Comp 305 Spring 2021 Algorithms and Complexity — Final Project

1 Submission and Grading

Projects information will be stored Google Sheet and your progress will be followed via GitHub. Please open this link Google Sheet Link, and enter the reuired information in the google sheet. Please email to *agharelar20@ku.edu.tr* or *COMP305-staff group*, if there is any problem. You solution will be evaluated by some Test cases to measure both your algorithm's correctness and efficiency. Therefore, try to do your best and make your code efficient.

As a next step in the project settings, please open a GitHub public repository, include all of your members as contributors and add the repository link to the given Google Sheet link. This step is quite important for us to see your progress and has to be done quickly. Therefore, in the README file please keep a list of completed steps, a TO DO list and the results retrieved if there are any. You will also prepare a presentation and present to the TAs. Therefore, you should also create a Google Slides presentation and include its link to the Google Sheet document.

2 Presentation Details

Each of the presentations should take ∼10 minutes and there will be a 5-minute Q&A session afterwards. If a presentation lasts longer than 10 minutes, then it will be interrupted. During the presentation each of the groups should explain and report:

- The algorithm you designed to solve the problem, the choices of the data structures you used and your reasoning.
- The time complexity of your algorithm (and the space complexity if applicable).
- Your run times for each of the test cases.
- Further improvements that can be done as future works.

This project does not expect from you to come up with just one solution and then test only that solution. For each of the problems you can start with some baseline approaches with more complexity and improve the baseline algorithm step by step. Be as creative as possible. Report different approaches you tested and why did you decide on the final algorithm you present. Your grading will be based on your creativity, your cumulative progress and how well did you present your approach.

3 Deadlines

You can work on your project until the day of presentation. The project presentations will be held between ***24th-28th of May, 2021***.

In the following pages, you can see each of the available projects:

Counting Beautiful Strings

A **beautiful** string is defined as a string that can be constructed by concatenating two copies of a same string. For example, **"abaaba"** is a beautiful string, since it can be created by concatenating two copies of **"aba"**, however, **"aaa"** is not a beautiful string.

One day, Amir, who has been bored recently, decides to play with beautiful strings. More specifically, he wants to count the number of sub-strings in a predefined string. It is important to note that a sub-string of a string can be obtained by eliminating some characters (it can also be 0) from the original string, provided that the relative order of elements in sub-string does not changed.

Please help Amir to count the number of beautiful sub-strings of a string.

Input

The first line contains one integer **T**. ($1 \leq \mathbf{T} \leq 20$) – where **T** denotes the number of test cases.

In the next **T** lines, you will be given a string in each line.

Output

Print **T** lines, one for each test case, containing the number of the beautiful sub-string.

Example 1

Input

3

aaa

abab

baaba

Output

3

3

6

Explanation

First string:= we can eliminate any of the 'a' to get a beautiful sub-string.

Second string := ”abab”, ”aa”, ”bb”
Third string := ”bb”, ”baba”, ”baba”, ”aa”, ”aa”, ”aa”

Deliveries

You are suppose to find efficient algorithm that can solve the question in $O(n^3)$. You also need to make a document that clearly explain your approach, proof of correctness of the algorithm, and your reasoning for asymptotic complexity. You also need to manually write 10 test cases (both input and output) to check correctness of your program. Your test cases should be such that it consider a variety of possibilities.