Untangle String

You have two strings, **S** and **R**, each with a length of ***n***, where every character in both strings is unique (no character is repeated in either string). You also have an initially empty string P.

You need to perform the following operations on string S. In each operation, you will do the following:

1. Choose two integers l and r($1 \le l \le r \le k$)(where k represents the current length of string S).
2. Append the substring S[l]+S[l+1]+...+S[r] to the end of string P.
3. Delete the substring S[l]+S[l+1]+...+S[r], from string S, and then merge the substrings S[1]+a[2]+...+a[l−1] and S[r+1]+S[r+2]+...+S[k](new length of string S will be k−r+l−1).

A substring of a string is defined as a sequence of consecutive characters of the string.

Calculate the minimum number of operations required to make String P equal to string S.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1e6$). The description of the test cases follows.

The first line of each test case contains a single integer n($1 \le n \le 26$) — the length of the string S.

The second and third line of each test case consists of string S and R respectively, consisting of lowercase English letters.

It is guaranteed that all characters in S are pairwise distinct, and the same is true for R (no character repeats within S, and no character repeats within R).

It is guaranteed that for every character in S, there is a corresponding matching character in string R.

Output

For each test case, output the minimum number of operations.

Example

Input

2
5
abdec
abcde
5
ceyfg
cefgy

Output

3
3

Explanation

First test case

S [abdec], R [abcde], P[]
Operation 1 : S[dec] , P[ab]
Operation 2 : S[de] , P[abc]
Operation 3 : S[] , P[abcde]

Second test case

S [ceyfg], R[cefgy], P[]
Operation 1 : S[yfg] , P[ce]
Operation 2 : S[y] , P[cefg]
Operation 3 : S[] , P[cefgy]