Apriori Algorithm

For a given set of transactions, the main aim of Association Rule Mining is to find rules that will predict the occurrence of an item based on the occurrences of the other items in the transaction.

For example, given the 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

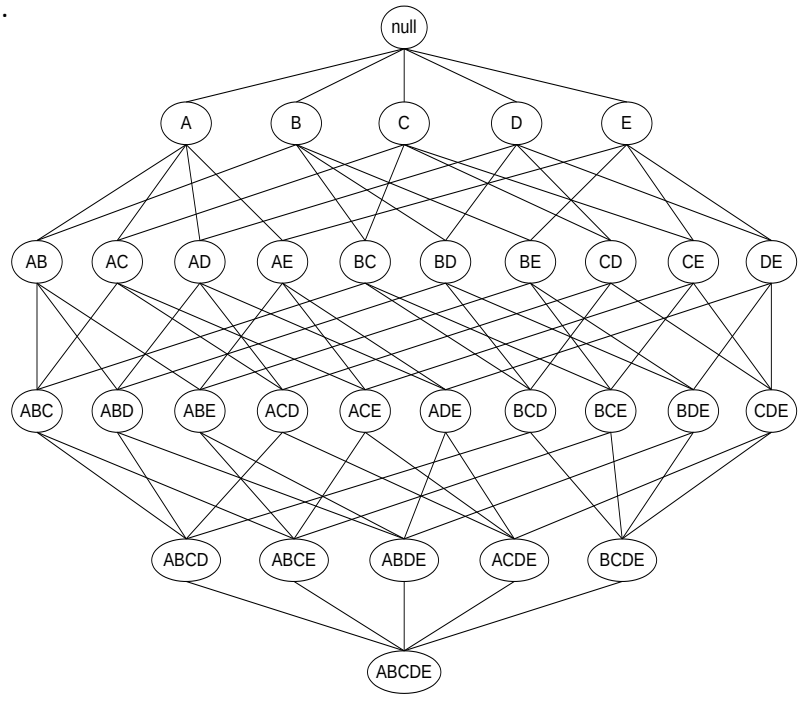
For the given Table, examples of Association Rules are:

{Diaper} -> {Beer}
{Milk, Bread} -> {Eggs, Coke}
{Beer, Bread} -> {Milk}

In general, mining association rules involves two steps.

1. **Frequent Itemset Generation**
   - Generate all the itemsets whose support ≥ *minsup*
2. **Rule Generation**
   - From each frequent itemset, generate the association rules which satisfy the *minsup* and *min_conf* thresholds.

The first step involved is the generation of frequent itemsets which is computationally very expensive. For a given ***d*** number of items, there are $\mathbf{2^d}$ possible candidate itemsets from which we need to identify the frequent itemsets.



Each itemset in the lattice is called as a candidate itemset. For generating the frequent itemsets from the lattice, a brute force approach is:

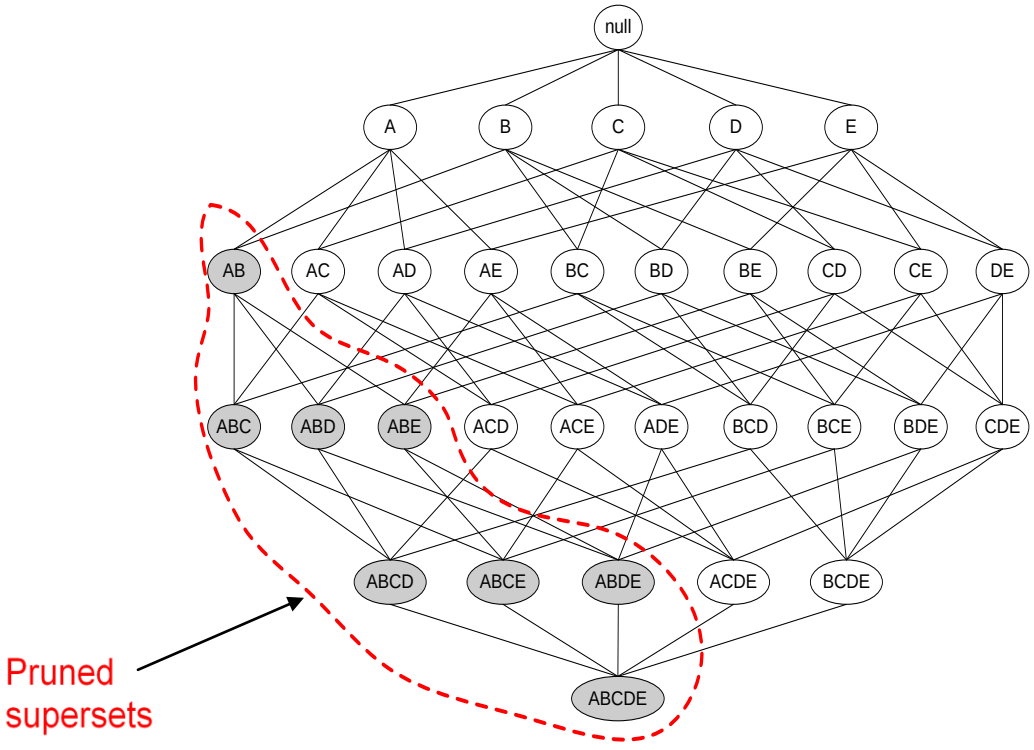
- Calculate the support of each candidate itemset in the lattice by scanning through the given transactional dataset.

This approach is very expensive because, we need to consider 2^d number of candidate sets. For improving the efficiency, we can reduce the number of candidate itemsets by pruning them. There are other techniques which help in improving the efficiency but currently we will concentrate on how to reduce the number of candidate itemsets to improve the efficiency.

There is a principle named as **Apriori principle** which states that: "*If an itemset is frequent, then all of its subsets must be frequent.*"
We can say that, support of an itemset will never exceed the support of its subsets. This property is also known as anti-monotone property of support.

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

When the Apriori Principle is applied on the lattice mentioned above, we can see the pruning as follows:



In the lattice shown above, **AB** is **infrequent** which made all the **supersets** of **AB** infrequent and hence they will be **pruned** from the candidate itemset.

Apriori Algorithm is a basic algorithm which is used to find the frequent itemsets was developed by Agrawal and Srikanth in 1994. The important steps in the generation are as follows:

- Join Step
    - Candidate itemsets are found
- Prune Step
    - Identify the candidate itemsets which are not frequent and make the frequent itemset

The pseudo code of the Apriori algorithm is as follows:-

```
Ck: Candidate itemset of size k
Lk : frequent itemset of size k

L1 = {frequent items};
for (k = 1; Lk !=∅; k++) do begin
        Ck+1 = candidates generated from Lk;                    →JOIN STEP
        for each transaction t in database do
                increment the count of all candidates in Ck+1 that are
contained in t
        Lk+1  = candidates in Ck+1 with min_support             →PRUNE STEP
        end
return ∪k Lk;
```

Let us now consider the working of this algorithm with an example:

Given a dataset, we need to identify the frequent itemsets that can be generated from the dataset and generate the association rules.

Let us consider the dataset as

TID	Items
10	A, B, C, E
20	C, D, E
30	A, B
40	B, C, E
50	C, E

Let us suppose that *minsup* = 2. (For an item to be frequent, its support count should be ≥2)

In the first scan, support count for each item in the itemset is calculated. So, the candidate set (***C_1***) looks like:

Itemset	Sup
{A}	2
{B}	3
{C}	4
{D}	**1**
{E}	4

We need to identify the items which don't satisfy the *minsup* condition and then prune them. After pruning, the frequent Itemset of size-1(L_1) looks like:

Itemset	Sup
{A}	2
{B}	3
{C}	4
{E}	3

Since there is a possibility of generating the 2-itemsets, the algorithm will be executed again. So, in the join-step, we will identify all the possible 2-itemsets from the frequent 1-itemsets. Candidate Itemset of size-2(C_2) from this table is:

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

Identify the itemsets which are not satisfying the *minsup* condition and then prune them. Considering the Apriori principle here, since the itemsets are not frequent, the super-set of these itemsets won't be frequent.

Itemset	Sup
{A, B}	2
{A, C}	1
{A, E}	1
{B, C}	2
{B, E}	2
{C, E}	3

After pruning, frequent Itemset of size-2(L_2) obtained is:

Itemset	Sup
{A, B}	2
{B, C}	2
{B, E}	2
{C, E}	3

In the join step, when two itemsets of same length let us say k here, are considered to form a (k+1)-itemset, check if the two sets share the same (k-1) values. If they satisfy, then merge these two sets. By following this method, further computations can be reduced while identifying the candidate itemsets in the join step.

So, now if we consider the same procedure, 4 candidate itemsets are available here. Only {B, C} and {B, E} each of size-2, has a common value B. So, now considering the join of these two sets will lead to {B, C, E} and automatically all the other subsets are included in this. So, no need to again consider {B, E} and {C, E} again and compute.

Since there is further possibility of generating 3-itemsets, the possible candidate 3-itemsets from L_2 is C_3 which is:

Itemset
{B, C, E}

The itemset {B, C, E} has the support count of 2. It is satisfying the *minsup* condition. Hence, the identified L_3 is:

Itemset	Sup
{B, C, E}	2

Since there are no further possible combinations of itemsets, the algorithm will end here. The frequent itemsets identified are:

{A}, {B}, {C}, {E}, {A, B}, {B, C}, {B, E}, {C, E}, {B, C, E}

To generate the strong association rules which satisfy both minimum support and minimum confidence from these frequent itemsets is a straightforward approach. Generation is as follows:

- For each frequent itemset ***l***, generate all the possible non-empty subsets of ***l***.
- For every non-empty subset of ***l***, output the rule "***s*** -> (***l*** **-** ***s***)" if *support_count(**l**)/support_count(**s**)* ≥ *min_conf,* the minimum confidence threshold.

There is no separate condition to check if the rule satisfies *minsup* condition. All the rules will satisfy the *minsup* condition since they are all identified from the frequent itemsets.

To show a working example for how to generate the association rules, let us consider the frequent itemset being generated from the previously mentioned example which is **{B, C, E}**. Let us identify all the possible nonempty subsets of this itemset. They are: {B}, {C}, {E}, {B, C}, {C, E}, {B, E}. The resulting association rules are as shown below, each listed with its confidence:

{B} -> {C, E}	confidence = 2/3 = 0.67
{C} -> {B, E}	confidence = 2/4 = 0.5
{E} -> {B, C}	confidence = 2/3 = 0.67
{B, C} -> {E}	confidence = 2/2 = 1
{B, E} -> {C}	confidence = 2/2 = 1
{C, E} -> {B}	confidence = 2/3 = 0.67

If the *min_conf* is, say, 75% then only 4^{th} and 5^{th} rules are generated as they are the only ones which are strong.