****

WILL TAKE TIME…A LOT OF IT

To be done in groups of 2 --- NOT INDIVIDUAL

****

Implement the *Apriori* algorithm to find all frequent itemsets given a support threshold (as a percentage between 0 and 100) as input.

a. Your program should take as input
(1) data file containing transactions, each on a separate line, with items separated by spaces, and
(2) support percentage threshold (***minsupp***) a decimal number between 0 and 100,

b. Your program should output all frequent itemsets into a text file called **`FIS.out`** with the following format:

`FIS.out`: e.g.

```
1 10 14 (0.89)    //items are ordered ascendingly and are SINGLE SPACE separated
22 31 45 (0.09)
…
```

Please pay close attention to the format of your input and output files.

c. Subfolder **`datasets`** contains a number of popular ARM datasets that you will use for this project. First, test your implementation on **`simpledataset.dat`**. Generate all frequent itemsets for different thresholds (by hand or use Weka) in order to make sure that your implementation is producing the correct results.

d. Experimental study: Once you are confident that your implementation is working properly, proceed to conduct an experimental study that shows how your frequent itemset mining step performs on real-world datasets. The table below shows four popular datasets (full descriptions can be found on http://fimi.cs.helsinki.fi/data/) available inside subfolder **`datasets`**.

Dataset name	*Description*	*# of items*	*Avg. length*	*# of transactions*
`chess.dat` **(334.3KB)**	a set of chess moves --- Game domain theory and inductive reasoning	75	37	3,196
`retail.dat` **(4.0MB)**	market basket data from an anonymous Belgian retail store	16,470	10.3	88,126
`accidents.dat` **(33.9MB)**	traffic accident data from a Belgian study on car accidents	468	33.8	340,183
`kosarak.dat` **(30.5MB)**	click-stream data for a Hungarian on-line news portal	41,270	8.1	990,002

All datasets, except for **`retail.dat`**, have their items numbered starting from 1 up to the total number of items (e.g., in **`chess.dat`**, items are named 1, 2, 3, … 75); items in **`retail.dat`** start at **`0`**. The datasets do not record other information directly; for example, if you need the total number of items or the total number of transactions, you will have to scan the dataset. Please recall that I WILL RUN YOUR PROGRAM ON OTHER DATASETS THAT YOU HAVE NOT SEEN --- Your program should run smoothly ON ANY TRANSACTIONAL DATASET formatted as described above.

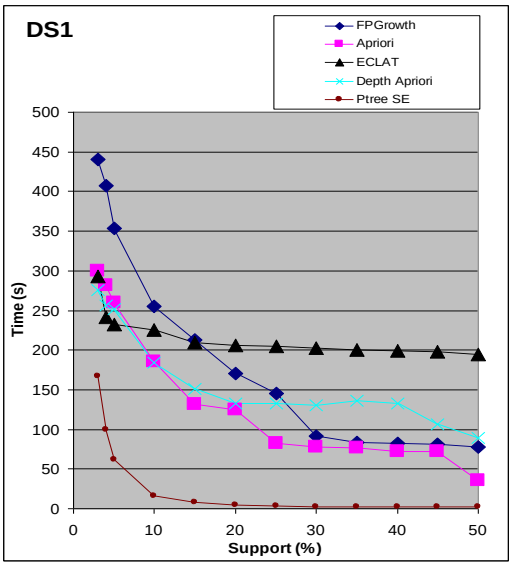
In ARM, experimental studies focus on showing the effect of varying the support threshold on the execution time needed. The two tables below show what support thresholds to use for testing your implementation on each dataset (shown below in percentage as well as actual numbers). *Dense* datasets contain longer transactions (e.g., containing a large number of items) and tend to produce large frequent itemsets, unlike *sparse* datasets. This explains why I have chosen high support thresholds for the dense datasets and very low support thresholds for sparse ones.

Dense dataset name	95%	75%	55%	35%
chess.dat	3036	2397	1758	1119
accidents.dat	323174	255137	187101	119064

Sparse dataset name	1%	0.75%	0.50%	0.25%
retail.dat	881	661	441	220
kosarak.dat	9900	7425	4950	2475

For each of the above datasets, run your implementation for the specified support thresholds and record
(1) the execution time it took to complete (in seconds), and
(2) the total number of frequent itemsets generated.

Find another team to compare execution times. If you prefer not to work with other teams, compare your performance against any of the implementation shown on this page: *http://fimi.ua.ac.be/src/* (**you will have to download the code and figure out how to run it on your machine**). Use descriptive graphs, such as the one below, to highlight the effects of varying the support threshold on a given dataset for different algorithm implementations (in your case, you will have only two implementations: yours and the one you are comparing against). You will need one such graph per dataset. Also, include tables and similar graphs showing the effect of varying support on the total number of produced frequent itemsets.



****

In addition to the items requested earlier, your report MUST include a 1-page description detailing what EACH MEMBER did along with a percentage value describing the member's share of the work. If you disagree with your percentage, I ask that you meet with me ASAP.

Save your report inside a folder with your last names (e.g., MILLER_DRAKE_PROJECT3) and submit to
/usr/people/handins/CS332/

Please note that your program should work for ANY DATASET formatted as described above. **DURING THE IN-CLASS DEMO, YOU WILL BE ASKED TO RUN YOUR PROGRAM ON DATA FILES THAT YOU HAVEN'T SEEN AND TO PROVIDE EVIDENCE THAT YOUR IMPLEMENTATION UTILIZES THE ANTI-MONOTONE PRUNING STEP THAT PRECEDES SUPPORT COUTING.**

****