Dam you, autocorrect!

You work for a publishing company and they have finally decided to enter the twenty first century. They have piles and piles of old books just laying around, but they need to get them into digital form. They acquire some hefty scanners and some OCR software and get to work. However, after a few days of scanning old books and documents, someone notices that the OCR software is doing a poor job of analyzing the text. It seems to be randomly changing letters in words to other letters, and will sometimes even add or remove letters.

You are tasked with writing some software that will determine just how badly the OCR software performed on a given page of a text using an edit distance. An edit is defined as an addition, subtraction, or substitution of a letter in a word. Any given word's edit distance is the number of these edits required to make the word match another word. A word's error number is its minimum edit distance when compared against an entire dictionary.

For example, if the word "bein" appears, its error number would be 1, since you can change the 'i' to 'e' to make "been" (or the 'b' to 'v' to make "vein"), or add a 'g' between the 'e' and 'i' to make "begin" (or to the end for "being"), or remove the 'e' to make "bin." You only care about the minimal edit distance for a given word and not the actual edits that were made or which word you matched in the dictionary.

Input: You will be given both the dictionary and the page of text on standard in. You are guaranteed that all letters in both the dictionary and the page of text will be lower case and will all be within the normal ASCII letter range ('a' to 'z'). The format is as follows:
<number of words in the dictionary>
dict_word1
dict_word2
…
last_dict_word
<number of words in the page of text>
text_word1
text_word2
…
last_text_word

Output: For each word in the page of text, output its error number on its own line. There will be <number of words in the page of text> lines in your output. The format is as follows:
min_edit_distance_word1
min_edit_distance_word2
…
min_edit_distance_last_word

See the next page for an example.

Example

Input:

```
6
character
optical
recognition
software
sucks
this
6
thiss
opptciall
chraactir
recognizition
softtware
sycks
```

Output:

```
1
4
3
2
1
1
```

BONUS

There is a bonus problem available as well. The input and output is exactly the same as described above, but the test case for the problem is much more brutal. Expect your initial solution to run into the time limit. If you think your solution is optimized, or at least fairly efficient, try submitting your solution to the bonus problem for an extra point!