Problem L. Fenwick Tree

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	1024 megabytes

Prof. Pang is giving a lecture on the Fenwick tree (also called binary indexed tree).

In a Fenwick tree, we have an array $c[1 \dots n]$ of length n which is initially all-zero ($c[i] = 0$ for any $1 \le i \le n$). Each time, Prof. Pang can call the following procedure for some position pos ($1 \le pos \le n$) and value val:

```
def update(pos, val):
    while (pos <= n):
        c[pos] += val
        pos += pos & (-pos)
```

Note that `pos & (-pos)` equals to the maximum power of 2 that divides `pos` for any positive integer `pos`.

In the procedure, val can be **any real** number. After calling it some (zero or more) times, Prof. Pang forgets the exact values in the array c. He only remembers whether $c[i]$ is zero or not for each i from 1 to n. Prof. Pang wants to know what is the minimum possible number of times he called the procedure assuming his memory is accurate.

Input

The first line contains a single integer T ($1 \le T \le 10^5$) denoting the number of test cases.

For each test case, the first line contains an integer n ($1 \le n \le 10^5$). The next line contains a string of length n. The i-th character of the string is `1` if $c[i]$ is nonzero and `0` otherwise.

It is guaranteed that the sum of n over all test cases is no more than 10^6.

Output

For each test case, output the minimum possible number of times Prof. Pang called the procedure. It can be proven that the answer always exists.

Example

standard input	standard output
3 5 10110 5 00000 5 11111	3 0 3

Note

For the first example, Prof. Pang can call `update(1,1)`, `update(2,-1)`, `update(3,1)` in order.

For the third example, Prof. Pang can call `update(1,1)`, `update(3,1)`, `update(5,1)` in order.