Code Jam 2014 - Round 1C

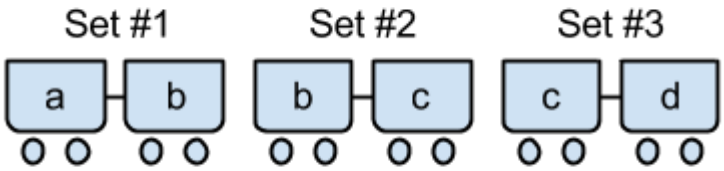
Reordering Train Cars

Problem

Yahya is a brilliant kid, so his mind raises a lot of interesting questions when he plays with his toys. Today's problem came about when his father brought him a set of train cars, where each car has a lowercase letter written on one side of the car.

When he first saw the gift, he was happy and started playing with them, connecting cars together without any particular goal. But after a while he got bored (as usual) from playing without having any goal. So, he decided to define a new interesting problem.

The problem is that he currently has **N** sets of connected cars. He can represent each set of connected cars as a string of lowercase letters. He wants to count the number of ways he can connect all **N** sets of cars to form one *valid* train. A train is valid if all occurrences of the same character are adjacent to each other.



The previous figure is one way Yahya could connect the cars "ab", "bc" and "cd" to make a valid train: "ab bc cd". If he had connected them in the order "cd ab bc", that would have been invalid: the "c" characters would not have been adjacent to each other.

You've surely noticed that this is not an easy problem for Yahya to solve, so he needs your help (and he is sure that you will give it!). That's it; go and help Yahya!

Note: letters are written only on one side of the cars, so you can not reverse them. For example, if a car has "ab" written on it, it could not be changed to read "ba".

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. The first line of each test case contains a single integer **N**, the number of sets of connected cars. The following line contains **N** strings separated by a single space. Every given string represents a set of connected cars and is composed of lowercase English letters only.

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 number of different ways of obtaining a valid train. As this number may be very big, output the number of ways *modulo 1,000,000,007*.

Limits

Memory limit: 1 GB.
1 ≤ **T** ≤ 100.
1 ≤ Set of connected **Cars'** lengths ≤ 100.

Small dataset

Time limit: 60 seconds.
1 ≤ **N** ≤ 10.

Large dataset

Time limit: 120 seconds.
1 ≤ **N** ≤ 100.

Sample

Sample Input	Sample Output
3 3 ab bbbc cd 4 aa aa bc c 2 abc bcd	Case #1: 1 Case #2: 4 Case #3: 0

Sample Explanation

In the first case, there is only one way to form a valid train by joining string *"ab"* to *"bbbc"* to *"cd"* in this order.

While in the second case, there are 4 possible ways to form a valid train. Notice that there are two different sets of connected cars represented by the string *"aa"*, so there are two different ways to order these two strings and to group them to be one set of connected cars *"aaaa"*. Also there is only one way to order set of cars *"bc"* with *"c"* in only one way to be *"bcc"*. After that you can order *"aaaa"* and *"bcc"* in two different ways. So totally there are 2*2 = 4 ways to form a valid train.

In the third sample case, there is no possible way to form a valid train, as if joined in any of the two ways *"abc"+"bcd"* or *"bcd"+"abc"*, there will be two letters of *"b"* and *"c"* not consecutive.