COL 761 - Homework 1

- You need to do the homework in your already formed team of 3. Make sure to upload your code to the GitHub repo you mentioned in HW0.
- Due: **26th August 11:59PM.**
- Your code must compile and execute on HPC. You may get 0 for compilation or execution errors
- No multi-threading or parallelization is allowed.
- **Do not copy code from your friend or from the internet. We have previous year's submissions as well as all available libraries of Apriori algorithm and FP-Tree from the web and all submitted codes will be checked against these as well as those submitted in this homework. Any plagiarized code will result in an F grade for the course.**
- **30% penalty for each late day**

Submission Instructions:

- Add **col761-22** as a collaborator for your private repository. (should be done as part of HW0).
- There should be **only one** zip uploaded per assignment in the github repo.
- Upload HW1_<RollNo>.zip file to the root directory of your GitHub repository. This RollNo should be of one of your team members. Ex. HW1_MCS162913.zip. On unzipping it should **produce one folder** of the same **name as the zip file.** This folder should contain all the source files (your code for the assignment) **&** 2 bash scripts (<RollNo>.sh and compile.sh). **DETAILS ON THE 2 BASH SCRIPTS BELOW.**
- In addition, the unzipped folder should also contain a README.txt explaining all the files you have bundled, explanation of q4(details below) and entry numbers and names of **all team** members. The text file must always mention the **contributions of each student** in the assignment. Additionally it should mention what is the **% contribution per student** to the assignment. Marks will be assigned to individual members based on their percentage contribution.
- For example, if someone has contributed 20%, then his/her marks will be 20/33*(marks obtained). However, no extra marks will be given for contributing more than 33%. In case of disputes, we will go by majority vote within that team.
- **Make sure to use the same roll number to give the name of all submission files (HW1_<RollNo>.zip , RollNo.sh), etc.**
- You also need to submit another script (**<RollNo>_install.sh), ONLY on Moodle (Do not zip it). Only one per team. Multiple submissions will be penalized.**
- There should be only **one** submission per group (on github and Moodle).
- **Testing shall be done on different dataset. Make sure not to hardcode the dataset name anywhere. Always read the dataset name from the command line as shown in examples. All files/datasets must be stored in the base folder only.**
- Since your submissions will be auto graded, it is essential to ensure your submissions conform to the format specified.
- Latest timestamp of your last pushed commit and Moodle submission of install.sh will be treated as your submission time.

---

1. [**5 points**] Mention your GitHub repo. Make sure that this is the same GitHub repo as what you mentioned in HW0.

2. [**20 + 30 points**] This question is on frequent itemset mining. Implement Apriori and FP-Tree Algorithm to mine frequent itemsets. Apply it on the Dataset: http://fimi.uantwerpen.be/data/webdocs.dat.gz. You may assume all items are integers.

   [NOTE]: All **Files for which roll number is required, must be named after the same member for all homeworks.**
   Create a bash file **<RollNo>.sh.** For example, if MCS162913 is your roll number, your file should be named **MCS162913.sh**.
   Executing the command **"./<RollNo>.sh -apriori <dataset_name> X <outputFilename>"** should generate an output text file containing the frequent itemset at >=X% support threshold with the Apriori algorithm. Similarly **"./<RollNo>.sh -fptree <dataset_name> X <output filename>"** should use FP- Tree. **Notice that X is in percentage and not the absolute count.**

   IMPORTANT: Your implementations must ensure that the transactions are **NOT** loaded into main memory. This means that it is **not allowed** to parse the complete input data and save it into an array or similar data structure. However, the frequent patterns and candidate sets can be stored in memory.

   The output file should strictly follow the following format follow the below rules:

   i. **Each** frequent itemset must be on a **new line.**
   ii. The items in the itemset must be **space separated**.
   iii.All itemset must be sorted in **ascending order lexicographically.**

   Your grade will be based on the following formula: (F-score)*20 and (F-score)*30.
   Example:
   - there are **M** itemsets in the correct output (C_1, …, C_M)
   - Your output has **N** itemsets (I_1, ..., I_N)
   - Further, your output has **n** of the correct itemsets (I_α=C_ω, …, I_β=C_Σ) **&** size({I_α, …, I_β})=n)
   - We calculate precision and recall and F-score as below:
     A. Precision(p) := n/N and
     B. Recall(r) := n/M.
     C. F-score is F := 2pr/(p+r) = 2*n/(M+N).
     D. Read more about F-score here(https://en.wikipedia.org/wiki/F-score).

   In order to help you get the output format right, you'll be provided with a few small dummy transaction datasets, along with expected output and the script that will be used to check whether your output file matches the expected output file.
   Run it as follows:

```
,----
| $ python3 checker.py --expected out_30.dat_sorted --output your_output.dat
`----
```

Note: this code just performs one-to-one checks of the output and only works if the output is properly sorted(lexicographically). You can use this to check if your output is in the right format. The F-score calculating script is not released, we'll use it to calculate your final score. For more details about the correct format, go to Q3.

3. [**15 points**] Compare the performance between your implemented Apriori and FP-Tree algorithms.

   Executing the command **"./<RollNo>.sh -plot webdocs.dat <output_filename>"** should generate an output ".png" file with name as specified in command line argument. Output file contains a plot using matplotlib where the x axis varies the **support threshold,** and the y-axis contains the **corresponding running times**. It should plot the running times of FP-tree and Apriori algorithms at support thresholds of **5%, 10%, 25%, 50%, and 90%**. Explain the results that you observe. You may add a timeout if your code fails to finish even after 1 hour.

Bash scripts you need to provide:

- **compile.sh** that compiles your code with respect to all implementations. Specifically, running ./compile.sh in your submission folder should create all the binaries that you require. Any optimization flags like O3 for g++ should be included here itself
- <RollNo>.sh as specified earlier
- install.sh(**ONLY ON MOODLE**) that should **execute**
    1. cloning of your team's submission in the current directory, followed by
    2. executing bash commands to load all the required HPC modules.
    3. Inside the repository we should be able to locate HW1_<RollNo>.zip corresponding to your Homework 1 submission.
    4. Note that this script will be run as **source install.sh** (to make use of HPC module load alias).

Compiler Specification:

- GCC version 7.1.0
- Java version 1.8
- Python3 version 3.6.5
- Python2 version 2.7.13

Following steps will be used to evaluate your code :-

1. install.sh script will be downloaded from moodle.
2. This will be run as

   ,----
   | $ source install.sh
   `----

   Running as source keeps the environment variables created by script and changes directory commands executed by a script in the current scope.

   (https://superuser.com/questions/176783/what-is-the-difference-between-executing-a-bash-script-vs-sourcing-it)

3. Expected output after running of this script :-
   - Your github repository will be cloned.
   - Directory will be changed to your github repo's directory.
   - The correct HW1 zipfile will be unzipped in the directory.
   - HPC modules required by your script for compiling the source will be loaded.

4. compile.sh (if exists) will be run. Expected result :-
   - Your source code will be compiled into an executable binary.

5. <RollNo.sh> will be run with appropriate arguments, it should make use of your compiled binary to give the correct outputs based on the arguments provided to the <RollNo.sh> command (Refer Q3). This output can be used with the sanity check code to check for the correct format.
   For Q4, the correct plot will be checked against the generated png.

An example install.sh file might contain the following lines:

```
1 #!/usr/bin/bash
2 git clone https://github.com/myuser/myrepo
3 cd myrepo
4 unzip HW1_MyRollNo.zip
5 module load compiler/python/3.6.0/ucs4/gnu/447
6 module load compiler/gcc/7.1.0/compilervars
7 g++ -O3 -Wall -o mybinary mysource.cpp
~
~
~
~
~
~
~
```

GOOD LUCK