COL761 (Session 2023-2024) Assignment 1

Instructions

- Submission deadline is **26 Jan. 23:59 IST 2024**
- You need to do the homework in your already formed teams. You will upload the code to the GitHub repository mentioned in A0 (or HW0).
- **Your code must compile and execute on HPC.**
- You will submit a script to **moodle**. Details are in section 4.
- Do not copy the code from your friends or from the internet. **Plagiarism will result in -X marks where X is the total of the assignment.**
- You may choose any of the following languages to write your code in: C, C++, Python.
- Please read the full document clearly.

1 Problem statement

Dataset compression plays a curcial role in modern data management and analytics. As the volume of ata generated and collected continues to skyrocket, the need for efficient storage and processing becomes more pressing. While reducing the size of the dataset, it is essentail to ensure that the compression is lossless, i.e., the original dataset can be retrieved efficiently form the compressed dataset without any loss in information.

You are given a large transactional dataset D. Each row (called a transaction) of the dataset contains a list of items (I1, I2, . . .) that were purchased together by some users. The task is to compress this dataset D, by reducing repetition. A natural candidate for this dataset is to use frequent itemset mining.

Example 1

Consider the following dataset as example –
T1 – A, B, C, D, E
T2 – A, B, C, D, F
T3 – A, B, C, D, E, G
T4 – A, B, C, D, E, F, G

Here T1, T2, T3, T4 are the transaction IDs and A, B, C, D, E, F, G are items.

Let us say we create the following mapping to re-label some itemsets –
X: {A, B, C, D},
Y: {E, G}

The new dataset becomes –
T1 – X, E
T2 – X, F
T3 – X, Y
T4 – X, Y, F

The storage cost is taken as the total number of items across all transactions in the dataset and the number of keys and items in the decoder mapping. For example 1, the size of the original dataset is 23. The size of the mapping table is 8, and the size of the compressed dataset is 9. Hence, we are able to reduce the storage size from 23 to 17

(8 + 9). This corresponds to roughly 26% compression. Note that creating the mapping differently will lead to a different compression rate.

To reconstruct the original data, the mapping can directly be used while decoding the compressed dataset. Note that a decompression is considered lossless if and only if the set of items in the original transaction is exactly the same as the set of items in the decoded transaction.

Given the dataset D, your task is to provide the compressed dataset D$'$ and the decoder mapping M, such that the compression is completely lossless.

Note, in order to verify that the compression is lossless, you must also provide a decompression code that will re-generate the dataset given just the compressed mapping dataset.

You'll be working on datasets found here[1].

2 Important points

- Your dataset will consist only of non-negative integers. Think of them as item IDs. Duplicate transactions are possible. Duplicate items in transactions are also possible. Your output should also contain just integers (or long long) and spaces. No other characters are allowed.
- **Your output should be just one file containing both the new dataset and the mapping.** How to combine them is upto you, a suggestion is to store some meta-data in the header. One such organization could be as shown in the figure below.
- Definition of size: as informally defined in the example above, we will count the number of space separated tokens! Thus, the size of the dataset is the number of space separated tokens in the dataset. Thus,

$$\text{compression ratio}(D) = \frac{\text{size}(D) - \text{size}(D')}{\text{size}(D)}$$

- Your output file should have a compression ratio of $\geq 10\%$! **If its size is larger than that of the original dataset, or has a compression ratio of less than 10%, you will not get compression marks.**
- To ensure efficient evaluation of assignments, your programs will get 1 hour to run (both compression and decompression should be done within this window).

3 Marking scheme

Total marks: 100

The marking will be competitive. It will be based on two factors: (a) the compression ratio (70%), and (b) the efficiency (time of running) of the algorithm (30%). Note that to be considered for (b) your algorithm should decompress the dataset loss-lessly.

Penalty for error

$$\text{Error} = \frac{|D \cup D'| - |D \cap D'|}{|D|}$$

where D is the original dataset seen as a set of transactions, and D' is the re-constructed dataset, also seen as a set of transactions. Note, transactions will also be seen as sets of items, and equality of transactions will be defined based on set equality when performing $D \cup D'$ and $D \cap D'$. More precisely $T1 = \{1,2\}$ is the same as $T1' = \{1,2\}$, but different from $T2' = \{2,3\}$, $T3' = \{1\}$, and $T4' = \{1,2,3\}$.

$$\text{Penalty} = \begin{cases} 0, & \text{Error} = 0 \\ 1.25^{100*\text{Error}}\%, & \text{Error} \leq 0.2 \\ 100\%, & \text{otherwise} \end{cases}$$

[1]https://drive.google.com/file/d/1Aij48f2k-gtJIrqQnFUpbXsOwrnXo-8F/view?usp=sharing

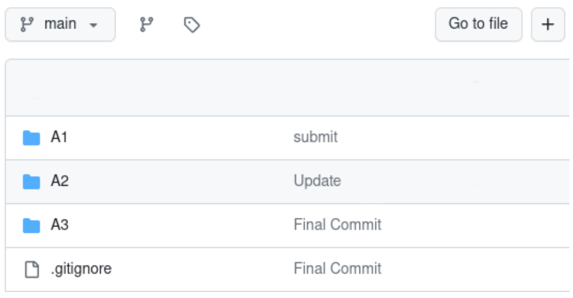
So, finally the marks will be, **marks for (a) + marks for (b)** if your code performs loss-less compression and **marks for (a) – Penalty** if it performs lossy compression.

4 Submission format

These instructions will be constant across assignments, so please read them carefully. The correct evaluation of your assignment depends on the accuracy of you following these instructions since your assignment will be automatically graded which means the automatic grading script will expect certain path structures to be followed, so please read them carefully.

If student S makes a correct submission, then:

1. S and her team-mates have uploaded her code to the GitHub repository she mentioned as part of A0 (or HW0).
2. Within the repository, S has put the code inside a folder called "A1". All the code that's inside this folder and nothing else will be used to evaluate this assignment.



A1	submit
A2	Update
A3	Final Commit
.gitignore	Final Commit

3. If S's code requires compilation (C, C++, etc.), then she has placed the compilation command in a script named "compile.sh" inside the folder "A1".
4. "compile.sh" provided by S loads required modules to add their language's compiler to the path. She found them using the command `module avail 2>&1 | grep <compiler name>` on HPC. She also purged all existing modules before loading her own modules by placing `module purge` as the first line of "compile.sh".
5. For S's code to be checked by automatic compilation, she has placed a script named "interface.sh" that runs the code. "interface.sh" takes two command line parameters, (1) absolute or relative path to the input dataset, and (2) absolute or relative path to the output dataset. "interface.sh" also loads the required HPC modules after purging everything that was previously loaded.
6. On moodle, S (or exactly one of her team-mates, no repetitions) has submitted a bash file named "clone.sh" that just contains the single command to clone her repository, `git clone https://github.com/Srepo`. This is the http clone URL, do not place the SSH URL provided by GitHub. Also, do not provide your PATs or credentials to clone the URL, we'll take care of it.
7. S did not assume any absolute paths to refer to other parts of her code or dataset, S's code works for relative paths.
8. S's code does not modify the original input dataset.

5 A few useful hpc modules

- compiler/gcc/9.1.0
- compiler/python/3.9.13/ucs4/gnu/447
- compiler/java/JDK/13.0.2

You may use `module show <module-name>` to get more details about the module, about the paths it adds to `$PATH`, `$LD_LIBRARY_PATH`, etc. Note, the name of the module is what appears to be like the full path above, so to get the details of gcc-9.1.0, you'll use `module show compiler/gcc/9.1.0`.

6 Checklist

- ☐ Did not cheat.
- ☐ Code uploaded to GitHub repo in "A1" directory.
- ☐ "clone.sh" uploaded to moodle. Did not put SSH URL or PATs in the git clone command by exactly one person in the team.
- ☐ "compile.sh" put inside "A1", if needed.
- ☐ "interface.sh" put inside "A1". It accepts CLI params as specified and produces output file.
- ☐ Correct modules loaded in "clone.sh" and "interface.sh". Pre-existing modules purged prior to that.
- ☐ Code tested on HPC.
- ☐ Code results in $\geq 10\%$ compression.