127. Word Ladder

Hard 5611 1467 Add to List Share

A **transformation sequence** from word `beginWord` to word `endWord` using a dictionary `wordList` is a sequence of words `beginWord -> s1 -> s2 -> ... -> sk` such that:

- Every adjacent pair of words differs by a single letter.
- Every `si` for `1 <= i <= k` is in `wordList`. Note that `beginWord` does not need to be in `wordList`.
- `sk == endWord`

Given two words, `beginWord` and `endWord`, and a dictionary `wordList`, return *the **number of words** in the **shortest transformation sequence** from `beginWord` to `endWord`, or `0` if no such sequence exists.*

Example 1:

```
Input: beginWord = "hit", endWord = "cog", wordList =
["hot","dot","dog","lot","log","cog"]
Output: 5
Explanation: One shortest transformation sequence is "hit"
-> "hot" -> "dot" -> "dog" -> cog", which is 5 words long.
```

Example 2:

```
Input: beginWord = "hit", endWord = "cog", wordList =
["hot","dot","dog","lot","log"]
Output: 0
Explanation: The endWord "cog" is not in wordList,
therefore there is no valid transformation sequence.
```

Constraints:

- `1 <= beginWord.length <= 10`
- `endWord.length == beginWord.length`
- `1 <= wordList.length <= 5000`
- `wordList[i].length == beginWord.length`
- `beginWord`, `endWord`, and `wordList[i]` consist of lowercase English letters.
- `beginWord != endWord`
- All the words in `wordList` are **unique**.

Accepted 613,812 Submissions 1,857,051

Seen this question in a real interview before? Yes No

Companies

Related Topics

Similar Questions

```java
class Solution {
    public int ladderLength(String beginWord, String endWord, List<String> wordList) {

    }
}
```