COL761 A1 - Transactional Data Compression

Instructions

- Submission deadline is **20/08/2023 23:59 IST.**
- You need to do the homework in your already formed teams. You will upload the code to the github repository mentioned in HW0.
- You will submit a script to **moodle** which will clone the repository.
- Your code must compile and execute on HPC.
- Do not copy the code from your friends or from the internet. **Plagiarism will result in an -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++, Java, and Python.

Submission Instructions

- Add **col761-23** as a collaborator for your private repository.
- As already mentioned, you will be uploading your code to github. The organisation of the repository for multiple assignments is up to you. You may have a different branch for your assignments, or just the main branch having all the code.
- You will be submitting a script A1_<RollNo>.sh (e.g., A1_MCS162913.sh) to moodle which will clone your github repository and leave just a folder containing all your A1 code. Running the bash script from a folder should result in the clone of your github repository in the same directory from where the code was run. Inside your github repository, you must have a folder named “A1” which will contain all the source code of your assignment.
- **The evaluation will be automatic, using a simple scripting language. So please follow the naming conventions exactly. Inducing manual work will result in a penalty. In case you need clarity, confirm on piazza, well before submission.**
- We want equal contribution from all the team-members. Therefore, in a text file, you should mention the percentage contribution of each of your team-members. The marks will be in proportion to contribution.
- Testing will be done on a different dataset. Do not submit the dataset.
- Submission time will be `max(submission time on moodle, last commit date and time)`.
- For automatic evaluation you will be required to create an automatic bash script interface to run the assignment (`interface.sh`). This should be inside the folder A1.
- You will also need to provide a compilation bash script (`compile.sh`). This should also be inside the folder A1. This script is optional, since python doesn’t require compilation.
- You will submit a PDF writeup that will document your algorithm in detail. This should also be inside the A1 folder. It should be named `writeup.pdf`.

Problem Description

Dataset compression plays a crucial role in modern data management and analytics. As the volume of data 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 essential to ensure that the compression is lossless i.e. the original dataset can be retrieved efficiently from 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 order in which the items occur within a transaction is not important. The task is to compress this dataset `D`, by reducing repetition. A natural candidate for this dataset is to use frequent itemset mining.

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 – In this case, 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 (size of original dataset) to 17 (size of mapping table + size of compressed dataset). 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. The order in which the items occur within a transaction does not matter.

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.

There are various design decisions you can think about that will affect the compression ratio. In the above sample dataset, instead of grouping `(A, B, C, D)` as X, what will happen if `(A, B, C, D, E)` is set as X. Which one will result in higher compression and why? When should you choose to add a mapping entry for an itemset? Secondly, if you have constructed your mapping as `{W: (A, B), X: (C, D, E), Y: (A, B, C), Z: (D, E)}`, will you decompose the transaction `T = A, B, C, D, E,` as `W, X` or as `Y, Z`?

Datasets

Your code will be tested on a different dataset than provided. Please find some datasets to verify your code here.

Evaluation

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

Penalty for Error

Let us say the original dataset is `D = {T1, T2 ...}` and your decoded dataset is `D' = {T1', T2', ...}`

```
Error = (|D ∪ D'| - |D ∩ D'|) / |D|
```

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

Automatic Evaluation

Your assignment will be automatically evaluated. The following steps will be run.

1. `$ bash A1_<RollNo>.sh` will be run. This should clone your repository in the same folder from where the script is run.
2. The only other directory present after this step should be your repository. It will automatically be detected and will be `cd`-ed inside. There should be a directory named A1 in there, everything else will be removed.
3. `$ cd A1` will be run.
4. If the folder contains a compilation script named “compile.sh” then it will be run as: `$ bash compile.sh` (with no arguments).
5. `$ bash interface.sh C <path/to/dataset.dat> <path/to/output.dat>` will be run. Note that capital C will be provided as the first command-line argument to the script. This should output a compressed dataset saved to disk in a human readable format. We will count the number of space separated item-ids (they will be integers) to compute compression ratio.
6. `$ bash interface.sh D <path/to/compressed_dataset.dat> <path/to/reconstructed.dat>` will be run. The first command-line argument is a capital D. This will decompress the output from the previous step to produce the re-constructed dataset. This will be used to compute the loss penalty, if any. To ensure fairness, during this step, all the access to the original dataset will be removed (basically, the dataset itself will be deleted from your folder, and any other output generated by your code will also be removed).
7. Finally, a score will be automatically assigned to the assignment. The distribution of marks according to team members' contribution will be done manually afterwards.
8. If automatic evaluation fails at any step due to not conforming to the naming convention or not accepting arguments in the right format, the evaluation will be penalised 10 marks per hour of manual work.