**Problem A. Big Buttons**

### Problem

You are a contestant on a popular new game show and are playing for the grand prize!

There are two big buttons, a red one and a black one. You will make a sequence of exactly **N** button presses.

There are lots of different sequences of presses you could make, but there are **P**forbidden prefixes, each of length no greater than **N**. If you make a sequence of presses which begins with *any* of the forbidden sequences, you will not win the grand prize. It is fine for your sequence to contain one or more forbidden prefixes as long as they do not appear at the start of your sequence.

A *winning* sequence must consist of exactly **N** button presses and must not begin with one of the forbidden prefixes. How many different winning sequences are there?

### Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing two integers **N** and **P**, as described above. Then, there are **P** more lines, each of which contains a string of between 1 and **N** characters, inclusive, describing one of the forbidden sequences of presses. An R represents pressing the red button, whereas a B represents pressing the black button.

### 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 winning sequences, as desribed above.

### Limits

1 ≤ **T** ≤ 100.  
1 ≤ **P** ≤ min(2**N**, 100).  
Each forbidden prefix is between 1 and **N** characters long, inclusive.  
No two forbidden prefixes will be the same.

#### Small dataset

1 ≤ **N** ≤ 10. **(In the contest, this condition is 1<=N<=5)**

#### Large dataset

1 ≤ **N** ≤ 50. **(In the contest, this condition is 1<=N<=30)**

### Sample

|  |  |
| --- | --- |
| Input | Output |
| 4  3 2  BBB  RB  5 1  R  4 3  R  B  RBRB  50 5  BRBRBBBRBRRRBBB  BRBRBRRRBRRRBRB  BBBRBBBRBRRRBBB  BRBRBRRRBRRRB  BRBRBBBRBBBRB | Case #1: 5  Case #2: 16  Case #3: 0  Case #4: 1125556309458944 |

Note that the last Sample case would not appear in the Small dataset.

In the first case, you must make a sequence of 3 presses. There are 8 possible sequences of three presses, but some of them will cause you to lose the game. They are listed below:

* RBB. This is forbidden since it starts with the first forbidden sequence (RB).
* RBR. This is forbidden since it starts with the first forbidden sequence (RB).
* BBB. This is forbidden since it starts with the second forbidden sequence (BBB).

Thus, there are only 5 winning sequences.

In the second case, you must make a sequence of 5 presses. There is only one forbidden sequence, which is R. This means that the first press must be B, and the next 4 presses can be either button. This gives a total of 16 different button presses.

In the third case, you must make a sequence of 4 presses. There are three forbidden sequences, but since every possible sequence begins with either R (the first forbidden sequence) or B (the second forbidden sequence), there are no winning sequences. So the answer is 0.

**Contest Analysis**

Overview  |  Problem A  |  Problem B  |  Problem C

**Big Buttons: Analysis**

Small dataset

Since we have 2 choices at every step, there are 2**N** possible strings of length **N**. We can generate all of these strings in O(2**N**) time using a naive recursion. For each string, we need to check if any of the **P** strings is its prefix. A simple implementation of the above would run in O(2**N** × **P** × **N**) time for each case. Since **N** ≤ 5 in this dataset, this is fast enough.

分析程序的复杂度，简单情况用简单方法求解即可

Large dataset

In this dataset, since **N** can be as large as 30, **it is not feasible to generate all 2N strings**. The main idea in solving the Large dataset is to find the number of **invalid** strings of length **N** and **subtract it** from the total number of strings (2**N**).

If a forbidden prefix has length L, we have 2(**N**- L) invalid strings of length **N** with that prefix. Also, if we have two strings A and B, and A is a prefix of B, then all strings that have B as their prefix will also have A as their prefix. So we can **remove all forbidden prefixes that themselves begin with a different forbidden prefix** to avoid overcounting the sum of invalid strings.

We can now easily count the sum of invalid strings by finding the number of invalid strings for each remaining forbidden prefix. We can **remove redundant forbidden prefixes** in O(**P** × **P** × **N**) time, and find the counts of strings with the remaining forbidden prefixes in O(**P** × **N**) time, so the overall running time is O(**P**2**N**). This is fast enough to solve the Large data set, but can you see how to improve the first step to O(**P** × **N**) time by using a [trie](https://en.wikipedia.org/wiki/Trie)?