A subsequence of a string $T = t_0 t_1 t_2 \dots t_{n-1}$ is $T' = t_{i_0} t_{i_1} \dots t_{i_m}$ where $i_0 < i_1 < \dots i_m$ and $m < n$.

A substring of a string is a subsequence of the string where every element is consecutive.

You will be given a string S. P is the set of all the distinct **substrings** of S of length 2. Now the elegancy of each element of P is the square of the index (1-based) in S of the first letter of that substring. If a substring occurs multiple times only the first occurrence should be considered for the elegancy. Suppose, $S =$ `abcabd`. This means P is consisted of the substrings `ab`, `bc`, `ca` and `bd`. And the elegancies of those substrings are 1, 4, 9 and 25 respectively.

Now you will be given another string T. You have to split T to minimum amount of strings such that every string is a **subsequence** of T, any of the strings should not contain any **substrings** of length 2 which don't belong to P. Every character of T should belong to **exactly** one string. If multiple ways to divide T to minimum amount of strings, you have to consider that which minimizes the total elegancy of all the strings. Elegancy of a string is the sum of elegancy of all the length 2 substrings of that string. For a one letter string the elegancy is 0. Total elegancy is the sum of elegancy of all the strings.

Lets say, $S =$ `abcabd` and $T =$ `bcadzb`. One of the valid ways to split T is: {`bc`,`ab`, `d`, `z`}. Note that {`acb`, `d`, `z`, `b`} is not a valid way because `acb` is not a subsequence in T. Also {`cab`, `bdz`} is not a valid way either because the string `bdz` contains `dz` which don't belong to P although all the elements are subsequences. Now the optimal subsequences for this are {`bcab`, `z`, `d`} which has total elegancy of $(14 + 0 + 0) = 14$. For this case you cant split T to less than 3 subsequences and with 3 subsequences it is the minimal total elegancy.

Input

First line of the input contains a number X, the number of test cases which is at most 20. Each case starts with S. The next line contains T. Both S, T contains only lowercase letters. S consists of at most 1000 characters and T consists of at most 100 characters. There won't be any blank lines between two lines.

Output

You have to output two numbers K and C separated by a space where K is the minimum amount of strings possible by splitting T according to the above rules and C is the minimum total elegancy.

Sample Input

```
1
abcabd
bcadzb
```

Sample Output

```
3 14
```