**1.**

A python script (part1.py) is written to run all the algorithms 1) Apriori, (2) FP-Growth, and (3) ECLat for different minimum supports. Result graphs for computation time for each algorithm is part1/result.pdf

**2.**

Results for different runs for (1) FP-Growth, (2) FPClose, and (3) FPMax are saved in part2/results.xlsx.

**3.**

Implemented MSAprori (with spmf library as reference) with following constraints:

- Multiple minimum support

- Support difference constraint, and

- Item constraint

To run the algorithm run the file RunModifiedMSApriori.java in customMSApriori package.

Use following commands in current folder

* javac customMSApriori\\*.java
* java -cp . customMSApriori.RunModifiedMSApriori

**4.**

Modified MSApriori to take the SPD difference constraint in to consideration and changes are added to **AlgoMSApriori.java** in customMSApriori package.

Use the following commands to run and give support difference constraint in RunModifiedMSApriori.java

**5.**

1. These are the top 5 item sets with at least size 2 and 3 (calculated for retail1.txt with FPgrowth for min support : 0.01)

*1534 1943 #SUP: 29159*

*225 1816 #SUP: 8295*

*1097 1534 #SUP: 5714*

*225 1816 1834 #SUP: 6959*

*1816 1834 #SUP: 20265*

These can be called “cannot-be-together” item sets. For “must-have” we take item with least number in each set and take union of them

So, it’ll be

*{ 1534, 225, 1097, 1816}*

1. For below run below parameters are used:

minimum support parameter 𝛿=0.5

support difference constraint 𝜑=5%

Both these parameters are passed in RunModifiedMSApriori.java.

By using above can’t be together and must-have lists, here is the computed list of frequent item sets with its support and minimum item support

Results are in **part5/b.txt**

1. By removing item constraints in part3 algorithm, we’ll get same frequent itemsets for both runs in part3 and part4 algorithms because both algorithms runs on same minimum item support and support difference constraints (𝛿=0.5, 𝜑=5%)

Output:

============= MSAPRIORI - STATS =============

The algorithm stopped at level 4, because there is no candidate

Frequent itemsets count : 105

Maximum memory usage : 147.49331665039062 mb

Total time ~ 72590 ms

Results are in **part5/c.txt.**

1. In part3 algorithm, these are the stats for computation time.

Without item constraints : 72590 ms

With item constraints : 73463 ms

Using the library we get the computation time as 69451 ms.