**Day 81 coding Statement :**

You have a **binary** string *S* of length *N*. In one operation you can select a substring of *S* and **reverse** it. For example, on reversing the substring [2,4]*S*[2,4] for *S*=11000, we change 11000→10010.

Find the **minimum** number of operations required to sort this binary string.  
It can be proven that the string can always be sorted using the above operation finite number of times.

**Input Format**

* The first line of input will contain a single integer *T*, denoting the number of test cases.
* Each test case consists of 22 lines of input.
  + The first line of each test case contains a single integer *N* — the length of the binary string.
  + The second line of each test case contains a binary string *S* of length *N*.

**Output Format**

For each test case, output on a new line — the minimum number of operations required to sort the binary string.

**Sample Input**

4

3

000

4

1001

4

1010

6

010101

**Sample Output**

0

1

2

2

import java.util.Scanner;

public class RatanPrajapati\_day81 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int T = sc.nextInt();

        while (T-- > 0) {

            int n = sc.nextInt();

            String st = sc.next();

            int ans = 0;

            for (int i = n - 2; i >= 0; i--) {

                if (st.charAt(i) == '1' && st.charAt(i + 1) == '0') {

                    ans++;

                }

            }

            System.out.println(ans);

        }

    }

}