Code Jam 2020 - Virtual World Finals

Replace All

Problem

Banana Rocks Inc is coming up with a revolutionary technology to perform the common edit operation "replace all". Their implementation replaces every occurrence of a character within a given text with another character. (If the character does not appear in the text, then the operation occurs but has no effect.)

For example, if the starting text is `CODEJAMWORLDFINALS` and an operation is performed to replace `A` with `O`, the new text would be `CODEJOMWORLDFINOLS`. If another operation is performed on that result to replace `O` with `Y`, the final text would be `CYDEJYMWYRLDFINYLS`.

Unfortunately, the implementation is incomplete, so it can only perform replacements from a specific list of **N** pairs of characters. Also, even if a replacement of a specific character c_1 with another character c_2 is implemented, the reverse replacement of c_2 with c_1 may or may not be implemented.

You want to try all the implemented replacements. You are given some initial string **S** to use as the initial text. You can perform any number of replacements in sequential order: the first replacement is performed on **S**, and the (i+1)-th replacement is performed on the result of performing the i-th replacement. The only requirement is that each implemented replacement is performed at least once during this process. There is no upper limit on how many times you may perform each replacement.

The allowed characters are decimal digits and uppercase and lowercase English alphabet letters. In this problem, uppercase and lowercase versions of the same letter are treated as distinct characters.

What is the maximum number of unique characters that can appear in a text that is the result of the last replacement performed?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of two lines. The first line contains a string **S** and an integer **N**: the initial text and the number of implemented replacements. The second line contains **N** two-character strings $\mathbf{R_1}$, $\mathbf{R_2}$, ..., $\mathbf{R_N}$, representing the implemented replacements. $\mathbf{A_i}$ and $\mathbf{B_i}$ are the first and second characters of $\mathbf{R_i}$, respectively. The i-th implemented replacement corresponds to replacing all occurrences of $\mathbf{A_i}$ with $\mathbf{B_i}$.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the maximum number of unique characters that can appear in a text that is the result of performing all implemented replacements to **S** one or more times each, in some order.

Limits

Time limit: 60 seconds per test set.
Memory limit: 1GB.
$1 \le \mathbf{T} \le 100$.
$2 \le$ length of $\mathbf{S} \le 1000$, for all i.
Each character of **S** is an uppercase or lowercase English alphabet letter or a decimal digit.
$\mathbf{A_i}$ is an uppercase or lowercase English alphabet letter or a decimal digit, for all i.
$\mathbf{B_i}$ is an uppercase or lowercase English alphabet letter or a decimal digit, for all i.
$\mathbf{A_i} \neq \mathbf{B_i}$, for all i.
$(\mathbf{A_i}, \mathbf{B_i}) \neq (\mathbf{A_j}, \mathbf{B_j})$, for all $i \neq j$. (Each replacement is unique.)

Test Set 1 (Visible Verdict)

$2 \le \mathbf{N} \le 62$.
$\mathbf{B_i} \neq \mathbf{B_j}$, for all $i \neq j$.

Test Set 2 (Hidden Verdict)

$2 \le \mathbf{N} \le 62 \times 61$.

Sample

Sample Input

```
4
CODEJAMWORLDFINALS 2
AO OY
xyz 3
xy zx yz
CJ 4
20 2O HC KS
AB 2
Ab bA
```

Sample Output

```
Case #1: 14
Case #2: 2
Case #3: 2
Case #4: 2
```

The above cases meet the limits for Test Set 1. Another sample case that does not meet those limits appears at the end of this section.

Sample Case #1 is the one in the statement. Notice that if we perform the replacements in the order mentioned in the statement, we get 13 different characters in the final text. If we perform them both once in the other order, however, we can get `CYDEJOMWYRLDFINOLS`, which has 14 different characters.

In Sample Case #2, one way to get 2 different characters in the final text is to perform the replacements in the order given from left to right, once each.

In Sample Case #3, none of the replacements affect the text at all, so it does not matter how we apply them. We will always be left with the original two letters. Notice that replacements can contain characters not appearing in the initial text, and the initial text can contain characters not appearing in the implemented replacements.

In Sample Case #4, remember that uppercase `B` is not the same character as lowercase `b`.

The following additional case could not appear in Test Set 1, but could appear in Test Set 2.

```
1
1234 5
12 2X X3 31 X2
```

The correct output is `Case #1: 4`.

In this additional sample case, one possibility is to perform the replacements in the following order: `X3 2X X2 2X 12 31`. This process goes through the following strings, starting with **S**: `1234 1234 1X34 1234 1X34 2X34 2X14`.