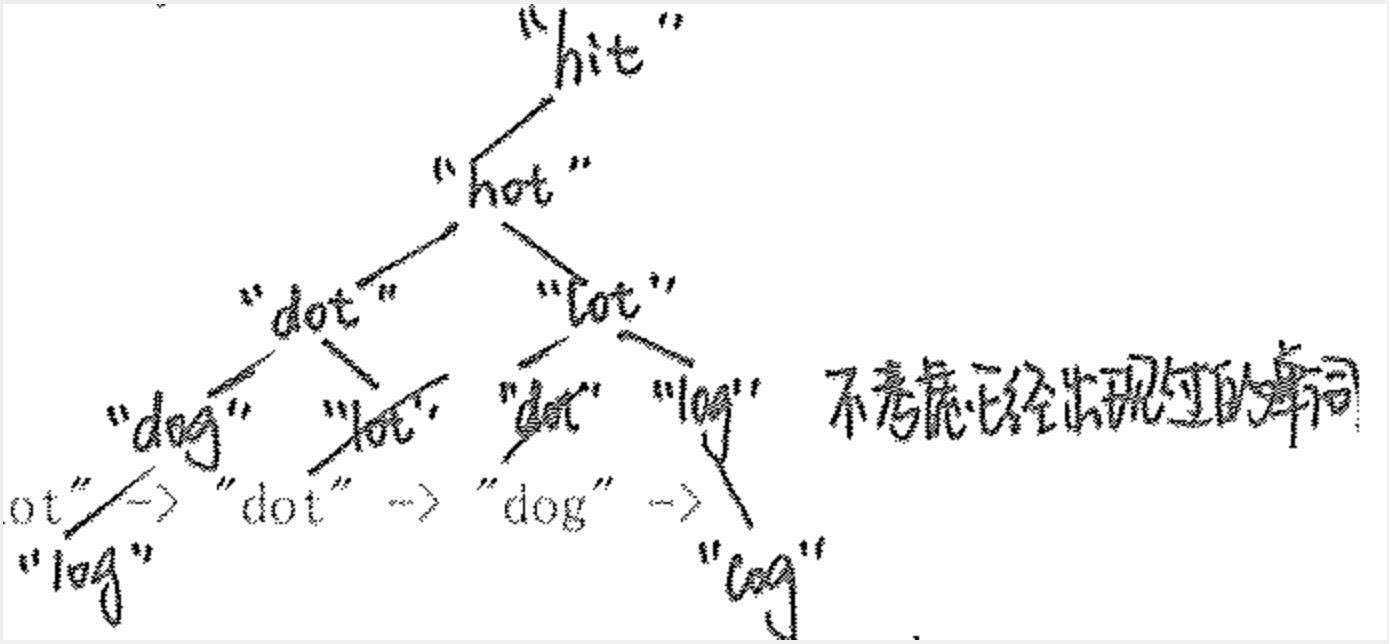
127. Word Ladder

Question | My Sub

Total Accepted: **61534** Total Submissions: **315545** Difficulty: **Medium**

> Given two words (*beginWord* and *endWord*), and a dictionary's word list, find the length of shortest transformation sequence from *beginWord* to *endWord*, such that:
>
> 1. Only one letter can be changed at a time
> 2. Each intermediate word must exist in the word list
>
> For example,
>
> Given:
> *beginWord* = `"hit"`
> *endWord* = `"cog"`
> *wordList* = `["hot","dot","dog","lot","log"]`
>
> As one shortest transformation is `"hit" -> "hot" -> "dot" -> "dog" -> "cog"`,
> return its length `5`.

思路： 广度优先搜索 Bredth First Search.



首先建立一个字母表 = [a, b, c, d, …, z, y, z]

首先用一个队列存储访问过的元素，从上图可以看出，广度优先搜索从hit这个单词（节点）开始。
queue = [hit]

开始运算，如果queue不为空的话一直进行下去：

把queue中的第一个元素取出，也就是hit

从第一个字母开始loop through the word string，用字母表中的字母替.
把替换后的单词和endWord对比，如果相同则找到了答案。如不相同进行下一步。
把替换后的单词和WordList中的单词挨个对比，如果找到相同的话就保存起来，并且把wordList中的那个单词删除掉，表示这个单词已经访问过并且已经被保存了。 例如， hit, 先替换换第一个字母 ait, bit, cit, …., xit, yit, zit, 这些单词中既没有endWord也不在wrodlist中，略过去并替换下一个。hat, hbt, …, hot,..., hyt, hzt. 其中hot在wordlist里面，这时候后我们把hot保存起来。queue = [hot], 然后把wordlist中的hot 删除掉 wordlist = [dot, dog, lot, log]. 继续替换最后一个字母， hia, hib,...hiz. 没有满足条件的单词，略过。

这样就访问了两个节点了。接续queue中的元素继续进行搜索，记住知道queue为空为止。拿出queue中的第一个元素 hot, 继续上述的步骤，对每一个字母进行替换， aot, bot, ...dot, …, lot,... 发现 dot, 与 lot 在wordlist中，加入到queue 中。queue = [dot, lot]。同时在wordlist中删除相应的单词， wordlist = [dog, log]

queue不为空， 继续操作。
对于dot进行替换操作， aot, bot, ...zot, 略过。dat, dbt, …, dzt, 略过。doa, dob, …, dog, ...找到dog. 放进queue中。 queue = [lot, dog], 同时在wordlist中删除dog, wordlist = [log]

queue不为空，对lot进行操做，重复上述步骤，找到log， 放进queue中， 删除wordlist中的log, wordlist = [], queue = [dog, log]

queue不为空，继续找！ dog为queue中第一个元素， 替换！ aog, bog, cog!!! 找到了 cog! 停止所有循环即可，queue里面剩下的那个就也不用考虑了，因为我们只需要找到“length of shortest transformation sequence from beginWord to endWord” 即可。

```python
class Solution(object):
    def ladderLength(self, beginWord, endWord, wordList):

        if not wordList or len(wordList) == 0:
            return 0

        queue = []
        queue.append(beginWord)
        wordList.remove(beginWord)
        result = 1     # an index to keeptrack on the  number of steps to find the result
        # create an alphabet for replacing letter in the word
        alphabet = [chr(x) for x in range(ord('a'), ord('z')+1)]

        while queue:  # keep doing these steps untill the queue is empty
            n = len(queue)
            for i in range(n):
                pop = queue.pop(0)

                for letter in alphabet:
                    for j in range(len(pop)):
                        if letter != pop[j]:          # replacing one letter in the word at every time
                            tmp = pop[:j] + letter + pop[j+1:]

                            if tmp == endWord:        # compare the newWrod with the endWord
                                return result + 1

                            if tmp in wordList:       # check if the newWord is in the wordlist
                                queue.append(tmp)
                                wordList.remove(tmp)
            result += 1   # increate the index by one

        return 0
```