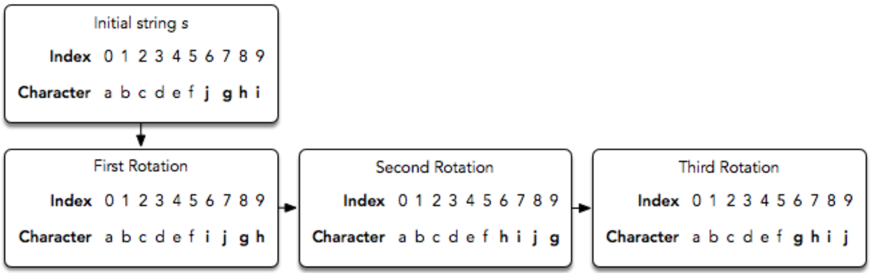
Suffix Rotation

by forthright48

Megan is playing a string game with the following rules:

- It starts with a string, s.
- During each turn, she performs the following move:
  - Choose an index in s. The chosen index must be strictly greater than any index chosen in a prior move.
  - Perform one or more circular rotations (in either direction) of the suffix starting at the chosen index.

  For example, let's say $s =$ `abcdefjghi`. During our move, we choose to do three right rotations of the suffix starting at index 6:



Initial string s

Index	0	1	2	3	4	5	6	7	8	9
Character	a	b	c	d	e	f	j	g	h	i

First Rotation

Index	0	1	2	3	4	5	6	7	8	9
Character	a	b	c	d	e	f	i	j	g	h

Second Rotation

Index	0	1	2	3	4	5	6	7	8	9
Character	a	b	c	d	e	f	h	i	j	g

Third Rotation

Index	0	1	2	3	4	5	6	7	8	9
Character	a	b	c	d	e	f	g	h	i	j

  Note that this counts as *one* move.

- The goal of the game is to convert s into the lexicographically smallest possible string *in as few moves as possible*. In other words, we want the characters to be in alphabetical order.

Megan plays this game g times, starting with a new string s each time. For each game, find the minimum number of moves necessary to convert s into the lexicographically smallest string and print that number on a new line.

Input Format

The first line contains an integer, g, denoting the number of games.
Each of the g subsequent lines contains a single string denoting the initial value of string s for a game.

Constraints

- $1 \leq g \leq 100$
- $1 \leq |s| \leq 1000$
- s consists of lowercase English alphabetic letters only.

Output Format

For each game, print an integer on a new line denoting the minimum number of moves required to convert s into the lexicographically smallest string possible.

Sample Input 0

```
3
abcdefghij
acab
baba
```

Sample Output 0

```
0
1
2
```

Explanation 0

We play the following $g = 3$ games:

1. In the first game, `abcdefghij` is already as lexicographically small as possible (each sequential letter is in alphabetical order). Because we don't need to perform any moves, we print 0 on a new line.
2. In the second game, we rotate the suffix starting at index 1, so **`acab`** becomes **`aabc`**. Because the string is lexicographically smallest after one move, we print 1 on a new line.
3. In the third game, we perform the following moves:
   - Rotate the suffix starting at index 0 (i.e., the entire string), so **`baba`** becomes **`abab`**.
   - Rotate the suffix starting at index 1, so **`abab`** becomes **`aabb`**.

   Because the string is lexicographically smallest after two moves, we print 2 on a new line.

Submissions:79

Max Score:80

Difficulty: Expert

Rate This Challenge:

☆ ☆ ☆ ☆ ☆

More

Current Buffer (saved locally, editable) Java 7

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int q = in.nextInt();
        for(int a0 = 0; a0 < q; a0++){
            String s = in.next();
            // your code goes here
        }
    }
}

```

Line: 1 Col: 1

Upload Code as File □ Test against custom input Run Code Submit Code

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature