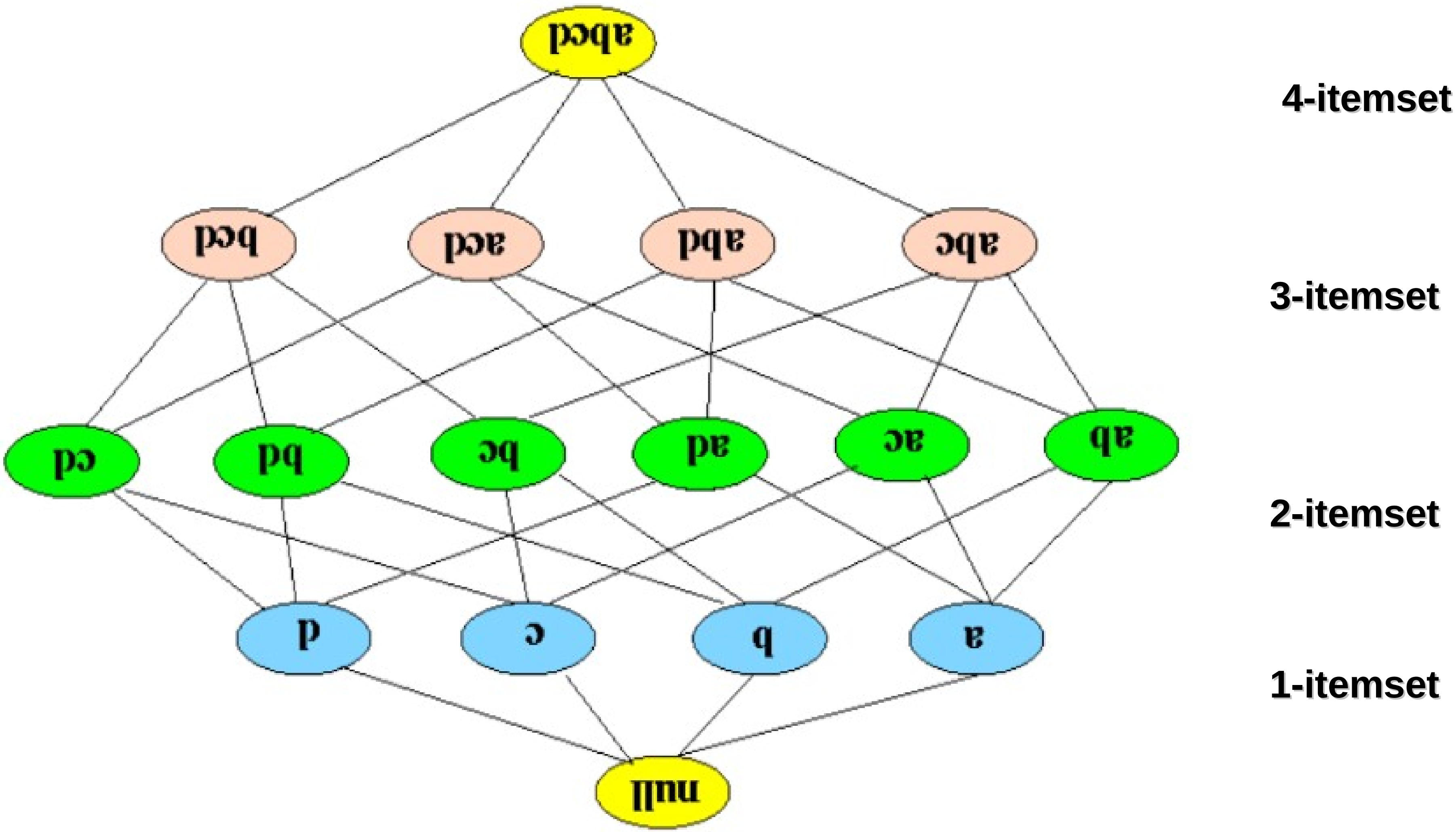
DATA MINING
&
DATA WAREHOUSING

Module III

- **Association Rule Mining**
  - **What is AR**
  - **Methods to discover AR**
  - **Apriori algo**
  - **Partition algo**
  - **Pincer seaarch algo**
  - **FPtree growth algo**
  - **Incremental algo**
  - **Border algo**
  - **Generalized ARs**

- The Aprirori algo has the following properties
  - Bottom-up and Breadth-first methods
  - Computation
    - Starts from smallest frequent itemsets
    - Moves upward till it reaches largest FI
    - Number of DB passes depends on largest size of FI
  - If FI becomes larger
    - many iterations required (disadvantage)
    - So performance decreases

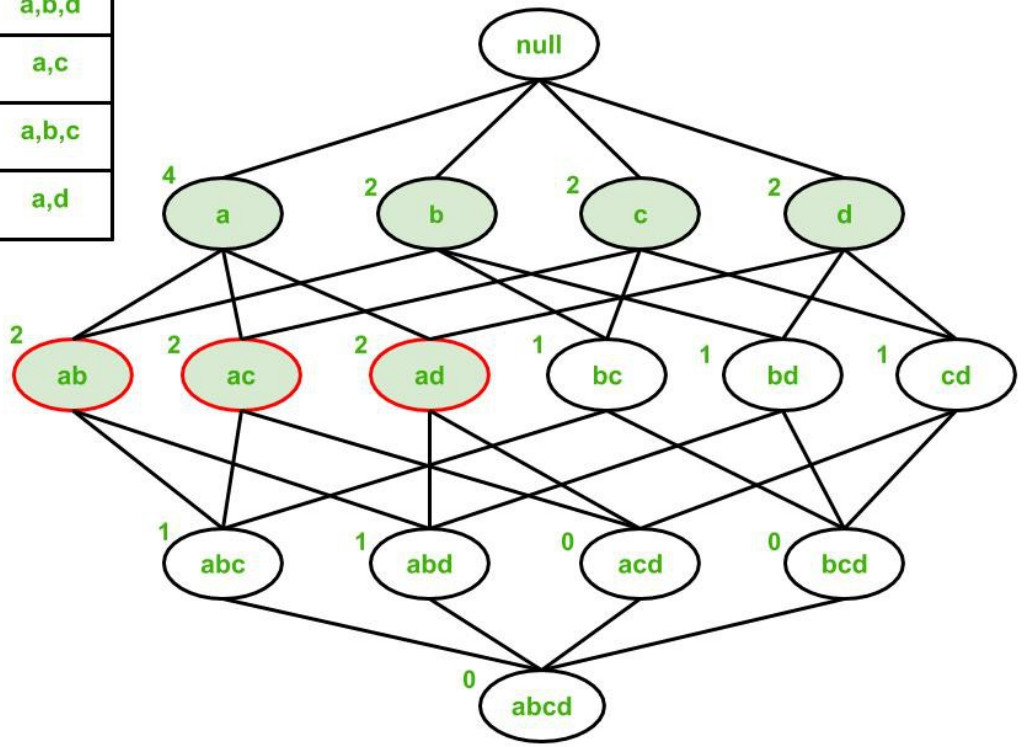

abcd
bcd
acd
abd
abc
cd
bd
bc
ad
ac
ab
d
c
b
a
null
4-itemset
3-itemset
2-itemset
1-itemset

- **To overcome the above said disadvantage of many iterations**
  - **Pincer Search Algorithm was proposed by,**
    - **Dao-I Lin & Zvi M. Kedem of New York University in 1997**

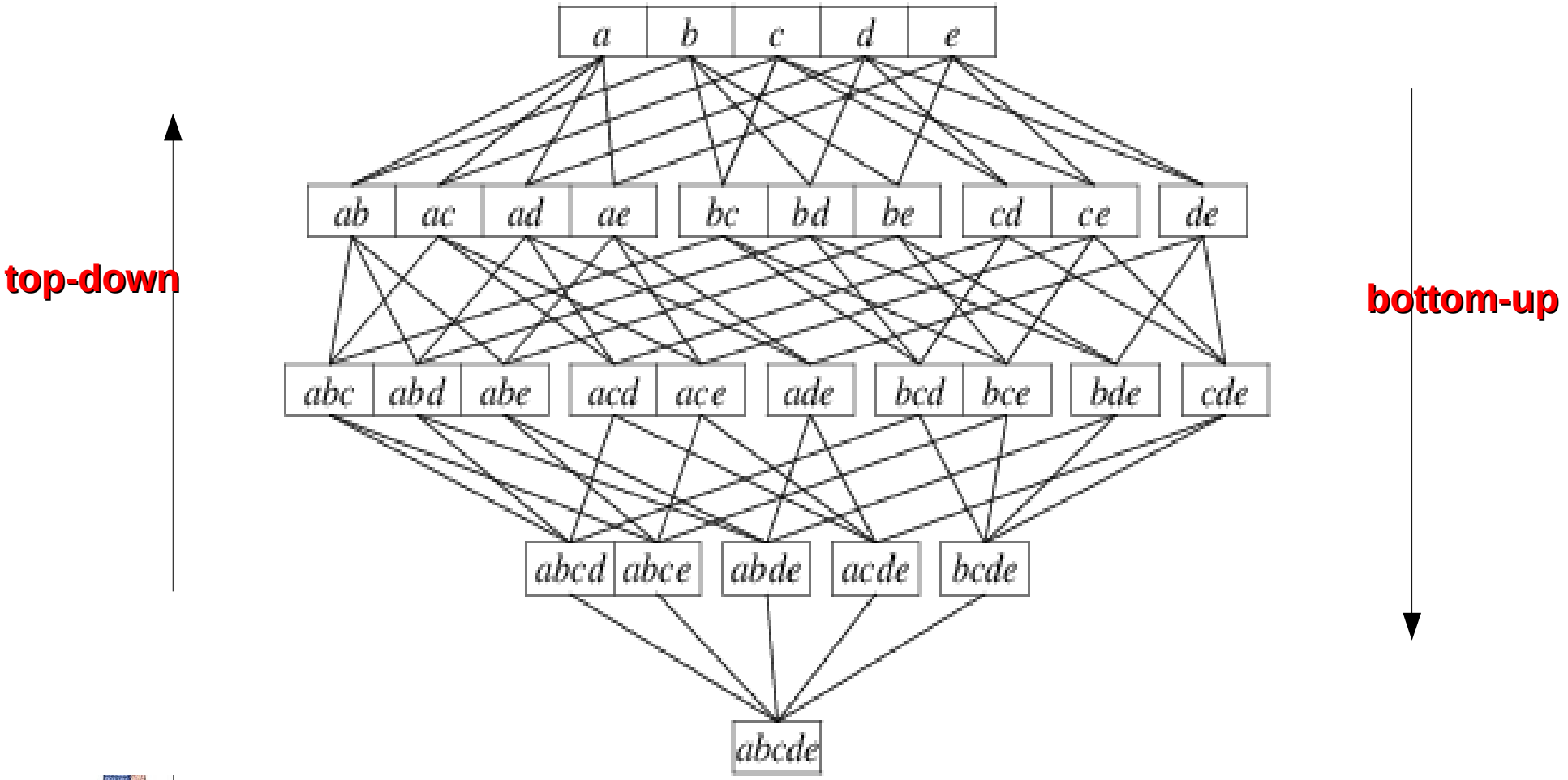
Pincer Search Algorithm

- **It is a slight modification to original Apriori Algorithm**

TID	Items
1	a,b,d
2	a,c
3	a,b,c
4	a,d



- **The algorithm uses both, the top-down and bottom-up approaches to Association Rule mining.**



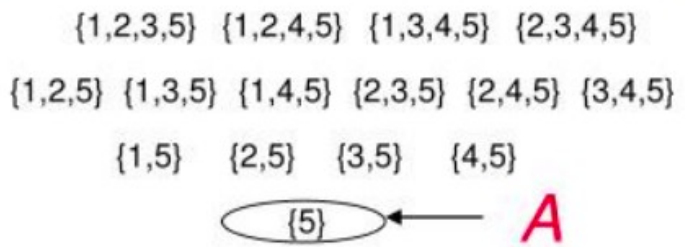
- **In this algorithm**
  - **the main search direction is bottom-up (same as Apriori)**
  - **& also conducts simultaneously a restricted top-down search,**

- **The Main Idea of this algorithm is**
  - **the information gathered in one direction is used to prune more candidates or passes in the other direction**

- **Top down search basically is used to maintain data structures called**
  - **Maximum Frequent Candidate Set (MFCS)**
  - **it produces the Maximum Frequent Set –**
    - **the set containing all maximal frequent itemsets**
- **The algorithm specializes in dealing with maximal frequent itemsets of large length.**

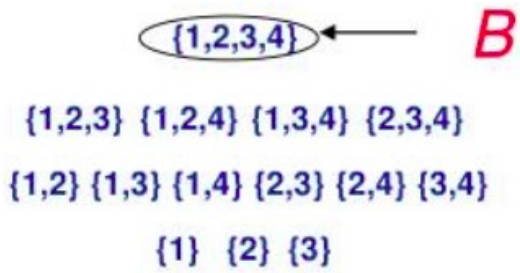
- Let A and B be two itemsets and $A \subseteq B$

- Observation-1: A infrequent $\Rightarrow$ B infrequent
  (if a transaction does not contain A, it cannot contain B)



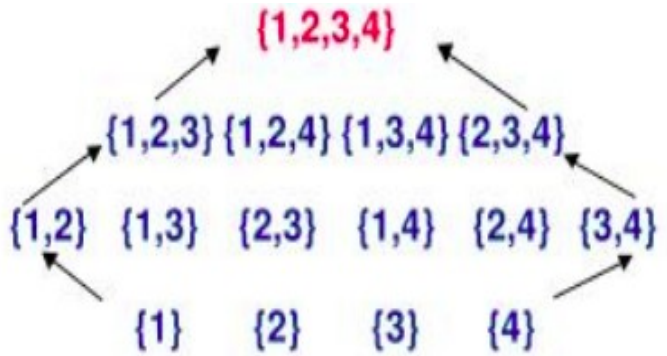
Upward Closure Property

- Observation-2: B frequent $\Rightarrow$ A frequent
  (if a transaction contains B, it must contain A)



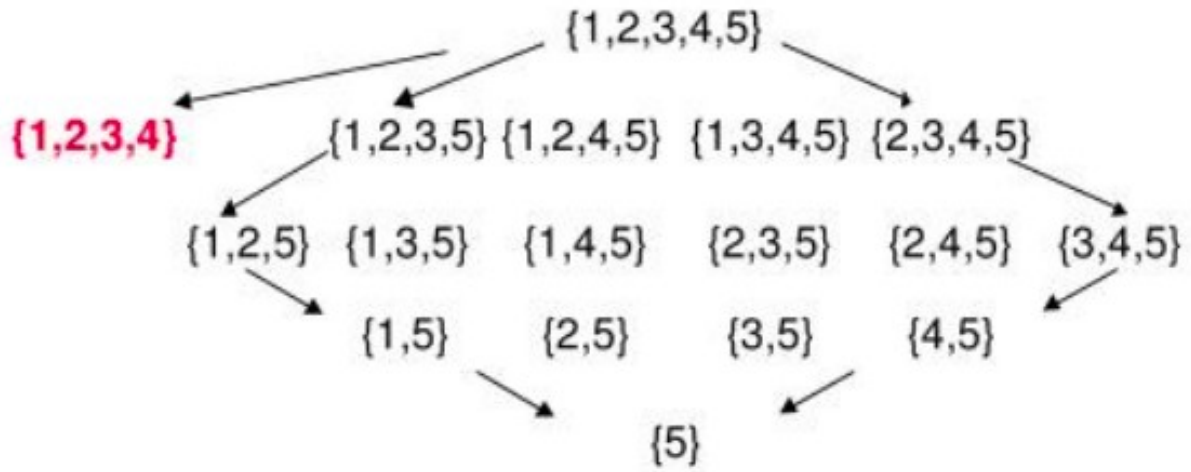
Downward Closure Property

- Observation-1 leads to bottom-up search algorithms,



Blue: frequent itemsets
Red: maximal frequent itemsets
Black: infrequent itemsets

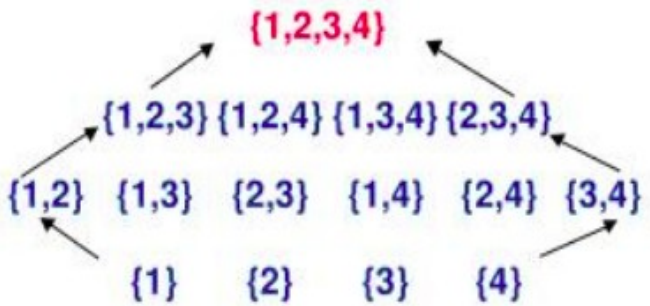
- Observation-2 leads to top-down search algorithms,



Data Structures Maintained

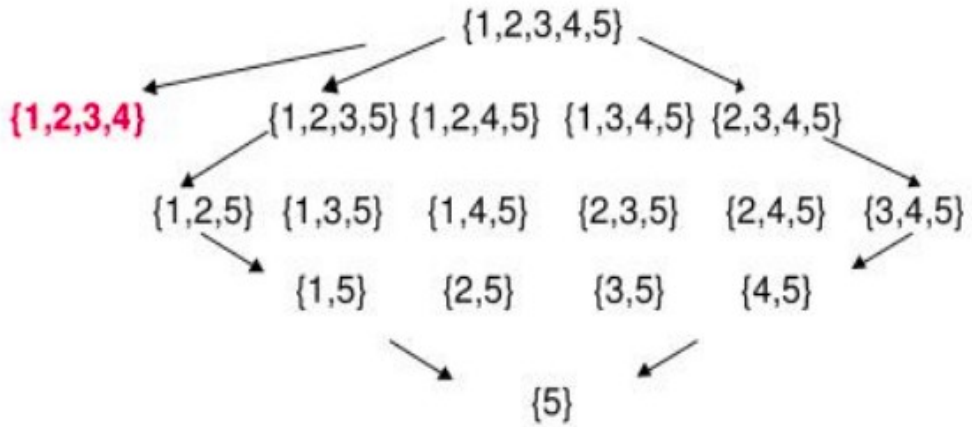
- For bottom-up search: Candidate set (as usual)
- For top-down search: Use a new dynamically maintained data structure: maximum frequent candidate set (MFCS)
- MFCS is a set of itemsets:
  - Union of its subsets contains all known frequent itemsets
  - Union of its subsets does not contain any currently known infrequent itemsets
  - It is of minimum cardinality
- MFCS supports efficient coordination between bottom-up and top-down searches

- For bottom-up search, every frequent itemset is explicitly examined (in the example, until {1,2,3,4} is examined)



Blue: frequent itemsets
Red: maximal frequent itemsets
Black: infrequent itemsets

- For top-down search, every infrequent itemset is explicitly examined



- **Both direction searches are used for pruning in following way:**
  - **If some maximal frequent itemset is found in the top down direction,**
    - **then this itemset can be used to eliminate (possibly many) candidates in the bottom-up direction.**
      - **Because the subsets of this frequent itemset will be frequent and hence can be pruned**
  - **If an infrequent itemset is found in the bottom up direction,**
    - **then this infrequent itemset can be used to eliminate the candidates found in top-down search found so far**

- In this algo in each pass
  - it counts the support in the bottom-up direction
  - and also count the support of some itemsets using top-down approach
    - which are called as MFCS
- So that early pruning can be done

1. $L_0 := \emptyset$; $k := 1$; $C_1 := \{\{ i \} \mid i \in I\}$
2. MFCS := {{1,2, ...,n}}; MFS := $\emptyset$
3. while $C_k \neq \emptyset$
4. read database and count supports for C_k and MFCS
5. MFS := MFS $\cup$ { frequent itemsets in MFCS }
6. determine frequent set L_k and and infrequent set S_k
7. use S_k to update MFCS
8. generate new candidate set C_{k+1} (join, recover, and prune)
9. $k := k + 1$
10. return MFS