COMP0005 Algorithms
Academic Year 2022-23

Group Coursework: Set Membership

Released: February 20th, 2023
Submission deadline: March 20th, 2023 @ 4pm GMT

The problem.

Consider the set membership problem: "Given a set of elements *A*, determine whether element *a* belongs to *A*". Many real-life problems can be abstracted as the set membership problem. For example, to register a new username on a website, one needs to determine whether such username is unique or whether it is already in use; to determine whether to block a certain URL, web browsers need to determine whether such URL belongs to the set of URLs deemed potentially malicious; to prevent choosing a weak password, one needs to check whether it belongs to a dictionary; and many more. This conceptually simple problem can be difficult to solve computationally, if the set *A* is very large, the amount of memory storage very limited, and the need to answer the query "is *a* in *A*?" very fast.

The task.

Your task is to design, implement, and *thoroughly evaluate* a variety of algorithms and associated data structures to solve the set membership problem (that is, to find an element in a set) under the conditions set above (i.e., limited storage, large *A*, need to answer queries in real-time). In particular, you should consider: (a) sequential search, (b) binary search tree, (c) balanced search tree, and (d) bloom filter [1]. The first three have been covered in lectures; you are expected to study the last one on your own (Wikipedia link provided above, as well as reference to the original paper). Assume the elements in your set are strings, and that your set grows over time (you must be able to both *search* and *insert* elements in a set) but do not shrink (there is no need to support a delete operation).

Evaluation should be conducted both *theoretically* (in terms of space and time complexity), and *experimentally*. For experimentation, use both real and synthetic data. For real data, use the test files provided as follows: read file `test1-mobydick.txt`; insert one by one and in the given order each of its space-separated words (i.e., strings); compute and report the total amount of time it takes to perform all these insert operations, for each of the four data structures under consideration. Subsequently read file `test-search.txt` and search one by one and in the given order each of the words (i.e., strings) it contains within the sets previously constructed; compute and report the total amount of time it takes to perform all these search operations for each of the four data structures under consideration. Repeat using files `test2-warpeace.txt` and `test3-dickens.txt` to create sets (insert operations); always use file `test-search.txt` to search strings in these sets. For synthetic data, you are expected to generate data that will enable you to stress-test and compare the performance of the various algorithms and data structures under different conditions (for you to determine).

Constraints. You are expected to implement your own algorithms and data structures. You are NOT allowed to use (import) Python libraries, except for `bitarray` which you may use to implement bloom filters. You may further use `timeit`, `random` and `string` to realise your experimental framework (as explained below). If in doubt about what you can and cannot use, ask in Moodle "Ask a question" forum. **Your code must successfully run using Jupyter Notebook with Python 3.8.**

Submission instructions.

The coursework has both a group submission and an individual submission.

For the GROUP submission. Using the Moodle course website, submit two files:

- A Jupyter notebook `SetMembership.ipynb` containing your implementation of all data structures and algorithms required for this coursework, including code to generate synthetic data and to conduct experimentation with it. This notebook MUST follow the structure of the skeleton code provided and MUST successfully run using Python 3.8. Do not submit the synthetic data you created; rather, your notebook should contain code to generate it; you may use the Python `random` and `string` packages for this purpose. Use the Python `timeit` library to measure the execution time of your implementation (do not include the time to generate test data in your measurements).

- A PDF document `report.pdf` of maximum 4 pages (font style: Arial or Times New Roman, font size: 12pt), where you present, for each of the four algorithms, their theoretical and experimental analysis. For theoretical analysis, cover both space and time complexity, both in the average and worst-case scenarios. For experimental analysis, briefly describe your experimentation framework (e.g., what synthetic data you have generated and why, what you have measured), then critically and thoroughly discuss your results, both when using the real data provided, and when using the synthetic data you generated. Include graphs to illustrate your results. Conclude the report with a brief summary and comparison of main pros and cons of the four algorithms; highlight what scenario(s) they are each best suited for.

Only one member of the group is required to submit the above two files.

For the INVIDUAL submission. Using the Moodle course website, each member of the group is required to submit a one-page PDF document, containing a concise assessment of the contributions made to the coursework by each member of the group, including yourself. The document should strictly follow the template provided `sample-groupeval-0.docx`. When submitting the file, please use the following naming convention: [surname]`-groupeval-`[number]`.pdf` where [surname] is your surname and [number] is your group number.

Submission deadline: Wednesday, March 20th 2023, 4pm GMT.

Assessment.

This coursework represents 40% of the final mark for COMP0005 Algorithms. Submissions will be evaluated in terms of the following marking criteria:

- Quality of your code (i.e., is your implementation *correct*? Is your code *readable* and *well documented*? Is it time and space *efficient*?) [20 points]
- Theoretical analysis (e.g., is your theoretical analysis *correct*? Is it *complete*?) [20 points]
- Experimental analysis (e.g., is your experimental framework comprehensive? have edge-cases been analysed? Have suitable stress-test conditions been experimented with? Are the results of your experimentation correct? Are they thoroughly discussed?) [60 points]

Marks for each criterion will be assigned following the UCL CS Grade Descriptor (criteria 4, 5 and 6).

Note: students within the same group should not expect to receive the same mark. Marks will depend on each student's individual contribution to the work of the group.

Academic Integrity.

UCL has a very clear position about academic integrity – what it is, why it is important, and what happens if you breach it. Make sure you have read UCL position on this matter before attempting this coursework.

References.

[1] Bloom, Burton H. (1970), "Space/Time Trade-offs in Hash Coding with Allowable Errors", Communications of the ACM, 13 (7): 422–426. [PDF][Wikipedia]