Morse (`walrus`)

In an unspecified zoo, there are n walruses (in romanian "*morse*") sitting neatly in a line.

Figure 1: *Odobenus Rosmarus*, also known as *walrus*.

The walruses are denoted by a string of n characters $\overline{c_1 c_2 \ldots c_n}$, where:

- $c_i =$ `'.'`, if the i-th walrus is sleeping; or
- $c_i =$ `'-'`, if the i-th walrus is awake.

Each second, the following things can happen:

1. You can choose to wake up any walrus yourself; and
2. Every walrus which was awakened **in the previous second** will wake up its neighbours. As an exception, the walruses which were awake at the start will never wake up their neighbours.

Since waking up the walruses yourself is dangerous, you ask yourself the following questions:

1. What is the smallest number of walruses you need to wake up yourself in order to guarantee that all walruses will eventually be awake?
2. While waking up as few walruses yourself as possible, what is the earliest possible time (in seconds) when all of the walruses will be awake?

☞ Among the attachments of this task you may find a template file `morse.*` with a sample incomplete implementation.

Input

Each test contains multiple test cases. The first line of input contains a single integer t — the number of test cases.

The first line of each test case contains a single integer n — the number of walruses.

The second line of each test case contains a string of n characters $\overline{c_1 c_2 \ldots c_n}$, where:

- $c_i =$ `'.'`, if the i-th walrus is initially asleep; or
- $c_i =$ `'-'`, otherwise.

Output

For each test case, print two integers:

1. The smallest number of walruses you need to wake up yourself in order to guarantee that all walruses will eventually be awake;
2. While waking up as few walruses yourself as possible, the earliest possible time (in seconds) when all of the walruses will be awake.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

Constraints

- $1 \le t \le 10\,000$.
- $1 \le n \le 300\,000$.
- It is guaranteed that **at least one walrus is initially asleep** and that the sum of n across all test cases does not exceed $300\,000$.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (10 points)	All walruses are initially asleep.
– **Subtask 3** (20 points)	$n \le 10$.
– **Subtask 4** (35 points)	$t \le 10, n \le 2000$.
– **Subtask 5** (35 points)	No additional constraints.

Examples

input	output

3
5
`..-..`
3
`...`
20
`...---....---.--...-`
	2 3
	1 2
	4 4

Explanation

An optimal strategy for the first test case:

In the first second, you can wake up the second walrus yourself:

$$''..-..'' \rightarrow ''.--..''$$

In the second second, you can wake up the fifth walrus yourself. Additionally, the second walrus will wake up the first walrus:

$$''.--..'' \rightarrow ''.--.-'' \rightarrow ''---.-''$$

In the third second, you will not wake up any walruses yourself. However, the fourth walrus will be awakened by the fifth walrus:

$$''---.-'' \rightarrow ''-----''$$

In total, you will have awakened 2 walruses yourself, which is the minimum required to wake up every walrus.

While waking up 2 walruses yourself, the minimum total time needed so that every walrus wakes up is 3 seconds.

An optimal strategy for the second test case:

In the first second, you will wake up the second walrus yourself. The other walruses will be woken up in the next second:

$$''...'' \rightarrow ''.-.'' \rightarrow ''---''$$