PermRLE

You've invented a slight modification of the run-length encoding (RLE) compression algorithm, called PermRLE.

To compress a string, this algorithm chooses some permutation of integers between 1 and k, applies this permutation to the first k letters of the given string, then to the next block of k

letters, and so on. The length of the string must be divisible by k. After permuting all blocks, the new string is compressed using RLE, which is described later.

To apply the given permutation p to a block of k letters means to place the p[1]-th of these letters in the first position, then p[2]-th of these letters in the second position, and so on.

For example, applying the permutation {3,1,4,2} to the block "abcd" yields "cadb". Applying it to the longer string "abcdefghijkl" in blocks yields "cadbgehfkilj".

The permuted string is then compressed using run-length encoding. To simplify, we will consider the compressed size of the string to be the number of groups of consecutive equal letters. For example, the compressed size of "aabcaaaa" is 4; the first of the four groups is a group of two letters "a", then two groups "b" and "c" each containing only one letter, and finally a longer group of letters "a".

Obviously, the compressed size may depend on the chosen permutation. Since the goal of compression algorithms is to minimize the size of the compressed text, it is your job to choose the permutation that yields the smallest possible compressed size, and output that size.

Input

The first line of input gives the number of cases, N. N test cases follow.

The first line of each case will contain k. The second line will contain S, the string to be compressed.

N = 20 S will contain only lowercase letters 'a' through 'z' The length of S will be divisible by k

2 <= k <= 16 1 <= length of S <= 50000

Output

For each test case you should output one line containing "Case #X: Y" (quotes for clarity) where X is the number of the test case and Y is the minimum compressed size of S.

Example

Input:

2
4 abcabcabcabc
3 abcabcabcabc

Output:

Case #1: 7
Case #2: 12