Randomized Algorithms (RA-MIRI): Assignment #3

1 Statement

In this programming assignment, you will have to study experimentally the performance of two different cardinality estimation algorithms, namely, HyperLogLog [1] and Recordinality [2].

You will need to write programs for HyperLogLog (HLL) and for Recordinality (REC), but you shall also write routines to generate synthetic data streams $Z = z_1, z_2, \ldots, z_N$, for instance, following a Zipfian law of parameter $\alpha \geq 0$ for n distinct elements $\{x_1, \ldots, x_n\}$ in which

$$\mathbb{P}\{z_j = x_i\} = \frac{c_n}{i^\alpha}, \qquad 1 \leq j \leq N, \quad 1 \leq i \leq n,$$

and

$$c_n = \frac{1}{\sum_{1 \leq i \leq n} i^{-\alpha}}.$$

You will try your programs with some real datasets (`https://mydisk.cs.upc.edu/s/8KswTJasg5q4xx7`), as well as with synthetic data. The datasets (`.txt` files) come from publicly available novels downloaded from the Project Gutenberg Free eBooks page (`https://www.gutenberg.org/`); the original files have been processed to remove punctuation, words of less than 3 letters, every uppercase letter has been converted to its corresponding lowercase, etc.

For each dataset, prepare a table that compares HLL, REC and the true cardinalities. The lists of unique words and their frequencies of each dataset can be found in the corresponding .dat files. These files have been produced using the Unix command

```
sort file.txt | uniq -c | awk '{ print $2 " " $1 }' > file.dat
```

Remember that these cardinality estimation algorithms are randomized, hence you should run each several times to get estimates of their expected performance. Both algorithms rely upon using good hash functions[1]. This implementation of a family of random hash functions by J. Lumbroso can be of help for your work (`https://github.com/jlumbroso/python-random-hash`);

[1]Recordinality can actually work without using hash functions and directly use the items in the data stream. You might want to carry out some experiments to check this.

it's in Python (also in Java), and you can easily adapt to C++ if you prefer. Other sites worth looking at might be https://xxhash.com/ or https://github.com/lemire/StronglyUniversalStringHashing/.

You should also study how the amount of memory (number of counters m in HLL, number of elements k kept in REC) impacts the quality of the estimations; do this for dataset `dracula.txt` at least, maybe for others. Check that the standard error behaves as predicted by the theory.

The same kind of experiments can be conducted using synthetic data streams. They allow you to do the same type of experiments, but now with total control on your hands of the length N and number of distinct elements n. You will notice that the value of α makes little difference—but you can conduct some small experiment just to confirm this hypothesis!

Do not forget to include in your report some details about your choice of hash functions and all references that you have used for this assignment. Make sure to include proper citation to the references in the main body of your report: the bibliography section is not just an isolated collection of references that appear at the end of your report.

Bonus: Add more cardinality estimation algorithms (e.g., Probabilistic Counting (PCSA), KMV (K Minimum Values), MinCount, Adaptive Sampling, variants of HLL, . . .) to the comparative study.

2 Instructions to deliver your work

Submit your report in PDF using the FIB-Racó. The deadline for submission is January 7th, 2026 (8:00). The submitted file should be called

`*username(s)*-cardest.pdf`

Your username(s) are the first part of your institutional email address. **Do not submit anything else but the PDF file with the name as above**. Deviation from these guidelines will be penalized.

Your PDF must also include a link to a repository (in Github or GitLab, for example) or shared folder which contains all the source files of your program(s) and a `README` file with instructions to compile and execute the program(s) to reproduce the experiments. Make sure that the link works. All these source files should be easily downloaded as a `.zip` or `.tar` file (this is the case, for instance, with Google Drive folder), otherwise create such a file inside the repo or folder, ready for downloading. Do not submit the compressed file to *Racó*.

You can choose to work in the assignment in teams of two students, or individually. I encourage you to do it in team with a classmate. Let me know if you are interested in finding a classmate to do the assignment but you do not know who could be in a similar situation, I will try to help.

I also encourage you to use LaTeX to prepare your report. For the plots you can use any of the multiple packages that LaTeX has (in particular, the bundle TikZ+PGF) or use independent software such as matplotlib and then include the images/PDF plots thus generated into your document.

References

[1] Ph. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In Ph. Jacquet, editor, *Proc. 2007 Conference on Analysis of Algorithms (AofA)*, volume AH of *Discrete Mathematics & Theoretical Computer Science (Proceedings)*, pages 127–146, 2007.

[2] A. Helmi, J. Lumbroso, C. Martínez, and A. Viola. Data streams as random permutations: the distinct element problem. In N. Broutin and L. Devroye, editors, *Proc. 23rd Int. Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA)*, volume AQ of *Discrete Mathematics & Theoretical Computer Science (Proceedings)*, pages 323–338, 2012.