Problem Solving Workshop #18 Tech Interviews and Competitive Programming Meetup

January 21, 2016 https://www.meetup.com/tech-interviews-and-competitive-programming/

Instructor: Eugene Yarovoi (can be contacted through the group Meetup page above under Organizers)

More practice questions: leetcode.com, glassdoor.com, careercup.com, geeksforgeeks.org
Books: Elements of Programming Interviews, Cracking the Coding Interview
Have questions you want answered? Contact the instructor, or ask on Quora. You can post questions and follow the instructor and other people who write about algorithms.

Try to find optimized solutions, and provide a time and space complexity analysis with every solution.

Problem #1, "Time Traveling Counter"

(i) **[Easy]** Design a class whose purpose is to accumulate counts of Strings. It should have the following methods:

```
int getCount(String str)
void add(String str)
```

`add` inserts one more instance of the given string. `getCount` returns how many times the given string was added. If you try to get the count of a string that was never added, 0 should be returned (it should not be an exception or cause a problem for your code -- this is an expected use case).

(ii) **[Easy-Medium]** Now, you have access to a `getTime()` function that returns a `long` representing the current time (strictly increasing every time you call it). How would you modify your implementation to have the following method signatures instead:

```
int getCount(String str, long timestamp)
void add(String str)
```

`getCount` should now return the count as it would have been at the specified timestamp. If that timestamp is at the same time as when an `add` happens, the `add` is included in the number returned by `getCount`.

For example, if the following sequence of calls is made at the following times:

T = 0 `add("foo")`
T = 2 `add("bar")`
T = 3 `add("foo")`
T = 5 `getCount("foo", 2)` -> returns 1, because the action at time T=3 hadn't happened yet
T = 6 `getCount ("foo", 3)` -> returns 2, because at T = 3 both "foo" have been inserted

(iii) **[Easy-Medium]** Same as (ii), but support the `void reset(String str)` method, which resets the count of `str` at the current time. Concatenating to the above example, if then we have:

T = 7 `reset("foo")`

T = 8 `getCount("foo", 7)` -> returns 0, because the foos were already deleted at T = 7
T = 9 `getCount("foo", 6)` -> returns 2, because the foos had not yet been deleted at T = 6

(iv) **[Medium-Hard]** Same as (iii), but now we wish to allow for ***retroactive corrections*** to the counters by supporting passing a timestamp in `add` and `reset`. So, now there are additional overloaded methods that look like this:

```
void add(String str, long timestamp)
void reset(String str, long timestamp)
```

The effect of these methods should be that, for all `getCount` queries that follow, the data structure should act as though the `add` and `reset` actions had been enacted at the specified timestamps.

Problem #2, "Reshuffle"

[Medium] People are sitting in a row at a bar. Some seats are occupied and some seats are empty. We want to know: what is the smallest number of people who would need to move for everyone to be sitting in one contiguous block?

We can represent the bar as a string where each character is either E (seat is empty) or O (seat is occupied). The i-th character indicates the status of the i-th seat.

Example Input: EEEOEEEOOEOOEOOEEO
Output: 2
Explanation: The input is an 18-seat bar where (0-indexed) seats 3, 7, 8, 10, 11, 13, 14, 17 are occupied. In this case, if we have the following moves:
17 moves to 12
3 moves to 9
The people are now sitting in seats 7, 8, 9, 10, 11, 12, 13, 14 which are one continuous block of seats. There is no way to move fewer people and still have everyone sit together.

Problem #3, "Printing Directories"

Consider a command like `ls` in a shell. Suppose we want it to print all the files in a directory in multiple columns, so that the columns are aligned. For example, if we have the filenames A, Boo, Bzarr, and Gamma, they might get printed like this:

```
A    Bzarr
Boo  Gamma
```

The font is always monospace (each character takes up the same width), the width of a column is always the length of the longest string in that column plus 2 characters for padding (assume there's padding even for the rightmost column for simplicity), and the files are always printed in alphabetical order, going first from top to bottom, and then from left to right. If the N files are printed using R rows, then there

must be ceiling (N / R) columns. The last column may have some empty spaces. For example, if we have files named A, B, C, D and decide to use 3 rows, we will print:

```
A  D
B
C
```

In a terminal window, the maximum width of a line is typically limited by the size of the window. We don't want to exceed the line width limit. We also want to print the directory files as compactly as possible -- that is, using the smallest possible number of rows.

Given a line width limit L, and a list of files in alphabetical order (they must also be printed in that order), what is the minimum number of rows you need to print all the files, aligned in columns as shown earlier, without exceeding the line width limit?
(To make the problem more interesting, don't assume L is a small number).

Complexity guidance: [Easy] $O(N^2)$, **[Hard]** O(N log N)

Follow-up: What if you instead have to print in left-to-right, then top-to-bottom? E.g.

```
A  B  C
D
```