Final Project

RELEASE DATE: 05/01/2014

DUE DATE: 06/25/2014 (**Wednesday!!**), **noon** ON GitHub

Unless granted by the instructor in advance, no late submissions will be allowed. Also, the gold medals cannot be used on the final project.

Introduction

The main theme of the final project is a email searcher. The email searcher helps retrieving precious emails from the mail pool, which contains mails that came in a streaming manner. Of course, we expect the searcher to effectively use the computational and storage resources of your computer. So the data structure (and the associated algorithm) used for representing the mail pool can be crucial.

Problem Description

The email searcher needs the following basic functionality. You can use your own function prototype for implementing the functionality.

- `add`: adds a mail into database.
- `remove`: removes a mail from database.
- `query`: return mails that fits the search query.

Survey Report

You are asked to study at least **THREE** data structures for dealing with the email searcher. Then, you should make a comparison of those data structures according to some different perspectives, such as average speed, worst speed, space, implementation, popularity, etc.. Based on the results of your comparison, you are asked to **recommend** the best one for the email searcher, and provide the "cons and pros" of the choice.

The survey report should be less than or equal to **eight** A4-pages with readable font sizes and formats. Criteria for evaluating your survey report would include, but are not limited to, clarity, strength of your reasoning, "correctness" in using the data structures, and the work loads of team members.

Competition

We will hold a mini-competition for the project. Each team is asked to submit the source code of your email searcher to be automatically compiled on the CSIE Linux machines for the mini-competition. The details will be announced soon.
Two things will be tested in the competition:

- the accuracy of the searcher
- if the accuracy is 100%, the speed of your email searcher

The fast track of the competition ends at 6/15 noon; the slow track ends at 6/25 noon.
Every team is asked to submit to the mini-competition at least three times (with the three data structures used) and list the results in the survey report. Of course, more submissions are encouraged and welcomed.

Submission File

Please push your program to your team's repository on GitHub (details to be announced later) before the deadline. We will take a snapshot at the deadline as yous submission. For your program, please **DO NOT PUT BINARY FILES** in your repository.

Your repository should contain the following items:

- the source files of your final email searcher, including any package you use (see below); those source files can be different from what you submitted to the mini-competition
- the report (`report.pdf`) with at most eight A4 pages in PDF format. The report should contain the following items:

  (1) the team members' names and school IDs

  (2) how you divide the responsibilities of the team members

  (3) the data structures you compared, including the results submitted to the mini-competition site

  (4) the data structure you recommend

  (5) the advantages of the recommendation

  (6) the disadvantages of the recommendation

  (7) how to compile your code and use the email searcher

  (8) the bonus features you implement and why you think they deserve the bonus

  You do not need to submit a printed version of your report.

Misc Rules

Report: No, you do not need to submit a hard-copy.

Teams: By default, you are asked to work as a team of size three. A one-person team is allowed only if you are willing to be as good as a three-people team. It is expected that all team members share balanced work loads. Any form of unfairness in a three-people team, such as the intention to cover the other member's work, is considered a violation of the honesty policy and will cause both members to receive zero or negative score.

Data Structures and Algorithms: You can use any data structures and algorithms, regardless of whether they were taught in class.

Packages: You can couple your email searcher with any software package (as long as you are not violating any copyright) but you need to *clearly cite where you get the code* and *clearly describing what the source code does* in your report.

Platform and Language: As usual, you can only use C/C++ to design your main program. If you use packages from other languages, you still need to call them from C/C++. You can either use Linux or Windows as the running platform of your email searcher. But the submission to the mini-competition needs to be Linux-compatible with a Makefile (details to be announced).

Grade: The grading TAs would grade qualitatively with letters: A++[210], A+[196], A[186], B+[176], B[166], C+[156], C[146], D+[136], D[126], F+[116], F[76], F-[36], Z[0]. The score of the team would be the average of all the grading TAs. We reserve the possibility to adjust individual scores in the team based on performance/workload if necessary. The final project is equivalent to four usual homework sets; the midterm exam is equivalent to three. Your raw score in the class would be calculated by

$$\frac{\text{best homework} * 1.5 + \text{worst homework} * 0.5 + \text{other homework} + \text{midterm} * 3 + \text{final} * 4}{13}.$$

If your email searcher meets the basic functionality and you write down every item reasonably in the survey report, you will get at least B.

Bonus: We encourage everyone to think about making your email searcher better. The room between B[162] and A++[210] is basically left for bonus. To get bonus points, you need to justify that the additional features/functionality of the email searcher is worth being the bonus in your report. For instance, you can try to compare more data structures/algorithms or add your creativity in designing some good data structures/algorithms for the email searcher. Some more functionality (such as being Gmail-search compatible?) is also worth thinking. A fancy GUI may be another possibility, but not the only way and

very likely not an important way. After all, *we are seeking for better data structures and algorithms in this class, not just better GUI.*

Collaboration: The general collaboration policy applies. In addition to the competitions, we still encourage collaborations and discussions between different teams.