

Data Structure and Algorithm, Spring 2021

Final Project - Email Searcher 2.0

Competition & Report Due: 23:59:59, Tuesday, June 22, 2021

TA E-mail: dsa_ta@csie.ntu.edu.tw

Rules and Instructions

- This document describes the theme of final project. A more detailed spec will be released later.
- Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.
- In this Final Project, you are required to team up and participate a programming competition on *DSA Judge* (<https://dsa-2021.csie.org>). You are also required to upload a report about your implementation and analysis to *Gradescope* (<https://www.gradescope.com>).
- By default, you are asked to work as a team of size three. A one-person or two-people 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.
- In the programming competition, each team will have 5 quotas per day to submit your programs in the C language to the judge system. The score of your submission will be evaluated and shown right after your submission is judged. We will score your programming part depending on your rankings on the scoreboard (see Grading section below).
- The PDF file for the report part should be submitted to Gradescope before the deadline. Unless granted by the instructor in advance, no late submissions will be allowed.
- If you have questions about the Final Project, please go to the course forum and discuss (**strongly preferred**, which will provide everyone a better learning experience). If you really need an email answer, please follow the rules outlined below to get a fast response:
 - The subject should contain "[Final Project]". For example, "[Final Project] Failed to compile the test environment in my environment". Adding the

tag allows the TAs to track the status of each email and to provide faster responses to you.

- If you want to provide your code segments to us as part of your question, please upload it to [Gist](#) or similar platforms and provide the link. Screenshots or code segments directly included in the email are discouraged and may not be reviewed.
- You are allowed to use the functionalities provided by the C Standard Libraries. But we ban several system calls that can cause unfairness, such as `fork`. Those calls will result in **Security Error**.
- Discussions are encouraged, but you should write the final solutions alone and understand them fully. In order to maximize the level of fairness in this class, lending and borrowing solutions to other team are both regarded as dishonest behaviors and will be punished according to the honesty policy. Books, notes, and Internet resources can be consulted, but not copied from. That is, **you cannot copy the code that you find online**. In order to maximize the level of fairness in this class, lending and borrowing solutions to other team are both regarded as dishonest behaviors and will be punished according to the honesty policy.
- If you found some security vulnerabilities (e.g. ways to bypass the timing mechanism) in the final project. Please report the vulnerabilities to the TAs (and only to the TAs) by email immediately. This helps us fix the issue as soon as possible. We surely will not punish you and will reward your honesty in some way.

Introduction

The main theme of this project is an upgraded version of [email searcher](#) (the theme of last year). The upgraded email searcher helps retrieve precious mails and perform analysis on mails in a distributed way, which is done with the help of many workers. These workers fetches mails and queries from the server, and reply the answers to get rewards within a limited time. In this final project, you need to develop a efficient worker to earn the points as much as you can.

Problem Description

In this project, you will be given a set of emails (will be made public) and many queries regarding to the contents of the mails, each has its own reward. Your job is to develop a strategy which helps you earn the most points, either by solving as many queries as possible, or by cleverly choosing the queries to solve. You may answer queries in any order you like. The three types of queries you may encounter are:

- `EXPRESSION-MATCH(EXPRESSION, CONSTRAINTS)`

In this type of query, you will be given an expression and other optional constraints. You need to answer all the mail IDs that match the expression and other constraints, sorted in increasing order.

- `FIND-SIMILAR(MESSAGE-ID, THRESHOLD)`

In this type of query, you will be given an message ID and a threshold. You need to answer all the other mail IDs that shares similarity greater than the threshold with such mail and sorted in increasing order of mail IDs.

- `GROUP-ANALYSE(MESSAGE-IDS)`

In this type of query, you will be given a set of mail IDs. Assume a mail represents a transitive relationship between the sender and recipient, and a group is defined by users that can directly or indirectly have relationship with each other. You need to answer the number of groups and the size of the largest group among the mails' senders and recipients.

Each query will have its own reward R and penalty P . If you answer a query with the exact right answers, you earn the reward R . However, if your answers are wrong, then you will be charged with a penalty of P . The total points of your submission will be the sum of all the rewards and penalties.

Competition

A competition will be hold on the good old DSA Judge. Your team will have 5 quotas per day to submit programs for the above problem. In each submission, your program will be granted 5 seconds to process and answer queries right after getting all the mails and queries by calling the initialization function in header file. The score of your submission will be the sum of all the rewards and penalties (yes, your score can be negative for low accuracy) your program accumulated within the time limit. A rank list with all team's score will be made public.

The performance of your program is measured only by accuracy (whether your program returns the right answers) and efficiency (the number of queries the program can answer within the time limit). We somehow need to put a limit on memory usage for security and fairness. Each submission will be allow to use at most 10 gigabytes, and exceeding the limit will result in **Memory Limit Error** or **Runtime Error** on the judge system.

After the competition begins, you may see that the time limit on judge system is little longer than the 5 seconds above. This is because we want to ensure your program has the entire time limit duration to execute, excluding the time spent on data reading and writing, which is provided and unified by the functions in the header file. Each calls to answering function will trigger a time limit check, if your program has run more than 5 seconds after the initialization function returns, the answering function will terminate the execution of your function by invoking the `exit` system call. Noted that calling the initialization function after the first 15 ms will result in **Security Error**, in order to prohibit you from stealing the time before the timer is actually initialized.

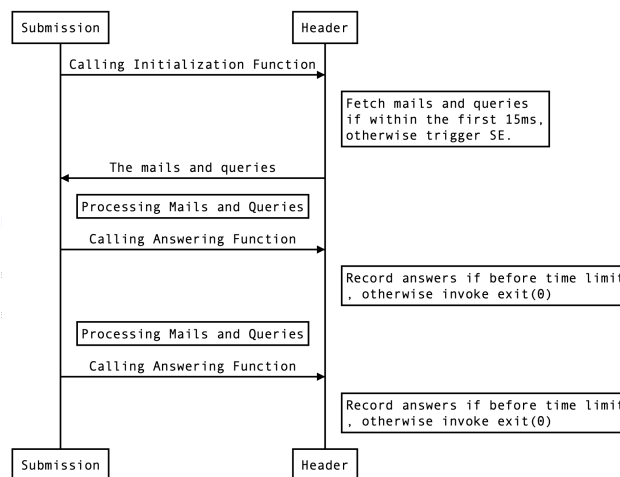


Figure 1: Example Execution Flow

Report

In addition to participating the programming competition, we also require each team to hand in a report to briefly explain your work. The report will account for 60 percent of your final project grade (see next Section). Please write the report in human-readable English (preferred) or Chinese, lest the grading TAs should not be able to fully understand your work.

You are suggested to use the [latex report template](#) to formulate your report, but please feel free to use other software to produce the report. The report you submit to the Gradescope should contain below information and sections in **no more than 3 A4 pages**:

- Information
 - Team ID (eg. Team No.0)
 - Student IDs of all members (eg. B08902071, B08902143)

- Sections

1. **Data Structures & Algorithms**

Explain what data structures and algorithms you have tried and why you choose to use them. For example: We use _____ to _____, so that we can _____ in $O(__)$.

2. **Cost Estimations of Queries**

Provide and compare the cost estimation of your implementation on each types of queries. Using a table to show difference is encouraged. For example: Our implementation solves _____ in $O(__)$, which is faster than solving _____ in $O(__)$.

3. **Scheduling Strategy**

Describe how you design the strategy to answer queries. Does your strategy work as well as you imagined? If so/not, why? For example: We decide the priorities depending on _____, which is more relevant to _____ than _____, since _____.

4. **Additional Notes (*optional)**

Supply any other efforts/findings you have tried/discovered that (might) improves your implementation. For example: We also tried _____, and the result is _____.

Grading

By default, final project accounts for 10 percent of your raw score of this course. We will map the final score you got in final project to that 10 percent by a monotonic (increasing) function. The total score of final project is designed to be 1000, and split into the competition part and report part.

$$\text{Score}_{\text{Final}} = 1000 \cdot (0.4 \cdot \text{Score}_{\text{Competition}} + 0.6 \cdot \text{Score}_{\text{Report}})$$

The score within each part will be percentages, graded with different criteria. The Competition Score will be assigned depending on your ranking and points of your best submission. You may find the baselines released by the TAs on the scoreboard, eg: Baseline (30%), Baseline (70%) or TA Submission (Jason). Those baselines with percentage attached means if you surpass such baseline, you will at least get such percentage of Competition Score. The baselines without percentages are usually submitted by TAs just for fun or to show how good they can solve such problem. However, you may receive secret gifts from some TAs by surpassing their submissions.

The Report Score will be qualitatively graded with letters: A[0.95], B[0.8], C[0.65], D[0.5], F[0.3], with +/- denoting +0.05/-0.05. The qualitative description of each letter is listed below:

- A. Explain the implementation detailed and concisely. Provide profound and accurate analysis on performance and costs. Being able to design effective and explainable strategies based on the analysis.
- B. Some implementation details are missing. Small mistakes in assumptions and analysis. Slightly flawed explanation of strategy.
- C. Mediocre explanation with flawed analysis and unconvincing strategy explanation.
- D. Vague explanation of implementation, seriously wrong analysis on performance or costs. Unable explain the strategy design.
- F. Some part of the report is basically not understandable, missing, seriously incorrect, or found committing plagiarism.