Programming Test for Engineering Candidates

MatrixOrigin Technologies

June 8, 2021

1 Overview

There are three problems in this test. In each problem, please provide well-documented source code files, a makefile, and a readme file. All programs should compile in Linux. You have to use the specified programming language for each problem. Every problem chooses C++ as its designated programming language. You can use standard C++ librariess. No other third party code is allowed in this test. During the test you can use your books, use Internet for research, or any other means to come up with proper solutions.

2 Evaluation

You must produce a correct output to these questions given a random input according to the input file format. If your program does not produce a correct output file, you fail in this test. You must pass all three questions in the correctness test. We will also measure your performance in terms of execution speed. If your program takes more than 1 minute for a reasonable-size input file, you will also fail. In addition to the correctness and performance evaluation, we will also evaluate your documentation skill. All source files must provide Doxygen documentation. Also a well-documented PDF readme file must be provided. In readme, overview of your solution, algorithm, data structures, and an architecture diagram of your program must be provided. We will also evaluate your English writing skill through this documentation. If your documentation is hard to understand or does not provide enough details in each area of the above, you will also fail in this test.

3 Submission

Please use Linux tar command to archive your source code, Makefile, Readme files. No object files or executables should be included in the archive. An archive name should be firstname.lastname.tar. A Linux "tar xvf firstname.lastname.tar" should be able to extract your archive successfully by creating appropriate problem directories. The archive should have the following directories and files.

./*Problem_A* source code files, readme.PDF, makefile, and Doxygen files.

./*Problem_B* source code files, readme.PDF, makefile, and Doxygen files.

./*Problem_C* source code files, readme.PDF, makefile, and Doxygen files.

Please submit your solution to "yingfeng.zhang @matrixorigin.cn" or "yingfeng.zhang @gmail.com". Please note that you can submit your solution only once. Any subsequent submissions will be ignored.

4 Deadline

There is no deadline to this test. You can take as much time as you need to produce a correct output for any random input file. You also need to spend a good amount of time in documentation. Normally, a good engineer takes 2 weeks to finish this test. Some engineers take a month or just a week. It depends on each candidate's personal workload and schedule. We do not take into consideration about how fast you produce a solution. Instead, we focus on

the quality of your solution and documentation and ensure you do a thorough job. An engineer must produce a nice result.

5 Problem A

A paper "Deterministic Skip Lists" is referenced for this problem. Your task is to understand the algorithm in the paper and implement search, insert, and remove functions. Both search and insert function codes are shown in the paper, but no remove function code. Your job is to implement the proposed data structure including remove function and create a program using the data structure to process the input and output file as specified below. You are welcome to do additional optimization beyond this paper.

5.1 Input and Output

Input file contains a series of commands to insert/remove/find some strings. Your program should execute the commands as it sees and produce an output file according to each command. Insert command, denoted by "INS 'some string'" inserts the "some string" into the data structure. If there is already a duplicate string already in the data structure, you should print "DUP 'some string'", otherwise "INS 'some string'". Remove command, denoted by "REM 'some string'" removes the 'some string" from the data structure. If there is no such string in the data structure, you should print "NONE 'some string'", otherwise "REM 'some string'". Find command, denoted by "FIND 'some string'" finds the "some string" in the data structure. If founds, you should print "FIND 'some string'", otherwise "NONE 'some string'".

5.2 Sample Input

INS abc
FIND abc
FIND lll
REM lll
INS lll
FIND lll
REM lll
INS abc

5.3 Sample Outputg

INS abc
FIND abc
NONE lll
NONE lll
INS lll
FIND lll
REM lll
DUP abc

Please do not assume any limit on the number of commands in the input file. If your program cannot allocate more memory, then just print out an error message that "no more available memory" in the output file; at the time of such memory allocation error, you should have executed enough number of

commands in the input file. Before exiting, please ensure that you close both input/output files.

5.4 Program Usage

Problem_A ⟨input_file⟩ ⟨output_file⟩

6 Problem B

Your task in this problem is to produce edit operations for two random strings to align based on the algorithm provided in the Chapter_12.pdf file. The input contains two random strings in each line delineated by a space. The output produces the edit operations in order to make the first string align (equal to) the second string. For this problem, you will have to fully comprehend the provided Chapter_12.pdf. A detail pseudo code and some relevant background information are presented in that article.

The output of Problem B contains two lines for each line of input. Also note that there is an additional newline character in between solutions for input lines. A character '-' means no character. So the sequence (- a) means inserting a character 'a', while (a -) means deleting 'a'. The Sample Output should give you a clear idea of how an output should be processed.

Please make sure that you produce an optimal path of edit operations. There could be multiple optimal paths. In such a case, it suffices to produce one of the optimal paths.

6.1 Sample Input

abc lajfa
abc aljdfla
abc aldabc
asdf ajaf
af asdf

6.2 Sample Output

-a-bc
lajfa

a—-bc
aljdfla

a—bc
aldabc

asdf
ajaf

a–f
asdf

6.3 Regarding optimal path vs. non-optimal path

Please note that there can be many edit distance operations for any two random strings. Not all the edit distance operations are optimal edit distance operations (paths). In this problem, we are seeking for finding optimal paths. Please also

note that there can be many optimal paths given any two random strings. As long as you produce an optimal path, your solution will be accepted.

For example, let us consider the following example where both optimal and non-optimal paths are provided.
Input file:
abcd acdba

Solution One:
a b c d–
a - c d a b
Solution Two:
a b c d -
a c d a b

The first solution has the optimal path with edit distance of 3, while the second solution, although still producing an aligned pair, has a non-optimal path with edit distance of 4.

6.4 Program Usage

Problem_B ⟨input_file⟩ ⟨output_file⟩

7 Language Independent Extractive Summarization

After reviewing the attached paper Language Independent Extractive Summarization, your job is to implement the proposed text summarization schemes in the paper. The paper uses PageRank and HITS. Your program should take an option (-pagerank, -hits) to produce a correct output depending on the algorithm used. Your program takes a text file as an input and produces its summary to output file. In this problem, we represent a summary of a document as a list of sentences in order of importance (or weight) computed based on the algorithm in the paper. The weight should be normalized in range [0, 1] where 1 signifies the highest weight.

7.1 Input and Output

The input is the filename that is a text document. You can assume it is a pure text document without tags. The output is the summary of the input text document according to the summarization algorithm chosen, PageRank or HITS. The output format should show each sentence with its weight indicated in parenthesis at the start of a sentence. For example, an output could be like this:

```
(0.94323) this is the most important key sentence . . .
(0.82434) this is the second important key sentence . . .
(0.63232) another sentence . . .
(0.4232) yet another one . . .
(0.00002) least unimportant sentence . . .
```

7.2 Program Usage

```
Problem C <-pagerank|-hits> <input text file> <output summary file>
```