Time is the longest distance between two places.
Tennessee Williams

In computer science, edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. In bioinformatics, it can be used to quantify the similarity of DNA sequences, which can be viewed as strings of the letters A, C, G and T.

Different definitions of an edit distance use different sets of string operations. The Levenshtein distance operations are the removal, insertion, or substitution of a character in the string. Being the most common metric, the Levenshtein distance is usually what is meant by "edit distance".

Formal definition and properties

Given two strings a and b on an alphabet Σ the edit distance $d(a, b)$ is the minimum-weight series of edit operations that transforms a into b. One of the simplest sets of edit operations is that defined by Levenshtein in 1966:

Insertion of a single symbol. If $a = uv$, then inserting the symbol x produces uxv. This can also be denoted $\varepsilon \to x$, using ε to denote the empty string.

Deletion of a single symbol changes uxv to uv ($x \to \varepsilon$).

Substitution of a single symbol x for a symbol $y \neq x$ changes uxv to uyv ($x \to y$).

In Levenshtein's original definition, each of these operations has unit cost (except that substitution of a character by itself has zero cost), so the Levenshtein distance is equal to the minimum *number* of operations required to transform a to b.

Example

The Levenshtein distance between "flow" and "grown" is 3. A minimal edit script that transforms the former into the latter is:

1. **f**low →**g**rown (substitution of "f" for "g")
2. f**l**ow →g**r**own (substitution of "l" for "r")
3. flow →grow**n** (insertion of "n" at the end).

Given two strings a and b on an alphabet Σ (the set of standard ASCII characters), obtain the Levenshtein distance between a and b, where $0 \leq length(a), length(b) \leq 100$.

Input

The first line of the input contains an integer, t, indicating the number of test cases. For each test case, two lines appear, the first one containing the first string a and the second one containing the second string b.

Output

For each test case the output should contain a single line, which consists of the corresponding Levenshtein distance.

Sample Input

```
6
impossible
possible

possible
sorry
scared
excused

counted
proud
two people
to
```

Sample Output

```
2
8
4
7
5
8
```