Escaping from escaping

[Due to SPOJ restrictions, this problem has been modified with respect to the original version used in the Argentinian Programming Tournament of 2013 in order to have multiple test cases per input file. The original version of this problem (in Spanish) can be found at http://www.dc.uba.ar/events/icpc/download/problems/tap2013-problems.pdf]

A communications protocol is a set of rules designed to transfer information in a communications system. Elisa's job is to write programs that implement parts of said protocols. It is often necessary for this to transfer sequences of fields, and in order to know where a field ends and the next one begins it is customary to insert a separator between each pair of consecutive fields. Using a simple separator such as a space, a comma or a semicolon is inconvenient because sometimes the fields to be transferred contain these same characters. The standard solution in these cases is to insert an "*escaping*" character just before every appearance of a separator inside a field, so that it can be thus distinguished from a real separator. Elisa thinks this solution will increase a lot the length of the data to be transmitted, so she has decided to use a separator that is complex enough for it to never appear inside the data. In this way she hopes to escape the inefficient alternative of escaping separators.

To choose the ideal separator, Elisa has compiled a *log*, which is nothing else than a long string of characters that is representative of the data that her protocol needs to manage. After thinking about the problem for a while, Elisa reached the conclusion that any non-empty string of characters that does not appear inside the log would be an acceptable separator for use within her protocol. But because she is interested in minimizing the length of the data to be transmitted, she would like to know the minimal length that an acceptable separator can have. She immediately wrote a program to calculate this length, and is now testing it for the particular case in which both the log and the acceptable separators only contain binary digits (**'0'** and **'1'**). Can you anticipate the results?

Input

The first line contains an integer number **T**, the number of test cases (**1 ≤ T ≤ 200**). Each of the following **T** lines contains a log, which is a non-empty string of at most 10^5 binary digits.

Output

For each test case, print a single line containing an integer number representing the minimal length of an acceptable separator for the given log.

Example

Input:

3
011101001
100010110011101
11111

Output:

3
4
1