CS2020: Data Structures and Algorithms (Accelerated)

Data Analysis Speed Demon

Competition!

Overview

We live in a world overwhelmed with information. Every two days, we create as much new information as was created in the entire history of human civilization up until 2003 (according to Eric Schmidt, the CEO of Google). We produce more than 3 exabytes of data per day! Much of this data is stored in large databases, and one of the challenges today is to rapidly process and analyze all the data. Since the databaes are so large, it requires very fast algorithms and data structures that are highly optimized for maximum efficiency. In this competition, you will try to develop the *fastest* algorithm for analyzing a large dataset.

When analyzing a large dataset, there are many different goals. We will focus on a particular type of data mining in which we want to discover properties and patterns of the underlying data. Frequently, we want to know various statistics: the average, the median, the mode. Often, we also want to know about patterns: how often do users of a certain type (e.g., males between the ages of 18 and 32) buy a certain item? Often, we want to know about correlations: how often does a user who buys item A click on link B?

For the purpose of this competition, we define an abstract data mining problem that involves finding correlations in our data set. The database consists of a large set of very large data entries. The goal is to find how many pairs of entries are identical, i.e., contain the same information. Your job is to implement a data mining program that reads in the database and performs this analysis *as fast as possible.*

Problem Details

We now describe the details of the competition more precisely.

Input. The input "database" is a file consisting of a set of lines of text, each of which represents one entry. Each line of text contains a large number of characters. (Notice that the lines may consists of thousands, or even tens of thousands, of characters.) Your program will be passed the name of the database file as a parameter. For example, if the database is stored in the file `database.in`, then we will execute your program with the string `database.in` as the first parameter.

The format of the input is as follows. The first line of the database contains a single integer i, which represents the number of entries in the database. It is followed by i lines, each containing an arbitrary number of characters and ended by an end-of-line character (ASCII character 10). Note that entries may consist of any 128 legal ASCII characters, except for 10 and 13 (which indicate a new line). Characters may be repeated, and entries may be of any length.

Output. Your program should calculate the number of pairs of entries that contain an identical set of characters. Notice that the characters may appear in any order: two entries e_1 and e_2 are equivalent if e_1 is equal to some permutation of e_2. You should write your output to `stdout`. It should consist of an integer, followed by a newline character.

Example. The following is an example of an input database:

```
7
BCDEFGH
ABACD
BDCEF
BDCAA
DBACA
DABACA
DABAC
```

The appropriate output in this case is:

```
6
```

In particular, note the following six pairs of equivalent entries:

```
(ABACD, BDCAA)
(ABACD, DBACA)
(ABACD, DABAC)
(BDCAA, DBACA)
(BDCAA, DABAC)
(DBACA, DABAC)
```

Rules

The following are the rules of the competition:

- You may work in teams of one or two.
- Your solution must be written in Java.
- There will be two categores of competition. The *JavaPro* submissions to the competition can use any functionality available in the Java libraries (e.g., `java.util.*`). The *ExpertDIY* submissions to the competition must be entirely written from scratch. They may not use any of the Java libraries, and every part of the program must by written by you. The only exception to this rule is related to file I/O: in either category, your submission may use `java.io.*` to read and write the file. You are allowed to submit one program in each category, however, you are encouraged to focus your time and effort on optimizing one program.
- All programs must be written entirely by you. You may use any ideas or algorithms that you find on the internet, in books, etc. However, all the submitted code must be written by you.
- The competition will end on Wednesday, March 23 at noon.

- We will begin uploading your solution on Monday, March 14. You may continue to update your solution up until the deadline. We will post occasional updates as to the current *leader* in each category based on our preliminary testing. The final winner will be determined by separate testing that occurs after the competition ends.
- Your submissions must be named either `JavaPro.java` or `ExpertDIY.java`, according to the appropriate category.
- All competitors will receive experience points, and the winners will receive an extra bonus.

Hints

A few hints toward achieving good performance:

- First, develop and test a working solution that achieves good asymptotic performance. Then improve it.
- Think about the performance of the data structures you are using and the actual costs of the operations involved.
- Remember that for large databases, memory usage is important. Maintaining big data structures that use a lot of memory can be slow.
- Think about data locality in memory: accessing elements in memory that are near to each other is much cheaper than accessing elements that are far away from each other.
- Beware of the small costs that add up. For example, declaring a variable leads to a memory allocation which has a cost.
- Beware the costs of recursion.
- Profile your solution extensively to determine what to optimize.