Lab 2 (Analysis of Algorithms)

CSC 172 (Data Structures and Algorithms)
Fall 2024
University of Rochester
Due Date: Sunday, September 15 @ End of Day

Introduction

The labs in CSC172 will follow a pair programming paradigm. Every student is encouraged (but not strictly required) to have a lab partner. Every student must hand in their own work but also list the name of their lab partner if any on all labs.

In this lab, we will work on our understanding of asymptotic analysis of algorithms. We will run implementation of four algorithms to determine their asymptotic runtimes and how close the findings are to the actual runtime. The main components of the lab are taken from `http://algs4.cs.princeton.edu/14analysis/`. For your convenience, I have uploaded all the necessary files from the authors' website. You can use the `Lab02.zip` file attached to the assignment on Blackboard or download the old zip files from `http://www.cs.rochester.edu/courses/172/spring2018/labs/Lab4.zip`. They contain the same files.

The zip file contains 8 java files. Four of these files are the focus of this lab.

- `TwoSum.java`: Read in n integers and counts the number of pairs that sum to exactly 0.
- `ThreeSum.java`: Read in n integers and counts the number of triples that sum to exactly 0.
- `TwoSumFast.java`: Functionality same as `TwoSum.java`. Applies a better algorithm
- `ThreeSumFast.java`: Functionality same as `ThreeSum.java`. Applies a better algorithm

The zip file also includes 7 data files:

- 1Kints.txt: contains 1K integers
- 2Kints.txt: contains 2K integers
- 4Kints.txt: contains 4K integers
- 8Kints.txt: contains 8K integers
- 16Kints.txt: contains 16K integers
- 32Kints.txt: contains 32K integers
- 1Mints.txt: contains 1M integers (Optional)

> Note: Using 1Mints.txt file is optional. For most of the problems, running your program on this file will take excessive time. You are welcome to try but it's not required.

Task 1 - mostly done in the workshop sessions

Answer the following questions:

1. What's the order of growth of the running time of `count` function in `TwoSum.java`? Provide an annotated code snippet stating asymptotic runtime for various blocks (in Big-Oh notation).
2. What's the order of growth of the running time of `count` function in `TwoSumFast.java`? Provide an annotated code snippet stating asymptotic runtime for various blocks (in Big-Oh notation).

3. What's the order of growth of the running time of `count` function in `ThreeSum.java`? Provide an annotated code snippet stating asymptotic runtime for various blocks (in Big-Oh notation).

4. What's the order of growth of the running time of `count` function in `ThreeSumFast.java`? Provide an annotated code snippet stating asymptotic runtime for various blocks (in Big-Oh notation).

Task 2 - mostly done in the lab sessions

1. Run `TwoSum.java` and provide screenshots of the output for all 6 (or 7 if you are running your program on 1Mints.txt) data files. Plot n vs $T(n)$. Use Logarithmic scale for x-Axis

2. Run `Three.java` and provide screenshots of the output for all 6 data files. Plot n vs $T(n)$. Use logarithmic scale for x-Axis

3. Run `TwoSumFast.java` and provide screenshots of the output for all 6 data files. Plot n vs T(n). Use logarithmic scale for x-Axis

4. Run `ThreeSumFast.java` and provide screenshots of the output for all 6 data files. Plot n vs T(n). Use logarithmic scale for x-Axis

5. Plot n vs $T(n)$ for `TwoSum.java` and `TwoSumFast.java` on the same figure. Use logarithmic scale for x-Axis. Describe your finding.

6. Plot n vs $T(n)$ for `ThreeSum.java` and `ThreeSumFast.java` on the same figure. Use logarithmic scale for x-Axis. Describe your finding.

Task 3 - mostly done on your own

1. Compare runtimes for each pair of the consecutive run for `TwoSum.java`. Estimate runtimes for 32K and 1M integers from the analysis. How close are you to the actual runtime?

2. Compare runtimes for each pair of consecutive runs for `TwoSumFast.java`. Estimate runtimes for 32K and 1M integers from the analysis. How close are you to the actual runtime?

3. Compare runtimes for each pair of consecutive runs for `ThreeSum.java`. Estimate runtimes for 32K and 1M integers from the analysis. How close are you to the actual runtime?

4. Compare runtimes for each pair of consecutive runs for `ThreeSumFast.java`. Estimate runtimes for 32K and 1M integers from the analysis. How close are you to the actual runtime?

You must estimate runtimes for both 32K and 1M integers irrespective of running your program for 1Mints.txt file. If any program takes excessively long time to run, you can provide the estimated runtime and explain why you did not complete running the program for any particular text file. We strongly encourage you to start early as running the programs may take longer than your estimation. Also, we recommend running the programs on the terminal (without using any IDE).

Grading (100 pts)

Your lab will be graded solely on the pdf file you have submitted. Though, you still need to submit the modified source code files.
Task 1: 20pts
Task 2: 60 pts
Task 3: 20 pts

Submission

Hand in the source code (with modified Netid) and the lab report from this lab at the appropriate location on the Blackboard system at `learn.rochester.edu`. You should hand in a single zip (compressed archive) `Lab2_TFP.zip` - with your personal initials instead of "TFP" - containing all your source files and Lab Report in PDF with detailed description for each task. Save the lab report as `Lab2Report_TFP.pdf` - with your personal initials instead of "TFP" . It should also contain information of both the team members if applicable.