Problem Statement for SentenceDecomposition

Problem Statement

Little Bonnie and her friends were dismayed to learn that their parents were reading all of their private communications. They decided to invent a new language that would allow them to talk freely. What they finally came up with was a language where sentences are built using a special method.

All the valid words that can be used in the new language are given in the String[] **validWords**. A sentence is a concatenation (with no spaces) of a sequence of valid words. Each valid word can appear 0 or more times in the sentence. What makes the language special is that each word can be transformed by rearranging its letters before being used. The cost to transform a word is defined as the number of character positions where the original word and the transformed word differ. For example, "abc" can be transformed to "abc" with a cost of 0, to "acb", "cba" or "bac" with a cost of 2, and to "bca" or "cab" with a cost of 3.

Although several different sequences of valid words can produce the same sentence in this language, only the sequence with the least total transformation cost carries the meaning of the sentence. The advantage of the new language is that the parents can no longer understand what the kids are saying. The disadvantage is that the kids themselves also do not understand. They need your help.

Given a String **sentence**, return the total cost of transformation of the sequence of valid words which carries the meaning of the sentence, or -1 if no such sequence exists.

Definition

Class:	SentenceDecomposition
Method:	decompose
Parameters:	String, String[]
Returns:	int
Method signature:	int decompose(String sentence, String[] validWords)

(be sure your method is public)

Notes

- If a word is used multiple times in a sentence, each occurrence can be transformed differently.

Constraints

- **sentence** will contain between 1 and 50 lowercase letters ('a'-'z'), inclusive.
- **validWords** will contain between 1 and 50 elements, inclusive.
- Each element of **validWords** will contain between 1 and 50 lowercase letters ('a'-'z'), inclusive.

Examples

0)
"neotowheret"
{"one", "two", "three", "there"}
Returns: 8
The following transformations can be made:
- "one" -> "neo" with a cost of 3
- "two" -> "tow" with a cost of 2
- "three" -> "heret" with a cost of 3
- "there" -> "heret" with a cost of 5

So the sequence {"one", "two", "three"} is the one carrying the meaning of "neotowheret". Its total transformation cost is 3 + 2 + 3 = 8.

1)
"abba"
{"ab", "ac", "ad"}
Returns: 2
The word "ab" is used twice, and each time, it is transformed differently.

2)
"thisismeaningless"
{"this", "is", "meaningful"}
Returns: -1

3)
"ommwreehisymkiml"
{"we", "were", "here", "my", "is", "mom", "here", "si", "milk", "where", "si"}
Returns: 10

4)
"ogodtsneeencs"
{"go", "good", "do", "sentences", "tense", "scen"}
Returns: 8

5)
"sepawaterords"
{"separate","words"}
Returns: -1
You are only allowed to rearrange letters within words, and not in the entire sentence.

This problem was used for:
Single Round Match 411 Round 1 - Division I, Level One
Single Round Match 411 Round 1 - Division II, Level Two

topcoder is also on