Analysis of Algorithms (CSCI 323 / 700)
Fall 2019

Final Project
Due Date: December 19, 2019

Introduction:

As mentioned in the syllabus and in the opening lecture, there are various projects worth a total of 30% of the course grade. The purpose of these projects is to give you practical experience with data structures, algorithms, and design techniques, and in the comparison of their empirical performance with the ones predicted by the analysis presented in class and found in the technical literature.

Overview:

This project should focus on 3-5 algorithms or alternatively, different implementations of the same algorithm. Generally, the algorithms should be at least as complex as those first studied in our course (as opposed to earlier courses like Data Structures.) Generally, sorting and order-statistics should be avoided as those topics were sufficiently covered in earlier projects. *The choice of algorithms must be approved by the instructor.*

Groups:

You may work individually or in groups of 2-3. The names of the student(s) must be typed into the top of the source code and in your accompanying article..

Stages:

The following stages are recommended:

1. Choose a group and a topic that your will work on together.
2. Abstract: write a one-paragraph description of your project, the algorithms and/or implementations you plan to compare. At this point, you should generally also be aware of the "predicted" time complexity of each algorithm or implementation. While there is no firm deadline for the Abstract, the sooner the better, since student projects should be distinct and first-come, first-serve. Approval of the project is required at this stage, and I will post the topics and the student(s) doing them as they have been approved.
3. Implementation: getting the algorithms, decided on in the Abstract stage, to work. The programming language is your choice. It is okay to get code from a textbook or the Internet, but you should cite from where you took the code. (see below “use of public code”).
4. Execution: your program should create random input of various sizes, appropriate for the nature of the algorithm. You should also be able to run the algorithm many times in succession, each time with random data, to get a sense of average case performance.
5. Tracking: - the algorithms should be modified to (a) keep track of actual (clock) time for a given run, as well as (b) counting the number of operations required. Deciding which operation(s) are track-worthy depends on the problem being solved, and should already be identified in the Abstract based on what the theoretical time complexity measures. Then track in a chart the number of operations and actual clock time.
6. Modification: are there any suggestions how to improve the algorithms, perhaps by creating a hybrid of various algorithms, integration of other design techniques, etc.
7. Analysis: you should discuss to what extent your empirical run time seems to agree with the theoretical runtime. If they differ, how do you account for that?

Use of public code:

Permission to use publicly available code for this assignment is with the following understanding:

- The code must be in the public domain (i.e. not someone's personal repository).
- You must comply with all copyright regulations.
- You should have a comment before each such algorithm acknowledging the URL and author (if known).
- This does not translate into the permitted use of public code for future assignments in this or other courses.
- You are responsible that the code works.
- You need to be familiar with code for the purpose of examinations.

Submission

You should submit your project by e-mail to Lawrence.Teitelman@qc.cuny.edu, with a single attachment - a zip-file which contains an "article" in PDF format, as well as your source code, raw output from your program, and any other relevant supporting material.

The format for your article should be as follows:

Header

Name for paper, e.g. "An Empirical Study of Three Minimum Spanning Tree Algorithms"
Name of student author(s)
Course Number and Course Name (CSCI 323 - Analysis of Algorithms)
Semester: Fall 2019

Introduction
A short paragraph describing what you sought to do in this project - list the three (or however many) algorithms you were comparing, what you compared them for, etc.

Algorithm Overview

For each algorithm/implementation mentioned in Introduction, give a paragraph about it. (This can generally be found on-line in Wikipedia and elsewhere). The name of the algorithm, high-level pseudocode, who invented the algorithm, and the theoretical time and space complexity.

Implementation Considerations

Any technical information about the implementation recursive vs. non-recursive, choices for data structures, choice of programming language, hardware / operating system used, how you keep track of operations, etc.

Data Considerations

As previously mentioned, to get a meaningful sense of actual performance, you will need to use random data, for a variety of input sizes (what we usually call n), for several tries (recommend 10) for each value of n, and then take the average number of operations for each value of n. Here you should describe how you generated your random data (it will be different for sorting than say for a graph problem like SSSP or MST), what n's you used, and how many tries in fact you used for each n.

Tracking of Operations

For each kind of operation tracked and for each algorithms, you would have a chart that looks something like this:

n	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	Iter 6	Iter 7	Iter 8	Iter 9	Iter 10	Avg.
10											
100											
1000											
10000											
100000											

Assessment

Assess your algorithms. Considerations might include

- performance (time / number of operations)
- accuracy - for approximation / randomized algorithms, to what extent did it find the optimal answer
- where applicable, there may be other measure of success. For example, if you are doing data compression, you might compare the compression ratios of the different algorithms and for various kinds of data

Conclusion

Some discussion as to what extent the predicted time complexities matched the actual (empirical) ones? If they didn't can you offer some insight as to why they are different?

Future Study

If you had more time, what further questions would you pursue on this topic? How would you extend the research?

Sources

Acknowledge sources (books, websites) you used for information, code, etc.

Exhibits (separate from PDF)

Source code

Raw output