University of Dhaka

Department of Computer Science and Engineering

CSE-4255: Introducing Data Mining and Warehousing Lab
4th Year 2nd Semester

Session: 2018 -19

Report Topic:

Comparison between Frequent Itemset Mining Algorithms: Apriori and FP-growth

Submitted by:

Amit Roy, Roll: JH- 40
Md. Tanvir Alam, Roll: SH-61

Submitted to:

Dr. Chowdhury Farhan Ahmed, Professor,
Department of Computer Science and Engineering, University of Dhaka

Abu Ahmed Ferdaus, Associate Professor,
Department of Computer Science and Engineering, University of Dhaka

Date of Submission:

26 August, 2019

Introduction:
Mining frequent patterns from a transaction database where a user defined support threshold is given is a well-studied problem in data mining. Two famous algorithms proposed for frequent pattern mining are Apriori and FP-growth.

In Apriori algorithm, the frequent 1-itemsets are found by scanning the database. A set containing frequent k-itemsets is denoted as L_k. Patterns in L_{k-1} is joined to find candidates for L_k. To improve the efficiency the algorithm reduces the search space. This is done by using a special property called “Apriori Property” or “antimonotonicity”. According to this property, Any (k-1)-itemset that is a nonempty subset of frequent k-itemset must be also frequent.

A candidate C will not be a member of frequent k-itemset L_k if any of its (k-1)-itemset is not frequent. So every member C is tested and added to the frequent k-itemsets L_k if it satisfies the minimum support threshold. The naive version of the algorithm needs to scan the entire database to test candidate itemsets. To improve efficiency, we have used trie data structure. Using this trie data structure requires the database to be scanned once per level and improves the runtime efficiently.

Apriori-based candidate-generate-and test method generated a large number of false candidates. It is costly to test more candidates. To solve this problem, FP-growth, a tree-based frequent itemset mining algorithm is proposed that eliminates the problems of candidate generation and testing method. In our experiment, we have run both Apriori and FP-Growth algorithm on several transactional databases and compared the performance.

Dataset Description:
In this section, we present the description of datasets. We have used 9 datasets from (http://fimi.uantwerpen.be/data/) named "T10I4D100K", "chess", "connect", "mushroom", "pumsb", "pumsb_star", "kosarak", "retail", “accident”. We have calculated total transaction count, average transaction length, distinct item count and density for every dataset. We defined the density of the dataset as the ratio of average transaction length and distinct item count. A dataset is dense if it’s density is over 10%, otherwise it is sparse.

In figure 1-4, we presented total transaction count of different datasets, distinct item count of different datasets, average Transaction Length of different datasets, density of different datasets respectively.

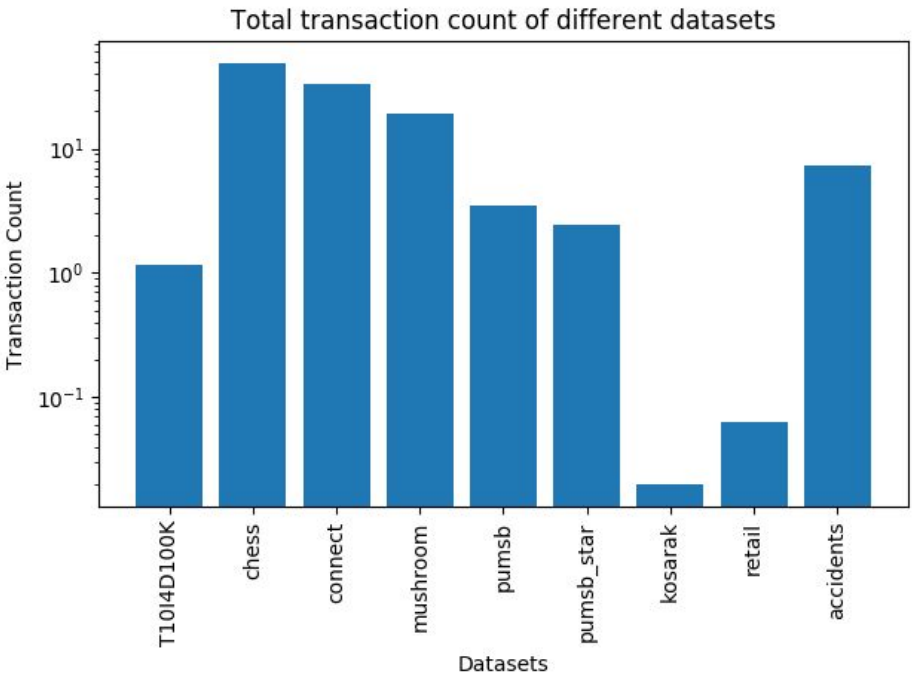


Figure 1: Total transaction count of different datasets

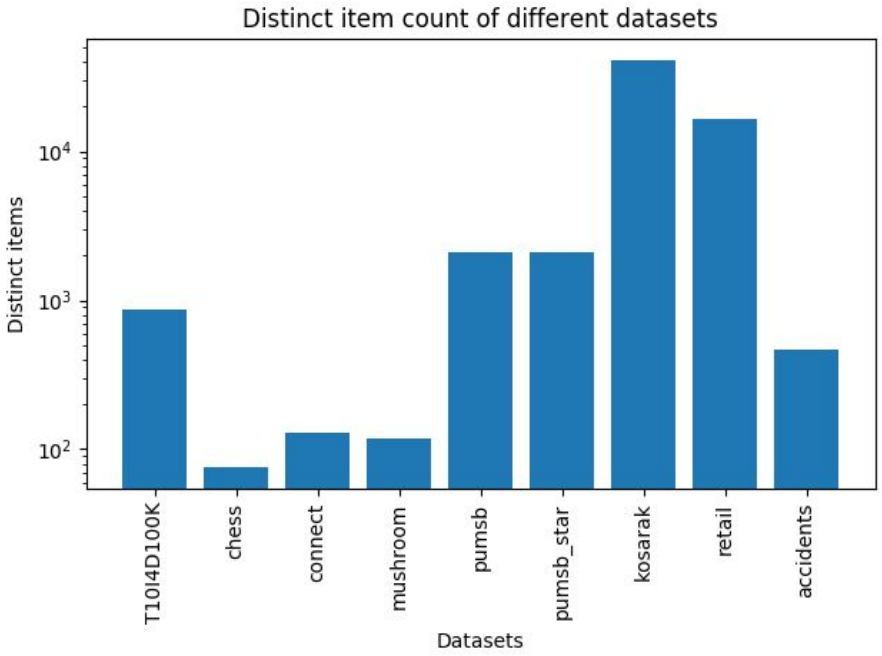


Figure 2: Distinct item count of different datasets

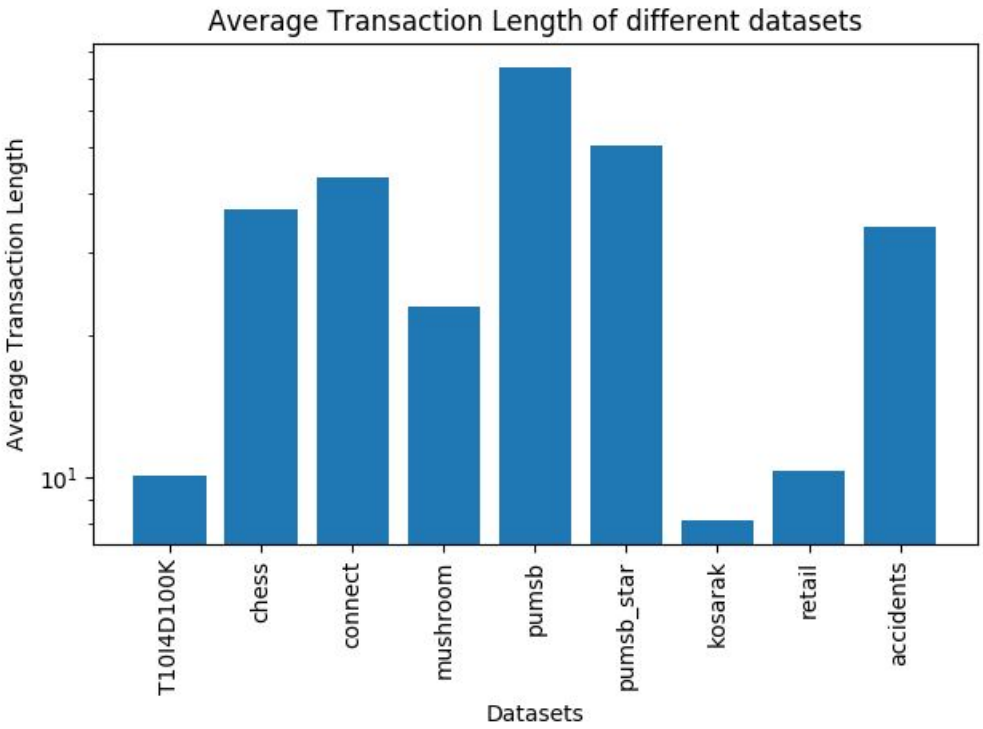


Figure 3: Average Transaction Length of different datasets

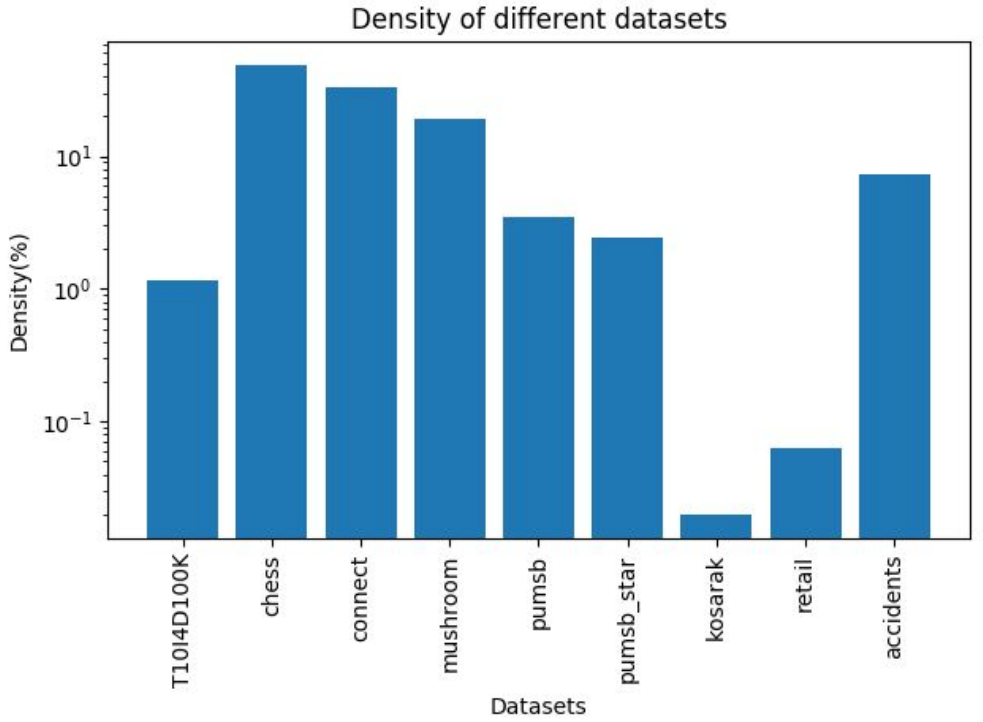


Figure 4: Density of different datasets

Implementation:
To implement the apriori algorithm, we used a trie-based approach in each level to update the support count candidates of that level, otherwise we had to scan the entire dataset for updating the support count of a single candidate. This modification in updating support count improves the running time performance of the apriori algorithm.

Since FP-growth does not blindly generate candidate like apriori and scans the database only twice, once while sorting the itemsets in decreasing frequency order and once while building the FP-tree. As we can observe from the comparison graphs, the curve of FP-growth is well below the curve of Apriori which is an evidence that FP-growth algorithm shows significantly better runtime performance than Apriori algorithm for mining frequent itemsets. As the user given minimum support threshold decreases the running time improvement becomes more clear from the comparison graphs.

Experimental Result:
To compare the running time analysis of Apriori and FP-Growth algorithm we have calculated the runtime of algorithms for different datasets and for different thresholds according to the density of the dataset and then plot them to visualize the performance difference. In figure 5-13, We present the runtime comparison of Apriori and FP-Growth on 9 different datasets for various minimum support thresholds.

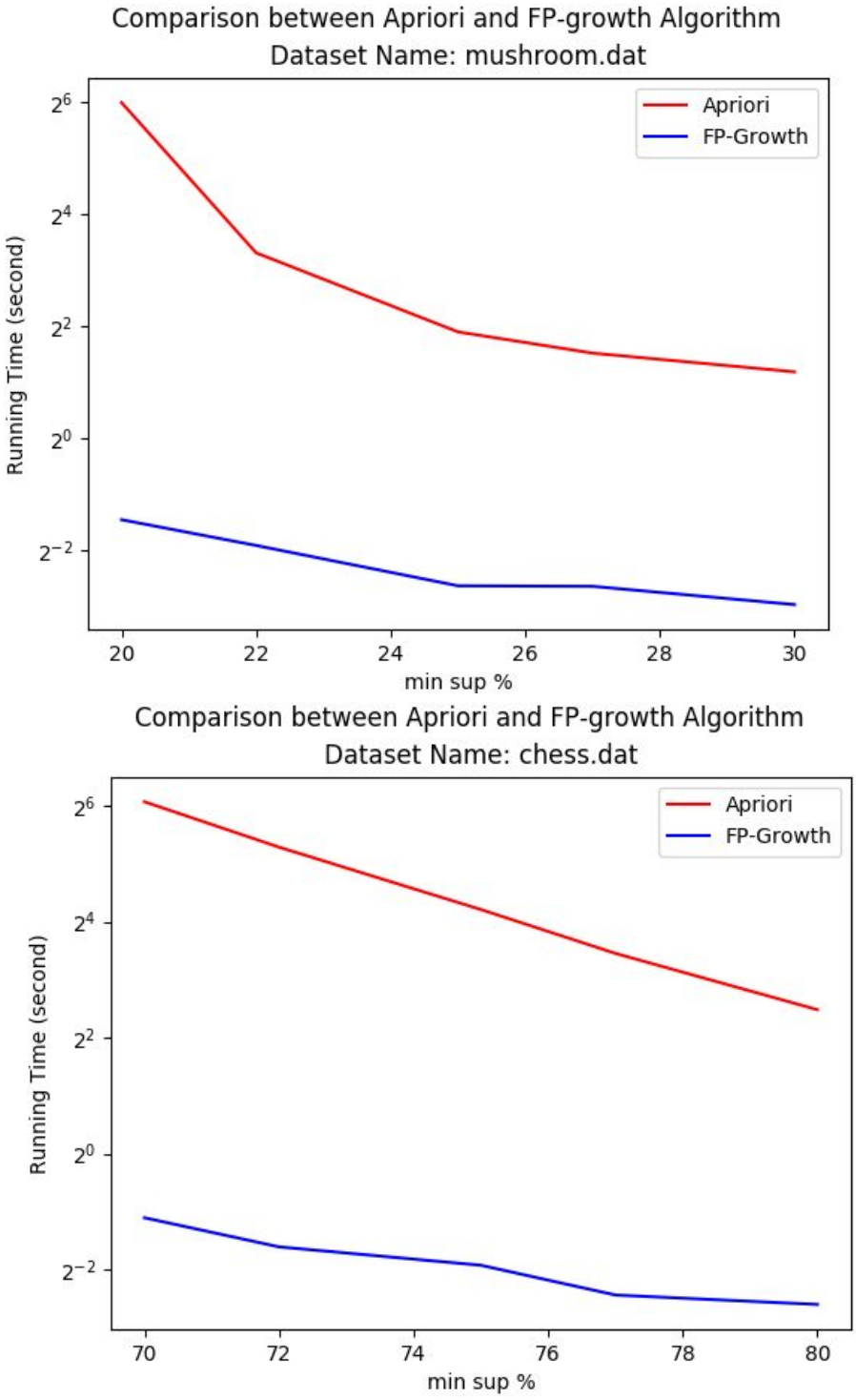
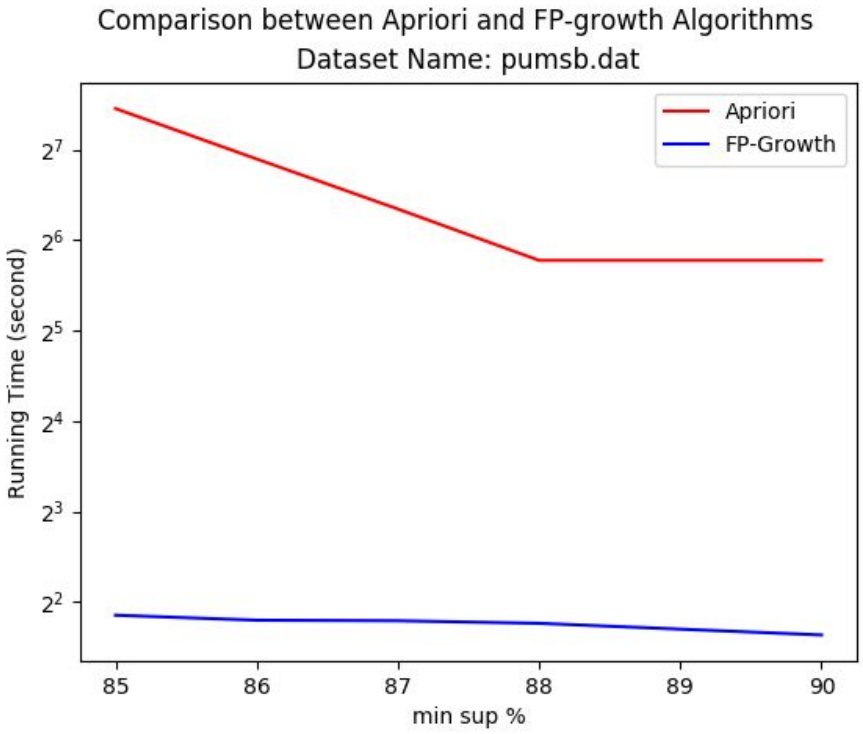


Figure 5-6: Comparison between Apriori and FP-growth Algorithm on dense dataset mushroom and chess



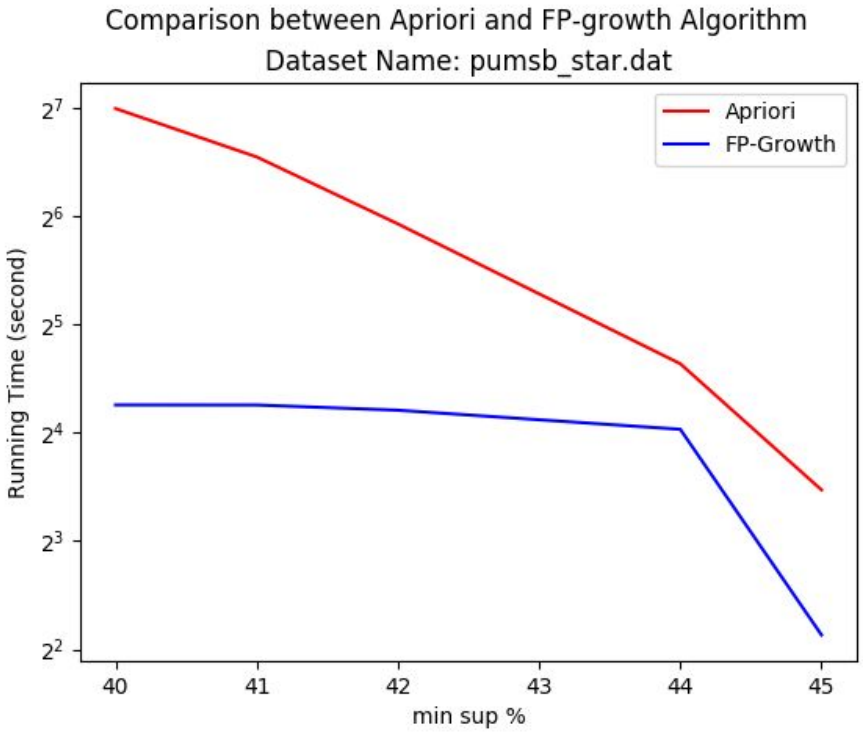


Figure 7-8: Comparison between Apriori and FP-growth Algorithm on dense dataset pumsb and pumsb_star

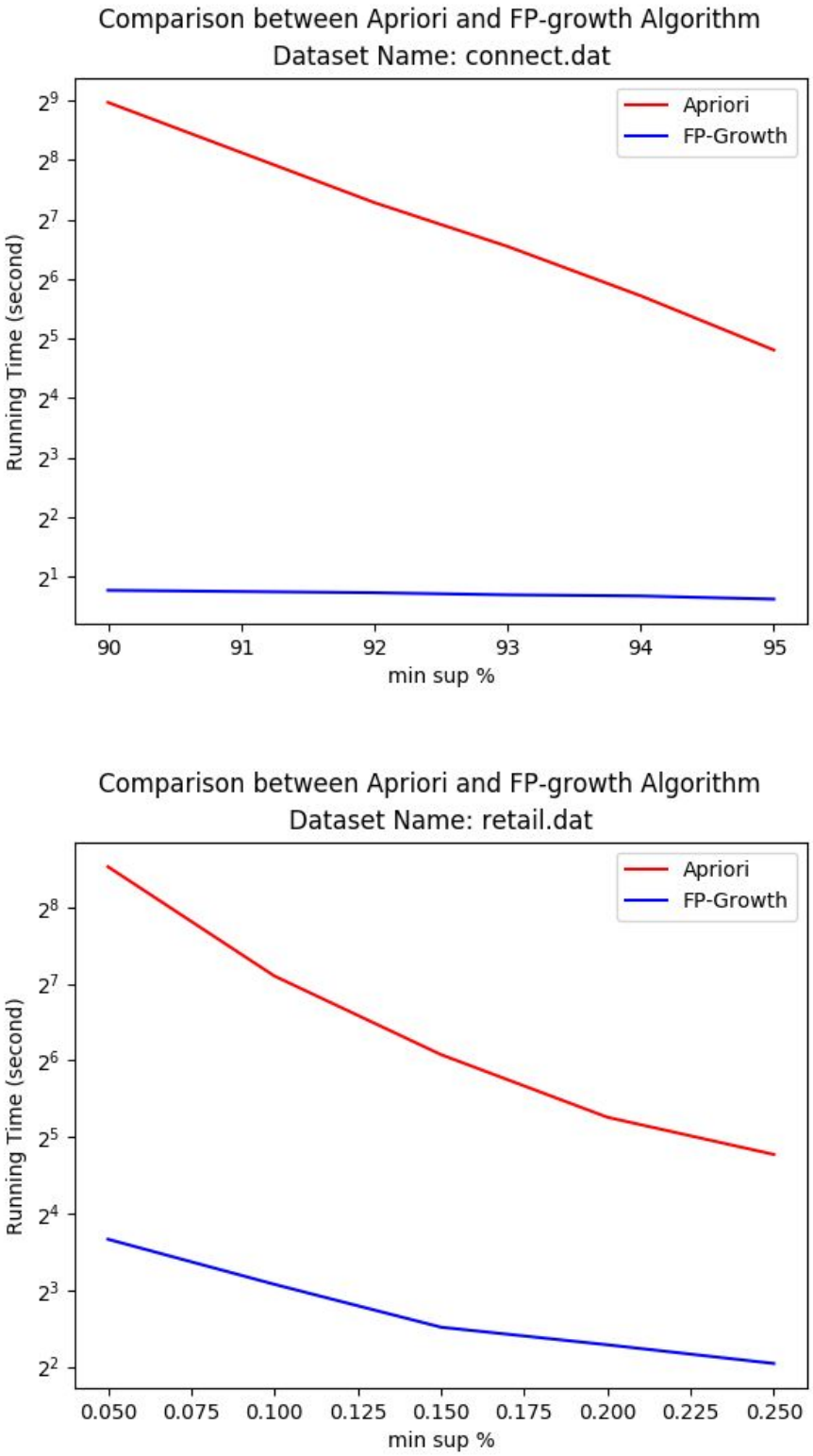
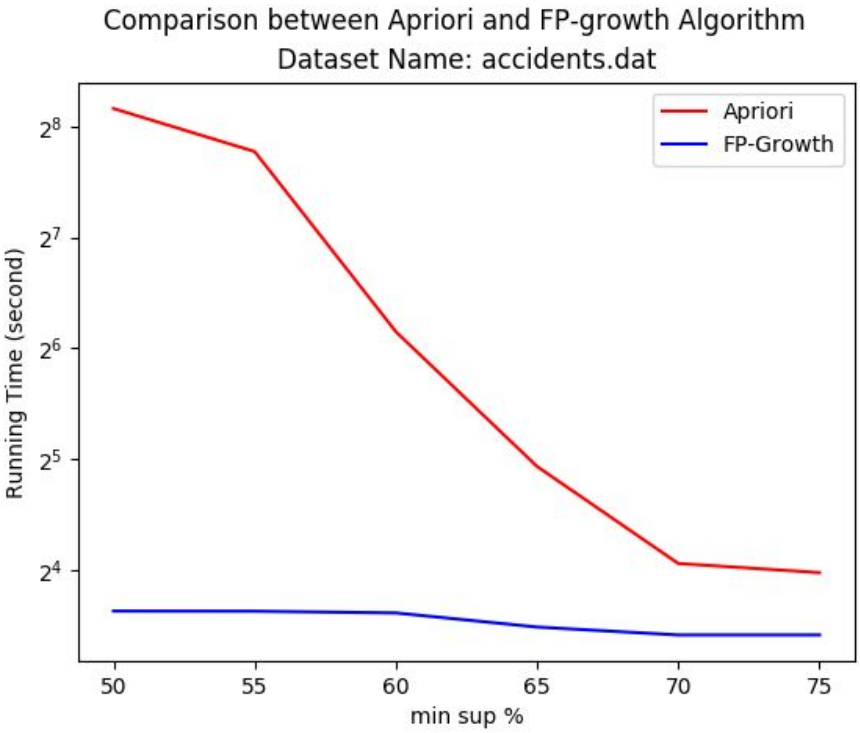


Figure 9-10: Comparison between Apriori and FP-growth Algorithm on dense dataset connect and sparse dataset retail



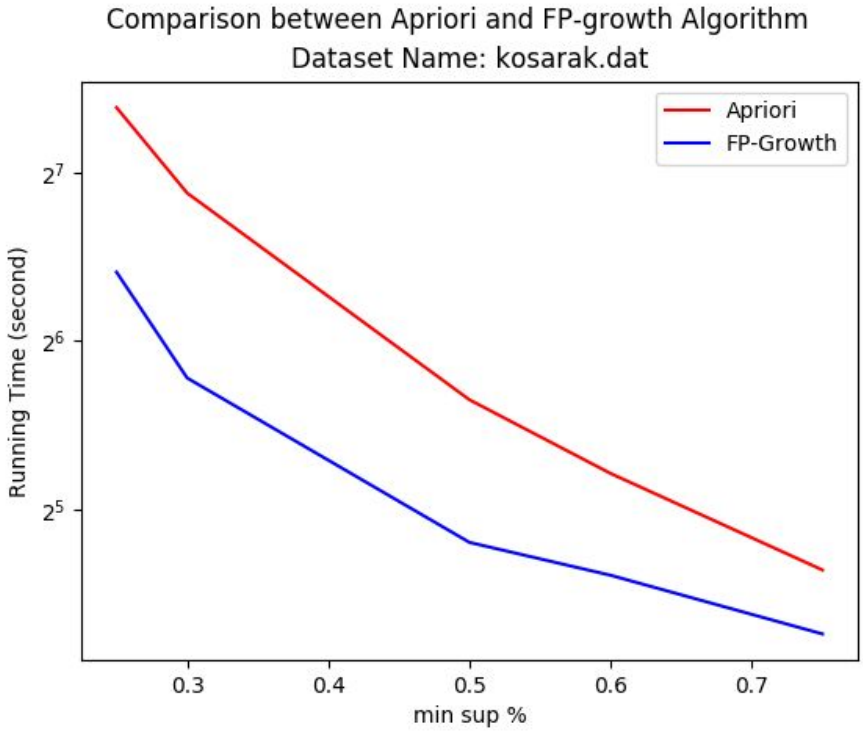


Figure 11-12: Comparison between Apriori and FP-growth Algorithm on large dataset accident and sparse dataset kosarak

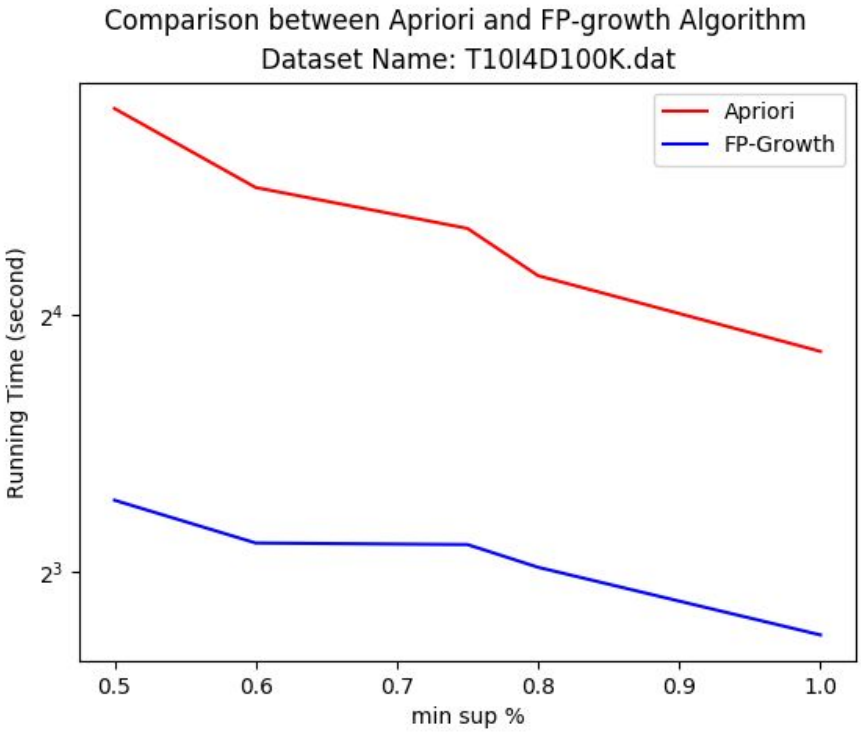


Figure 13: Comparison between Apriori and FP-growth Algorithm on sparse dataset T10I4D100K

In figure 15-19, we have presented the memory usage of two implementations on different datasets. Memory usage is defined by the sum of virtual memory(VM) and RAM allocated memory(RSS).

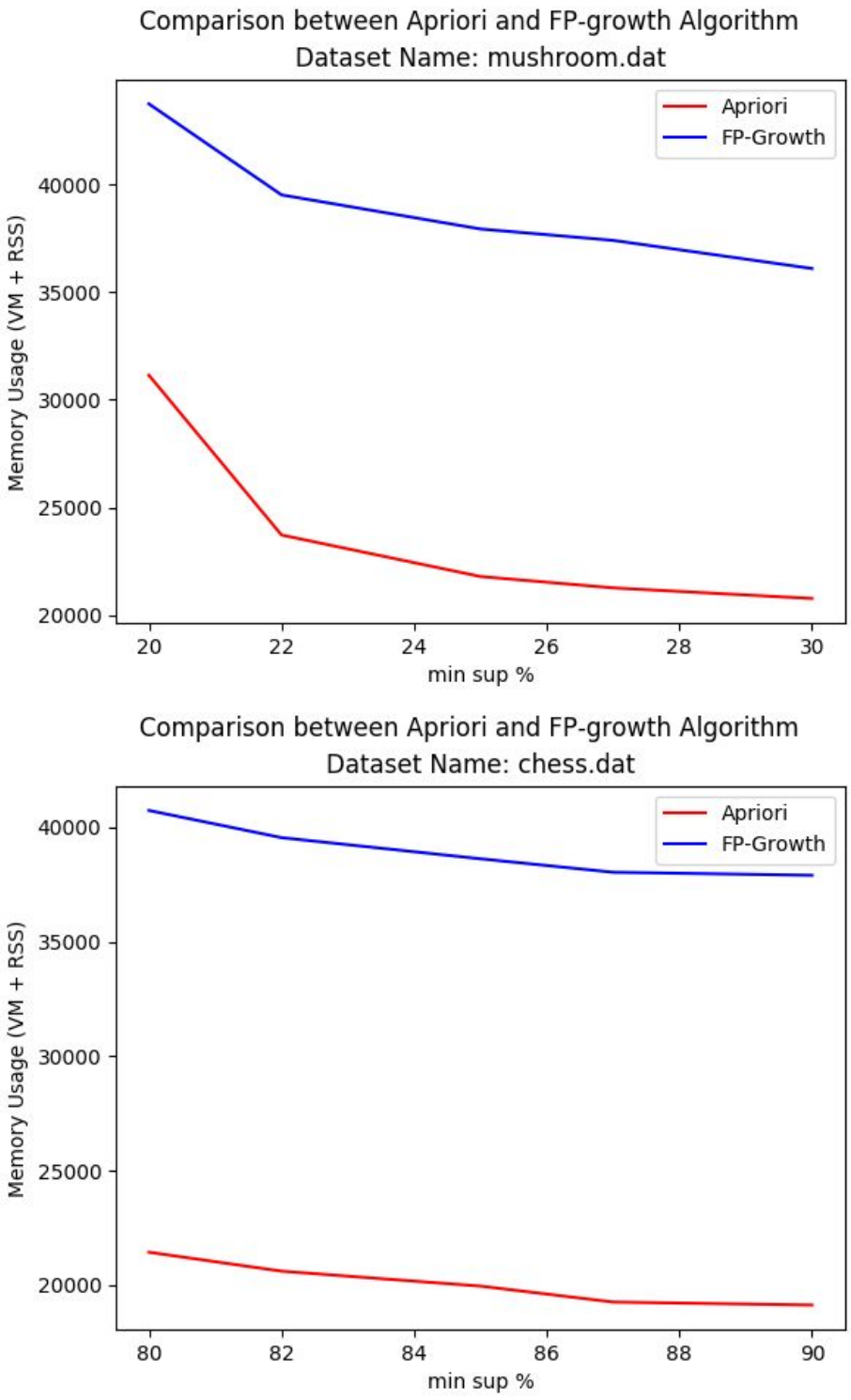


Figure 15-16: Comparison of Memory Usage between Apriori and FP-growth Algorithm on Dense Dataset mushroom and chess

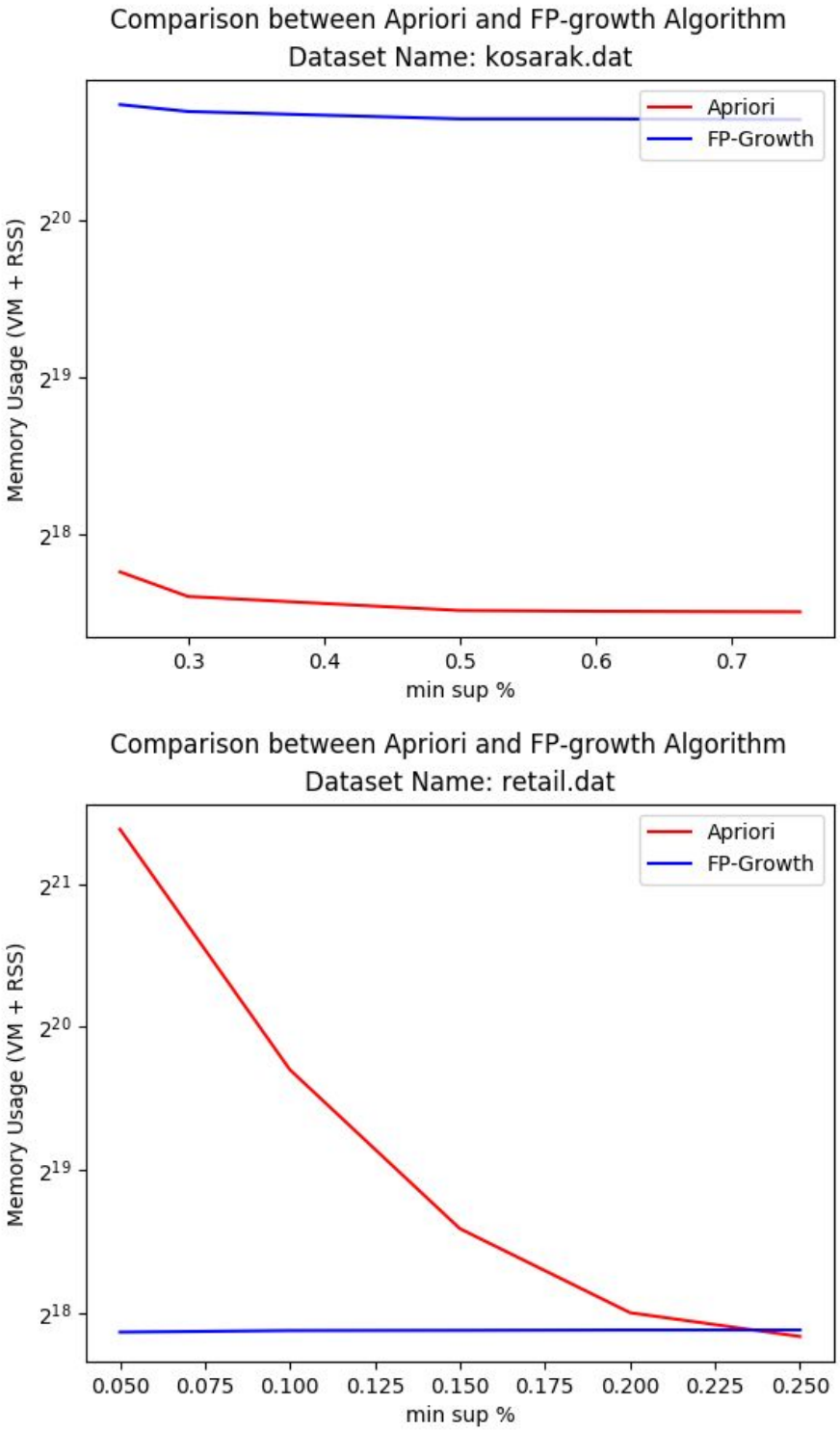


Fig 17-18: Comparison of memory usage between Apriori and FP-growth algorithm on sparse dataset kosarak and retail

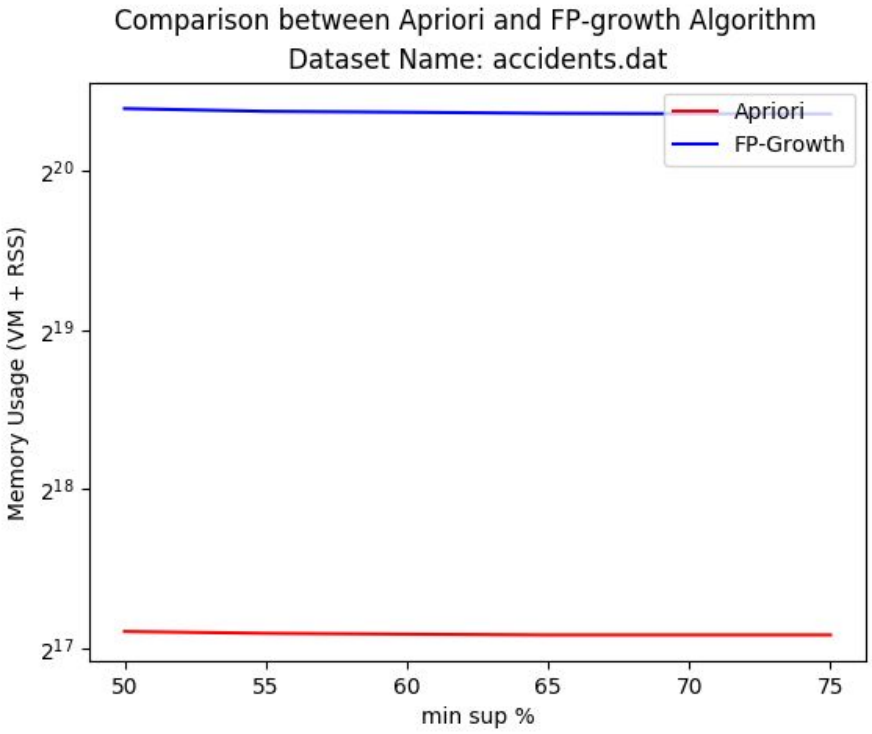


Fig 19: Comparison of memory usage between Apriori and FP-growth algorithm on dataset accident

Conclusion:

From our experimental results, we can say that FP-Growth outperforms Apriori. FP-Growth gets more efficient comparatively as minimum support threshold decreases. This is because as minimum support threshold decreases, the “Apriori Property” based pruning gets less effective and generates more false candidates.

We can also observe that the memory usage of Apriori algorithm is significantly less than FP-growth, the reason behind that is FP-growth algorithm creates a large number of conditional tree for each level which increases the memory usage.