C, A, E имеют единственные пути, позволяя получить популярные наборы $\{CB\,(4)\,, CE\,(3)\,, CBE\,(3)\}$, $\{AE\,(4)\,, AB\,(4)\,, AEB\,(4)\}$ и $\{EB\,(5)\}$ соответственно.



Алгоритм FPGrowth

```
// Initial Call: R ← FP-tree(D), P ← ∅, F ← ∅
FPGrowth (R, P, F, minsup):
Remove infrequent items from R
if IsPath(R) then // insert subsets of R into F
    foreach Y ⊆ R do
        X ← P ∪ Y
        sup(X) ← min_{x∈Y}{cnt(x)}
        F ← F ∪ {(X, sup(X))}
else    // process projected FP-trees for each frequent item i
    foreach i ∈ R in increasing order of sup(i) do
        X ← P ∪ {i}
        sup(X) ← sup(i) // sum of cnt(i) for all nodes labeled i
        F ← F ∪ {(X, sup(X))}
        R_X ← ∅ // projected FP-tree for X
        foreach path ∈ PathFromRoot(i) do
            cnt(i) ← count of i in path
            Insert path, excluding i, into FP-tree R_X with count cnt(i)
        if R_X ≠ ∅ then FPGrowth (R_X, X, F, minsup)
```

Детали алгоритма FPGrowth (1)

Если построено FP-дерево R, то рекурсивным образом могут быть построены проекции этого FP-дерева на каждый популярный предмет i в R в порядке возрастания поддержки.

Для проекции R на предмет i мы находим все появления i в дереве и для каждого появления определяем соответствующий путь из корня в i (строка 13). Число предметов i в заданном пути записывается в переменную $cnt\,(i)$ (строка 14) и путь вставляется в новое дерево-проекцию R_X, где X – это набор предметов, полученный добавлением предмета i к префиксу пути P. При вставке пути счетчик каждого узла в R_X вдоль заданного пути увеличивается на счетчик пути $cnt\,(i)$. Далее предмет i убирается из пути и рассматривается как часть префикса. Полученное в результате FP-дерево является проекцией набора предметов X, включающего текущий префикс, дополненный предметом i (строка 9). Далее алгоритм FPGrowth вызывается рекурсивно для проекции FP-дерева R_X и нового набора предметов X в качестве префикса (строка 16).

Детали алгоритма FPGrowth (2)

Базовым для рекурсии является случай, когда входное FP-дерево R содержит один путь. FP-деревья, которые содержат множественные пути, обрабатываются путем выделения всех наборов предметов, которые являются подмножествами пути, с поддержкой каждого такого набора, равной поддержке наименее популярного предмета в наборе (строки 2-6).

Сравнительные исследования алгоритмов Apriori и FPG показывают, что с увеличением числа транзакций временные затраты на поиск популярных наборов предметов растут для FPG намного медленнее, чем для Apriori.

Алгоритм FP-Max является вариантом алгоритма FPG, в котором определяются популярные наборы предметов максимального размера (максимальные популярные наборы предметов). Популярный набор предметов X называется максимальным, если X является популярным и не существует другого популярного набора предметов, содержащего X в качестве подмножества. Другими словами, популярный набор является максимальным, если он не является подмножеством другого популярного набора предметов.

Получение ассоциативных правил

Если дан популярный набор предметов $Z \in \mathcal{F}$, то для построения ассоциативных правил рассматриваются все его подмножества $X \subset Z$ и правила вида

$$X \longrightarrow Y,$$

где $Y = Z \setminus X \,(= Z - X)$.

Это правило должно быть популярным, так как его поддержка

$$s = \sup(X\,Y) = \sup(Z) \geqslant minsup.$$

Достоверность (confidence) правила вычисляется по формуле

$$c = \mathrm{conf}\,(X \to Y) = \frac{\sup(X \cup Y)}{\sup(X)} = \frac{\sup(Z)}{\sup(X)}.$$

Если $c \geqslant minconf$, то правило является сильным. С другой стороны, если $c < minconf$, то $\mathrm{conf}\,(W \to Z \setminus W) < minconf$ для всех подмножеств $W \subset X$, так как $\sup(W) \geqslant \sup(X)$. Таким образом, не нужно проверять подмножества X.

Алгоритм майнинга ассоциативных правил

AssociationRules ($\mathcal{F}$, $minconf$):

1 foreach $Z \in \mathcal{F}$, такого, что $|Z| \geqslant 2$ do
2 $\mathcal{A} \leftarrow \left\{X \mid X \subset Z, X \neq \emptyset\right\}$
3 while $\mathcal{A} \neq \emptyset$ do
4 $X \leftarrow$ максимальный элемент в $\mathcal{A}$
5 $\mathcal{A} \leftarrow \mathcal{A} \setminus X$ // удалить X из $\mathcal{A}$
6 $c \leftarrow sup(Z)/sup(X)$
7 if $c \geqslant minconf$ then
8 $Y \leftarrow Z \setminus X$
9 печатать $X \longrightarrow Y$, $sup(Z)$, c
10 else
11 $\mathcal{A} \leftarrow \mathcal{A} \setminus \left\{W \mid W \subset X\right\}$
  // удалить все подмножества X из $\mathcal{A}$

Пример: построение ассоциативных правил

Рассмотрим популярный набор предметов $ABDE\,(3)$ (в скобках указана поддержка) и допустим, что $minconf = 0.9$. Для построения сильных ассоциативных правил рассмотрим множество всех подмножеств

$$\mathcal{A} = \{ABD\,(3), ABE\,(4), ADE\,(3), BDE\,(3), AB\,(4), AD\,(3), AE\,(4),$$

$$BD\,(4), BE\,(5), DE\,(3), A\,(4), B\,(6), D\,(4), E\,(5)\}$$

$$X = ABD \Rightarrow \text{conf}\,(ABD \to E) = \frac{\sup(ABDE)}{\sup(ABD)} = \frac{3}{3} = 1 > 0.9$$

$$X = ABE \Rightarrow \text{conf}\,(ABE \to D) = \frac{\sup(ABDE)}{\sup(ABE)} = \frac{3}{4} = 0.75 < 0.9$$

Удаляем $X = ABE$ и все его подмножества, тогда

$$\mathcal{A} = \{ADE\,(3), BDE\,(3), AD\,(3), BD\,(4), DE\,(3), D\,(4)\}$$

$$X = ADE \Rightarrow \text{conf}\,(ADE \to B) = \frac{\sup(ABDE)}{\sup(ADE)} = \frac{3}{3} = 1 > 0.9$$

Пример: построение ассоциативных правил 2

$$X = ADE \Rightarrow \text{conf}\,(ADE \to B) = \frac{\sup(ABDE)}{\sup(ADE)} = \frac{3}{3} = 1 > 0.9$$

$$X = BDE \Rightarrow \text{conf}\,(BDE \to A) = \frac{\sup(ABDE)}{\sup(BDE)} = \frac{3}{3} = 1 > 0.9$$

$$X = AD \Rightarrow \text{conf}\,(AD \to BE) = \frac{\sup(ABDE)}{\sup(AD)} = \frac{3}{3} = 1 > 0.9$$

$$X = BD \Rightarrow \text{conf}\,(BD \to AE) = \frac{\sup(ABDE)}{\sup(BD)} = \frac{3}{4} = 0.75 < 0.9$$

Удаляем $X = BD$ и все его подмножества, тогда $\mathcal{A} = \{DE\,(3)\}$

$$X = DE \Rightarrow \text{conf}\,(DE \to AB) = \frac{\sup(ABDE)}{\sup(DE)} = \frac{3}{3} = 1 > 0.9$$

Итак, получаем набор сильных ассоциативных правил

$$ABD \to E, ADE \to B, BDE \to A, AD \to BE, DE \to AB$$

Меры оценки ассоциативных правил: Support

Поддержка (support) правила правила $X \to Y$ определяется как число транзакций, которые содержат как X, так и Y, т.е.

$$\sup(X \to Y) = \sup(XY) = |\mathbf{t}(XY)|$$

Относительная поддержка (relative support) правила $X \to Y$ – это доля транзакций, содержащих как X, так и Y, т.е. эмпирическая совместная вероятность выбора обоих наборов X и Y:

$$\mathrm{rsup}(X \to Y) = \mathrm{rsup}(XY) = \mathbb{P}[XY] = \frac{\sup(XY)}{|\mathbf{D}|}.$$

Поддержка (относительная поддержка) является симметричной мерой, так как

$$\sup(X \to Y) = \sup(Y \to X)$$

Support (поддержка): примеры

t	Предметы
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

Набор данных

sup	rsup	Наборы предметов
3	0.5	AD,CE,DE,ABD,ADE,BCE, BDE,$ABDE$
4	0.67	A, C, D, AB, AE, BC, BD, ABE
5	0.83	E, BE
6	1.0	B

Популярные наборы предметов для $minsup = 3$

Из набора предметов $ABDE$ выводится правило $AB \to DE$, которое имеет поддержку

$$\sup(AB \to DE) = \sup(ABDE) = 3,$$

а относительная поддержка равна

$$\text{rsup}(AB \to DE) = \frac{\sup(ABDE)}{|\mathbf{D}|} = \frac{3}{6} = 0.5$$

Меры оценки ассоциативных правил: Confidence

Достоверность (confidence) правила $X \to Y$ представляет собой условную вероятность того, что транзакция содержит последующий набор Y при условии, что она содержит предшествующий набор X:

$$\text{conf}\,(X \to Y) = \mathbb{P}\,[Y \mid X] = \frac{\mathbb{P}\,[X \wedge Y]}{\mathbb{P}\,[X]} = \frac{\text{rsup}\,(XY)}{\text{rsup}\,(X)} = \frac{\sup\,(XY)}{\sup\,(X)}$$

Как правило, мы заинтересованы в правилах с высокой достоверностью, а именно, когда

$$\text{conf}\,(X \to Y) \geqslant minconf,$$

где $minconf$ – минимально допустимый уровень достоверности. Достоверность не является симметричной мерой, так как по определению зависит от предшествующего набора предметов X.

Confidence (достоверность): примеры

t	Предметы
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

Набор данных

sup	rsup	Наборы предметов
3	0.5	AD,CE,DE,ABD,ADE,BCE, BDE,$ABDE$
4	0.67	A, C, D, AB, AE, BC, BD, ABE
5	0.83	E, BE
6	1.0	B

Популярные наборы предметов для $minsup = 3$

Правила	$conf$
$A \to E$	1.00
$E \to A$	0.80
$B \to E$	0.83
$E \to B$	1.00
$E \to BC$	0.60
$BC \to E$	0.75

Достоверность

Например, правило $E \to BC$ имеет достоверность $\mathbb{P}[BC \mid E] = 0.60$, т.е. при заданном предмете E имеется вероятность 60% найти предметы BC.
Однако безусловная вероятность набора BC равна

$$\mathbb{P}[BC] = \frac{4}{6} = 0.67,$$

что означает, что предмет E оказывает вредный эффект на предметы BC.

Меры оценки ассоциативных правил: Lift

Лифт (Lift) правила $X \to Y$ определяется как отношение наблюдаемой совместной вероятности X и Y к ожидаемой совместной вероятности в случае, если бы X и Y были независимы, т.е.

$$\text{lift}\,(X \to Y) = \frac{\mathbb{P}\,[XY]}{\mathbb{P}\,[X]\,\mathbb{P}\,[Y]} = \frac{\text{rsup}\,(XY)}{\text{rsup}\,(X)\,\text{rsup}\,(Y)} = \frac{conf\,(X \to Y)}{\text{rsup}\,(Y)}$$

Лифт показывает, насколько повышается вероятность нахождения Y в анализируемом случае, если в нем уже имеется X.

Значение лифта, близкое к 1, означает, что поддержка правила ожидается с учетом поддержки его компонентов. Обычно представляют интерес ситуации, когда лифт намного больше или намного меньше 1.

Лифт – это симметричная мера и лифт всегда больше или равен достоверности (confidence), потому что равен достоверности, деленной на вероятность нахождения набора Y.

Lift (лифт): примеры

t	Предметы
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

Набор данных

sup	rsup	Наборы предметов
3	0.5	AD,CE,DE,ABD,ADE,BCE, BDE,$ABDE$
4	0.67	A, C, D, AB, AE, BC, BD, ABE
5	0.83	E, BE
6	1.0	B

Популярные наборы предметов для $minsup = 3$

Правила	$lift$
$AE \to BC$	0.75
$CE \to AB$	1.00
$BE \to AC$	1.20

Лифты правил

Для набора предметов $ABCE$ поддержка равна $\sup(ABCE) = 2$, поэтому

$$\text{lift}(AE \to BC) = \frac{\text{rsup}(ABCE)}{\text{rsup}(AE)\,\text{rsup}(BC)} = \frac{\frac{2}{6}}{\frac{4}{6} \cdot \frac{4}{6}} = 0.75$$

$$\text{lift}(BE \to AC) = \frac{\text{rsup}(ABCE)}{\text{rsup}(BE)\,\text{rsup}(AC)} = \frac{\frac{2}{6}}{\frac{5}{6} \cdot \frac{2}{6}} = 1.2$$

Меры оценки ассоциативных правил: Leverage

Усиление или рычаг (Leverage) правила $X \to Y$ равен разности наблюдаемой совместной вероятности X и Y и ожидаемой совместной вероятности в случае, если бы X и Y были независимы, т.е.

$$\text{leverage}\,(X \to Y) = \mathbb{P}\,[XY] - \mathbb{P}\,[X]\,\mathbb{P}\,[Y] = \text{rsup}\,(XY) - \text{rsup}\,(X)\,\text{rsup}\,(Y)$$

Рычаг дает абсолютную меру неожиданности правила и должен использоваться вместе с лифтом.

Как и лифт, рычаг симметричен.

Leverage (рычаг): примеры

t	Предметы
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

Набор данных

sup	rsup	Наборы предметов
3	0.5	AD,CE,DE,ABD,ADE,BCE, BDE,$ABDE$
4	0.67	A, C, D, AB, AE, BC, BD, ABE
5	0.83	E, BE
6	1.0	B

Популярные наборы предметов для $minsup = 3$

Правила	$rsup$	$lift$	$leverage$
$ACD \to E$	0.17	1.20	0.03
$AC \to E$	0.33	1.20	0.06
$AB \to D$	0.50	1.12	0.06
$A \to E$	0.67	1.20	0.11

Рычаги правил

Для правила $ACD \to E$ рычаг равен

$$\text{leverage}\,(ACD \to E) = \mathbb{P}\,[ACDE] -$$

$$-\mathbb{P}\,[ACD]\,\mathbb{P}\,[E] = \frac{1}{6} - \frac{1}{6} \cdot \frac{5}{6} = 0.03$$

Правило $A \to E$ предпочтительнее первых трех, так как оно проще и имеет более высокий рычаг.

Меры оценки ассоциативных правил: Conviction

Пусть $\neg X$ – это отсутствие набора X в транзакции, аналогично для $\neg Y$. В зависимости от вхождения или отсутствия наборов X и Y в транзакциях возможны 4 случая, показанные в таблице:

	Y	$\neg Y$	
X	$\sup(XY)$	$\sup(X\neg Y)$	$\sup(X)$
$\neg X$	$\sup(\neg XY)$	$\sup(\neg X\neg Y)$	$\sup(\neg X)$
	$\sup(Y)$	$\sup(\neg Y)$	$\lvert\mathbf{D}\rvert$

Убежденность (conviction) правила $X \to Y$ измеряет ожидаемую ошибку правила, т.е. как часто X появляется в транзакциях, в которых Y отсутствует, т.е. это мера силы правила относительно дополнения к набору Y:

$$\text{conv}(X \to Y) = \frac{\mathbb{P}[X]\,\mathbb{P}[\neg Y]}{\mathbb{P}[X\neg Y]} = \frac{1}{\text{lift}(X \to \neg Y)} = \frac{1 - \text{rsup}(Y)}{1 - \text{conf}(X \to Y)}$$

Если совместная вероятность события $X\neg Y$ меньше ожидаемой при независимости X и $\neg Y$, тогда убежденность высока, и наоборот. Убежденность является асимметричной мерой.

Conviction (убежденность): примеры

t	Предметы
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

Набор данных

sup	rsup	Наборы предметов
3	0.5	AD,CE,DE,ABD,ADE,BCE, BDE,$ABDE$
4	0.67	A, C, D, AB, AE, BC, BD, ABE
5	0.83	E, BE
6	1.0	B

Популярные наборы предметов для $minsup = 3$

Правила	$rsup$	$conf$	$lift$	$conv$
$A \to DE$	0.50	0.75	1.50	2.00
$DE \to A$	0.50	1.00	1.50	∞
$E \to C$	0.50	0.60	0.90	0.83
$C \to E$	0.50	0.75	0.90	0.68

Убежденность

Для правила $A \to DE$ имеем

$$\text{conv}\,(A \to DE) =$$

$$= \frac{1 - \text{rsup}\,(DE)}{1 - \text{conf}\,(A \to DE)} = 2.0$$