Правила придруживања

Ненад Митић

Математички факултет
nenad@matf.bg.ac.rs

Увод

Сваки слог података садржи податке из једне *трансакције*. На почетку слога се налази (јединствени) идентификатор трансакције (у даљем тексту ИдТ, енг. *Transaction IDentifier, TID*) за којим је наведен скуп објеката који се налазе у тој трансакцији. У терминологији правила придруживања, објекти се називају *ставке*, а скуп објеката *скуп ставки*.

ИдТ	Скуп ставки
1	{Хлеб, Млеко}
2	{Хлеб, Пелене, Пиво, Јаја}
3	{Млеко, Пелене, Пиво, Кола}
4	{Хлеб, Млеко, Пелене, Пиво}
5	{Хлеб, Млеко, Пелене, Кола}

Увод

Одређивање правила придруживања је процес у коме се за дати скуп трансакција (слогова) проналазе правила која предвиђају појављивање ставке (објекта) на основу појављивања осталих ставки (објеката) у трансакцијама.

Основни појмови (из уводног дела)

- Скуп ставки
- Подршка
- Поузданост
- Правило придруживања

Увод

Примери правила придруживања

$$\{\text{Пелене}\} \longrightarrow \{\text{Пиво}\}$$
$$\{\text{Млеко, Хлеб}\} \longrightarrow \{\text{Јаја, Кола}\}$$
$$\{\text{Пиво, Хлеб}\} \longrightarrow \{\text{Млеко}\}$$

Импликација означава истовремено појављивање, не узрочно-последичну везу!

Тако је значење правила придруживања

$$\{\textit{Пиво, Хлеб}\} \longrightarrow \{\textit{Млеко}\}$$

ако купац купи пиво и хлеб, (вероватно) ће купити и млеко

Потрошачка корпа

Као класичан пример за илустрацију метода за одређивање правила придруживања користи се *потрошачка корпа*. Потрошачка корпа садржи скуп артикала које је појединачни купац купио у некој радњи/супермаркету. Свака потрошачка корпа (скуп артикала које је купио појединачни купац) представља једну трансакцију.

На основу садржаја потрошачих корпи (односно списка купљених артикала које су купили појединачни купци) власници продавница могу да извуку корисне информације помоћу којих могу да унапреде своје пословање.

Потрошачка корпа

Као пример података за илустрацију метода правила поред података из базе STUD2020 придруживања биће коришћени скупови

- Shopping dataset (http://dmg.org/pmml/pmml_examples/index.html) и
- Bakery (http://users.csc.calpoly.edu/~dekhtyar/466-Spring2018/labs/lab01.html)

Db2move верзије табела са материјалима су расположиве на сајту предмета

Потрошачка корпа

Неки од циљева налажења правила придруживања на основу потрошачких корпи су:

- Утврдити који се артикли продају заједно. У овом случају се такви артикли смештају један поред другог у циљу повећања продаје односно профита власника.
- Ако купац купи одређене артикле, који су још производи вероватни да се нађу у његовој корпи. Примена ових резултат је нпр. код организације промотивне кампање (распродаје) тако да се попуст не даје истовремено на артикле који се уобичајено купују заједно
- Поред уобичајених 'пакета' производа, наћи и изненађујуће (неуобичајене) придружене артикле (нпр. нови тренд)
- Предвиђање продаје и благовремено наручивање залиха
- ...

Неки појмови

Ефекат правила придруживања може да се опише као

- Утврђивање веза (придруживања, асоцијација) између података у великим базама података
- "Анализа веза између података"
  - откривање веза између појединачних података
  - при томе се не карактерише целокупна база података
- Опис релације између скупова елемената у подацима облика $A \longrightarrow B$ где су A и B скупови елемената

Неки појмови

- Скуп ставки (енг. itemset) садржи једну или више ставки (ставке, код потрошачке корпе артикли, енг. *items*)
- k-скуп ставки - скуп који садржи k ставки
- Дужина трансакције је број ставки које се налазе у трансакцији
- Подршка, поузданост - дефинисани у уводном делу
- Скуп ставки је чест ако се у улазним подацима јавља више од *minsup* пута, где *minsup* означава вредност минималног прага подршке који се задаје унапред
- Скуп ставки је поуздан ако се у улазним подацима јавља више од *minconf* пута, где *minconf* означава вредност минималног прага поузданости који се задаје унапред

Дефиниција правила придруживања

Нека су X и Y два скупа ставки. Тада се правило у ознаци $X \Rightarrow Y$ назива правило придруживања са минималном подршком *minsup* и минималном поузданошћу *minconf* ако важи:

1. Подршка скупа $X \cup Y$ је $\geq minsup$
2. Поузданост правила $X \Rightarrow Y$ је $\geq minconf$

Из уводног дела: Ако су A и B два скупа ставки, тада се

- подршка (енг. *support*) правила придруживања $A \Rightarrow B$ у ознаци *sup* дефинише као количник броја трансакција које садрже A и B у односу на укупан број трансакција $sup(A \Rightarrow B) = \frac{\#(A \cup B)}{N}$
- поузданост (поверење, енг. *confidence*) правила придруживања $A \Rightarrow B$ у ознаци *conf* дефинише као количник броја трансакција које садрже A и B у односу на број трансакција које садрже A $conf(A \Rightarrow B) = \frac{\#(A \cup B)}{\#(A)}$

Дефиниција правила придруживања

За дати скуп трансакција (слогова) T циљ одређивања правила придруживања је пронаћи CBA правила која имају

- подршку $\geq$ *minsup* (минимални праг подршке)
- поузданост $\geq$ *minconf* (минимални праг поузданости)

У случају јако великих база података

- minconf (минимални ниво поверења) се обично поставља високо (нпр. 80%)
- minsup (минимални ниво подршке) је уобичајено значајно нижи (нпр. 5-10%), због велике разноликости

Дефиниција правила придруживања

Висока подршка

- значи да се правило често појављује и да је мање вероватно да се правило случајно појавило
- издваја правила која нуде више могућности за добијање користих информација

Високо поверење (поузданост)

- означава *јако* правило, које указује на висок ниво узрочности, и јаку везу придруживања између артикала/елемената
- указује на правило које је од интереса и на које треба обратити пажњу

Формирање правила придруживања

Процес одређивања правила придруживања може се поделити на два корака:

1. Формирати све скупове ставки код којих је подршка $\geq$minsup
2. Формирати правила из сваког скупа честих ставки са поузданошћу $\geq minconf$

Теоријски доказ постојања алгоритма за одређивање правила придруживања је примена методе грубе силе:

- Излистати сва могућа правила придруживања
- Израчунати подршку и поузданост за свако правило. Када се зна подршка скупа ставки ово је једноставна опреација са становишта имплементације и утрошка ресурса.
- Одбацити правила која не задовољавају minsup и minconf праг

Формирање правила придруживања

Формирање свих честих скупова артикала је проблем јер

- у применама се обично појављују стотине хиљада различитих артикала (нпр. потрошачка корпа)
- за d артикала постоји $2^d - 1$ могућих скупова (+ празан скуп) који су потенцијално чести и које треба испитати
- ако се апликација често извршава (нпр. примена у хипермаркетима или великим системима) њихову учесталост треба проверити на основу милиона трансакција сваког дана/сата

Рачунарски врло захтеван процес:

- Потребно је $O(NMw)$ поређења где је N број трансакција, $M = 2^d - 1$ број кандидатских скупова, а w је максимална дужина трансакције
- Да ли је могуће смањити простор претраге за честе скупове?

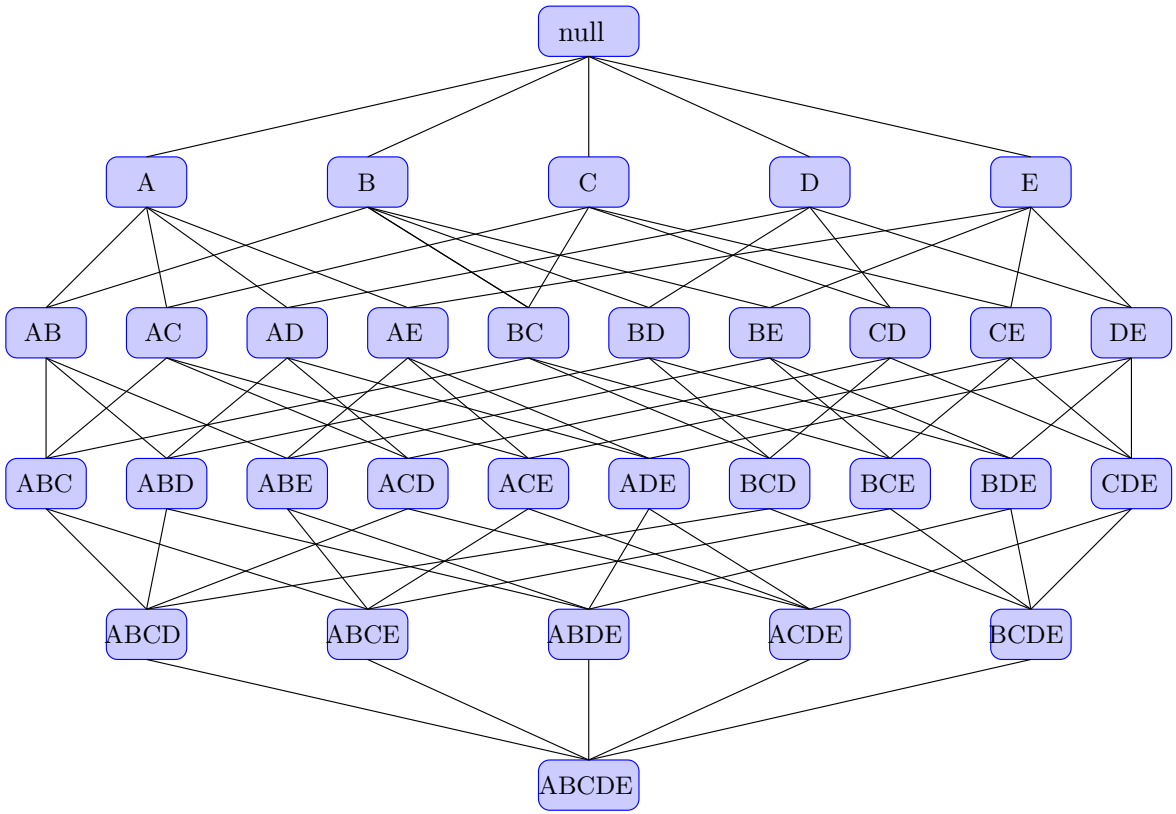
Стратегије за повећање ефикасности

Ефикасност претраживања се повећава смањењем броја потребних поређења које је могуће извести:

- Смањењем броја кандидатских ставки (M)
  - Уместо претраживања комплетног скупа $M = 2^d$ користе се технике поткресивања ради смањења M
  - Елиминација ниски које нису честе без пребројавања њихове подршке - *Априори принцип*
- Смањити број трансакција (N)
  - Са повећањем величине кандидатских скупова смањује се број трансакција које садрже скупове те величине
- Смањити број поређења (NM)
  - Користити ефикасније структуре података ради чувања скупова кандидата и/или трансакција
  - Није потребно упаривати сваки кандидат скуп са сваком трансакцијом

Решетка скупа ставки

Ставке из скупа трансакција који се испитује се најчешће представљају у облику *решетке*



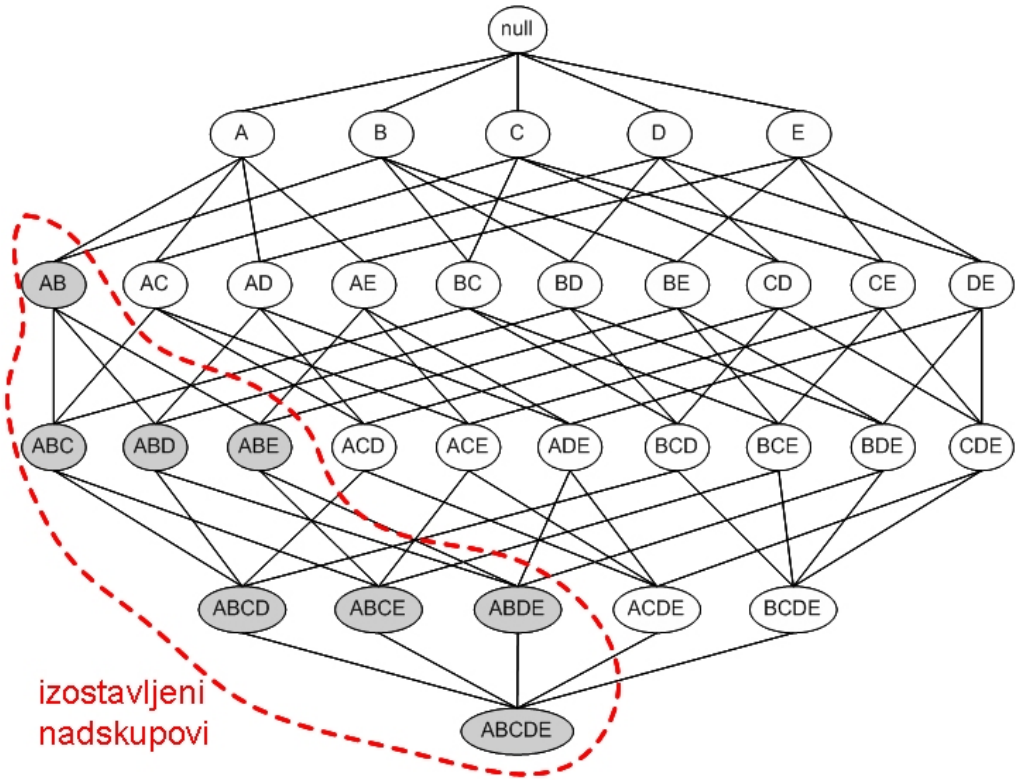
Априори принцип и анти-монотоност

- *Априори принцип*: ако је скуп ставки чест, тада су и сви његови подскупови такође чести
  - *Затворење на ниже (енг.* downward closure*)*: сваки подскуп честог скупа је такође чест
- *Особина анти-монотоности*: Мера f поседује особину анти-монотоности ако за све скупове ставки X, Y важи: $X \subset Y \Longrightarrow f(Y) \leq f(X)$

  Подршка задовољава особину анти-монотоности
- Последица (*анти-монотоности*): ако неки скуп артикала A није подржан (није чест), тада ни било који скуп B где важи $A \subseteq B$ није чест $\longrightarrow$ не треба да буде разматран

Априори принцип и анти-монотоност

Илустрација последице принципа анти-монотоности: ако скуп $\{A, B\}$ није чест, тада ни сви остали скупови који га садрже (означени сивом бојом) нису чести, односно не треба да буду разматрани у процесу одређивања правила придруживања



Монотоност и поузданост

Поузданост не задовољава ни једну од особина монотноности. Ако $X_1 \subset X \wedge Y_1 \subset Y$ тада поузданост правила $X \Rightarrow Y$ може бити мања, већа или једнака поузданости правила $X_1 \Rightarrow Y_1$. Међутим, за поузданост важи следеће правило:

Ако су X_1, X_2 и I скупови ставки такви да важи $X_1 \subset X_2 \subset I$, тада важи

$$conf(X_2 \Rightarrow I - X_2) \geq conf(X_1 \Rightarrow I - X_1)$$

Последица: могућност елиминације редундантних правила. Нпр. у случају правила $\{Hleb\} \Rightarrow \{Pivo, Mleko\}$ і $\{Hleb, Pivo\} \Rightarrow \{Mleko\}$ прво правило може да се уклони јер има мању поузданост од другог

Метода грубе силе

Алгоритам који показује да је (теоријски) могуће добити скуп правила придруживања која испуњавају услов $\geq minsup \wedge \geq minconf$
Алгоритам:

- Сваки скуп ставки у решетки је кандидат да буде чест
- Пребројати подршку за сваког кандидата прегледањем базе
- Упарује се свака трансакција са сваким кандидатом

Домаћи задатак: примените методу грубе силе за вредности minsup=2 и 3 и minconf=0.4 и 0.6 на улазни скуп

ИдТ	Ставке
1	Хлеб, Млеко
2	Хлеб, Пелене, Пиво, Јаја
3	Млеко, Пелене, Пиво, Кола
4	Хлеб, Млеко, Пелене, Пиво
5	Хлеб, Млеко, Пелене, Кола

Метода грубе силе

Проблем овог алгоритма је врло велики број операција (за иоле већи скуп трансакција, односно ставки), што га чини практично немогућим за извршавање, тако да се не користи у пракси

- Сложеност $O(NMw)$ - јако скупо јер је $M = 2^d - 1$- број ставки, w - максимална ширина трансакције, N - број трансакција
- Укупан број могућих правила придруживања за d ставки је

$$\sum_{k=1}^{d-1} \left[\binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \right]$$

Априори алгоритам

- R. Agrawal и R. Srikant 1994. године
- Показао да је могуће ефективно добијање скупа правила придруживања и у случају већих скупова података
- Користи Априори принцип и анти-монотност за поткресивање скупа кандидатских ставки и посебне структуре за смањење броја поређења
- Детаљан опис: Xindong Wu and Vipin Kumar: The Top Ten Algorithms in Data Mining - глава 4

Основна идеја је реализована у различитим имплементацијама. Касније су се појавили и други алгоритми који користе неке од идеја које су се први пут јавиле при дефиницији овог алгоритма.

Априори алгоритам (наставак)

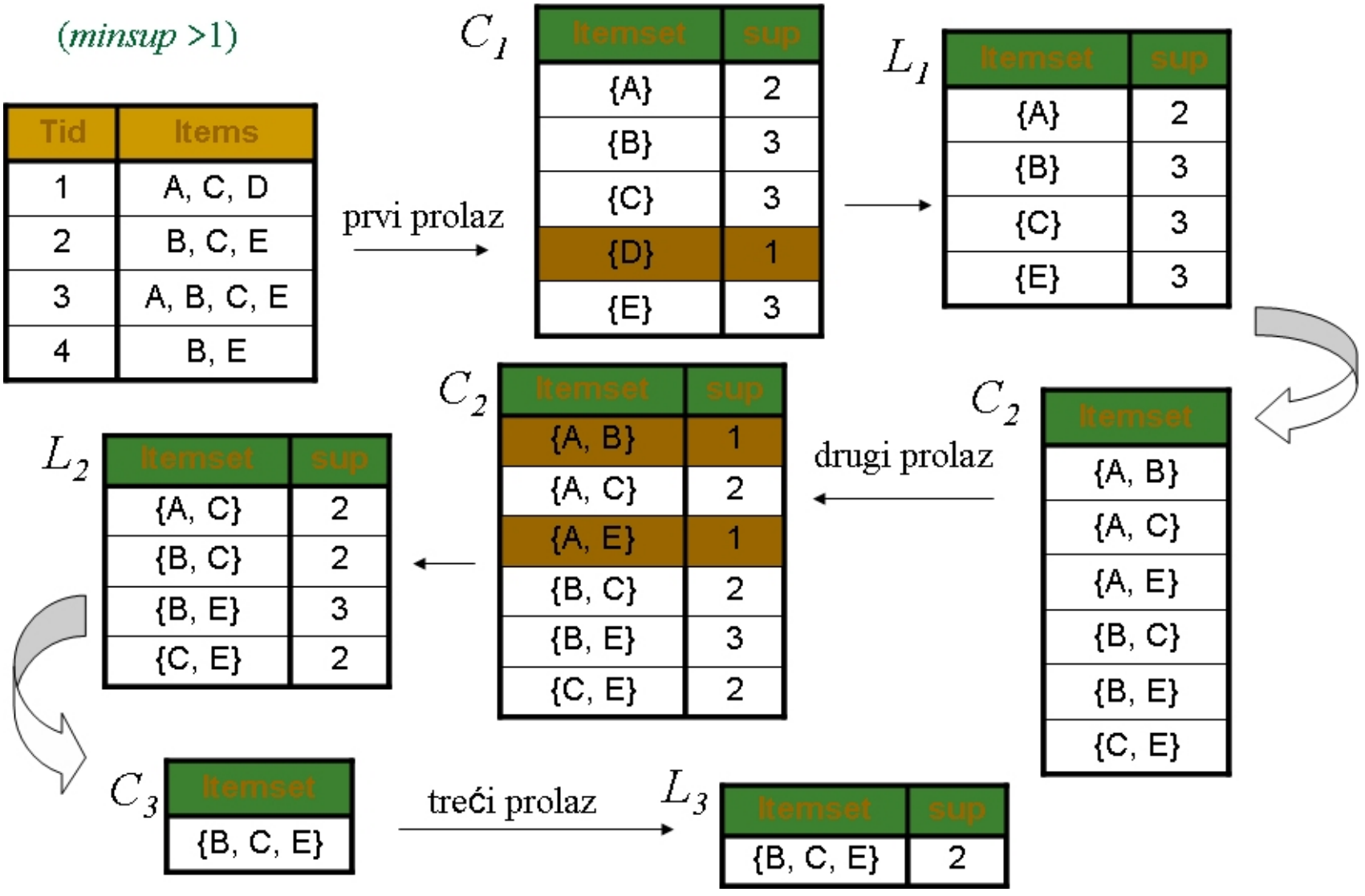
У Априори алгоритму се чести скупови ставки формирају извршавањем следећих корака:

- Елементи скупова ставки се уређују у лексикографски поредак
- Преброје се све 1-ставке и одбаце оне које нису честе
- На основу честих скупова ставки дужине k се формирају скупове дужине $k+1$
- Скупови k ставки се спајају и формирају скуп $k+1$ ставки акко су им једнаки првих $k-1$ елемената
- За формиране скупове дужине $k+1$ одређује се подршка
- Одбацују се ретки скупови $k+1$ ставки и не разматрају се њихови надскупове (последица затворења на ниже)
- Одбацује се скуп $k+1$ ставки ако неки од подскупова k ставки није чест

Априори алгоритам (наставак)

```
/* Ck+1 skup stavki kandidata duzine k+1 */
Apriori(Transakcije: T, Podrska: minsup)
begin
  k=1;
  F1={sve ceste 1-stavke};
  while Fk nije prazno do
    begin
       Generisati Ck+1 spajanjem stavki iz Fk;
       Odbaciti stavke iz Ck+1 koje ne
             zadovoljavaju zatvorenje na nize;
       Odrediti Fk+1 brojanjem podrske (Ck+1, T);
       Odbaciti one sa podrskom manjom od minsup;
       k=k+1;
    end;
  return (unija svih Fi, i=1;k);
end
```

Илустрација Априори алгоритма



Априори алгоритам (наставак)

На основу честих скупова ставки који су одређени врши се формирање правила придруживања:

- Сваки чест k-скуп ставки може да произведе до $2^k - 2$ правила придруживања (игноришу се правила са празном левом или десном страном)
- Правило се издваја делењем скупа ставки Y на два непразна подскупа X и $Y - X$, такве да $X \Rightarrow Y - X$ има већу поузданост од задатог прага
- Свако правило мора да задовољава и услов да има већу подршку од задатог прага подршке

Формирање и поткресивање скупова кандидата

Функција формирања нових нових кандидатских скупова у Априори алгоритму укључује

- Формирање кандидата: генеришу се нови к-скупови ставки на основу честих скупова ставки дужине (k-1) који су формирани у претходној итерацији
- Поткресивање скупа кандидата: елиминишу се одређени кандидатски к-скупови ставки коришћењем стратегије поткресивања на основу подршке
- Метод грубе силе - неефикасан јер разматра сваки подржани скуп дужине к као потенцијалног кандидата и затим примењује поткресивање
- Нека F_k означава скуп кандидата дужине k. F_k се формира на основу претходних честих F_i, $i < k$

Формирање и поткресивање скупова кандидата

Методе за формирање F_k

а) $F_{k-1} \times F_1$

- Проблеми - кандидатски скупови могу да се формирају на више начина: $\{X,Y\} \wedge \{Z\} \rightarrow \{X,Y,Z\}$; $\{X,Z\} \wedge \{Y\} \rightarrow \{X,Y,Z\}$
- Решење - кандидатски скуп се проширује честом ставком која је у лексикографском уређењу после ставки из кандидатског скупа
- Додатни недостатак: новоформирани k скуп може да садржи $k-1$ скупове који нису чести (а не постоје у скупу честих k скупова). Домаћи задатак: пронаћи пример оваквих скупова

б) $F_{k-1} \times F_{k-1}$

- Елементи $k-1$ скупова су уређени у лексикографском поретку
- Нека су $A = \{a_1, a_2, ..., a_{k-1}\}$ и $B = \{b_1, b_2, ..., b_{k-1}\}$ при чему је $a_{k-1} < b_{k-1}$ пар честих $k-1$-скупова ставки. Нови k скуп $C = \{a_1, a_2, ..., a_{k-1}, b_{k-1}\}$ се формира комбинацијом A и B ако је задовољен услов $a_i = b_i, \forall i = 1, 2, ..., k-2 \wedge a_{k-1} \neq b_{k-1}$
- Потребно је додатно испитивање да су сви (нови) $k-2$ скупови чести

Смањење броја поређења

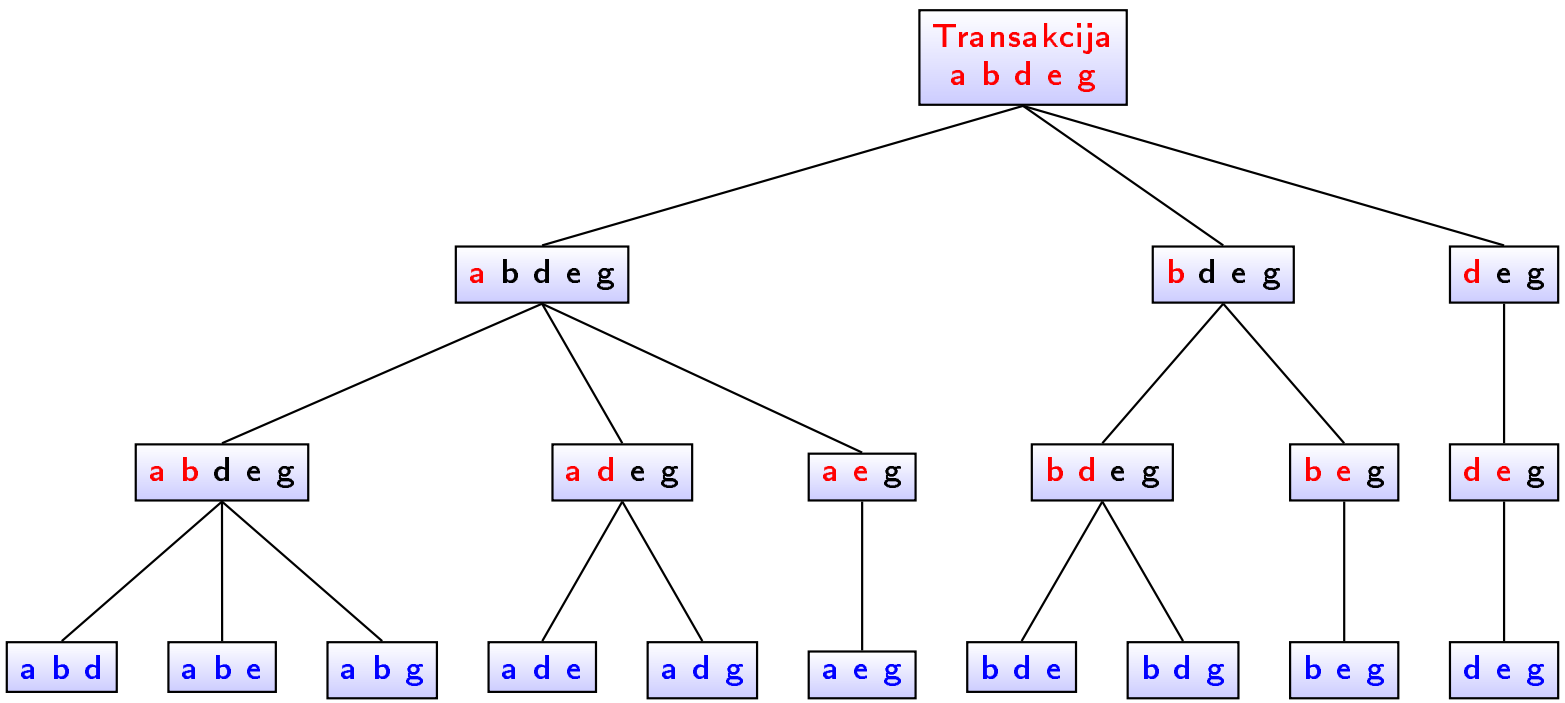
Априори алгоритам користи следећи механизам за смањење броја поређења

- Смешта ставке у решетку *у ширину*, ниво по ниво
- Групише податке у корпе (енг. *buckets*) по дужини скупова ставки
- Свака корпа се представља у облику хеш структуре са фиксним бројем грана у чвору
- На i-том нивоу хеш функција се примењује на i-ту ставку скупа
- Увећава се број ставки до којих се дође у листовима хеш структуре
- Ставке из трансакција се пореде са садржајем своје (кандидатске) корпе уместо са целим скупом кандидата чиме се смањује број поређења при одређивању броја појављивања (подршке) скупа ставки

Смањење броја поређења

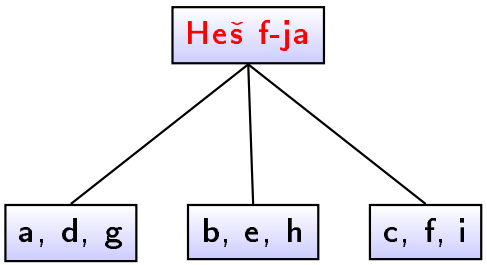
Смањење броја поређења при испитивању да ли се скупови ставки из трансакција налазе у скупу кандидатских ставки

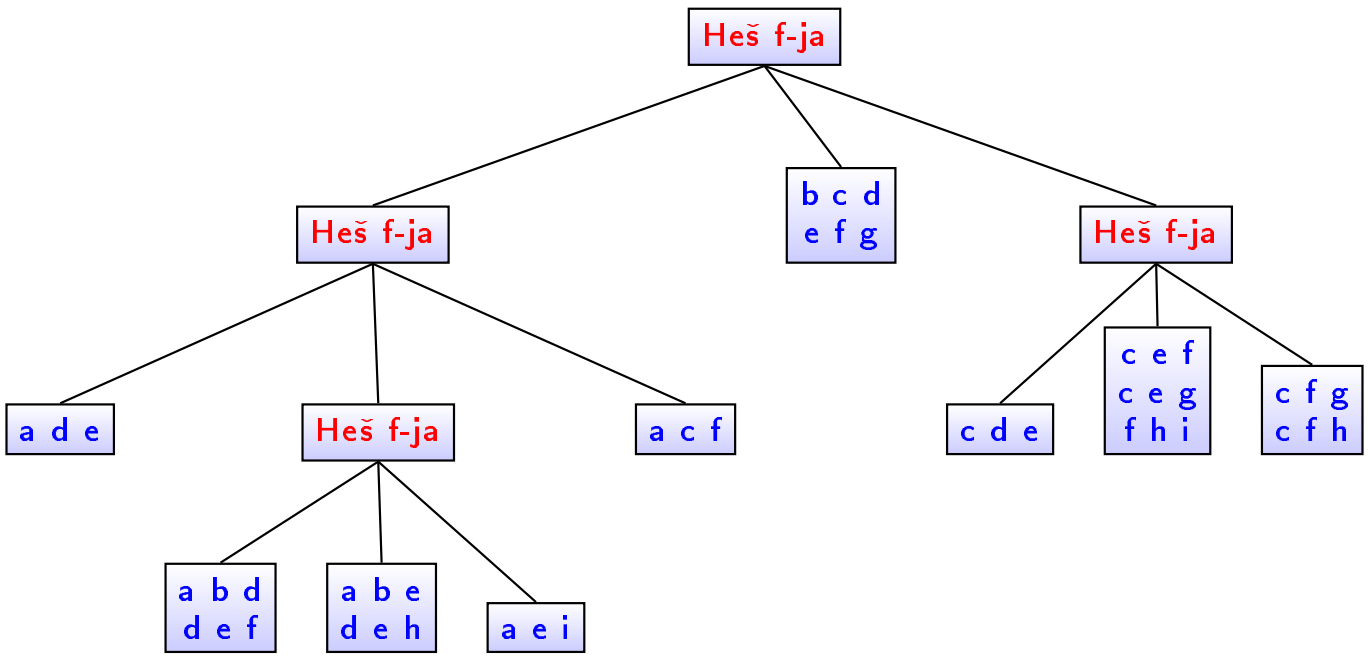
- ради ефикасности сви скупови ставки чувају ставке у растућем лексикографском уређењу
- кандидатске ставке се групишу у корпе и смештају у хеш дрво
- ставке из трансакције се такође групишу у корпе и упоређују са скупом кандидатских ставки из одговарајуће корпе (на слици 3-ставке)



Смањење броја поређења - хеш дрво

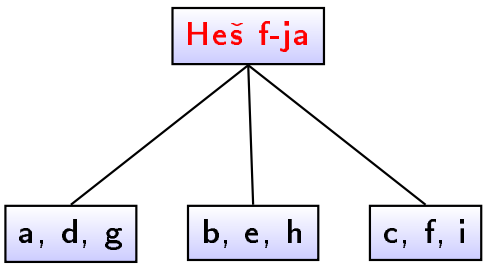
Хеш дрво дрво кандидатских ставки дужине 3 скупа ставки {a, b, c, d, e, f, g, h, i}

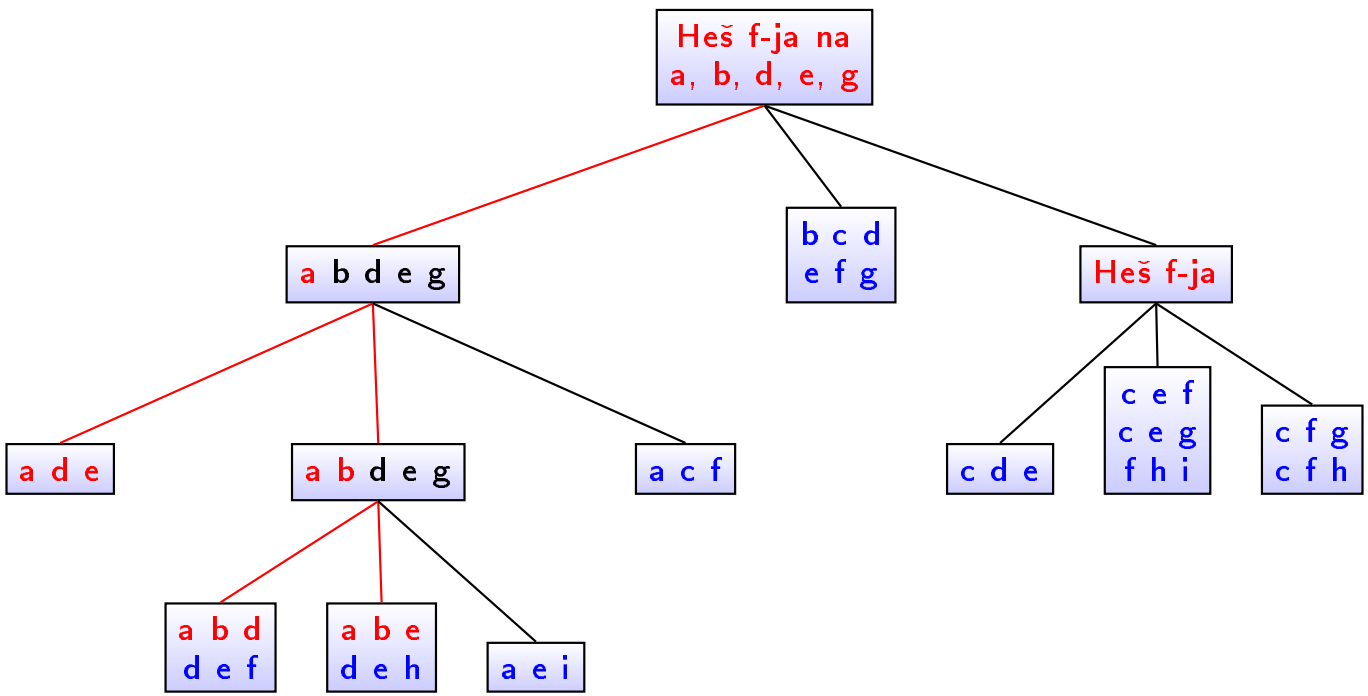




Смањење броја поређења - хеш дрво

У трансакцији која садржи {a, b, d, e, g} добијају се 3-ставке {*ade*}, {*abd*}, и {*abe*} које се поклапају са кандидатским ставкама





Максимално чести скупови ставки

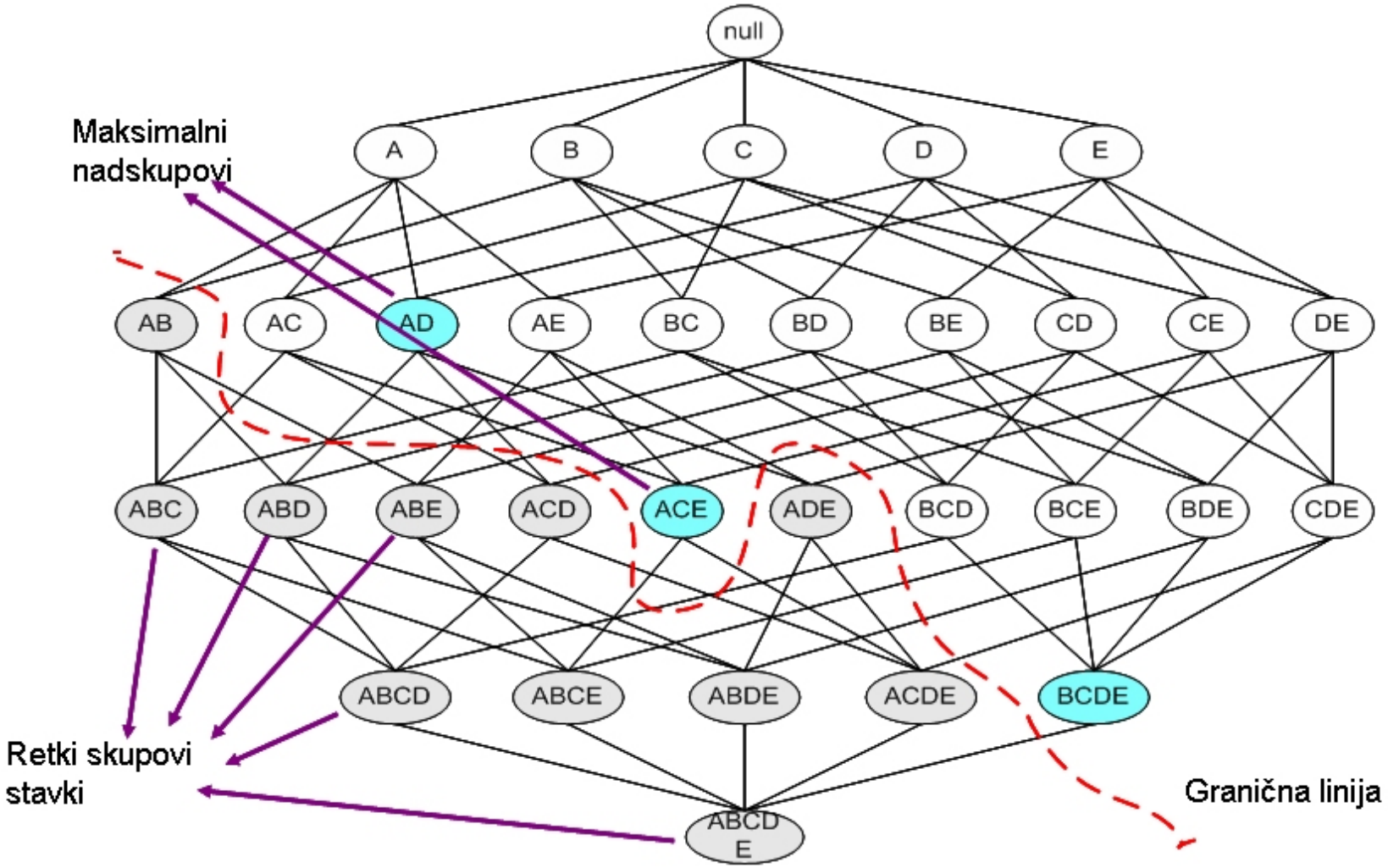
У пракси

- број честих ставки може да буде јако велики, поготову код великог скупа трансакција са великим бројем елемената
- потреба за компактним записивањем честих скупова ставки из кога могу да се добију одзив и прецизност за сваку ставку

Дефиниција: Скуп ставки је максимално чест ако ни један од његових непосредних надскупова није чест

1. Ако је X максимално чест скуп, тада су и сви његови подскупови чести
2. Било који од честих скупова ставки је подскуп неког од максимално честих скупова ставки $\Longrightarrow$ за добијање било ког честог скупа ставки довољно је чувати само скуп максимално честих скупова ставки
3. Скуп ставки који није подскуп неког од максимално честих скупова ставки није чест и може да се искључи из скупа ставки које се користе за формирање правила придруживања

Максимално чести скупови ставки



Затворени скупови ставки

Иако омогућују добијање свих честих скупова ставки, максимални скупови ставки не чувају информацију о подршци. У ту сврху се користе затворени скупови ставки.

Дефиниција: Скуп ставки је затворен ако ниједан од његових непосредних надскупова нема исту подршку
Последица: Сви подскупови затвореног скупа ставки имају исту или већу подршку

За скуп трансакција наведен у табели на левој страни, затворени скупови ставки су у табели приказани жутом бојом.

ИдТ	Ставке
1	Хлеб, Млеко
2	Млеко, Пелене, Пиво
3	Хлеб, Млеко, Пелене, Пиво
4	Хлеб, Млеко, Пиво
5	Хлеб, Млеко, Пелене, Пиво

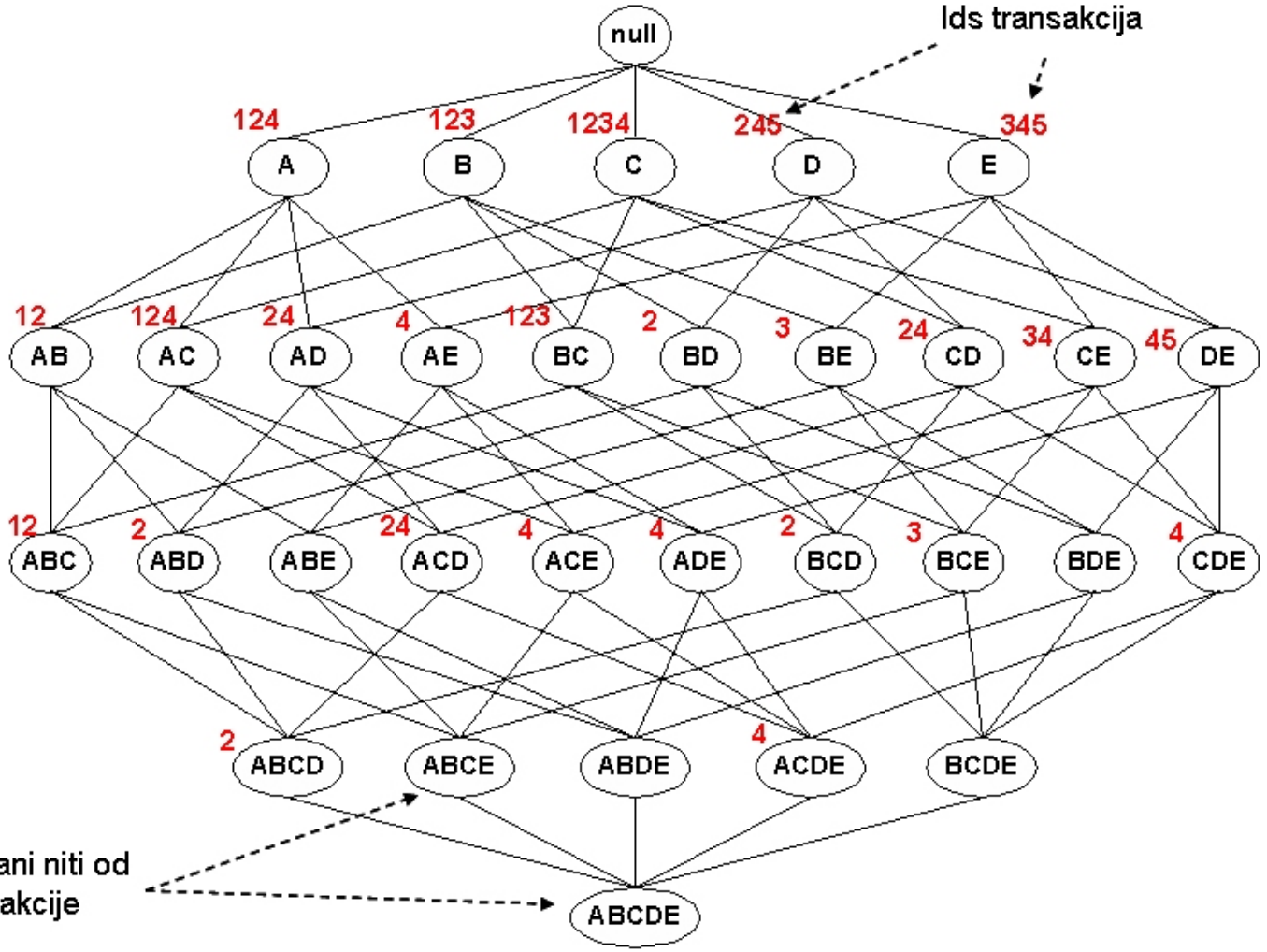
Ставка	Подршка
Хлеб	4
Млеко	5
Пелене	3
Пиво	4
Хлеб, Млеко	4
Хлеб, Пелене	2
Хлеб, Пиво	3
Млеко, Пелене	3
Млеко, Пиво	4
Пелене, Пиво	3
Хлеб,Млеко, Пелене	2
Хлеб,Млеко, Пиво	3
Хлеб, Пелене,Пиво	2
Млеко, Пелене, Пиво	3
Хлеб, Млеко, Пелене, Пиво	2

Компактно представљање честих ставки

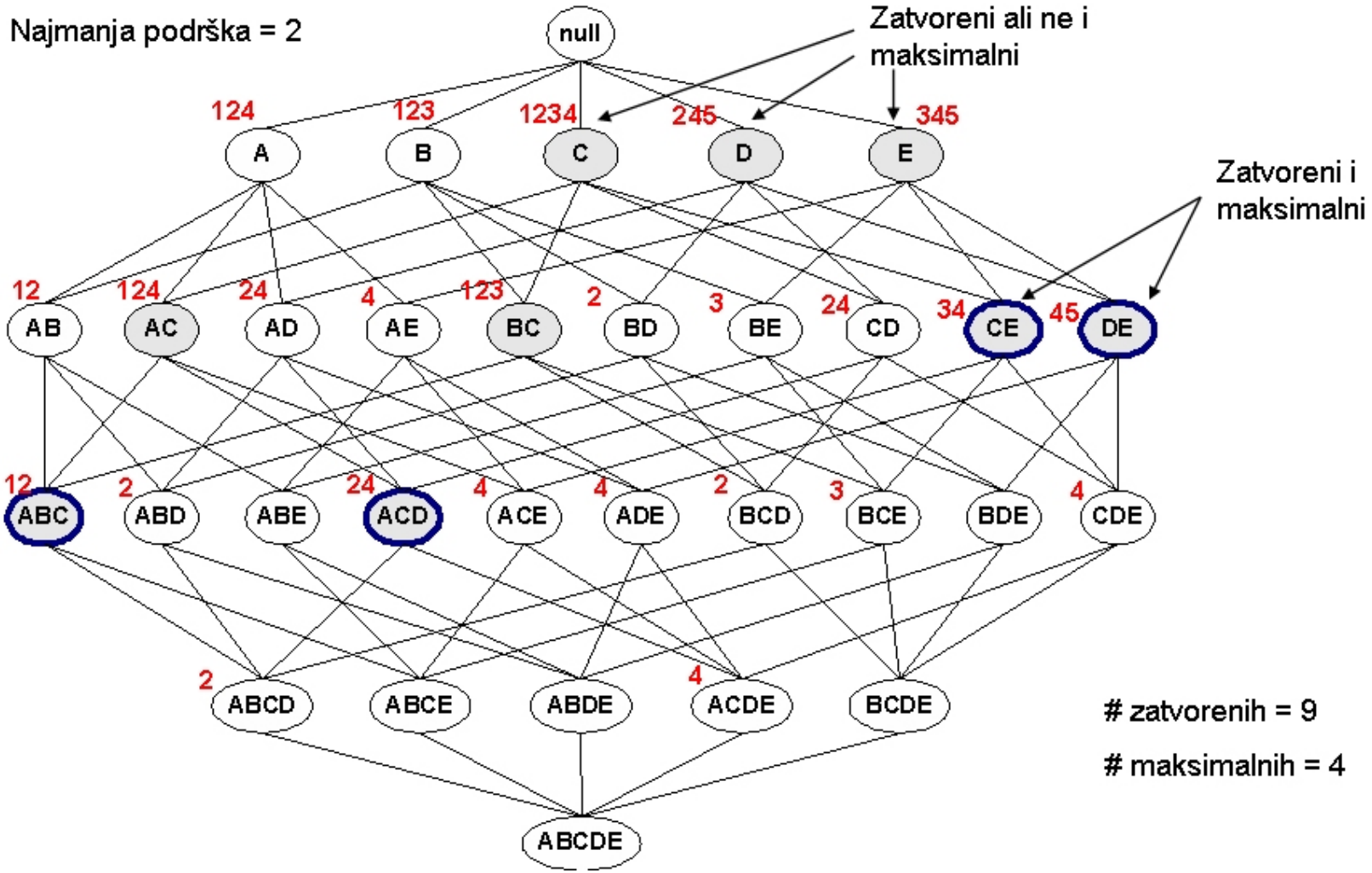
- Скуп ставки је максимално затворен ако је затворен и има подршку $\geq$ *minsup*
- Пошто је позната информација о подршци затворених скупова ставки, могуће је одредити подршку свих честих скупова ставки
- Уместо чувања свих честих скупова ставки и информација о њиховој подршци, довољно је чувати скуп максималних затворених скупова ставки и њихове подршке
- На наредној слици приказан је, за скуп трансакција наведен у табели, подршка сваког скупа ставки у решетки
- На другој слици, приказан је скуп максимално затворених скупова ставки за minsup=2

Компактно представљање честих ставки

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Компактно представљање честих ставки

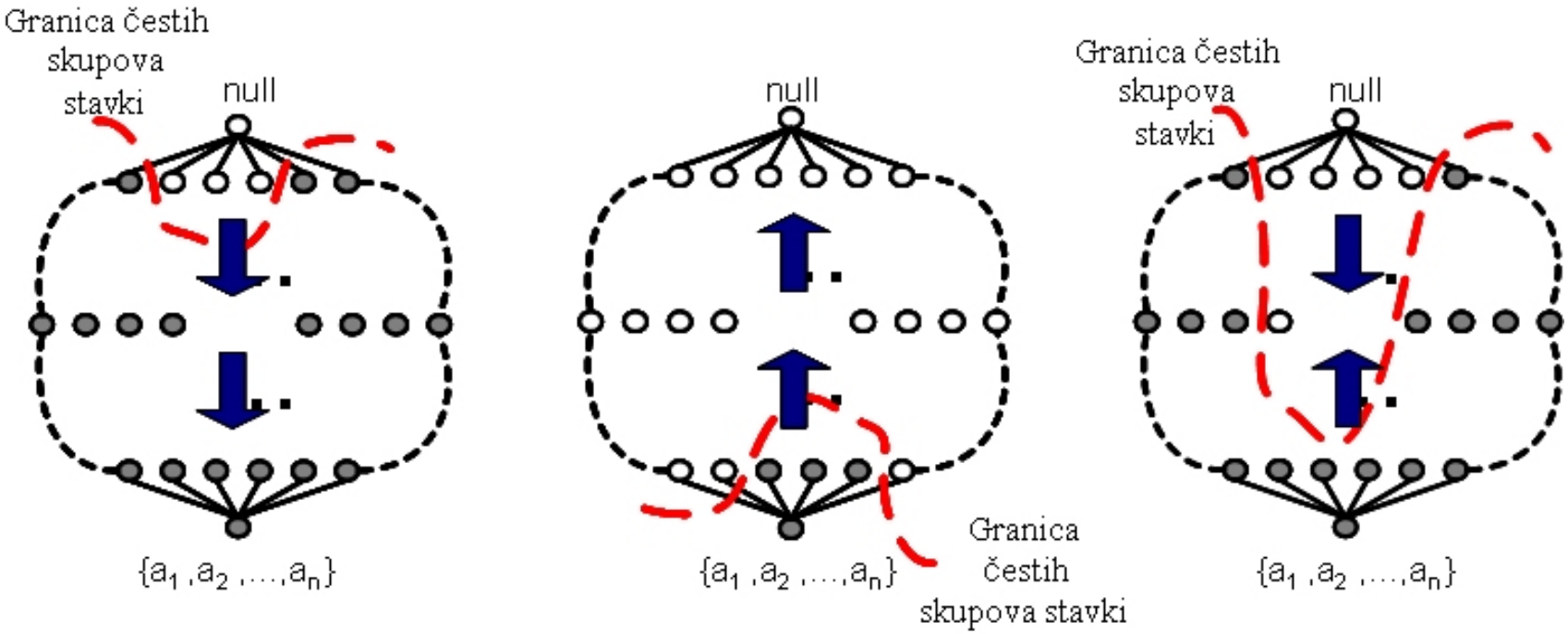


Начини обиласка решетке

- Чести скупови ставки се одређују обиласком решетке
- Велики број скупова ставки $\Longrightarrow$ потребна велика количина ресурса (првенствено меморије) за чување скупова ставки у решетки
- Проблем уочен код Априори алгоритма приликом обраде трансакција са великим бројем ставки
- Разлог - обрада решетке ниво по ниво ('у ширину'), захтева чување информације о скуповима са претходног нивоа
- Другачије технике обиласка решетке скупова ставки
- Свака техника има добре и лоше стране, и ситуације (у зависности од података) у којима се показује знатно боље од осталих

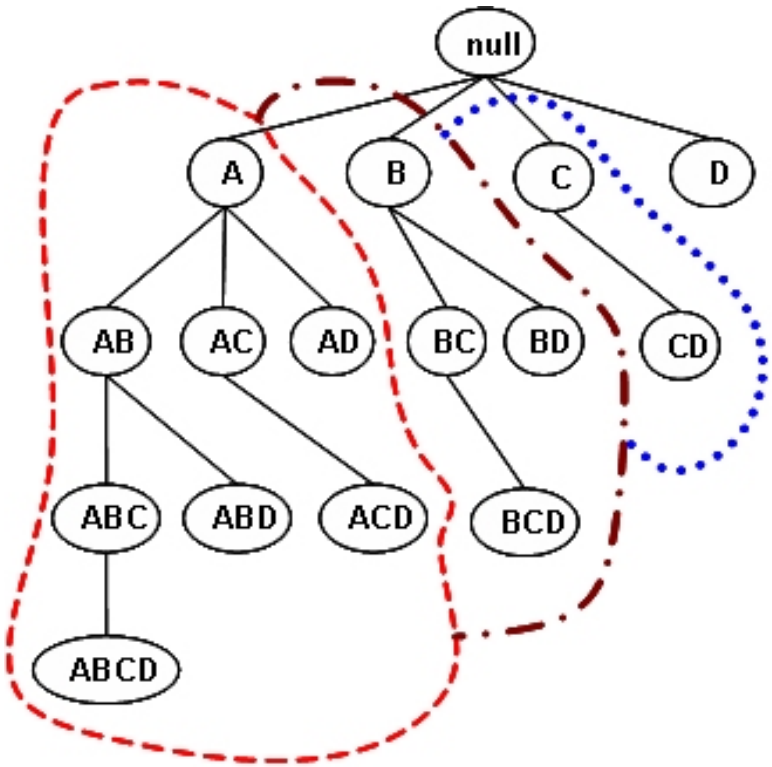
Начини обиласка решетке

Обилазак решетке: од општег ка посебном (лево), од посебног ка општем (средина) и двосмерно (десно)

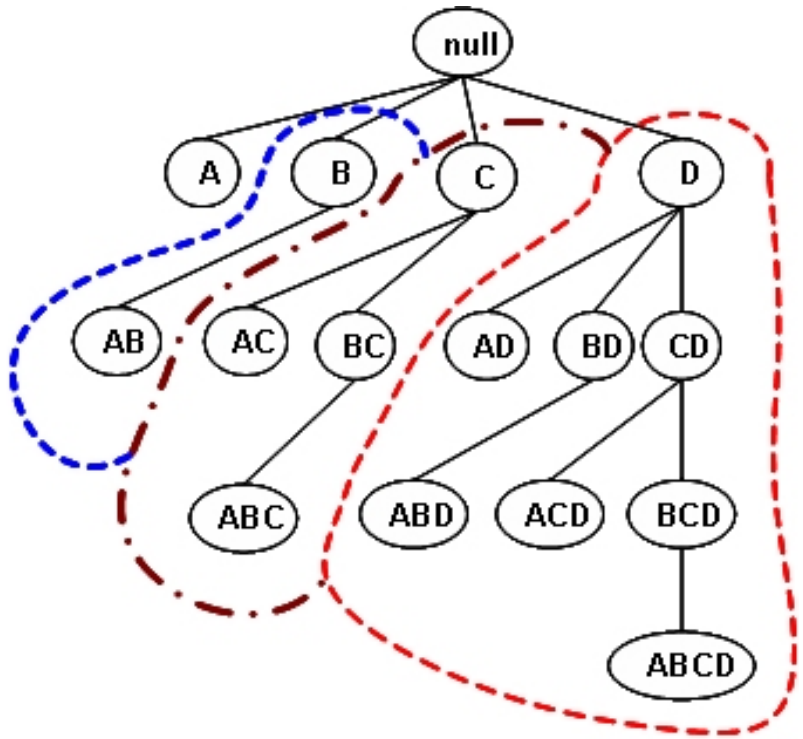


Начини обиласка решетке

Обилазак решетке: у еквивалентним класама које су засноване на префиксним или суфиксним деловима скупа ставки



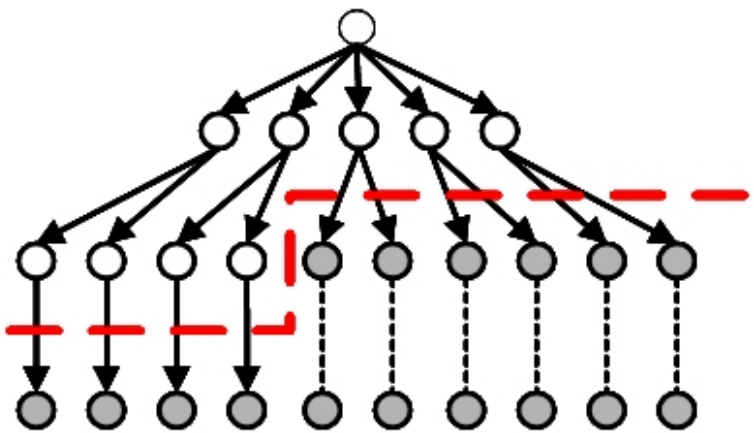
(a) Prefiksno drvo



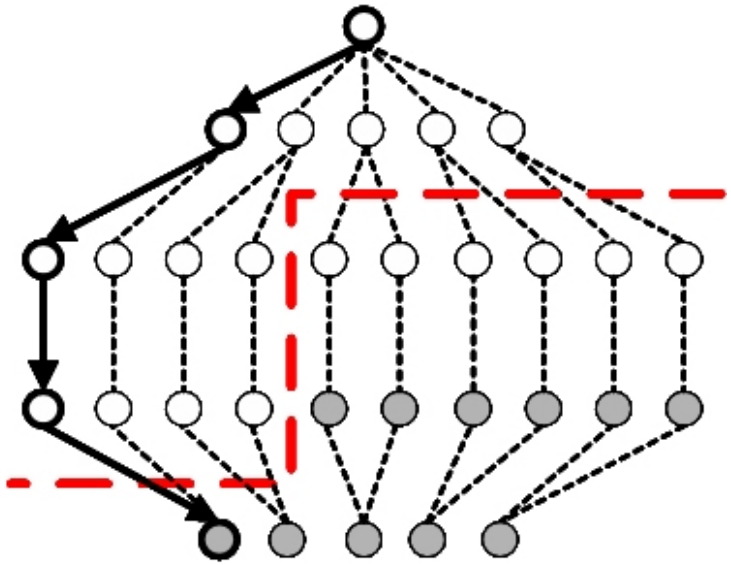
(b) Sufiksno drvo

Начини обиласка решетке

Обилазак решетке: прво у ширину (Apriori алгоритам) или прво у висину/дубину (нпр. SIDE - Simultaneous Depth-first Expansion algoritam)



(a) Obilazak 'prvo u širinu' (Apriori algoritam)



(b) Obilazak 'prvo u visinu'

Вертикални распоред података

Трансакције у бази података су уобичајено представљене хоризонтално. Међутим, могу бити представљене и вертикално, где се ставке налазе у заглављима колона, а вредности у пољима су бројеви трансакција.

Бр. транс.	хлеб	млеко	пелене	пиво	јаја	кола
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Ставка	Скуп идент. тр.	Бинарна репр.
хлеб	1, 2, 4, 5	11011
млеко	1, 3, 4, 5	10111
пелене	2, 3, 4, 5	01111
пиво	2, 3, 4	01110
јаја	2	01000
кола	3, 5	00101

Вертикални распоред података

- За пребројавање $k+1$-ставки користе се пресеци листи скупова трансакција k-ставки
- Овај начин чувања података захтева већу потрошњу меморије због чувања листи/мању потрошњу CPU
- *Eclat* и *VIPER* алгоритми користе рекурзивне пресеке листи трансакција

На наредном слајду је приказана модификација Априори алгоритма која користи вертикални запис трансакција.

Вертикални Априори алгоритам

```
/* Ck+1 skup stavki kandidata duzine k+1 */
VerikalniApriori(Transakcije: T, Podrska: minsup)
begin
  k=1;
  F1={sve ceste 1-stavke};
  Formirati vertikalnu listu transakcija za stavke;
  while Fk nije prazno do
    begin
       Generisati Ck+1 spajanjem parova stavki iz Fk;
       Odbaciti stavke iz Ck+1 koje ne
             zadovoljavaju zatvorenje na nize;
       Formirati tr_id liste svake kandidatske stavke
            iz Ck+1 kao presek tr_id listi para stavki
            iz Fk koriscnih za formiranje Ck+1;
       Odrediti podrsku stavki u Ck+1 brojanjem duzine liste;
       Fk+1=Ceste stavke Ck+1 zajedno sa  tr_id listama;
       k=k+1;
    end;
  return (unija svih Fi, i=1;k);
end
```

Алгоритам ФП раста

- Употребљава компримовану репрезентацију базе података помоћу ФП-дрвета (енг. *Frequent Pattern tree*)
- Што је већи број трансакција које имају заједничке почетне делове, то је већи степен компресије који се постиже овим алгоритмом
- Може да буде и вишеструко бржи од алгоритама који конструишу и обилазе решетку 'по ширини'

ФП дрво

ФП-дрво је дрволика структура са следећим карактеристикама:

- Садржи један корени чвор означен са *null*, скуп префиксних поддрвета који су деца кореног чвора, и табелу са заглављима честих ставки
- Сваки чвор у префиксном дрвету се састоји од три компоненте: имена ставке, бројача и чвор-везе. Име ставке садржи ставку која се обрађује у том чвору, бројач број њеног појављивања у трансакцијама које одговарају путањи до тог чвора, док чвор-веза садржи показивач на следећи чвор са истим именом у ФП-дрвету, или је *null* ако такав чвор не постоји
- Сваки упис у табелу заглавља садржи поља са именом ставке и почетним показивачем на чвор везу који показује на први чвор у ФП-дрвету са именом те ставке

Конструкција ФП-дрвета

Нека је дат скуп трансакција који садржи ставке А, Б, Ц, Д, и Е.

ИдТ	Ставке
1	{А, Б}
2	{Б, Ц, Д}
3	{А, Ц, Д, Е}
4	{А, Д, Е}
5	{А, Б, Ц}
6	{А, Б, Ц, Д}
7	{Б, Ц}
8	{А, Б, Ц}
9	{А, Б, Д}
10	{Б, Ц, Е}

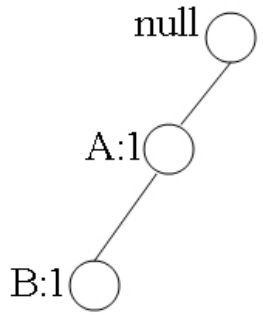
- Конструкција двета почиње од корена који не садржи ни једну ставку
- У чворовима дрвета који се конструишу уписује се ставка и број њеног појављивања
- У првом кораку се преброје све појединачне ставке и редослед ставки у трансакцијама сортира у опадајући редослед према њиховом броју појављивања (овде А,Б,Ц,Д,Е) (у зависности од начина сортирања добиће се другачији изглед дрвета!)

Конструкција ФП-дрвета

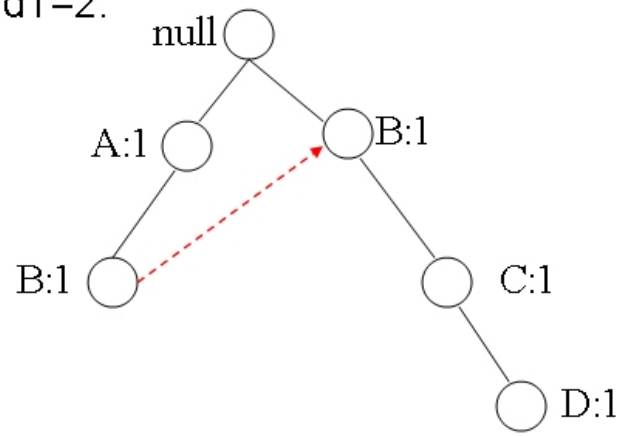
- На основу прве трансакције формирају се чворови означени са А и Б и уз њих се уписује тренутни број појављивања сваке ставке на тој путањи (овде 1)
- На основу наредне трансакције формира се нови скуп чворова (Б, Ц, Д) са одговарајућом граном. Уз сваки чвор се уписује 1 као број појављивања. Без обзира што у овој трансакцији постоји Б, формира се нова грана јер не постоји заједнички префикс са већ постојећим гранама

IdT	Stavke
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Posle čitanja IdT=1:



Posle čitanja IdT=2:



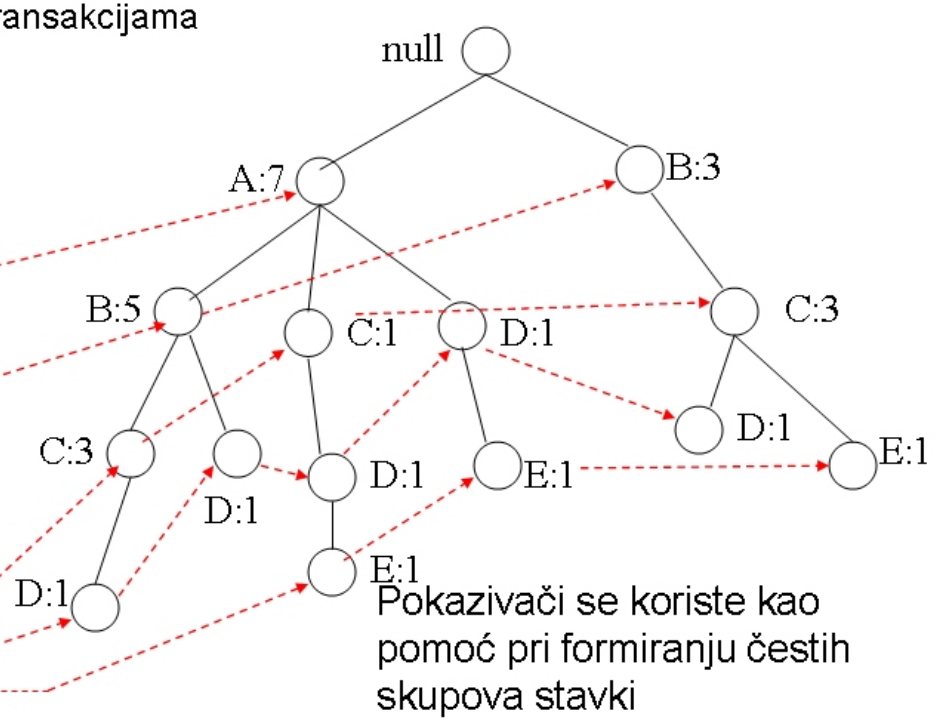
Конструкција ФП-дрвета

На крају конструкције добија се дрво приказано на слици. Формира се и табела са скупом показивача на листе истоимених ставки у различитим гранама дрвета.

IdT	Stavke
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Baza podataka sa transakcijama

Stavka	Pokazivač
A	
B	
C	
D	
E	



Да ли је решење исправно?

Конструкција ФП-дрвета - пример

Нека је дат скуп трансакција који садржи ставке a, b, c, d, и e.

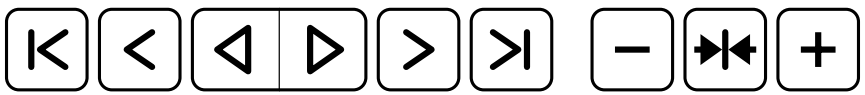
ИдТ	Ставке		Ставке
1	{a, b}		{a, b}
2	{d, b, c}		{b, c, d}
3	{a, d, e, c}		{a, c, d, e}
4	{e, d, a}		{a, d, e}
5	{a, b, c}		{a, b, c}
6	{a, c, d, b}		{a, b, c, d}
7	{a}		{a}
8	{b, a, c}		{a, b, c}
9	{a, b, d}		{a, b, d}
10	{e, c, b}		{b, c, e}

Први корак: пребројe се све појединачне ставке и редослед ставки у СВАКОЈ ОД ТРАНСАКЦИЈА СЕ СОРТИРА у опадајући редослед према њиховом броју појављивања (овде a, b, c, d, e). Број појављивања: a-8, b-7, c-6, d-5, e-3

Конструкција ФП-дрвета

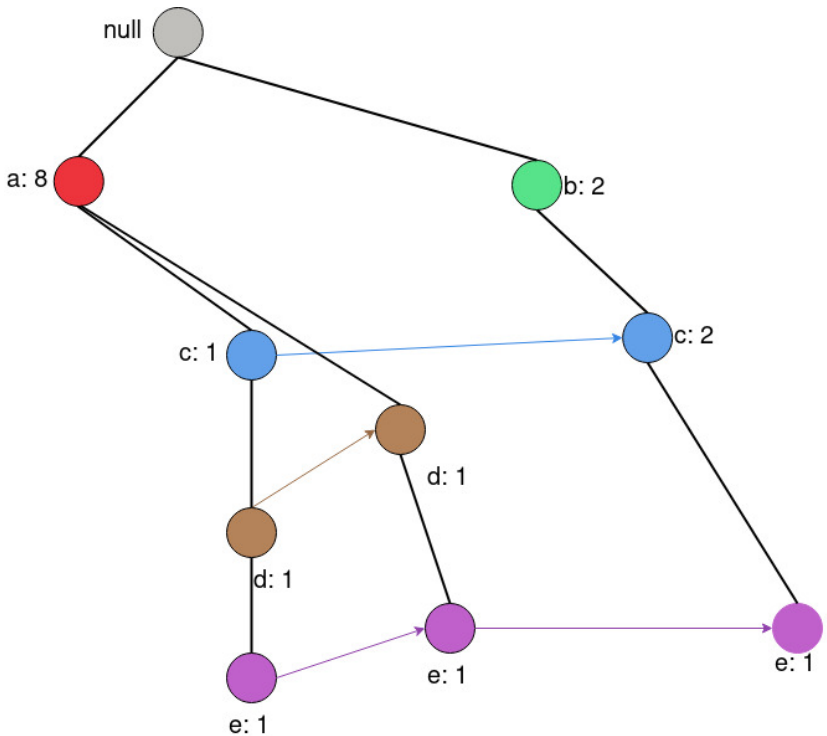
ИДТ	Ставке
1.	{a,b}
2.	{b,c,d}
3.	{a,c,d,e}
4.	{a,d,e}
5.	{a,b,c}
6.	{a,b,c,d}
7.	{a}
8.	{a,b,c}
9.	{a,b,d}
10.	{b,c,e}

Ставка	Показивач
a	
b	
c	
d	
e	



Одређивање честих скупова ставки

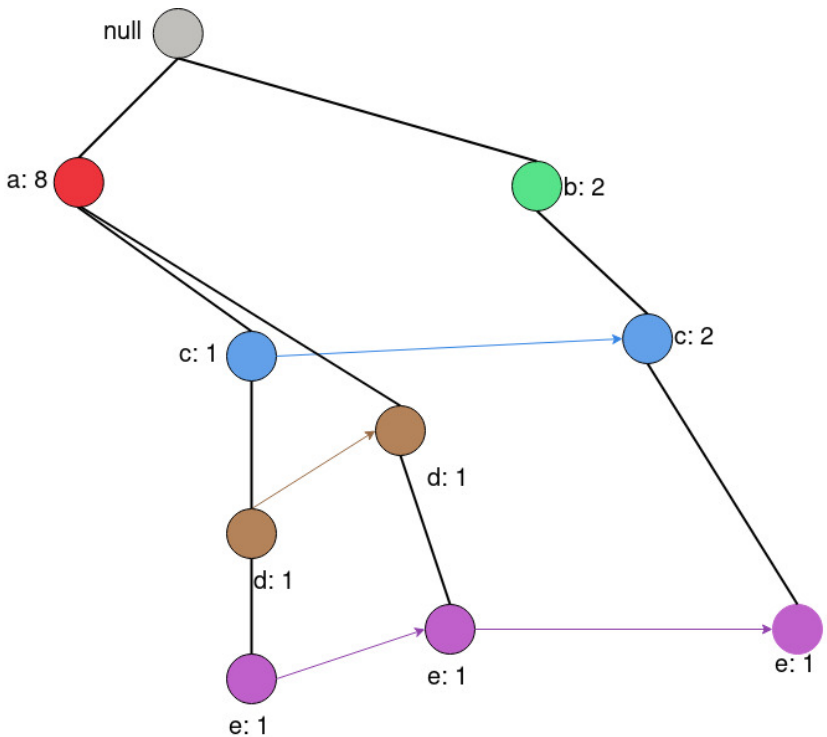
- Алгоритам ФП-раста одређује честе скупове ставки из ФП-дрвета методом одоздо навише
- Одређују се све честе ставке које се завршавају на e, затим оне које се завршавају на d, и редом на c, b и a
- До честих ставки које имају суфикс e доћи ће се испитивањем само оних грана које имају лист e.
- Поддрво са путањама које се завршавају на e је приказано на слици.



Одређивање честих скупова ставки

Декомпозиција на мање проблеме

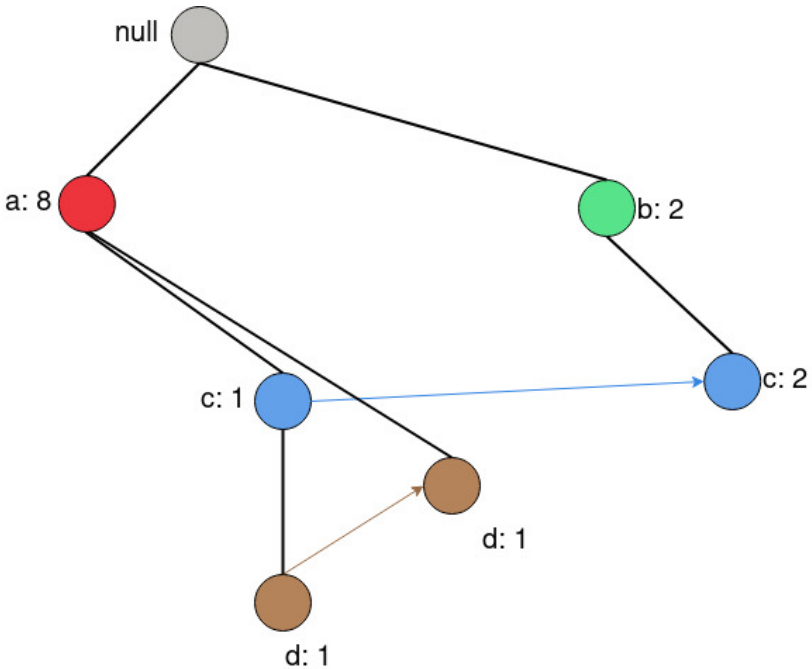
- Испита се да ли је често
- Ако јесте, одређују се честе ставке које се завршавају на *de*, и редом на *ce*, *be* и *ae*
- Унијом добијених скупова добијају се све честе ставке које се које имају суфикс *e*
- Испит поступак се спроводи за ставке са суфиксом *d*, *c*, *b*, и *a*, и унија добијених скупова представља скуп свих честих ставки

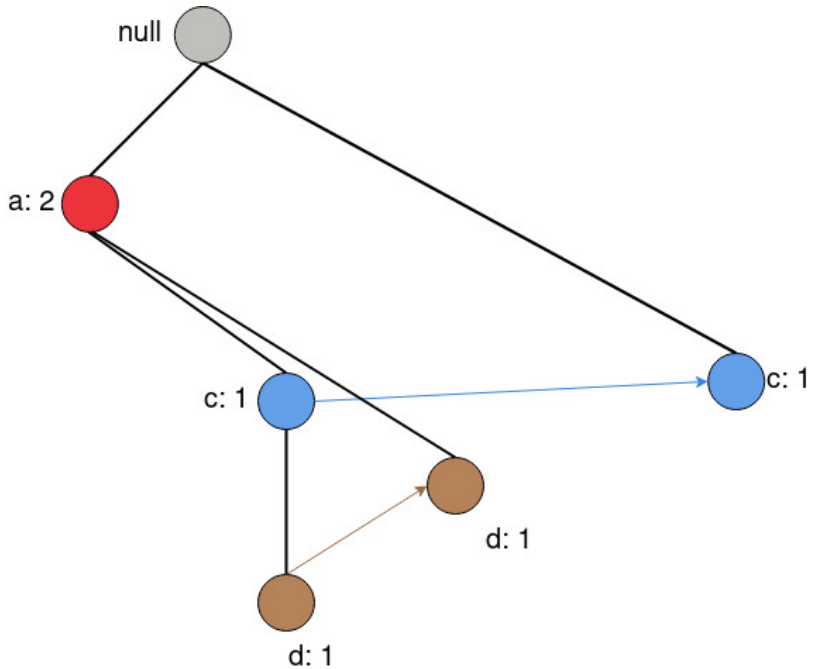


Одређивање честих скупова ставки

e је често (праг је 2, пребројавање броја појављивања ставке по листи из табеле заглавља)

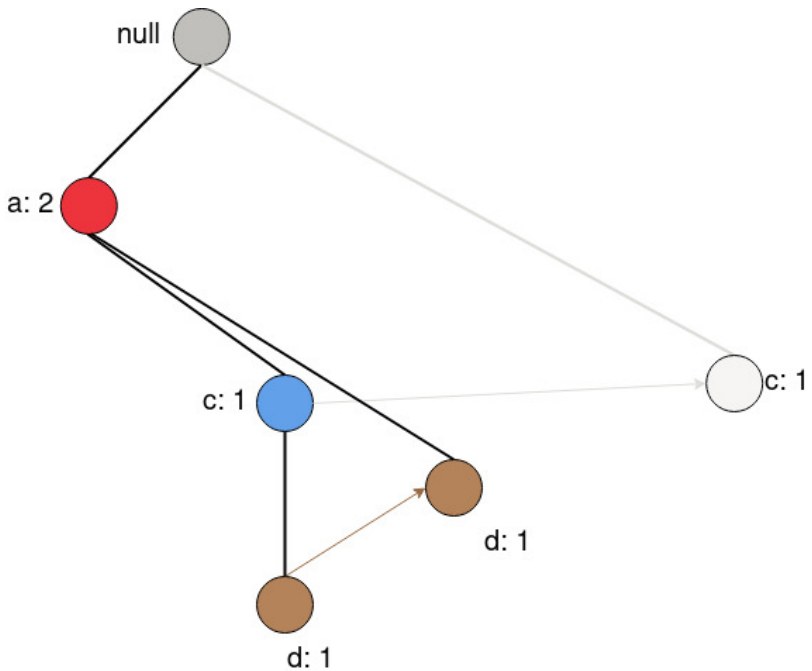
- Траже се честе ставке које се завршавају на *e*: *de*, *ce*, *be*, и *ae*
- Елиминишу се чворови који садрже *e* и изврши ажурирање броја појављивања ставки на вишим гранама дрвета (нпр. смањује се за 1 број појављивања b и c јер представљају трансакцију $\{b, c\}$ која не садржи e (иницијално део TID 2)
- Елиминишу се чворови који имају број појављивања у поддрвету мањи од прага (чвор *b*) и добија *условно* ФП-дрво за *e* (десна слика)

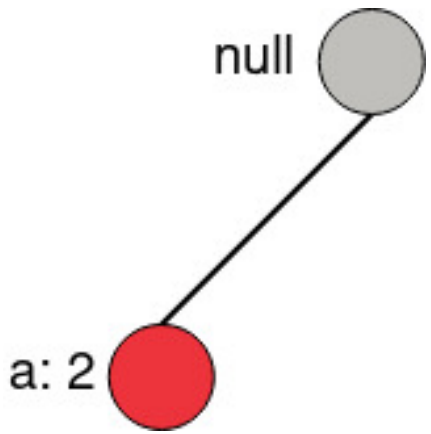




Одређивање честих скупова ставки

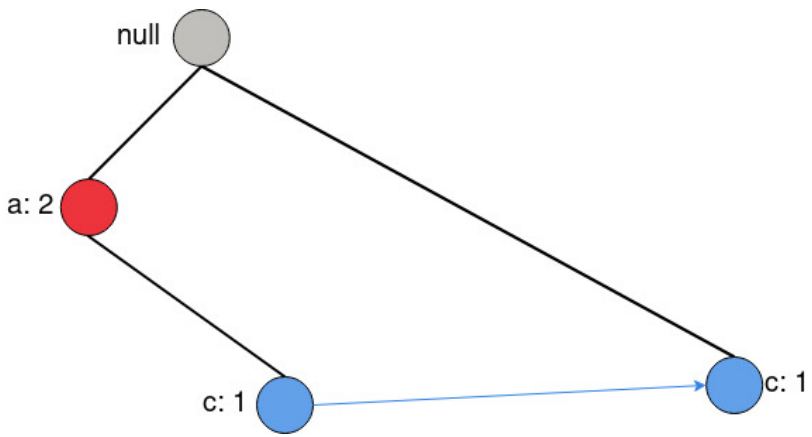
- Из условног ФП-дрвета за e добија се да је $\{d, e\}$ честа ставка (праг је 2!)
- Аналогно се формира условно ФП-дрво за de (на десној страни) које садржи само ставку (a) са бројем појављивања које је веће од прага, одакле се изводи закључак да је $\{a, d, e\}$ честа ставка.

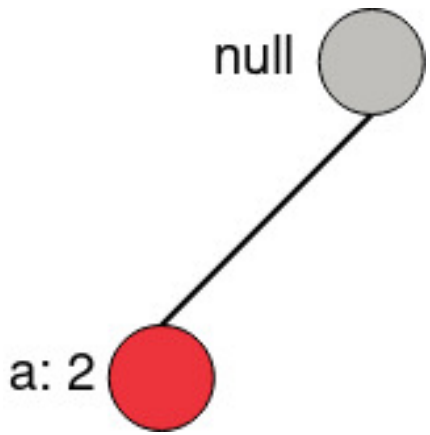




Одређивање честих скупова ставки

Наредни корак је одређивање честих ставки које се завршавају на c, e. Алгоритам ФП-раста се примењује аналогно и формира се дрво са префиксима које се завршава на ce одакле се види да је једина честа ставка $\{c, e\}$. На основу истог поступка види се да је честа и ставка $\{a, e\}$ (десна слика). Алгоритмом би се затим одредиле све честе ставке које се завршавају на d, итд.





Карактеристике алгоритама у SPSS Modeler V18.4 и Python-у

Видети

- IBM SPSS Modeler 18.2 Algorithms Guide
- Python ?