DATA GENERATION REPORT

Nitin K (2017CSB1093)

05.10.2020

CSE FINAL YEAR, UG

IIT ROPAR

INTRODUCTION

This report consists of how to generate Frequent Mining data. Here I have Implemented a program which would generate its own data Based on various **parameters** such as **number of transactions**, **size of maximal frequent itemset**, **average width of the transactions**, and **total number of items m** as input.

After **Data generation** I have used the Data as Input For three Mining Algorithms such as **Apriori** , **FPTree** , **Eclat**.

IMPLEMENTATION

I have created a Random Index array for inserting the max frequent Element. After this I have created another array of Total Transaction size to maintain the average using uniform distribution.

fun **generateData()**

Create a **data list**

generate a **random array of index of size FreqCount**

generate a **itemSet of MaxFreqSize**

Insert the itemSet at random Index in data list

Generate an other array of items count whose sum is equal to the **NO_OF_TRANSACTION * TRANSACTION_AVG_WIDTH - FREQ_COUNT * MAX_FREQ_ITEMSET_SIZE** to maintain the Avg Width of the Transactions.

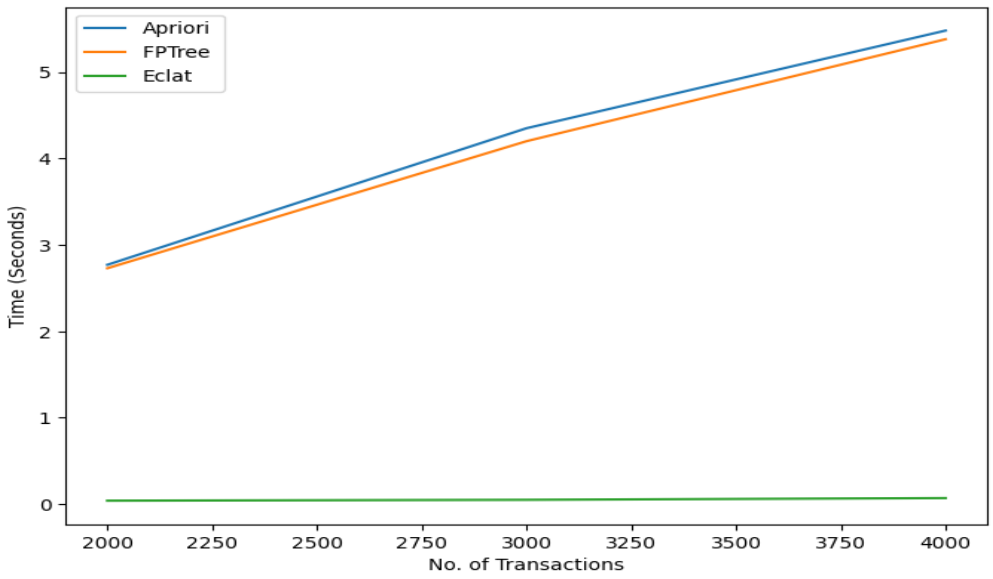
Insert item in the data list as per the above count using **Uniform Random distribution**

OBSERVATION

Lets See the Comparison Result For Different Dataset Generated.

1. Comparison Between Transaction and Run Time (In Seconds) In different Algorithms

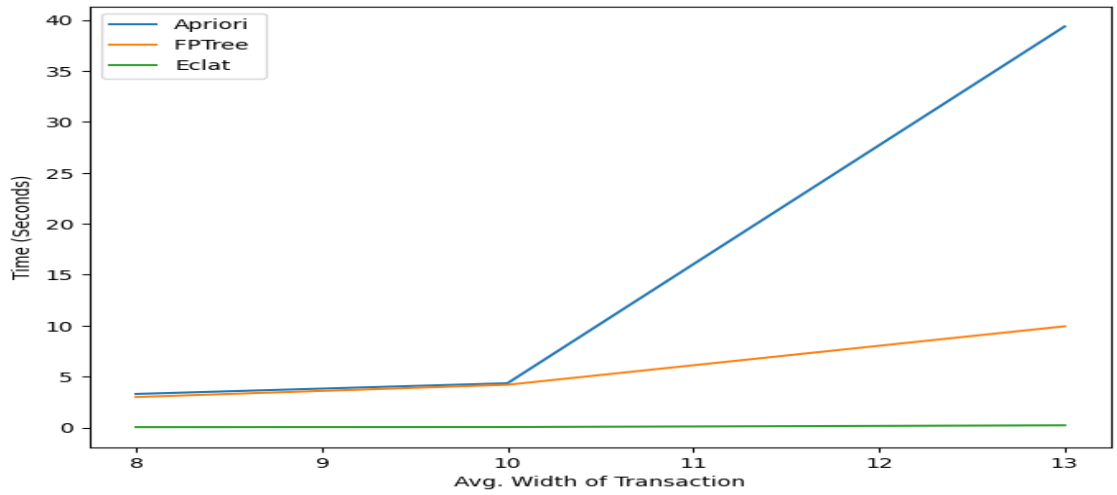
Transaction	2000	3000	4000
Run Time(Apriori)	2.77	4.35	5.48
Run Time(FPTree)	2.73	4.2	5.38
Run Time(Eclat)	0.04	0.05	0.07



It can be clearly seen from the above graph that Time Is Increasing For All the Algorithms as the number of transactions is increasing.But in **Apriori and FPTree** it is increasing very fast because In Apriori We have to traverse the whole transaction again and again which leads to high time complexity increase. And In FPTree It takes too much time in processing as we have to traverse for more transactions.

2. Comparison Between Average Transaction With and Run Time (In Seconds) In different Algorithms

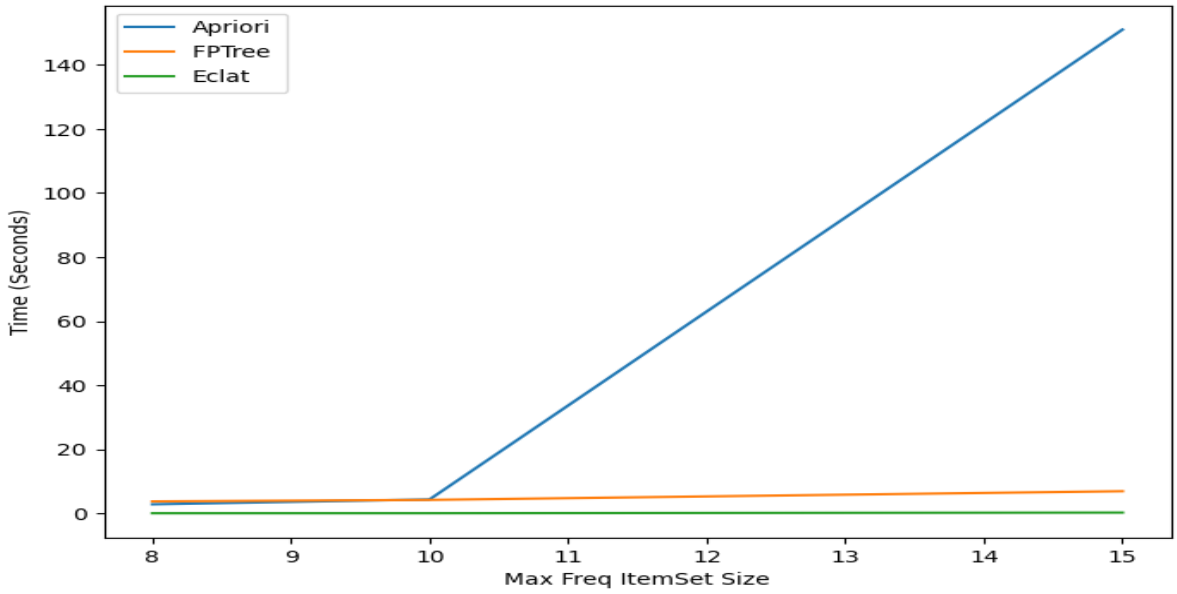
Avg Trans Width	8	10	13
Run Time(Apriori)	3.3	4.35	39.37
Run Time(FPTree)	3	4.2	9.94
Run Time(Eclat)	0.04	0.05	0.22



From the Above graph, It is clear that Time is Increasing with respect to Avg. Width of Transactions for All the Algorithms. But In Apriori It increases rapidly because time gets increased to find the frequent ItemSet in a Particular Transaction as transaction size gets increased.In FPtree the preprocessing times gets increased which results in increase in time.

3. Comparison Between Max Size Frequent Item Set and Run Time (In Seconds) In different Algorithms

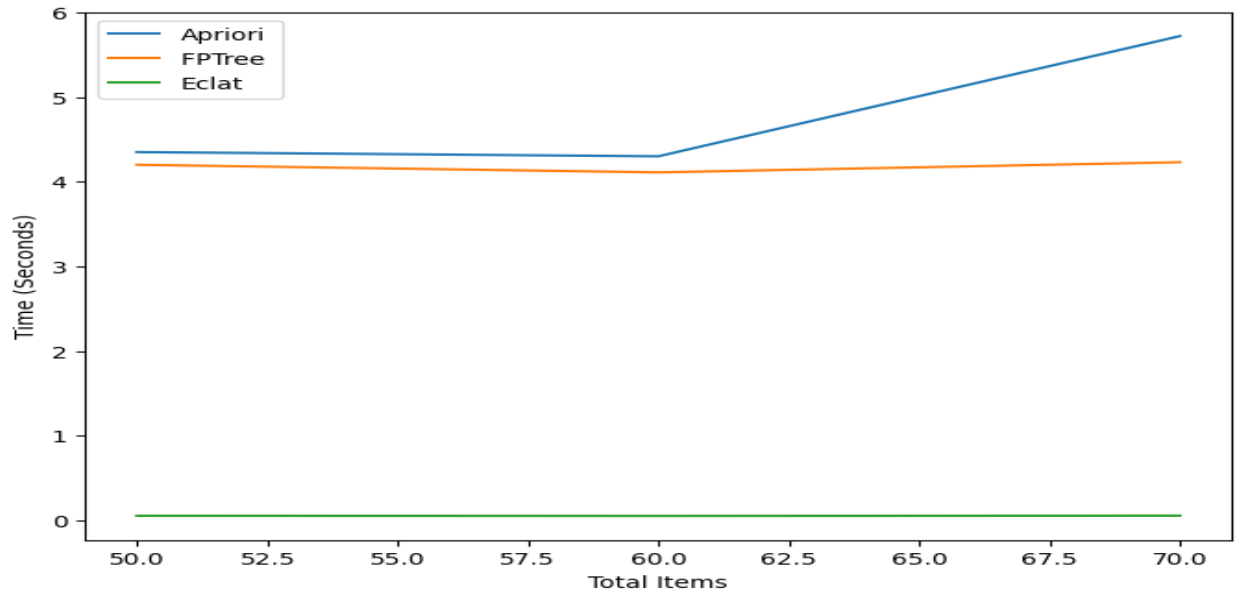
MaxFreqItem Size	8	10	15
Run Time(Apriori)	2.84	4.35	151
Run Time(FPTree)	3.74	4.21	6.92
Run Time(Eclat)	0.044	0.05	0.25



From the Above graph, It is clear that Time is Increasing with increase in Max Frequent ItemSet.But in Apriori it is rapidly increasing Because more the frequent ItemSet Size then Its obvious that their subset would also be frequent hence we have to find first all subset. That is why the time gets increased because of the increase in finding No. of Subsets.

3. Comparison Between Total Items and Run Time (In Seconds) In different Algorithms

Total Items	50	60	70
Run Time(Apriori)	4.35	4.3	5.72
Run Time(FPTree)	4.2	4.11	4.23
Run Time(Eclat)	0.052	0.05	0.22



From the Above graph, It is clear that Time is Increasing with increase in No of Items.

But it is higher in Apriori and FPTree. In Apriori , Increases in items cause an increase in width of the prefix tree which results in an increase in time rapidly. In FPTree, Increases in items cause The tree more Scattered which results in increase in time.

CONCLUSION

Observation From the Above Graph:

Time Complexity in Increasing Order for the given data on different Algorithms.

Eclat < FPTree < Apriori