**Day 73 coding Statement :**

A string is called *boring* if all the characters of the string are **same**.

You are given a string *S* of length *N*, consisting of lowercase english alphabets. Find the length of the longest *boring* substring of *S* which occurs **more than once**.

Note that if there is no *boring* substring which occurs more than once in *S*, the answer will be 00.

A substring is obtained by deleting some (possibly zero) elements from the beginning of the string and some (possibly zero) elements from the end of the string.

**Input Format**

* The first line of input will contain a single integer *T*, denoting the number of test cases.
* Each test case consists of two lines of input.
  + The first line of each test case contains an integer *N*, denoting the length of string *S*.
  + The next contains string *S*.

**Output Format**

For each test case, output on a new line, the length of the longest *boring* substring of *S* which occurs **more than once**.

**Sample Input**

4

3

aaa

3

abc

5

bcaca

6

caabaa

**Sample Output**

2

0

1

2

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

class RatanPrajapati\_day73 {

    public static void main(String[] args) throws IOException {

        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

        int T = Integer.parseInt(in.readLine());

        int m = 0;

        char[] c;

        while (T-- > 0) {

            m = Integer.parseInt(in.readLine());

            c = in.readLine().trim().toCharArray();

            int[] charCounter = new int[30];

            int longest = 0;

            char lastChar = c[0];

            int currLength = 1;

            for (int i = 1; i < c.length; i++) {

                if (c[i] == lastChar) {

                    currLength++;

                } else {

                    if (currLength >= charCounter[lastChar - 'a']) {

                        if (currLength > longest) {

                            if (currLength > charCounter[lastChar - 'a'] + 1) {

                                longest = currLength - 1;

                            } else {

                                longest = charCounter[lastChar - 'a'];

                            }

                        }

                        charCounter[lastChar - 'a'] = currLength;

                    }

                    lastChar = c[i];

                    currLength = 1;

                }

                if (i == (c.length - 1) && currLength > longest) {

                    if (currLength == c.length) {

                        longest = currLength - 1;

                    } else if (currLength >= charCounter[lastChar - 'a']) {

                        if (currLength > charCounter[lastChar - 'a'] + 1) {

                            longest = currLength - 1;

                        } else {

                            longest = charCounter[lastChar - 'a'];

                        }

                    }

                }

            }

            System.out.println(longest);

        }

    }

}