127. Word Ladder

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:

Only one letter can be changed at a time. Each transformed word must exist in the word list. Note that beginWord is not a transformed word.

Example 1:

```
Input:
beginWord = "hit",
endWord = "cog",
wordList = ["hot","dot","dog","lot","log","cog"]

Output: 5

Explanation: As one shortest transformation is
"hit" -> "hot" -> "dot" -> "dog" -> "cog",
return its length 5.
```

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

        Set<String> dict = new HashSet<>();
        for (String word : wordList) {
            dict.add(word);
        }

        if (beginWord.equals(endWord))
            return 1;

        HashSet<String> hash = new HashSet<String>();
        Queue<String> queue = new LinkedList<String>();
        queue.offer(beginWord);
        hash.add(beginWord);

        int length = 1;
        while(!queue.isEmpty()) {
            length++;
            int size = queue.size();
```

```
        for(int i = 0; i < size; i++) {
            String word = queue.poll();

            for(String nextWord : getNextWords(word, dict)) {
                if (hash.contains(nextWord))
                    continue;

                if (nextWord.equals(endWord))
                    return length;

                hash.add(nextWord);
                queue.offer(nextWord);
            }
        }
    }

    return 0;
}


private ArrayList<String> getNextWords(String word,
        Set<String> dict) {

    ArrayList<String> nextWords = new ArrayList<String>();

    for(char c = 'a'; c <= 'z'; c++) {
        for (int i = 0; i < word.length(); i++) {
            if (c == word.charAt(i))
                continue;

            String nextWord = replace(word, i, c);
            if (dict.contains(nextWord)) {
                nextWords.add(nextWord);
            }
        }
    }

    return nextWords;
}

private String replace(String s, int index, char c) {
    char[] chars = s.toCharArray();
    chars[index] = c;
    return new String(chars);
```

```
    }

}
```