Competitive Programming

Strings

B.Bhuvaneswaran, AP (SG) / CSE

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

Cyclic Palindrome

- A string is said to be palindrome, if it reads the same from both the ends.
- Given a string S, you are allowed to perform cyclic shifts.
- More formally, you can pick any one character from any end (head or tail) and you can append that character at the other end.
- For example, if the string is abc, then if we do a shift using the character at head position then the string becomes bca.
- Similarly, if we do the shift using the character at the tail then the input string becomes cab.
- Your task is to find out the minimum number of shifts needed to make the given string, a palindrome.
- In case, we can't convert the string to palindrome then print -1.

Sample input and output

Input

4

abbb

aaabb

aabb

abc

Output

-1

1

1

-1

Example

- aaabb → Right shift → baaab
- aabb → Left shift → abba (or) Right shift → baab
- abc → Never become a palindrome

Example

- aabb
- abcd

Logic

- Keep shifting left → Check for palindrome
- Keep shifting right → Check for palindrome
- Till when?
  - Number of shifts <= length / 2

Pseudo code

```
input word
length = len(word)
shifts = 0
flag = 0
word_left = word_right = word
while shifts <= length / 2:
        if isPal(word_left) or isPal(word_right):
                flag = 1
                break
        word_left = leftshift(word_left)
        word_right = rightshift(word_left)
        shifts++
if flag == 1:
        print shifts
Else:
        print -1
```

Pseudo code: isPal

```
n = len(string)
for i = 0 to n/2:
        if string[i] != string[n - i - 1]:
                return false
return true
```

Example

- abccba
- abcdba
- abccda

Pseudo code: leftShift

```
n = len(string)

shifted_string = string[1, n - 1] + string[0]

return shifted_string
```

Pseudo code: leftShift (C)

```
n = len(string)
for i = 0 to n - 2:
        shifted_string[i] = string[i + 1]
shifted_string[i++] = string[0]
shifted_string[i] = '\0'
return shifted_string
```

Example

- abcd → bcda

Pseudo code: rightShift

```
n = len(string)
shifted_string = string[n - 1] + string[0, n - 2]
return shifted_string
```

Pseudo code: rightShift (C)

```
n = len(string)
shifted_string[0] = string[n - 1]
for i = 1 to n - 1:
        shifted_string[i] = string[i - 1]
shifted_string[i] = '\0';
return shifted_string;
```

Example

- abcd → dabc

Lexi String

- Little Jill jumbled up the order of the letters in our dictionary. Now, Jack uses this list to find the smallest lexicographical string that can be made out of this new order. Can you help him?
- You are given a string P that denotes the new order of letters in the English dictionary.
- You need to print the smallest lexicographic string made by rearranging all the letters of the given string S.

Constraints

- 1 <= T <= 1000
- Length (P) = 26
- 1 <= length (S) <= 100
- All characters in the string S, P are in lowercase.

Input Format

- The first line contains number of test cases T.
- The second line has the string P.
- The third line has the string S.

Output Format

- Print a single string in a new line for every test case giving the result.

Example

Input

2

polikujmnhytgbvfredcxswqaz

abcd

qwryupcsfoghjkldezxvbintma

Ativedoc

Output

bdca

codevita

Explanation

- The transformed smallest lexicographical strings are in order they would be if order of letters are changed to string P.

Minimum String Rotations

- A rotation on a string is defined as removing first element and concatenating it at the end.
- Given N, and an array of N strings, print the minimum no. of cumulative rotations on the strings so as to make all the strings equal.
- If this is not possible return -1

Input Format

- The first line contains N, the number of strings
- This is followed by N strings

Output Format

- $2 <= N <= 10^4$
- 3 <= string length <= 100
- All characters are in uppercase

Example

Input

4

AABCD

CDAAB

DAABC

AABCD

Output

3

Explanation

- Finally, all the string will become aabcd. First and last strings require no rotations.
- Second string requires 2 rotations
- Last string requires 1 rotation
- Hence total rotations required are 3

Queries?

Thank You…!