Asszociációs szabályok keresése

Csima Judit

BME, VIK,
Számítástudományi és Információelméleti Tanszék

2014. április 23.

Alapfeladat

- adottak vásárlói kosarak (tranzakciók): miket vásároltak együtt
- cél: olyan szabályokat felállítani, hogy ha valaki vesz X-et, akkor esélyes, hogy vesz Y-t is
- X és Y lehet több elemből álló halmaz is
- egy ilyen szabály nem jelent ok-okozati összefüggést!
- de egy ilyen szabályból hasznot lehet húzni: pl. árazzuk le X-et kicsit, emeljük meg Y árát jobban

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

{Diaper} $\rightarrow$ {Beer},
{Milk, Bread} $\rightarrow$ {Eggs,Coke},
{Beer, Bread} $\rightarrow$ {Milk},

Implication means co-occurrence, not causality!

Jelölések, alapfogalmak

- elem (item): amit lehet venni, pl. tej, pelenka
- tranzakció: amiket együtt vettek, egy vásárlói kosár
- egyszerű modell: darabszám nem számít, csak az, hogy szerepel-e egy adott termék a kosárban
- cél: $X \to Y$ szabályok találása, ahol X, Y nemüres, diszjunkt elemhalmazok ($X \cap Y = \emptyset$)
- elnevezés: ha $|X| = k$, akkor X-et k-elemű elemhalmaznak (k-item set) hívjuk

Mikor jó egy szabály?

- ha X és Y sok tranzakcióban szerepel együtt (különben nem érdekes, pl. Hello Kittys papucs és motorosfűrész)
- ha az X-et tartalmazó kosarak jelentős része tartalmaz Y-t is
- lesz még valami más is, de először nézzük ezeket

Támogatottság, support

- $X \to Y$ abszolút támogatottsága (support count): hány kosárban van X és Y is, jele $\sigma(X \cup Y)$
- $X \to Y$ támogatottsága (support):
  $$supp(X \to Y) = \frac{\sigma(X \cup Y)}{number\ of\ transactions}$$
- csak olyan szabályokat akarunk, amikre a supp elég nagy (egy küszöbnél nagyobb, a küszöb neve min_sup)
- azért, mert ha kicsi a support, akkor arra nem lehet stratégiát építeni, az lehet, hogy véletlen egybeesés (Hello Kitty és fűrész)

Támogatottság, support

- $supp(X \rightarrow Y)$ csak $X \cup Y$-tól függ, attól nem, hogy X és Y hogy oszlik el a szabály két oldalára
- ha $supp(X \rightarrow Y)$ a küszöbnél nagyobb, akkor $X \cup Y$ neve gyakori elemhalmaz (frequent item set)
- az persze kérdés, hogy mi a küszöb ....

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Megbízhatóság, confidence

- $X \to Y$ megbízhatósága (confidence): $conf(X \to Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$
- azaz: az X-et tartalmazó kosarak mekkora részében van Y is
- az a szabály érdekes, aminél a conf egy küszöbnél (jele min_conf) nagyobb
- ez mutatja, hogy X eladásai befolyásolhatják Y eladásait

Definition: Association Rule

- **Association Rule**
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example:
    {Milk, Diaper} $\rightarrow$ {Beer}

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- **Rule Evaluation Metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Szabályok keresése

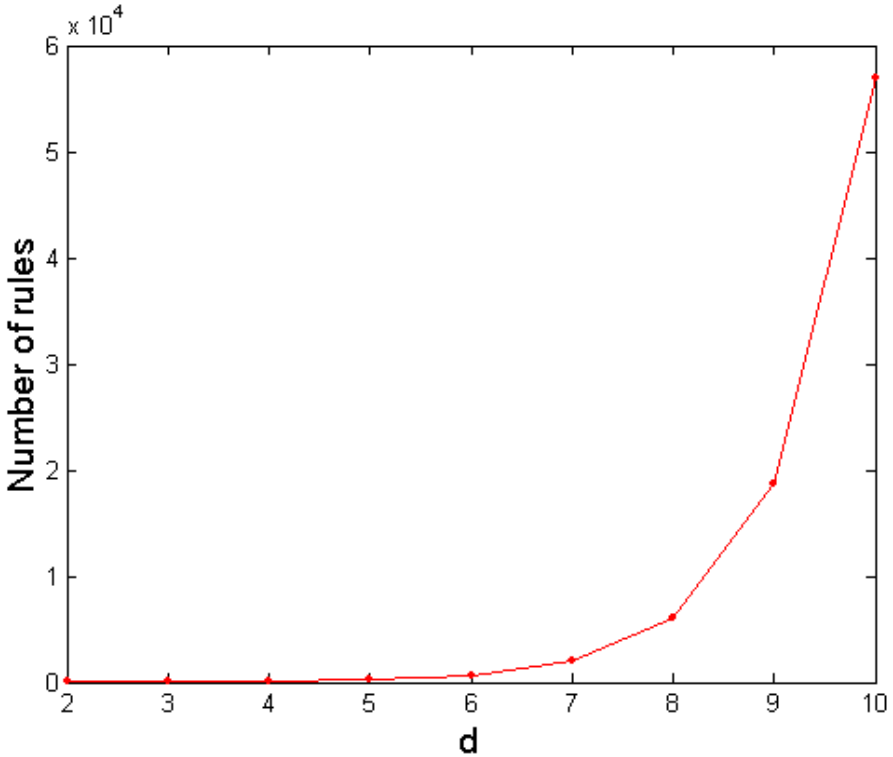
- olyan szabály kell, ahol $supp \geq min\_sup$ és $conf \geq min\_conf$
- supp mindig kisebb, mint conf egy adott szabály esetén
- két külön küszöb van, ez két külön feltétel
- egy szabály el tud bukni mindkét feltételen

Brute-force módszer

- minden olyan X és Y elemhalmaz végignézése, amikre $X \cap Y = \emptyset$
- minden ilyenre supp és conf számolása, rosszak kidobása
- ez sajnos túl sok: nagyjából $\sum_{k=1}^{d-1} \binom{d}{k} 2^{d-k}$, durván exponenciális (ahol d darab lehetséges item van)

Computational Complexity

- Given d unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:



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

$$= 3^d - 2^{d+1} + 1$$

If d=6, R = 602 rules

Észrevétel

- egy $X \to Y$ szabály akkor jó, ha elég nagy a supp és a conf is
- supp csak $X \cup Y$-tól függ, először ezt a lécet kell megugrania a potenciális szabálynak
- ha $supp(Z = X \cup Y)$ elég nagy, akkor jön az, hogy hogyan legyen Z szétosztva a szabály két oldalára, hogy a conf is elég nagy legyen
- válasszuk szét a két ellenőrzést:
  - először keressünk gyakori elemhalmazokat (Z), csak ilyenekből lehet jó szabály
  - nézzük meg, hogy egy gyakori elemhalmazból milyen nagy megbízhatóságú szabály gyártható le

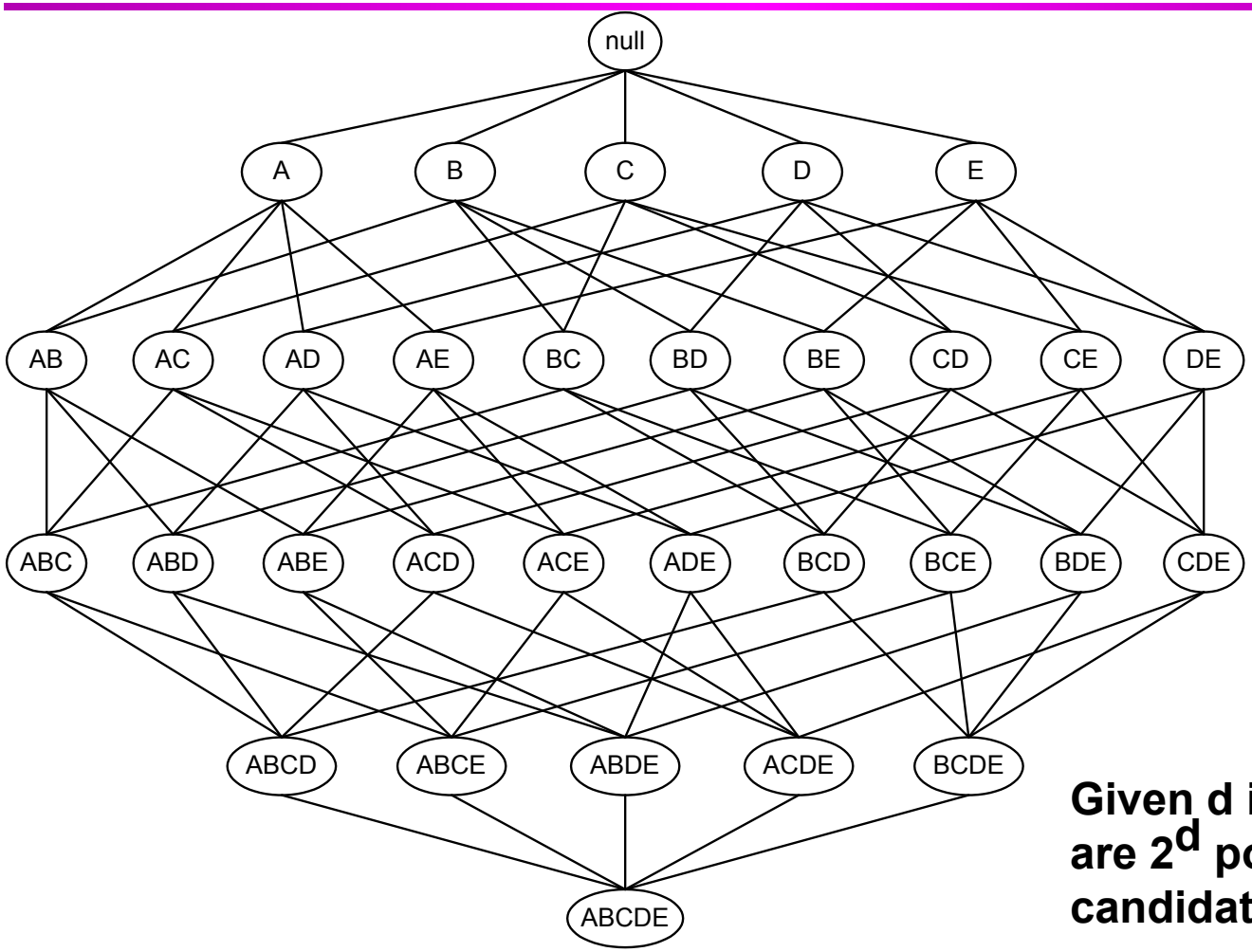
Általános algo

- először legenerálom az összes gyakori elemhalmazt (adott a *min_sup*)
- ezután minden egyes gyakori elemhalmazból megcsinálom a nagy megbízhatóságú szabályokat

Brute-force módszer gyakori elemhalmazok keresésére

- minden nem-üres részhalmazt végignézek
- ez nem jó, túl sok van: $2^d - 1$
- észrevétel: a részhalmazok háló-struktúrát alkotnak
- sőt: ha M jelölt van a gyakori halmazra és N tranzakció, akkor minden jelöltet össze kell vetni minden tranzakcióval (benne van-e a jelölt az adott kosárban)
- ez $O(NMw)$, ahol w a tranzakciók nagysága (hány elem van benne) és már csak M maga $2^d - 1$ a brute-force esetben

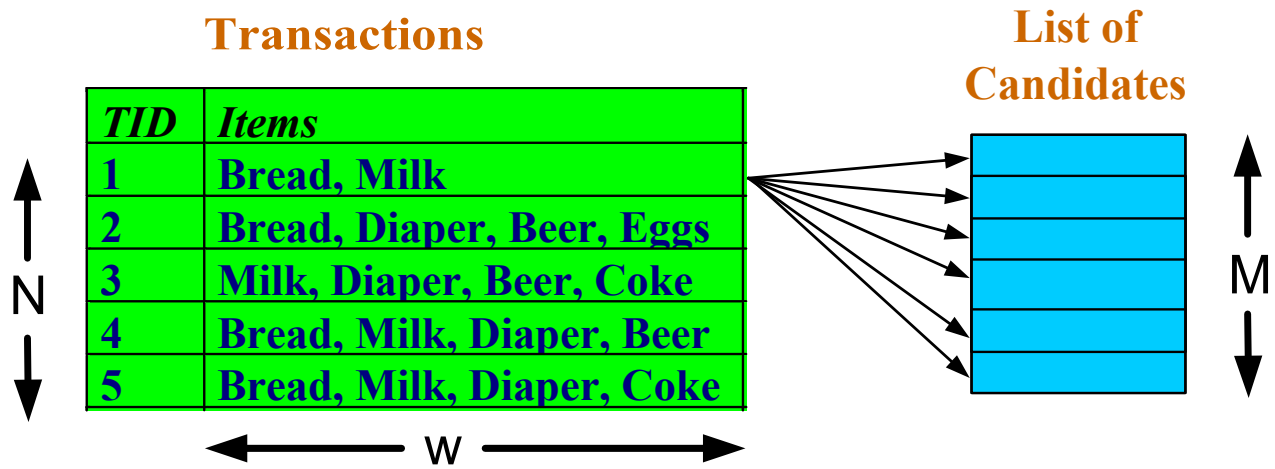
Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Hogyan lehetne gyorsítani?

- csökkentsük M-et: ne az összes részhalmazt nézzük ész nélkül, hanem szűrjünk valahogy mielőtt elkezdjk őket összevetni a tranzakciókkal
- csökkentsük N-t (a tranzakciók számát) vagy a hosszukat (w-t)
- használjunk valami ügyes adatszerkezetet a jelöltek és a tranzakciók összevetésére
- most először az első lehetőséget nézzük: M csökkentése

Jelöltek számának csökkentése

- cél: gyakori elemhalmazok keresése
- hogyan: a részhalmazokból álló hálót úgy bejárni, hogy minél több elemhalmazt ki tudjunk zárni, minél előbb
- követelmények:
  - minden gyakorit generáljunk végül
  - egyet csak egyszer
  - ne dolgozzunk túl sokat a generálás során
- egyszerűsítés: item-ek neve helyett számokat használunk

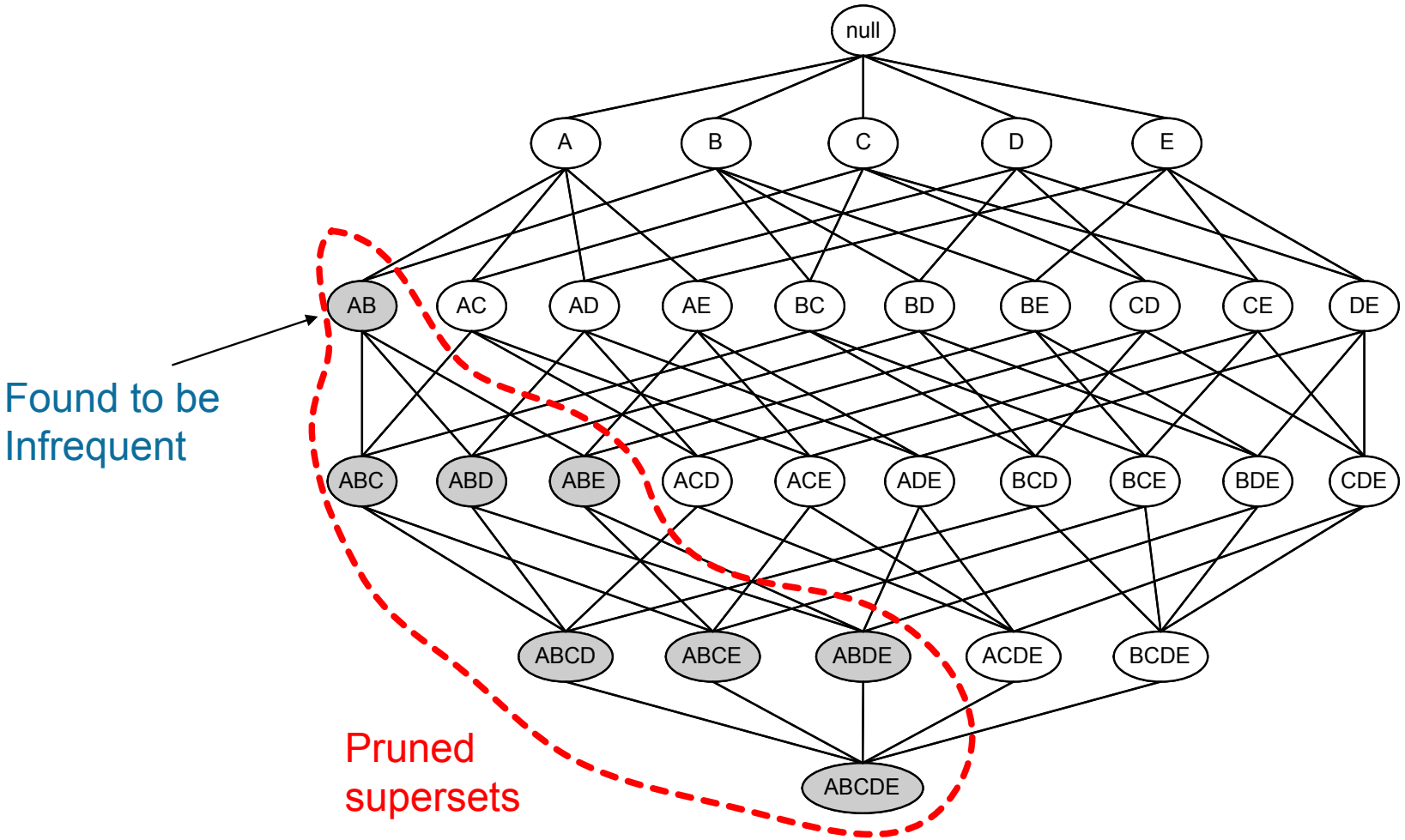
Apriori-elv

- Apriori-elv: ha X gyakori, akkor minden részhalmaza gyakori
- mert ha $Y \subseteq X$, akkor $supp(Y) = \frac{\sigma(Y)}{N} \geq \frac{\sigma(X)}{N} = supp(X)$ (itt N a tranzakciók száma)
- ugyanez máshogy: ha Y nem gyakori, akkor senki se gyakori, aki Y-t tartalmazza
- ezt úgy is szokták mondani, hogy a support függvény anti-monoton

Apriori algo

- haladjunk k szerint növően $k = 1$-től
- ha egy k elemű elemhalmaz nem gyakori, akkor minden nála bővebb elemhalmaz kizárható (infrequent)
- egy k elemű halmaz csak akkor lehet gyakori, ha minden $k - 1$ elemű részhalmaza gyakori

Illustrating Apriori Principle



Apriori algo

1. egyszer végignézek minden tranzakciót és kigyűjtöm az egy-elemű gyakoriakat (ehhez minden x elemre kiszámolom $\sigma(x)$-et), ez az F_1 halmaz
2. k $= 2$
   - $C_2 = 2$-elemű esélyesek: akiknek mindkét tagja F_1-ben van
   - $F_2 = C_2$-beli jelöltek összevetése a tranzakciókkal, a gyakoriak F_2
3. k $= 3$
   - $C_3 = 3$-elemű esélyesek: akiknek minden kételemű részhalmaza F_2-ben van
   - $F_3 = C_3$-beli jelöltek összevetése a tranzakciókkal, a gyakoriak F_3
4. k általában
   - $C_k =$ k-elemű esélyesek: akiknek minden $k-1$-elemű részhalmaza F_{k-1}-ben van
   - $F_k = C_k$-beli jelöltek összevetése a tranzakciókkal, a gyakoriak F_k

Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,
$^{6}C_{1} + {}^{6}C_{2} + {}^{6}C_{3} = 41$
With support-based pruning,
$6 + 6 + 1 = 13$

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

Észrevételek

- úgy könnyű F_{k-1}-ből C_k képzése, ha az elemhalmazokban növekvő sorrendben vannak
- mert ekkor könnyű C_k-ba tartozó k-elemű jelölteket előállítani
- úgy, hogy két olyan (rendezett) $(k-1)$-eleműt keresek F_{k-1}-ben, amiknek az első $k-2$ tagja ugyanaz
- így biztos, hogy minden k elemű gyakori bekerül C_k-ba, pontosan egyszer
- nem kell azzal foglalkozni, hogy C_k-ból kiszűrjük a duplikátumokat

Hogyan lesz tehát F_k F_{k-1}-ből?

- F_{k-1}-ben minden elemhalmazban rendezetten vannak az elemek
- két F_{k-1}-beli $k-1$ elemű elemhalmazból akkor csinálok egy k elemű jelöltet C_k-ba, ha
  - az első $k-2$ tagjuk ugyanaz
  - az így létrejött k elemű elemhalmaz többi $k-1$ elemű részhalmaza is F_{k-1}-ben van (ez még $k-2$ ellenőrzés)
- az így kapott C_k minden elemhalmazát összevetjük minden tranzakcióval (ténylegesen leszámoljuk a σ-kat)

További gyorsítás

- láttuk, hogy a nagy meló a jelöltek és a tranzakciók összevetése (az egyes jelöltek előfordulásainak kiszámolásához)
- eddig: mielőtt összevetjük a jelölteket a tranzakciókkal, csökkentsük a jelöltek számát (minden k elemű részhalmaz helyett csak C_k-beliek)
- további lehetőségek:
  - csökkentsük a tranzakciók hosszát: a nem gyakori egy eleműeket dobjuk ki az elején minden tranzakcióból
  - csökkentsük a tranzakciók számát: F_k előállítása közben, akkor dobjunk ki minden k-nál nem hosszabb tranzakciót

Tranzakciók és jelöltek ügyes összehasonlítása

- mikor? amikor már C_k elkészült és össze kell vetnem minden k-hosszú, C_k-beli jelöltet minden tranzakcióval, hogy benne van-e
- minden jelölt rendezetten tartalmazza az elemeit
- minden jelöltet összehasonlítok minden tranzakció minden k-elemű részhalmazával
- alapötlet: vödrös hash

Vödrös hash az összevetésre

- a jelölteket valahogyan belehash-elem egy táblába, az egyes vödrökben a jelöltek rendezetlenül vannak
- végigmegyek minden tranzakció minden k-elemű részhalmazán és az adott részhalmazt csak azokkal a jelöltekkel vetem össze, akikkel egy vödörbe kerülne
- egy konkrét megvalósítás $\leq k$ szintű fát és a mod 3 függvényt használva:
  - az i. szinten a jelölt i. elemének 3-mal vett osztási maradéka alapján megyek valamerre
  - szétszedem a vödröt, ha tudom, amennyiben 4-nél több jelölt lenne benne

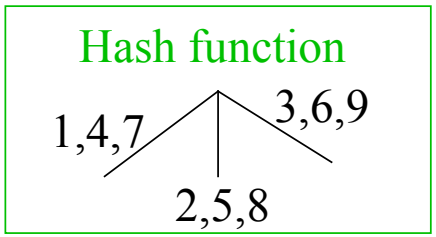
Generate Hash Tree

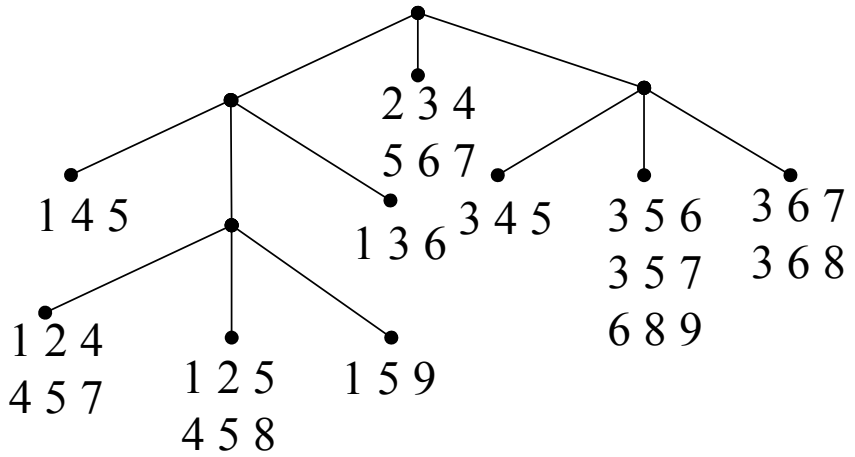
Suppose you have 15 candidate itemsets of length 3:

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

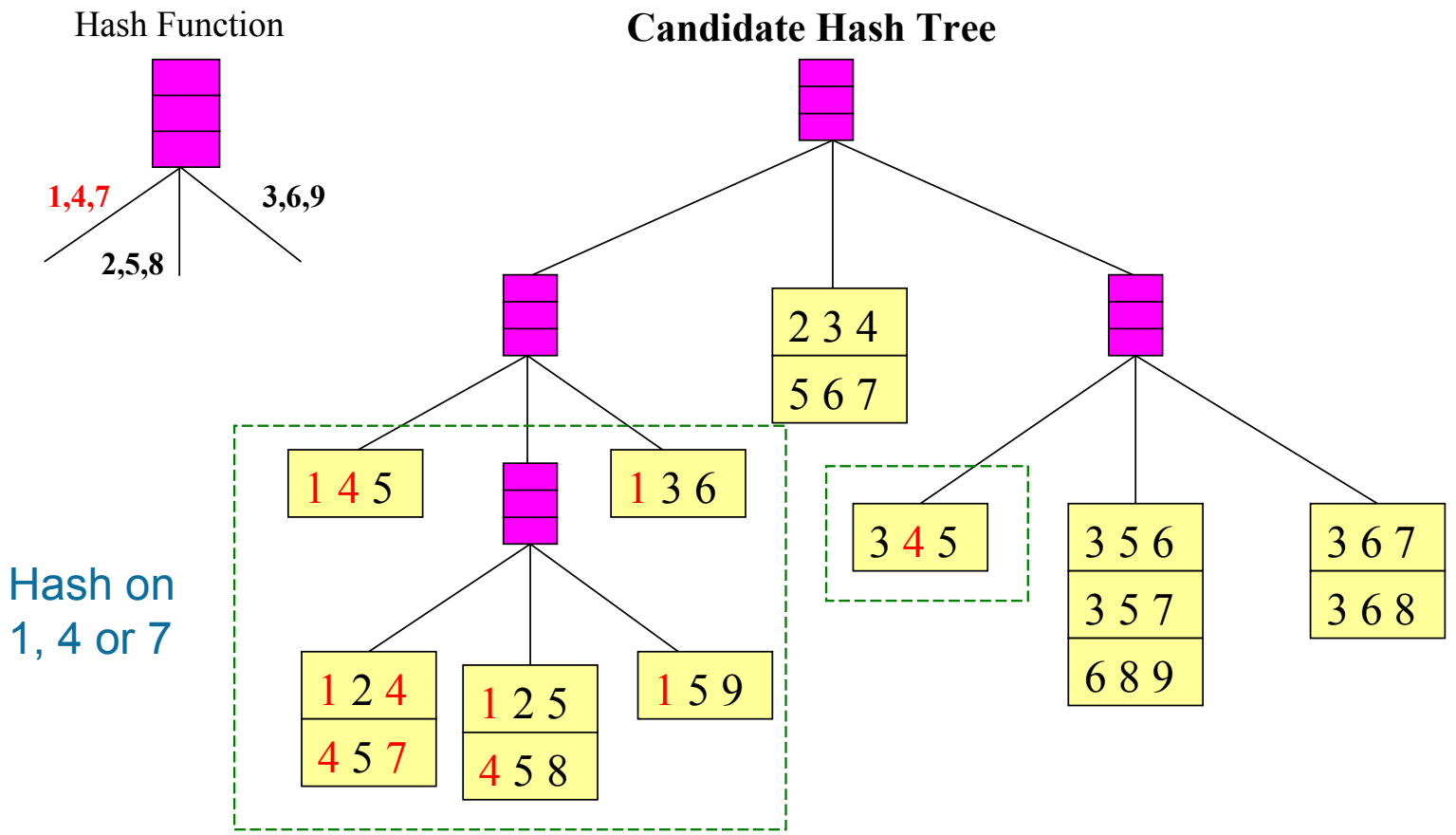
You need:

- **Hash function**
- **Max leaf size: max number of itemsets stored in a leaf node (if number of candidate itemsets exceeds max leaf size, split the node)**

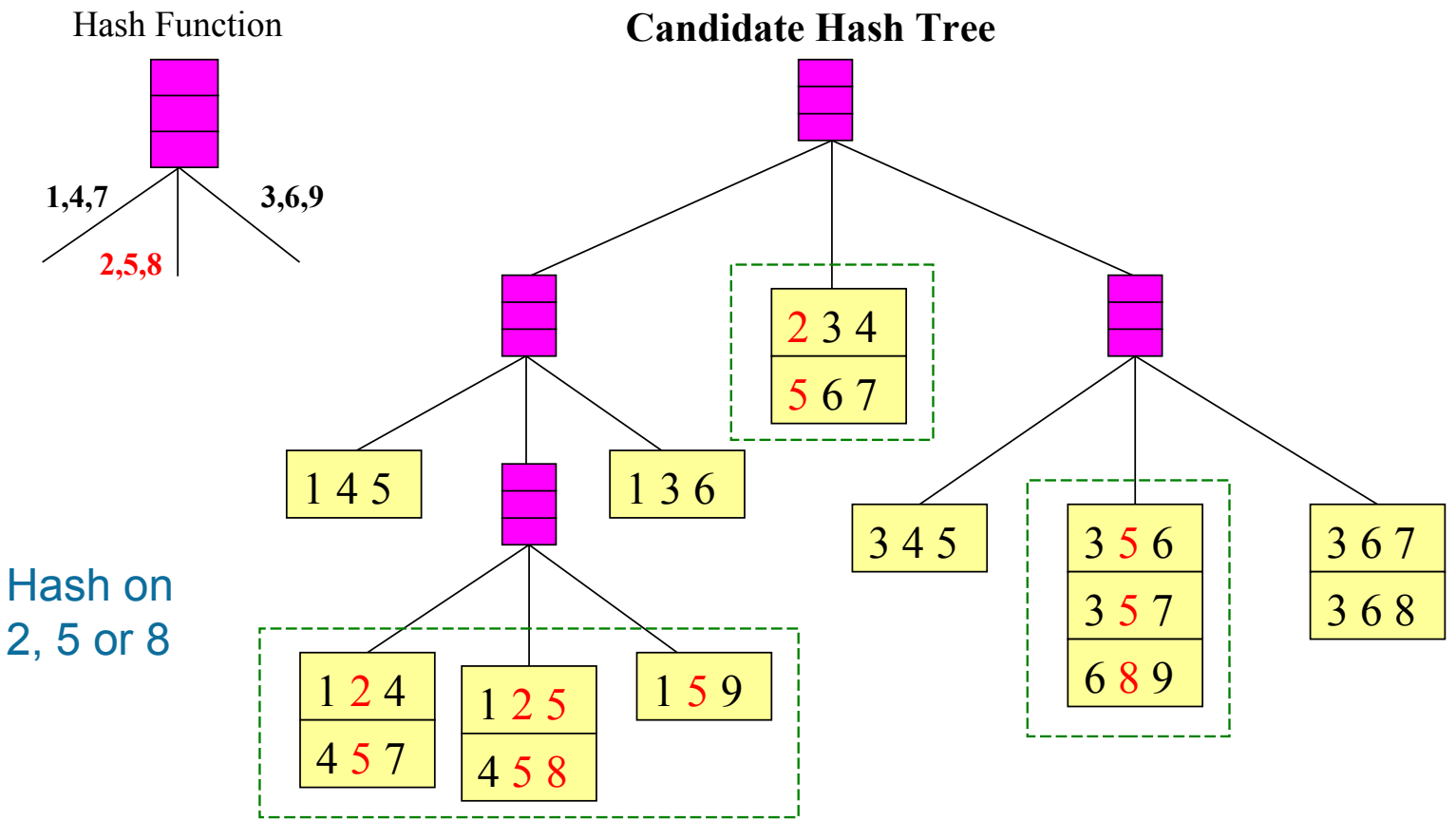




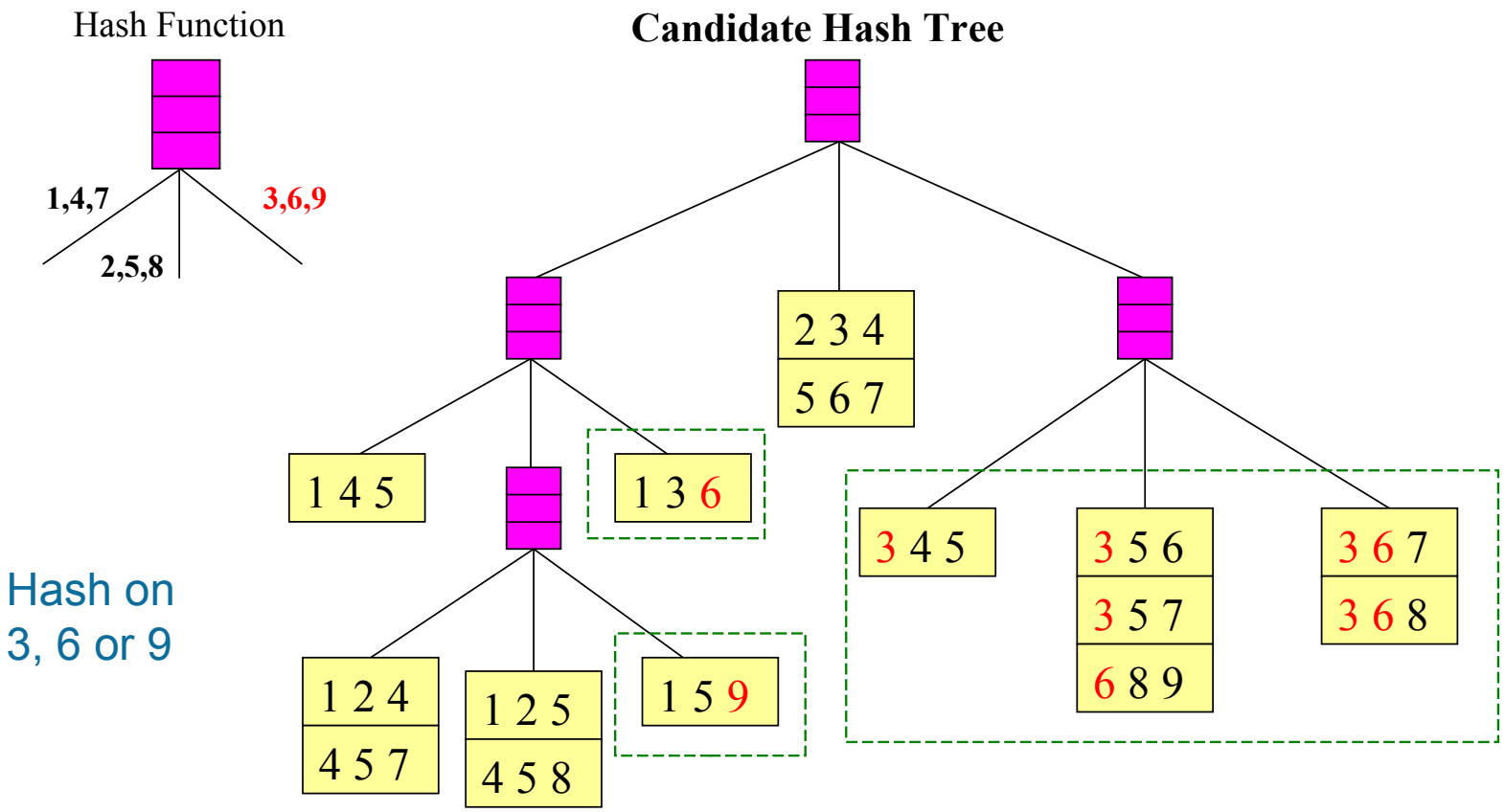
Association Rule Discovery: Hash tree



Association Rule Discovery: Hash tree



Association Rule Discovery: Hash tree

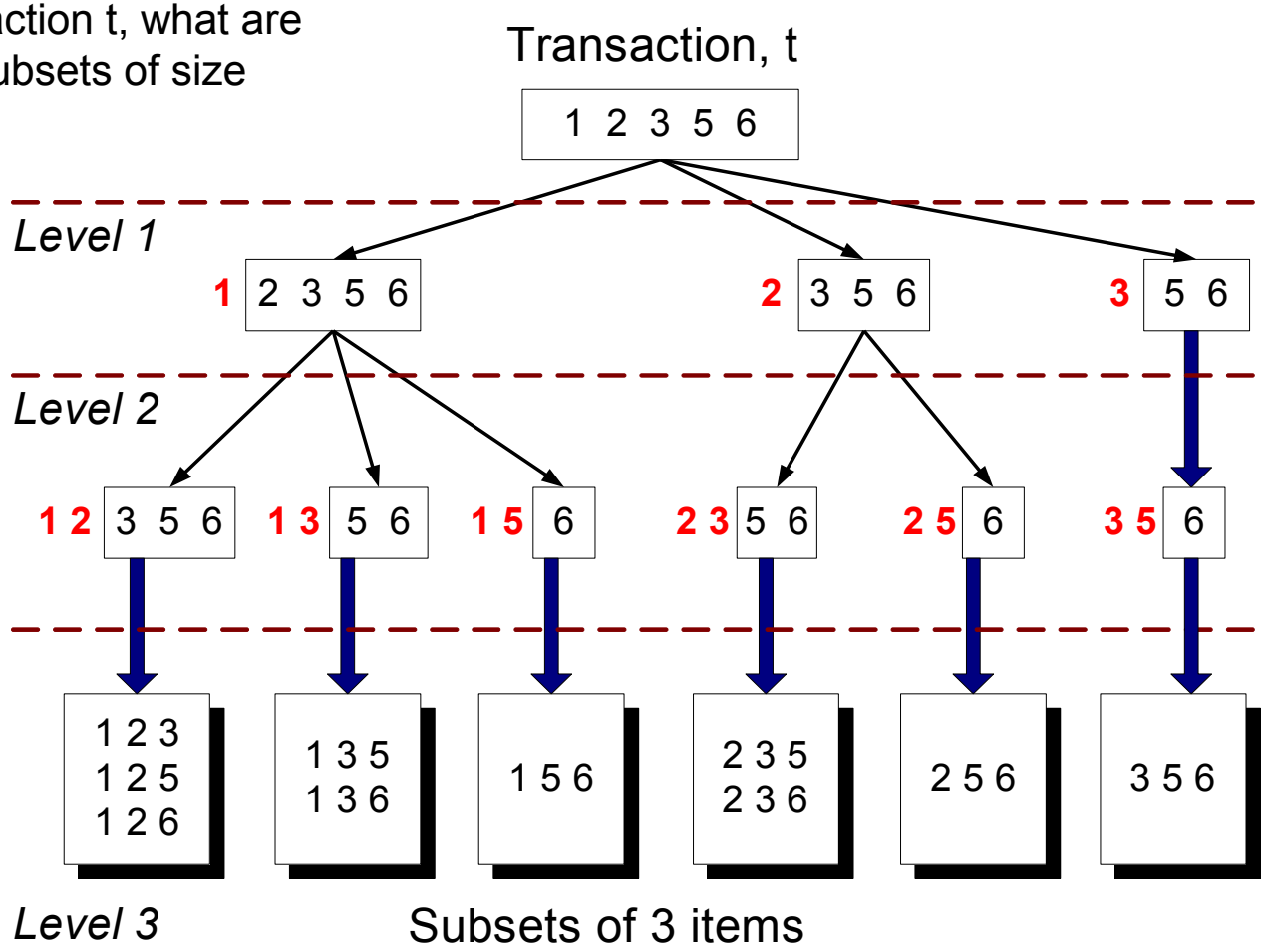


Tranzakciók feldolgozása összevetéshez

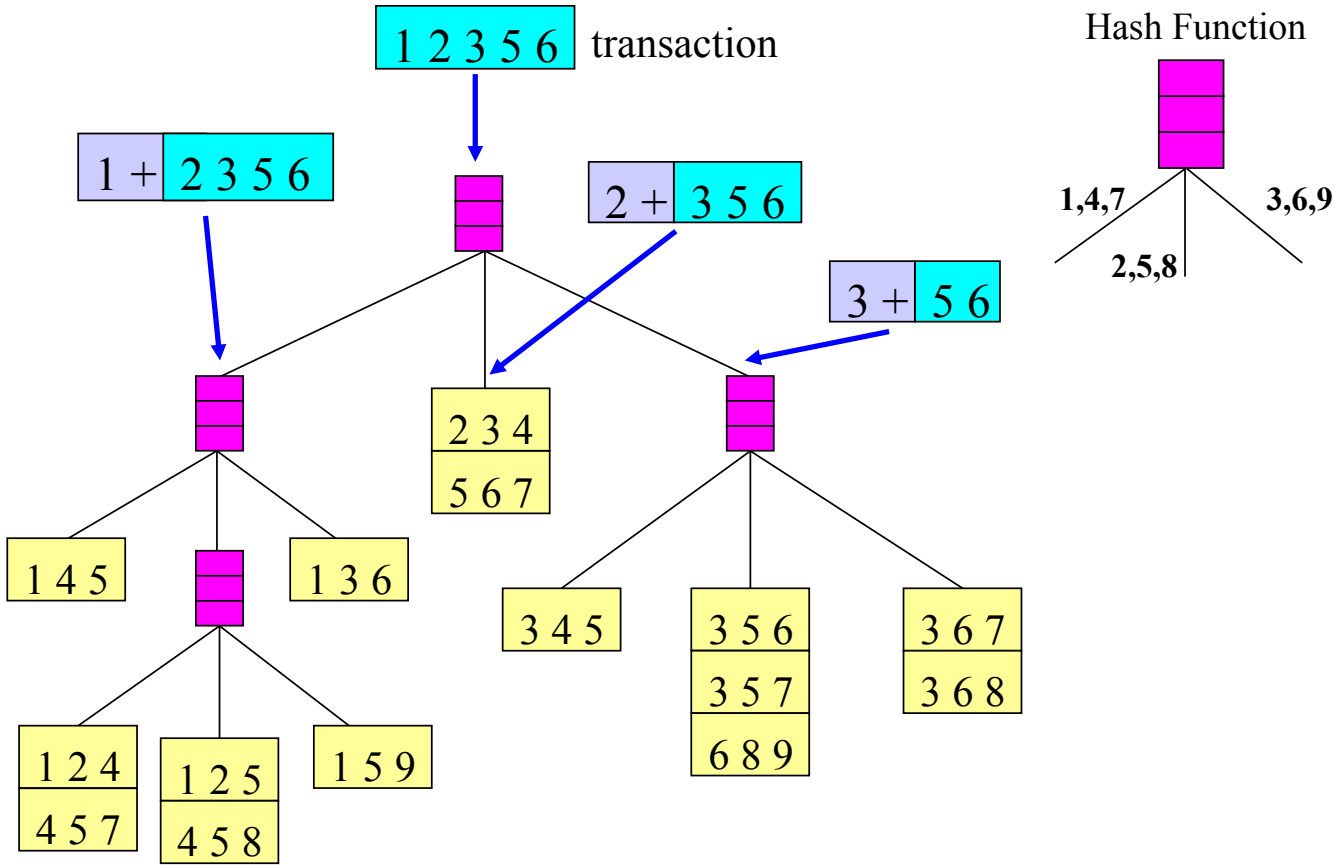
- egy adott tranzakcióra meghatározom az összes lehetséges k-elemű részhalmazt (ezeken belül sorrendben tartva az elemeket)
- ezeket a jelölteknek megfelelő struktúrában generálom (melyik vödörbe esnének?)
- minden vödörnél megnézem, hogy az oda eső jelöltek között van-e olyan, akihez passzol az adott részhalmaz

Subset Operation

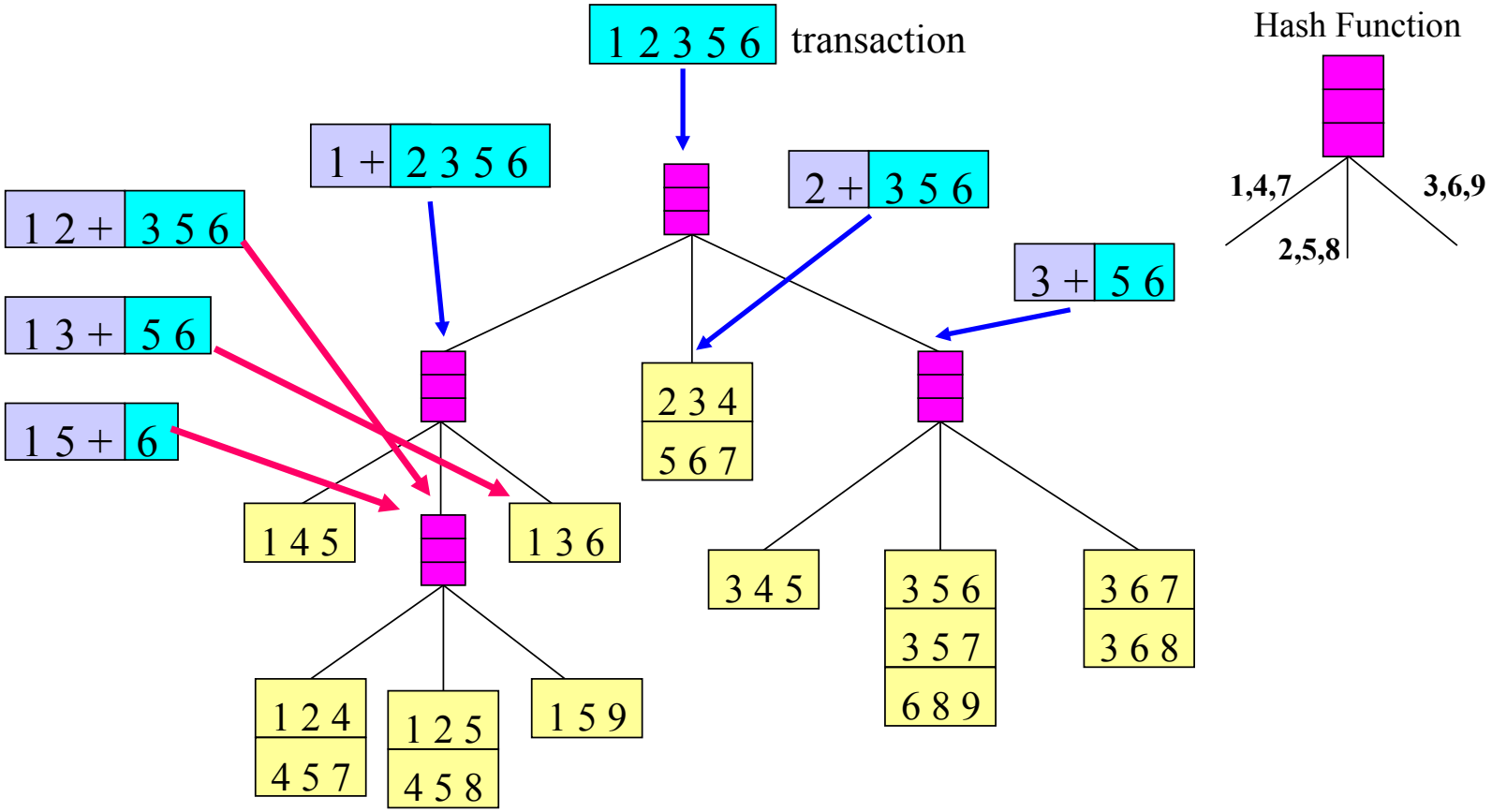
Given a transaction t, what are the possible subsets of size 3?



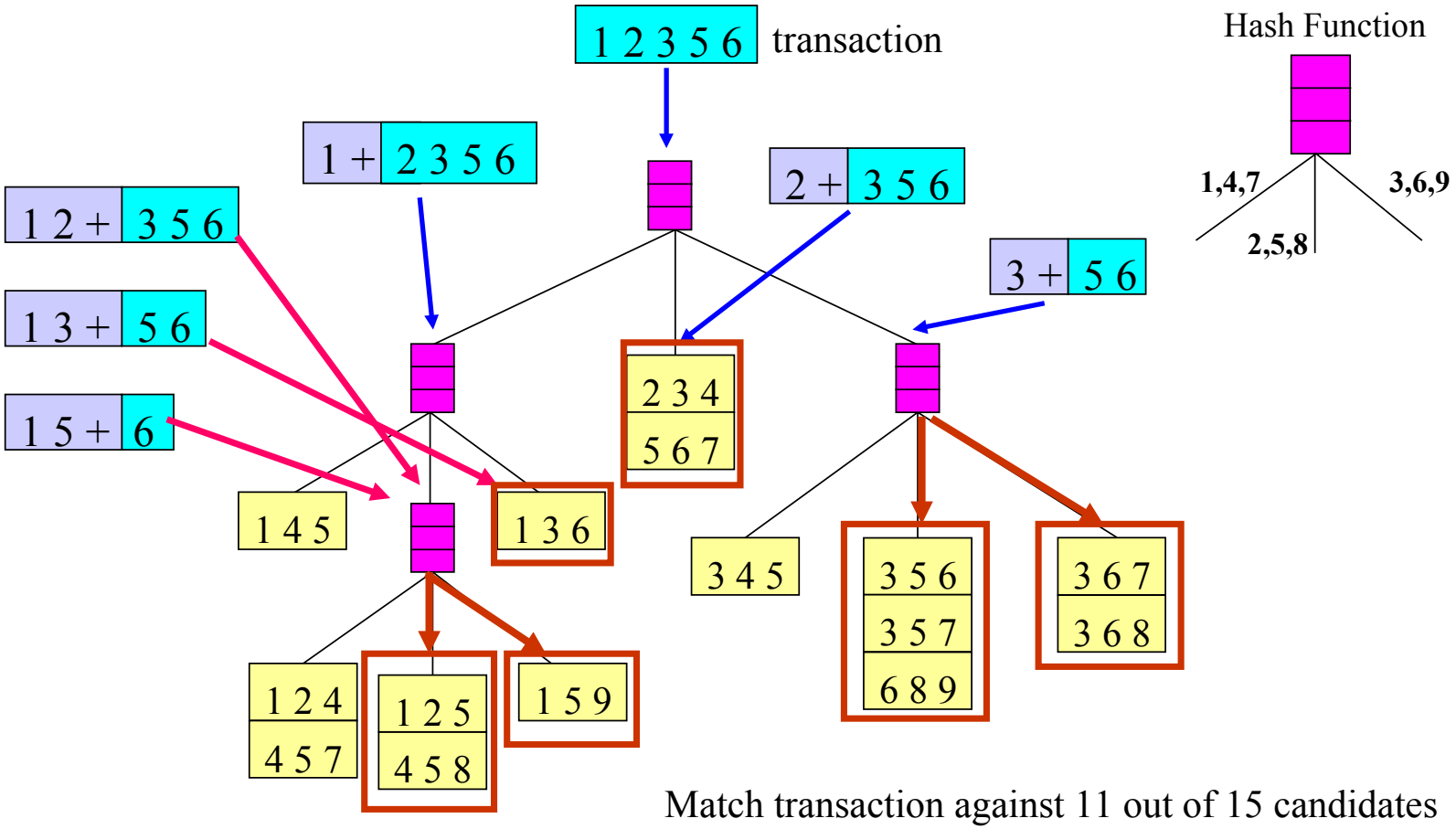
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Másik megoldás: szófa

- a jelölteket szófában tartjuk
- a tranzakciók k-elemű részhalmazait ebben keressük
- plusz haszon: C_{k-1}-es szófából kapott F_{k-1}-nek megfelelő szófán látszik, hogy kik lesznek esélyesek C_k-ba kerülésre: ahol az utolsó szinten van elágazás

Szaályok generálása gyakori elemhalmazokból

- tegyük fel, hogy megvannak a gyakori elemhalmazok
- minden Z gyakori elemhalmazból le szeretnénk generálni az összes olyan $X \to Y$ szabályt, ahol
  - $Z = X \cup Y$, X és Y sem üres
  - $supp(X \to Y) \geq min\_sup$
  - $conf(X \to Y) \geq min\_conf$
- a *min_sup*-os dolog Z gyakorisága miatt megvan
- a *conf*-os feltételt kéne teljesíteni

Brute-force algo

- adott Z esetén minden lehetséges módon $X, Z \setminus X$ kiválasztása
- minden választásra $conf(X \rightarrow Z \setminus X)$ számolása
- ehhez $\sigma(X)$ kell
- de $2^{|Z|}$ -2 lehetőség van X-re, ez túl sok

Észrevétel

Ha adott egy Z és ennek egy X részhalmazából, mint baloldalból származtatott szabály nem jó (conf-ja kisebb, mint *min_conf*), akkor az összes olyan X' baloldalból se lesz jó szabály, ahol $X' \subseteq X$.

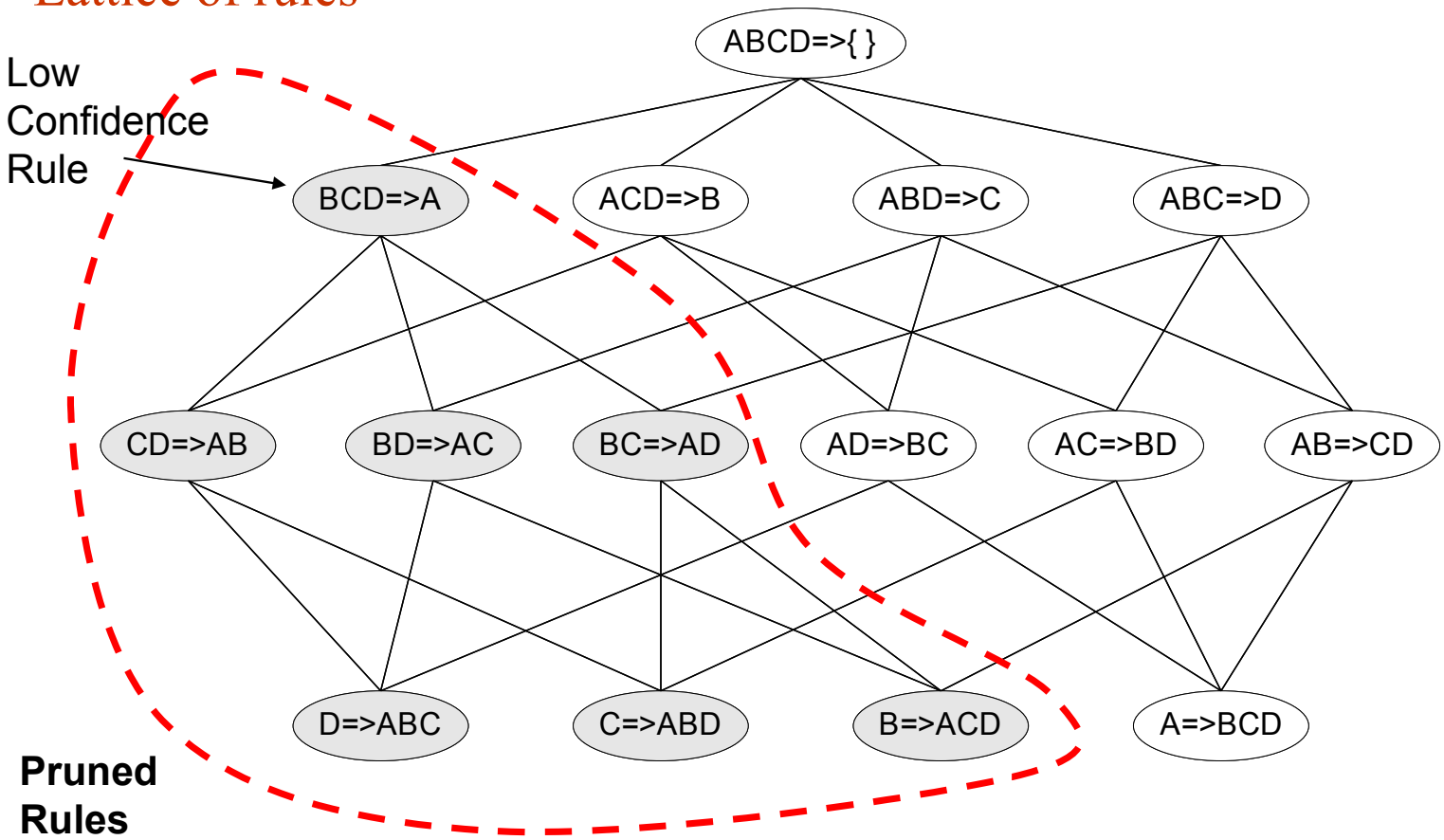
Biz.
$conf(X' \to Z \setminus X') = \frac{\sigma(Z)}{\sigma(X')} \leq \frac{\sigma(Z)}{\sigma(X)} < min\_conf$
A középen álló egyenlőtlenség azért igaz, mert $X' \subseteq X$ miatt $\sigma(X') \geq \sigma(X)$.

Észrevétel másként

- ha egy adott Z-ből generálok szabályokat és egy Y jobboldalú szabály rossz, akkor minden olyan szabály is rossz, ahol a jobboldal Y-nál bővebb
- ez hasonló az Apriori-elvhez
- csináljuk ugyanazt, amit az Apriori-algoban:
  - adott Z esetén először legeneráljuk az 1-elemű jobboldalú jó szabályokat
  - növeljük a szabályok jobboldalának hosszát, csak olyan jobboldalak jönnek be, amiknek minden eggyel kisebb részhalmazához tartozó szabály jó volt

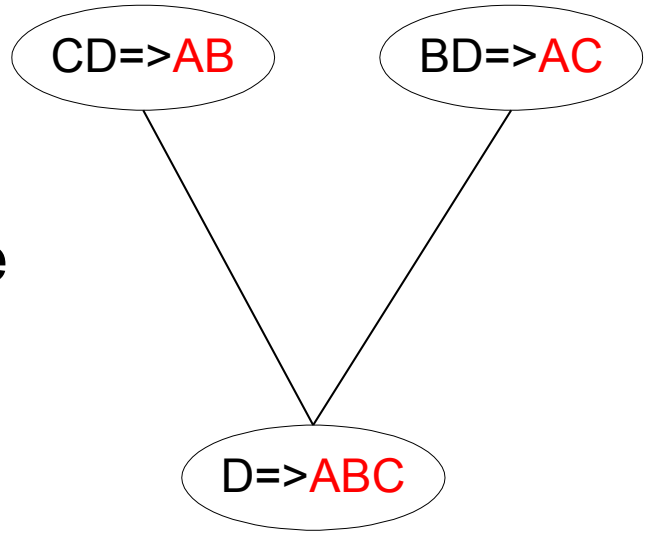
Rule Generation for Apriori Algorithm

Lattice of rules



Rule Generation for Apriori Algorithm

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent



- join(CD=>AB,BD=>AC) would produce the candidate rule D => ABC

- Prune rule D=>ABC if its subset AD=>BC does not have high confidence

Apriori-elven működő szabálygenerálás Z-ből

- egyelemű jobboldalú szabályokra conf számolása, csak a jók maradnak
- minden szabály jobboldalán rendezve tartjuk az elemeket
- $k-1$ hosszú jobboldalról k hosszú jobboldalra:
  - ha van két olyan $k-1$ hosszú jobboldal, akiknek az első $k-2$ tagja megegyezik, akkor ezekből unióval k hosszú jobboldalt képezünk (ezt minden lehetséges módon megtesszük)
  - leellenőrizzük, hogy a két, generáló $k-1$ hosszú részhalmazon kívüli többi $k-2$ darab $k-1$ elemű részhalmazhoz is jó szabály tartozott
  - aki ezen a szűrőn is átmegy, arra *conf*-ot számolok, aki ezt is túléli az lesz jó, k hosszú jobboldalú szabály

Mi kell $conf(X \to Z \setminus X)$ kiszámolásához?

- $conf(X \to Z \setminus X) = \frac{\sigma(Z)}{\sigma(X)}$
- nade Z és X is gyakoriak (Z def szerint, X meg ennek a része)
- ezeket az infókat már kiszámoltam a gyakori elemhalmazok generáláskor, onnan csak elő kell venni (nem kell újra nézni a tranzakciókat)

Hogyan tároljuk a gyakori elemhalmazokat és a hozzájuk tartozó σ-kat?

- Láttuk, hogy a szabályok generálásakor kellenek a gyakori elemhalmazok és a hozzájuk tartozó σ értékek is.
- Hogyan tároljuk ezeket?
- Gond, hogy nagyon sok gyakori elemhalmaz lehet.
- Kellene valami kompaktabb tárolás a gyakori elemhalmazoknak.

Compact Representation of Frequent Itemsets

- Some itemsets are redundant because they have identical support as their supersets

TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

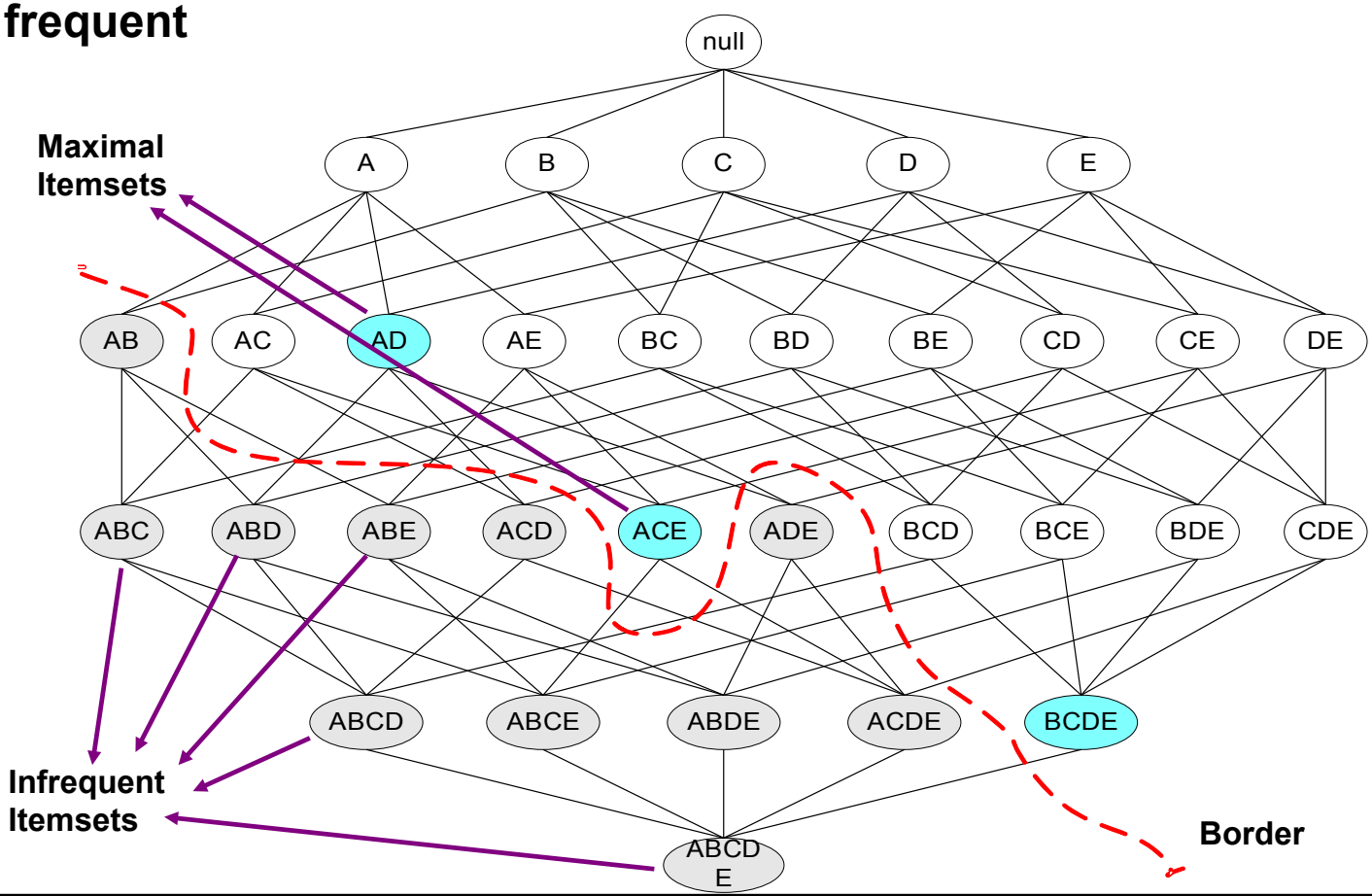
- Number of frequent itemsets $= 3 \times \sum_{k=1}^{10} \binom{10}{k}$
- Need a compact representation

Maximális gyakori elemhalmaz

- egy Z gyakori elemhalmaz maximális gyakori, ha gyakori és nincs őt tartalmazó, nála bővebb gyakori elemhalmaz
- miért jó ez?
  Igaz az, hogy a maximális gyakoriak részhalmazai alkotják az összes gyakori halmazt.
  biz.
  (1) Egy maximális gyakori minden részhalmaza gyakori.
  (2) Ha valaki gyakori, de nem maximális gyakori, akkor van nála bővebb gyakori. Ezt a bővítést folytatva, az elemszám végessége miatt előbb-utóbb egy maximális gyakorit fogunk kapni.
- Vagyis a gyakoriak tárolásához elég a maximális gyakoriakat tárolni.

Maximal Frequent Itemset

An itemset is maximal frequent if none of its immediate supersets is frequent



Mi a baj a maximális gyakoriak tárolásával?

- Ha tárolom a maximális gyakoriakat, akkor meg tudom határozni a gyakoriakat.
- De nekünk a σ értékek is kellenek, azok meg nem derülnek ki így.
- Azt lenne jó tudni, hogy amikor bővítek egy gyakorit és egy másik gyakorit kapok, akkor változik-e a σ.

Zárt elemhalmaz

- Egy X elemhalmaz zárt, ha bármely, nála egy elemmel bővebb elemhalmaz (X plusz valaki) támogatottsága kisebb, mint X támogatottsága (σ-ja)
- Azaz: X nem zárt, ha van legalább egy olyan nála csak egy elemmel bővebb halmaz, aminek σ-ja ugyanakkora, mint az övé
- ha Z zárt és gyakori egyszerre, akkor gyakori zárt elemhalmaznak hívjuk

Closed Itemset

- An itemset is closed if none of its immediate supersets has the same support as the itemset

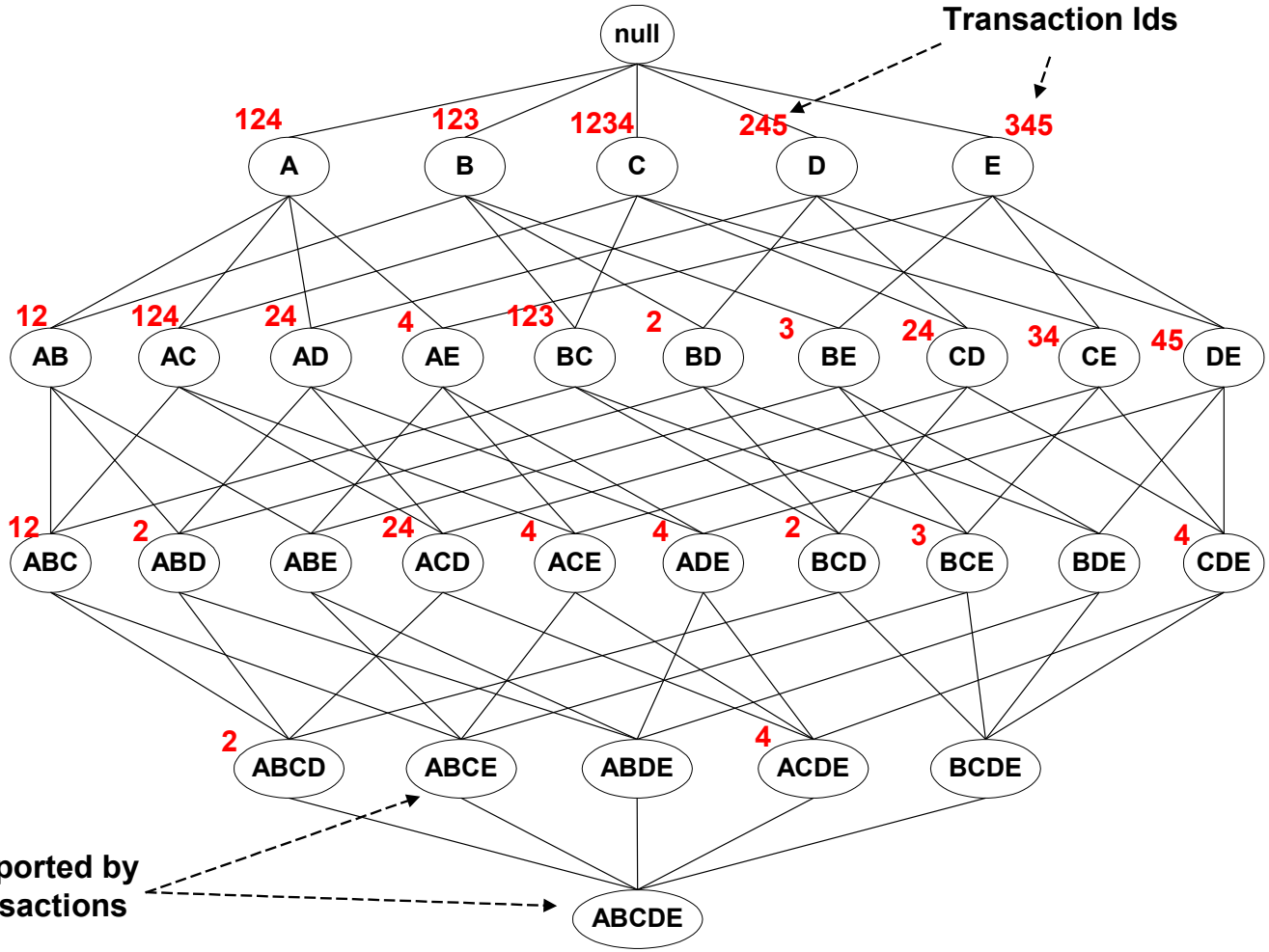
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

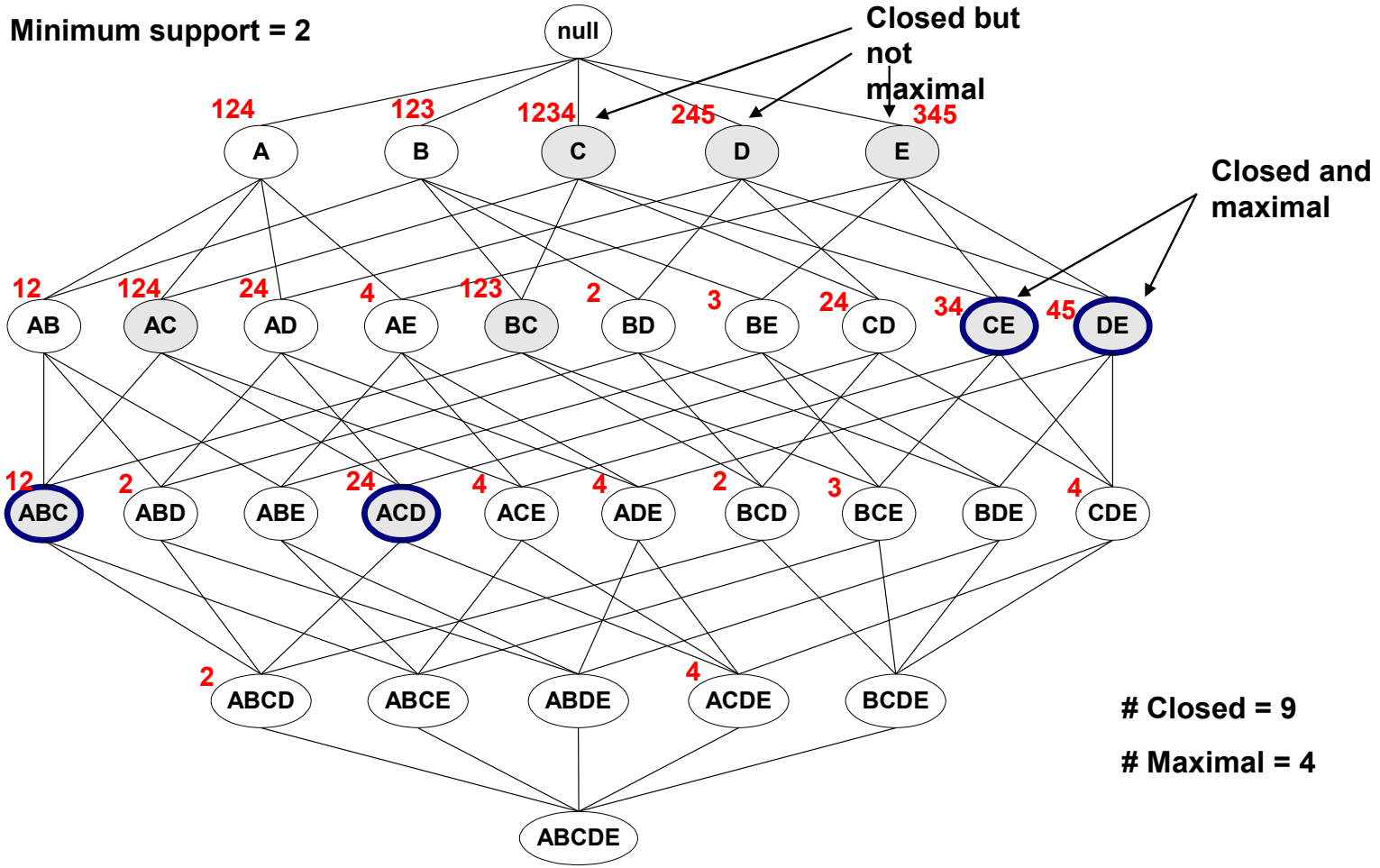
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

Maximal vs Closed Itemsets

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



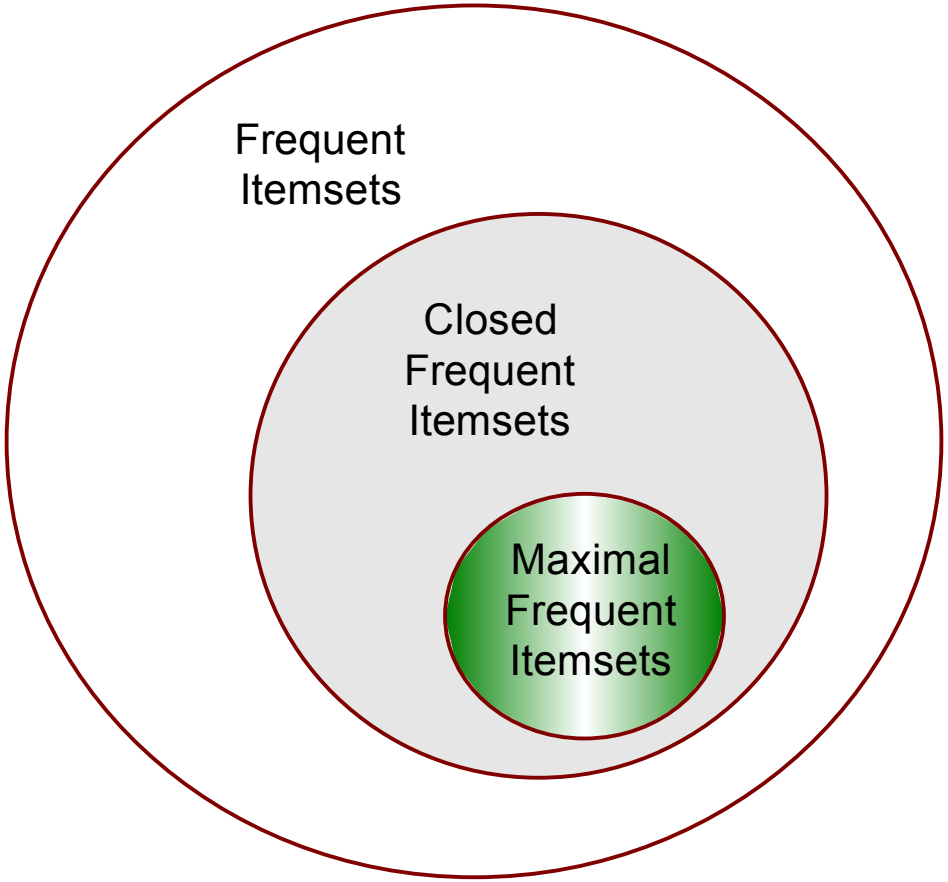
Maximal vs Closed Frequent Itemsets



Zártság és maximalitás kapcsolata

- ha valami maximális gyakori, akkor biztosan gyakori, de ha zárt, abból nem következik, hogy gyakori
- lehet valami zárt gyakori, de nem maximális gyakori
- de ha Z maximális gyakori, akkor egyben zárt gyakori is
  biz.
  ha Z maximális gyakori, akkor $\sigma(Z)$ *geq* küszöb, a gyakoriság miatt, de minden nála bővebb halmaz már nem gyakori, vagyis az ilyenekre a σ a küszöb alá, azaz ezzel együtt $\sigma(Z)$ alá is megy, vagyis Z zárt is egyben

Maximal vs Closed Itemsets



Miért jók a zárt gyakoriak?

- tároljunk minden zárt gyakorit a hozzájuk tartozó σ-val
- ezekből meghatározható
  - minden gyakori elemhalmaz
  - és a hozzájuk tartozó σ-k is

Gyakori elemhalmazok és σ-juk meghatározása a zártak segítségével

- Mivel minden maximális gyakori zárt is egyben, ezért ha a zárt gyakoriak adottak, akkor ezeknek az összes részhalmatai alkotják a gyakoriakat: a gyakori halmazok megvannak.
- honnan lesznek ezekhez a gyakoriakhoz σ-k?
  - Ha egy Z zárt és gyakori, akkor oda van írva.
  - Ha Z gyakori, de nem zárt, akkor az ő σ-ja kiszámolható a nála eggyel bővebb gyakoriak σ-jából: az ezek közül vett legnagyobbal egyenlő.
  - Ha tehát a σ-kat elemszám alapján csökkenő sorrendben számoljuk ki, akkor ez megtehető.