```
class Solution {
    /*
    public int ladderLength(String begin, String end, List<String> wordList) {
        Queue<String> q = new LinkedList<>();
        q.offer(begin);
        Set<String> visited = new HashSet<>();
        Set<String> all = new HashSet<>();
        all.addAll(wordList);
        visited.add(begin);
        int level = 1;
        while (!q.isEmpty()) {
            int size = q.size();
            while (size > 0) {
                size--;
                String s = q.poll();
                char[] cs = s.toCharArray();
                for (int i = 0; i < cs.length; i++) {
                    char c = cs[i];
                    for (int j = 'a'; j <= 'z'; j++) {
                        if ((char)j == c) continue;
                        cs[i] = (char)j;
                        String str = new String(cs);
                        if (all.contains(str)) {
                            if (str.equals(end)) {
                               return level+1;
                            }
                            if (!visited.contains(str)) {
                                q.offer(str);
                                visited.add(s);
                            }
                        }
                    }
                    cs[i] = c;
                }
            }
            level++;
        }
        return 0;
    }

    */
```

```
    public int ladderLength(String begin, String end, List<String> wordList) {
        Set<String> reached = new HashSet<String>();
        Set<String> wordDict = new HashSet<String>();
        reached.add(begin);
        wordDict.addAll(wordList);
        if (!wordDict.contains(end)) {
            return 0;
        }
        wordDict.add(end);
        int distance = 1;
        while (!reached.contains(end)) {
            Set<String> toAdd = new HashSet<String>();
            for (String each : reached) {
                for (int i = 0; i < each.length(); i++) {
                    char[] chars = each.toCharArray();
                    for (char ch = 'a'; ch <= 'z'; ch++) {
                        chars[i] = ch;
                        String word = new String(chars);
                        if (wordDict.contains(word)) {
                            toAdd.add(word);
                            wordDict.remove(word);
                        }
                    }
                }
            }
            distance++;
            if (toAdd.size() == 0) return 0;
            reached = toAdd;
        }
        return distance;
    }
}
```