GENERAL CONFEDERATION OF LABOR OF VIETNAM

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL PROJECT**

**THE DESIGN AND ANALYSIS OF ALGORITHMS**

**A SURVEY OF ITEMSET MINING**

*Instructing Lecturer*: **NGUYỄN CHÍ THIỆN**

*Student’s name*: **HUỲNH TRẦN MINH TIẾN – 520H0583**

**NGUYỄN TRUNG TÍN – 520H0589**

Class **: 20H50202**

Course  **: 24**

**HO CHI MINH CITY, 2022**

GENERAL CONFEDERATION OF LABOR OF VIETNAM

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL PROJECT**

**THE DESIGN AND ANALYSIS OF ALGORITHMS**

**A SURVEY OF ITEMSET MINING**

*Instructing Lecturer*: **NGUYỄN CHÍ THIỆN**

*Student’s name*: **HUỲNH TRẦN MINH TIẾN – 520H0583**

**TRẦN VĂN C – MSSV**

Class **: 20H50202**

Course  **: 24**

**HO CHI MINH CITY, 2022**

**ACKNOWLEDGEMENT**

In order to get a complete and good report like this, our team has received enthusiastic help from lectures and classmates.

My team would also like to thank for the helpful knowledge and enthusiastic help from lecturer Mr. Nguyễn Chí Thiện . Thank you, Mr. Thiện, for being so enthusiastic in teaching, educating, equipping us with the necessary knowledge and creating the most favorable conditions for us to complete this report.

And also thank Ton Duc Thang University for giving us a modern and

developed educational environment.

With hard work and effort we have successfully completed this report. But surely, this report cannot avoid mistakes. We are looking forward to receiving from teacher so that we can improve it better.

We sincerely thank you!

**THE PROJECT WAS COMPLETED**

**AT TON DUC THANG UNIVERSITY**

We pledge that this is a product of our own project and is under the guidance of Mr. Nguyễn Chí Thiện. The content of research, results in this subject is honest and not published in any form before. The data in the tables used for the analysis, comment, and evaluation were collected by the authors themselves from various sources indicated in the reference section.

In addition, many comments and assessments as well as data from other authors and organizations have been used in the project, with references and annotations.

**If any fraud is found, I am fully responsible for the content of my project.** Ton Duc Thang University is not involved in any copyright infringement or copyright infringement in the course of implementation (if any).

*Ho Chi Minh, December 3rd 2022*

*Author*

*Huỳnh Trần Minh Tiến*

*Nguyễn Trung Tín*

**EVALUATION OF INSTRUCTING LECTURER**

**Confirmation of the instructor**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh City, 2022

(sign and write full name)

**The assessment of the teacher marked**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh City, 2022

(sign and write full name)

# SUMMARY

Itemset mining is an important subfield of data mining, which consists of discovering interesting and useful patterns in transaction databases. The traditional task of frequent itemset mining is to discover groups of items (itemsets) that appear frequently together in transactions made by customers. Although itemset mining was designed for market basket analysis, it can be viewed more generally as the task of discovering groups of attribute values frequently co-occurring in databases. Due to its numerous applications in domains such as bioinformatics, text mining, product recommendation, e-learning, and web click stream analysis, itemset mining has become a popular research area.

This paper provides an up-to-date survey that can serve both as an introduction and as a guide to recent advances and opportunities in the field. The problem of frequent itemset mining and its applications are described. Moreover, main approaches and strategies to solve itemset mining problems are presented, as well as their characteristics. Limitations of traditional frequent itemset mining approaches are also highlighted, and extensions of the task of itemset mining are presented such as high-utility itemset mining, rare itemset mining, fuzzy itemset mining and uncertain itemset mining. The paper also discusses research opportunities and the relationship to other popular pattern mining problems such as sequential pattern mining, episode mining, sub-graph mining and association rule mining. Main open-source libraries of itemset mining implementations are also briefly presented.

**TABLE OF CONTENT**

**[ACKNOWLEDGEMENT iii](#_Toc18325)**

**[THE PROJECT WAS COMPLETED iv](#_Toc19405)**

**[AT TON DUC THANG UNIVERSITY iv](#_Toc6942)**

**[EVALUATION OF INSTRUCTING LECTURER v](#_Toc26354)**

**[SUMMARY vi](#_Toc11530)**

**[LIST OF TABLES, PICTURES, GRAPHS 1](#_Toc6452)**

**[Part 1: Frequent Itemset Mining 4](#_Toc10197)**

**[Part 2: Breadth-first search and depth-first search 6](#_Toc5961)**

**[Part 3: Apriori an breadth-first search algorithm 7](#_Toc30933)**

**[3.1 Horizontal database 7](#_Toc32424)**

**[3.2 Vertical database 11](#_Toc32574)**

**[Part 4: Eclat a vertical depth-first search algorithm 12](#_Toc21246)**

**[4.1 Definition 12](#_Toc29734)**

**[4.2 Example 14](#_Toc9379)**

**[4.3 Algorithm Presentation 16](#_Toc23481)**

**[Part 5: Frequent Pattern-growth algorithms 16](#_Toc3969)**

**[5.1 Definition 16](#_Toc4350)**

**[5.1.1 Shortcomings Of Apriori Algorithm 17](#_Toc32314)**

**[5.1.2 FP-Tree 17](#_Toc18654)**

**[5.1.3 Advantages Of FP Growth Algorithm 17](#_Toc15261)**

**[5.1.4 Disadvantages Of FP-Growth Algorithm 17](#_Toc3660)**

**[5.1.5 FP Growth vs Apriori Comparison 18](#_Toc27818)**

**[5.2 Example 18](#_Toc2238)**

**[5.3 Algorithm Presentation 25](#_Toc8106)**

**[REFERENCES 26](#_Toc10071)**

LIST OF TABLES, PICTURES, GRAPHS

**LIST OF TABLES**

[Table 1 : A transaction database 4](#_Toc8390)

[Table 2 : The complexity of Naive Search for FIM. 5](#_Toc4032)

[Table 3 : The vertical transaction database 13](#_Toc28877)

[Table 4 : FP Growth vs Apriori Comparison 18](#_Toc5457)

[Table 5 : Database 19](#_Toc21525)

[Table 6 : 1-itemset frequency 19](#_Toc9864)

[Table 7 : Database is sorted in alphabetical order 20](#_Toc16664)

**LIST OF FIGURES**

[Figure 1 : The search space for I = {a, b, c, d, e} 6](#_Toc24172)

[Figure 2 : Steps for Ck generation 12](#_Toc21083)

[Figure 3 : Conditional Pattern Base 24](#_Toc25021)

[Figure 4 : Conditional Frequent Pattern Tree 24](#_Toc21111)

[Figure 5 : Frequent Pattern Generated 25](#_Toc16459)

**CONTRIBUTION**

|  |  |  |
| --- | --- | --- |
| Name\Algorithm | Huỳnh Trần Minh Tiến | Nguyễn Trung Tín |
| Apriori | x |  |
| AprioriTID | x | x |
| Eclat | x |  |
| FPGrowth |  | x |

# Part 1: Frequent Itemset Mining

Itemset mining is an aspect of Data Mining. This is an important technique for predicting the future or to understand the past by discovering groups of frequent itemsets in the transaction databases.

Suppose, we have a transaction database:

|  |  |
| --- | --- |
| **TID** | **Transaction** |
|  | {a, c, d} |
|  | {b, c, e} |
|  | {a, b, c, e} |
|  | {b, e} |
|  | {a, b, c, e} |

Table 1: A transaction database

*I:* Set of items (I = {i1, i2, . . . im}).

*D:* Set of transactions(D = {T1, T2 . . . Tn}).

: Transaction q-th (with 1 ≤ q ≤ m).

q: each Transaction has a unique TID (Transaction Identifier).

|X|: X is a set of items (X ⊆ I.), |X| means the number of items in itemset X (length k).

*sup(X):* the number of transactions containing X in D. *(sup(X) = |{T|X ⊆ T ∧ T ∈ D}|)*

*relSup(X):* is **relative support**, relSup(X) = sup(X)/|D|.

The target of FIM aims to investigate associations between interesting items and discover all frequent itemsets in the given transaction database. The interest of a given itemset is defined by using *support.* The support (or absolute support) is the number of transactions containing specified items, which is denoted by *sup(X)*. For instance, sup(a) = 3, it’s contained in , and

An itemset is frequent if and only if its support is no less than a given “minimum support threshold” which is given by users. Following the given database, for example we have minsup = 4, the frequent itemsets are : {b} 4, {c}: 4, {b,e}: 4.

Searching for FIM is an enumeration problem. The goal is to list all itemsets which are specified the minimum support constraint. Thus, the naive method for solving this problem is usually infeasible and inefficient.

|  |  |
| --- | --- |
| Number of items (n) | Search space () |
| 100 |  |
| 1000 |  |
| 10000 |  |

Table 2: The complexity of Naive Search for FIM.

To avoid exploring the search space of all possible itemsets for improving efficiency, lots of algorithms were designed, such as **Apriori, FP-Growth, Eclat, H-Mine, and LCM** which we will discuss after. All of those algorithms have the same I/O, but it fits with differents the strategies and data structures:

1. They use a depth-first or breadth-first search.
2. The type of database representation that they are used internally or externally.
3. How they generate or determine the next itemsets to be explored in the search space.
4. How they count the support of itemsets to determine if they satisfy the minimum support constraint.

# Part 2: Breadth-first search and depth-first search

In FIM, Breadth-first search (BFS) and Depth-first search (DFS) are the most popular tools for searching. The BFS algorithm explores the search space by looking out the itemsets with an item, then 2 items, 3 items,... m items (m is the number of items in a database). Another algorithm is DFS, this method is different from BFS, it uses recursion, starting from an itemset with 1 item then appending items to the current itemset to create the larger itemsets.

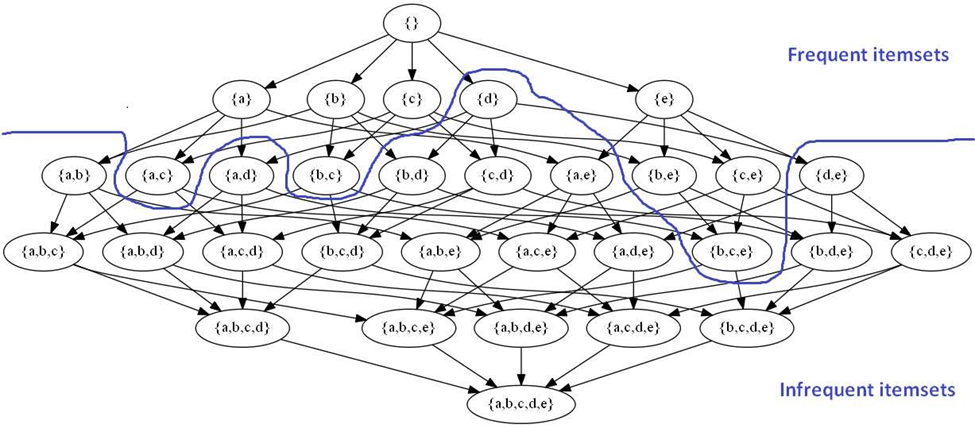


Figure 1: The search space for I = {a, b, c, d, e}

For example:

- BFS (popular algorithm is Apriori): {a}, {b}, {c}, {d}, {e} → {a,b}, {a,c}, {a,d}, … {d,e} → {a,b,c}, {a,b,d}, … → {a,b,c,d}, {a,b,c,e}, … → {a,b,c,d,e}.

- DFS (FPGrowth, H-Mine, LCM): {a} → {a,b} → {a,b,c} → {a,b,c,d} → {a,b,c,d,e} → {a,b,d} → {a,b,d,e} → {a,b,e} → {a,c} → … → {e}.

Avoiding the large search space is an important component in designing efficient algorithms. *The search space pruning techniques* will support us when we design FIM algorithms. Instead of finding all possible itemsets, this technique will ignore all supersets of infrequent itemsets by *sup.* For example, *minsup* = 3 and the support of {a,b} is 2, then all of the superitemsets of this are infrequent too, and the operations don’t need to explore it. This property is called the downward-closure property, anti-monotonicity-property or Apriori-property.

# Part 3: Apriori an breadth-first search algorithm

## 3.1 Horizontal database

Apriori is a Breadth-First search algorithm which is used to find frequent itemsets. In the Horizontal database, Apriori requires two inputs that are standard database (Table 1) and minsup threshold. The idea of this algorithm is an itemset which is frequent, all their non-subset must be frequent, if an itemset is infrequent, all its supersets will be infrequent. Based on that key, the progress to solve this problem will be separated into two stages: The first, finding an itemset that consists of all frequent itemsets, then generating all non-subset of its. For example, the transaction *database is table 1 and minsup = 3.*

|  |  |
| --- | --- |
| **TID** | **Transaction** |
|  | {a, c, d} |
|  | {b, c, e} |
|  | {a, b, c, e} |
|  | {b, e} |
|  | {a, b, c, e} |

**Step 1**: k =1, scanning the standard database, find the frequent 1-itemset called (candidate set).

|  |  |
| --- | --- |
| Items | sup\_count |
| a | 3 |
| b | 4 |
| c | 4 |
| d | 1 |
| e | 4 |

Removing all items which have support don’t satisfy the given minsup. This give us itemset .

|  |  |
| --- | --- |
| Items | sup\_count |
| a | 3 |
| b | 4 |
| c | 4 |
| e | 4 |

**Step 2:** K=2. Generating potentially frequent itemsets of length k (Called ) using (join step). Condition of joining and is that it should have (K-2) elements in common:

1. Check whether all (k-1)-subsets are frequent or not, if not frequent remove that itemsets..
2. Finding their support of each item in .

|  |  |
| --- | --- |
| Itemset | sup\_count |
| a, b | 2 |
| a, c | 3 |
| a, e | 2 |
| b, c | 3 |
| b, e | 4 |
| c, e | 3 |

1. Comparing their itemset’s support with minsup if less than it then remove those items, this give us itemset .

|  |  |
| --- | --- |
| Itemset | sup\_count |
| a, c | 3 |
| b, c | 3 |
| b, e | 4 |
| c, e | 3 |

**Step 3:** Generating using (join step). Condition of joining and is that it should have (K-2) elements in common. So, is matched. is generated by joining is {a, b, c}, {a, b, e}, {a, c, e}, {b, c, e}.

1. Check if all subsets of these itemsets are frequent or not, removing all infrequent itemsets.In this case, {a, b, c} is removed, because subset {a, b} is infrequent (Going on with the rest of itemsets).
2. Find the sup of these remaining itemset.

|  |  |
| --- | --- |
| Itemset | sup\_count |
| b, c, e | 3 |

1. Compare support with minsup (if less than minsup, removing those itemsets) this gives us itemset .

**Step 4:** Generating using (join step). Condition of joining and (k=4) is that it should have (K-2) elements in common. So, doesn’t match, then we stop the algorithm because no frequent itemsets are found further.

|  |
| --- |
|  |

## 3.2 Vertical database

Apriori-TID is a variant of the regular a priori association rule mining algorithm. The Apriori-TID algorithm converts transactional databases into itemset-based transactions to speed up search operations. This transformation eliminates the need to scan the entire data multiple times. In the Apriori-TID algorithm, the original database reading is only done the first time at the beginning . Furthermore, the reading will be transferred to the candidate database, which is a candidate itemset that has been pruned based on the minimum support. Therefore, the Apriori TID algorithm becomes very effective at the end of the process because the Ck size is getting smaller.

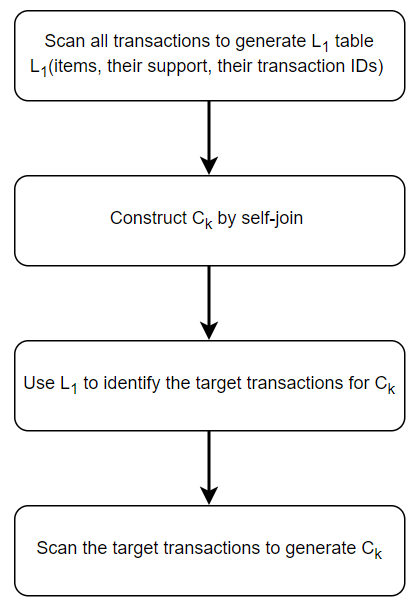


Figure 2: Steps for Ck generation

# Part 4: Eclat a vertical depth-first search algorithm

## 4.1 Definition

ECLAT means ***Equivalence Class Clustering and bottom-up Lattice Traversal.*** This algorithm inherited from Apriori algorithm, it's more efficient than the one. In Apriori, the limitation is large search space, because this generates candidates by combining itemsets without looking at the database. Thus, it can create some itemsets which don't exist in the transactions database. In Eclat, this problem is solved by using the vertical database.

|  |  |
| --- | --- |
| **Item** (x) | **TID-set (tid(x))** |
| a | {, , } |
| b | {, , , } |
|  | {, , , } |
|  | {} |
|  | , , , } |

Table 3: The vertical transaction database

In this algorithm, It only scans the standard database one time for transferring into a vertical database. The difference between Horizontal and Vertical databases is the way they show the information. In the Vertical database, it includes two values, item and its TID-set (which transaction this item appears in).

The vertical database has advantages. First, for any itemsets X and Y , the TID-list of

the itemset X ∪ Y can be obtained without scanning the original database by intersecting

the TID-lists of X and Y , that is: tid(X ∪ Y ) = tid(X) ∩ tid(Y ). Second, the TID-list of

an itemset X allows to directly derive its support without scanning the database, by using

the property that sup(X) = |tid(X)|.For example, the TID-list of {a, c} can be calculated

as tid({a, c}) = tid(a) ∩ tid(c) = {T1, T3, T5}, and it can thus be derived that the support

of {a, c} is |tid({a, c})| = 3.

The basic idea is to use TID-sets to compute the support value of a candidate and avoid generating subsets which do not exist in the database. All single items are used along with their TID-set. Then the function is called recursively and in each recursive call, each item- TID-set pair is verified and combined with another item-tid-set pairs. This will be stopped if no candidate item-tid-set is generated.

## 4.2 Example

Minsup = 3, database (table 1).

**Step 1:** In the first call, function scan the standard database and convert it into the vertical database. k = 1.

|  |  |
| --- | --- |
| **Item** (x) | **TID-set (tid(x))** |
| a | {, , } |
| b | {, , , } |
|  | {, , , } |
|  | , , , } |

**Step 2:** k = 2, recursively call the function to combined item-TID-set pairs which satisfied tid(X ∪ Y ) = tid(X) ∩ tid(Y ).

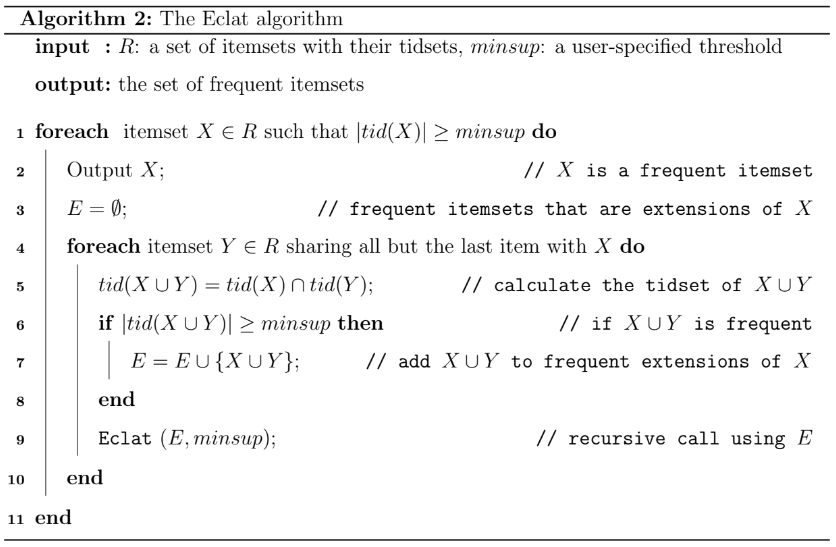
|  |  |
| --- | --- |
| **Item** (x) | **TID-set (tid(x))** |
| {a, b} | {, } |
| {a, c} | {, , } |
| {a, e} | {, } |
| {b, c} | {, , } |
| {b, e} | {, , } |
| {c, e} | {, , } |

**Step 3:** k = 3.

|  |  |
| --- | --- |
| **Item** (x) | **TID-set (tid(x))** |
| {b, c, e} | {, , } |
| {a, b, c} | {, } |
| {a, b, e} | {, } |
| {a, c, e} | {, } |

We stop at k =3, because there are no more item-tid-set pairs to combine.

## 4.3 Algorithm Presentation



**Part 5: Frequent Pattern-growth algorithms**

## 5.1 Definition

This algorithm is an improvement to the Apriori method. A frequent pattern is generated without the need for candidate generation. FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree.

This tree structure will maintain the association between the itemsets. The database is fragmented using one frequent item. This fragmented part is called “pattern fragment”. The itemsets of these fragmented patterns are analyzed. Thus with this method, the search for frequent itemsets is reduced comparatively.

### 5.1.1 Shortcomings Of Apriori Algorithm

1. Using Apriori needs a generation of candidate itemsets. These itemsets may be large in number if the itemset in the database is huge.
2. Apriori needs multiple scans of the database to check the support of each itemset generated => high costs.

These shortcomings can be overcome using the FP growth algorithm

### 5.1.2 FP-Tree

The frequent-pattern tree (FP-tree) is a small data structure that holds quantitative information about frequent patterns in a database. Each transaction is read and then mapped to a route in the FP-tree. This is repeated until all transactions have been read. Because their pathways intersect, different transactions with shared subsets keep the tree compact.

With the database's first item sets, a frequent Pattern Tree is created. The FP tree's goal is to find the most common pattern. Each node of the FP tree represents one of the item set's items.

The null node is represented by the root node, while the item sets are represented by the lower nodes. While forming the tree, the relationships of the nodes with the lower nodes, that is, the item sets with the other item sets, are maintained.

### 5.1.3 Advantages Of FP Growth Algorithm

1. Needs to scan the database only twice when compared to Apriori which scans the transactions for each iteration.
2. The pairing of items is not done in this algorithm and this makes it faster.
3. The database is stored in a compact version in memory.
4. It is efficient and scalable for mining both long and short frequent patterns.

### 5.1.4 Disadvantages Of FP-Growth Algorithm

1. FP Tree is more cumbersome and difficult to build than Apriori.
2. It may be expensive.
3. When the database is large, the algorithm may not fit in the shared memory.

### 5.1.5 FP Growth vs Apriori Comparison

|  |  |  |
| --- | --- | --- |
|  | FP Growth | Apriori |
| Pattern Generation | FP growth generates pattern by constructing a FP tree | Apriori generates patterns by pairing the items into singletons, pairs and triplets. |
| Candidate Generation | There is no candidate generation | Apriori uses candidate generation |
| Process | The process is faster as compared to Apriori. The runtime of the process increases linearly with an increase in the number of itemsets. | The process is comparatively slower than FP Growth, the runtime increases exponentially with increase in number of itemsets |
| Memory Usage | A compact version of database is saved | The candidates combinations are saved in memory |

Table 4: FP Growth vs Apriori Comparison

## 5.2 Example

|  |  |
| --- | --- |
| **TID** | **Items** |
|  | {a, b, c, d, e, f} |
|  | {i, h, a, c, b} |
|  | {a, i, g, b, d} |
|  | {f, e, a, d, b} |
|  | {a, e, g, c} |

Table 5: Database

The frequency of each individual item is computed

|  |  |
| --- | --- |
| Item | Frequency |
| a | 5 |
| b | 4 |
| c | 3 |
| d | 3 |
| e | 3 |
| f | 2 |
| g | 2 |
| h | 1 |
| i | 2 |

Table 6: 1-itemset frequency

**Step 1**: Set minsup = 3, scanning the standard database. These elements are stored in descending order of their respective frequencies. Removing all items which have support don’t satisfy the given minsup. This give us itemset L.

L = {A : 5, B : 4, C : 3, D : 3, E : 3}

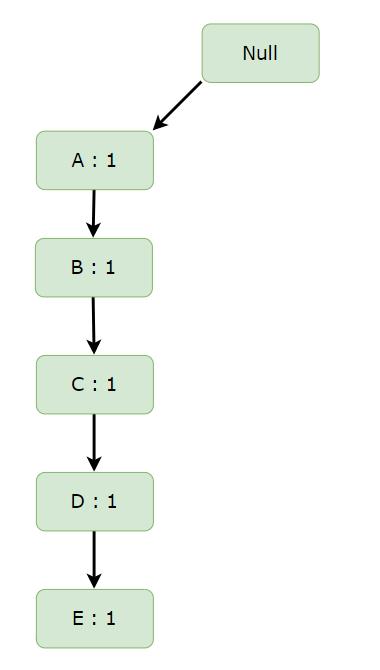
**Step 2**: For each transaction, the respective Ordered-Item set is built. It is done by iterating the Frequent Pattern set and checking if the current item is contained in the transaction in question. If the current item is contained, the item is inserted in the Ordered-Item set for the current transaction.

|  |  |  |
| --- | --- | --- |
| **TID** | **Items** | **Ordered-item Set** |
|  | {a, b, c, d, e, f} | {a, b, c, d, e} |
|  | {i, h, a, c, b} | {a, b, c} |
|  | {a, i, g, b, d} | {a, b, d} |
|  | {f, e, a, d, b} | {a, b, d, e} |
|  | {a, e, g, c} | {a, c, e} |

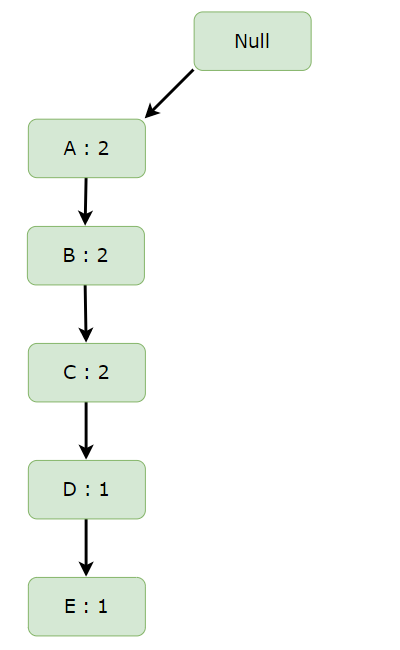
Table 7: Database is sorted in alphabetical order

**Step 3**: Build FP tree. All the items are simply linked one after the other in the order of occurrence in the set and initialize the support count for each item as 1.

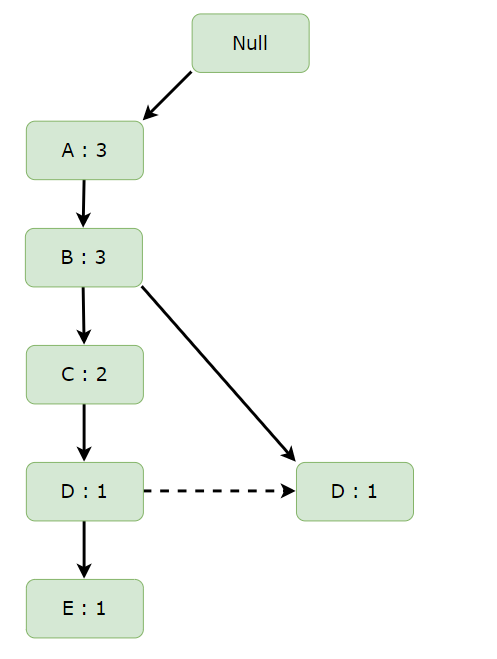
Inserting the set {a, b, c, d, e}



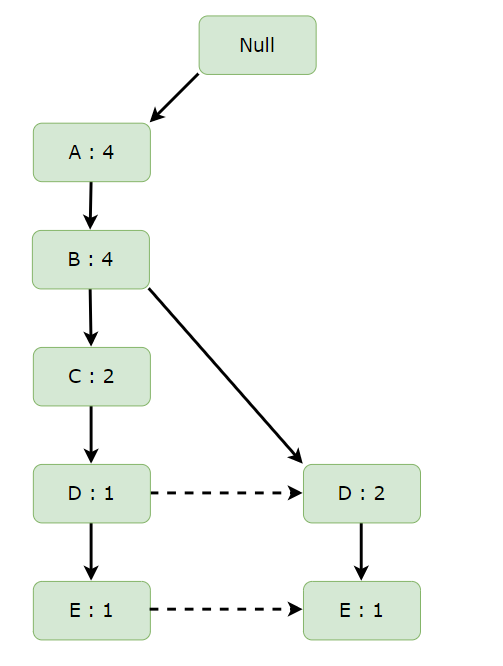
Inserting the set {a, b, c}: Till the insertion of the elements A, B and C, simply the support count is increased by 1



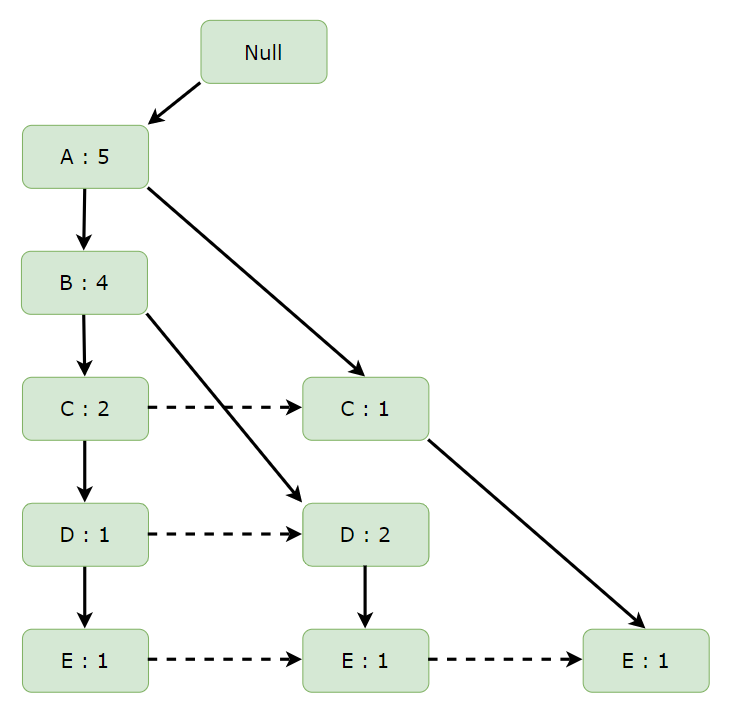
Inserting the set {a, b, d}: On inserting D we can see that there is no direct link between B and D, therefore a new node for the item D is initialized with the support count as 1 and item B is linked to this new node.



Inserting the set {a, b, d, e}: On inserting E, we first initialize a new node for the item E with support count as 1 and link the new node of D with the new node of E.



Inserting the set {a, c, e}: On inserting C, we can see that there is no direct link between A and C, therefore a new node for the item C is initialized with the support count as 1 and item A is linked to this new node. On inserting E, we first initialize a new node for the item E with support count as 1 and link the new node of C with the new node of E.



**Step 4**: Now, for each item, the Conditional Pattern Base is computed which is path labels of all the paths which lead to any node of the given item in the frequent-pattern tree.

|  |  |
| --- | --- |
| Items | Conditional Pattern Base |
| E | {{A,B,C,D : 1}, {A,B,D : 1}, {A,C : 1} |
| D | {{A,B,C : 1}, {A,B : 2}} |
| C | {{A,B : 2}, {A : 1}} |
| B | {A : 4} |
| A |  |

Figure 3: Conditional Pattern Base

Now for each item, the Conditional Frequent Pattern Tree is built. It is done by taking the set of elements that is common in all the paths in the Conditional Pattern Base of that item and calculating its support count by summing the support counts of all the paths in the Conditional Pattern Base.

|  |  |  |
| --- | --- | --- |
| Items | Conditional Pattern Base | Conditional Frequent Pattern Tree |
| E | {{A,B,C,D : 1}, {A,B,D : 1}, {A,C : 1} | {A : 3} |
| D | {{A,B,C : 1}, {A,B : 2}} | {A,B : 3} |
| C | {{A,B : 2}, {A : 1}} | {A : 3} |
| B | {A : 4} | {A : 4} |
| A |  |  |

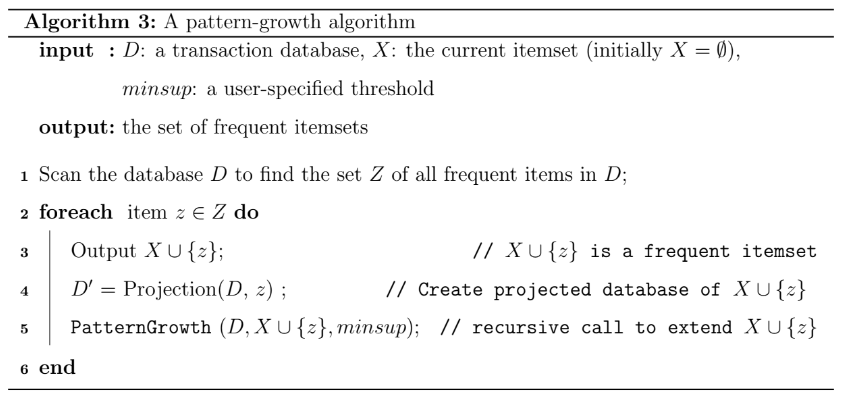
Figure 4: Conditional Frequent Pattern Tree

From the Conditional Frequent Pattern tree, the Frequent Pattern rules are generated by pairing the items of the Conditional Frequent Pattern Tree set to the corresponding to the item as given in the below table.

|  |  |
| --- | --- |
| Items | Frequent Pattern Generated |
| E | {<A,E : 3>} |
| D | {<A,D : 3>, <B,D : 3>, <B,A,D : 3>} |
| C | {<A,C : 3>} |
| B | {<B,A : 4>} |
| A |  |

Figure 5: Frequent Pattern Generated

**5.3 Algorithm Presentation**



**REFERENCES**

1. [AlindGupta, ML | ECLAT Algorithm,12 Jun 2019.](https://www.geeksforgeeks.org/ml-eclat-algorithm/)
2. [Mohammed Al-Maolegi, Bassam Arkok,](https://arxiv.org/ftp/arxiv/papers/1403/1403.3948.pdf) *[An Improved Apriori Algorithm for Association Rules,](https://arxiv.org/ftp/arxiv/papers/1403/1403.3948.pdf)* [pp. 21- 29.](https://arxiv.org/ftp/arxiv/papers/1403/1403.3948.pdf)
3. [Philippe Fournier- Viger, Jerry Chun-Wei Lin, Bay Vo, Tin Truong Chi, Ji Zhang, Hoai Bac Le (2017),](https://classroom.google.com/u/0/c/NTM4OTE4NzUxMzYy) *[A Survey of Itemset Mining.](https://classroom.google.com/u/0/c/NTM4OTE4NzUxMzYy)*
4. [Le Hoang,](https://viblo.asia/p/khai-pha-du-lieu-va-lop-bai-toan-khai-thac-cac-tap-pho-bien-p2-m68Z0W06KkG) *[Khai phá dữ liệu và lớp bài toán khai thác các tập phổ biến (P2).](https://viblo.asia/p/khai-pha-du-lieu-va-lop-bai-toan-khai-thac-cac-tap-pho-bien-p2-m68Z0W06KkG)*
5. [Nguyen Le Phong (2018),](https://www.youtube.com/watch?v=25VUeyh23WM&t=174s) *[[Data Mining 01] - Thuật Toán Apriori Chạy Như Thế Nào?](https://www.youtube.com/watch?v=25VUeyh23WM&t=174s)*
6. [Nguyen Le Phong (2018),](https://www.youtube.com/watch?v=qTYKaxqGLAo&t=1s) *[[Data Mining 01] - Thuật Toán](https://www.youtube.com/watch?v=qTYKaxqGLAo&t=1s)**[FP-GROWTH Khai Phá Luật Kết Hợp](https://www.youtube.com/watch?v=qTYKaxqGLAo&t=1s)*
7. Mohityadav[,](https://www.geeksforgeeks.org/apriori-algorithm/) *[Apriori Algorithm, 13 Jan, 2022.](https://www.geeksforgeeks.org/apriori-algorithm/)*