单词接龙 · Word Ladder

中文

Yelp (/problem/?tags=yelp) Snapchat (/problem/?tags=snapchat) 脸书 (/problem/?tags=facebook) 亚马逊 (/problem/?tags=amazon) 领英 (/problem/?tags=linkedin) 谷歌 (/problem/?tags=google) Breadth-first Search (/problem/?tags=breadth-first-search)

描述

给出两个单词（*start*和*end*）和一个字典，找出从*start*到*end*的最短转换序列，输出最短序列的长度。

变换规则如下：

1. 每次只能改变一个字母。
2. 变换过程中的中间单词必须在字典中出现。(起始单词和结束单词不需要出现在字典中)

- 如果不存在这样的转换序列，返回 0。 - 所有单词具有相同的长度。 - 所有单词只由小写字母组成。 - 字典中不存在重复的单词。 - 你可以假设 beginWord 和 endWord 是非空的，且二者不相同。

样例

样例 1:

```
输入：start = "a"，end = "c"，dict =["a","b","c"]
输出：2
解释：
"a"->"c"
```

样例 2:

```
输入：start ="hit"，end = "cog"，dict =["hot","dot","dog","lot","log"]
输出：5
解释：
"hit"->"hot"->"dot"->"dog"->"cog"
```

在线评测地址: https://www.lintcode.com/problem/word-ladder/ (https://www.lintcode.com/problem/word-ladder/)

收起题目描述 ^

语言类型 ALL (50) python (18) cpp (18) java (13) javascript (1) 上传题解

令狐冲
更新于 11/30/2020, 5:50:07 PM

使用分层遍历的BFS版本。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: start: a string
    @param: end: a string
    @param: dict: a set of string
    @return: An integer
    """
    def ladderLength(self, start, end, dict):
        dict.add(end)
        queue = collections.deque([start])
        visited = set([start])

        distance = 0
        while queue:
            distance += 1
            for i in range(len(queue)):
                word = queue.popleft()
                if word == end:
                    return distance

                for next_word in self.get_next_words(word):
                    if next_word not in dict or next_word in visited:
                        continue
                    queue.append(next_word)
                    visited.add(next_word)

        return 0

    # O(26 * L^2)
    # L is the length of word
    def get_next_words(self, word):
        words = []
        for i in range(len(word)):
            left, right = word[:i], word[i + 1:]
            for char in 'abcdefghijklmnopqrstuvwxyz':
                if word[i] == char:
                    continue
                words.append(left + char + right)
        return words
```

获赞 13　　15 条评论

令狐冲
更新于 11/27/2020, 5:40:50 PM

不使用分层遍历的版本。使用 distance 这个 hash 来存储距离来实现记录每个节点的距离。

python　java

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: start: a string
    @param: end: a string
    @param: dict: a set of string
    @return: An integer
    """
    def ladderLength(self, start, end, dict):
        dict.add(end)
        queue = collections.deque([start])
        distance = {start: 1}

        while queue:
            word = queue.popleft()
            if word == end:

                return distance[word]

            for next_word in self.get_next_words(word, dict):
                if next_word in distance:
                    continue
                queue.append(next_word)
                distance[next_word] = distance[word] + 1

        return 0

    def get_next_words(self, word, dict):
        words = []
        for i in range(len(word)):
            left, right = word[:i], word[i + 1:]
            for char in 'abcdefghijklmnopqrstuvwxyz':
                if word[i] == char:
                    continue
                next_word = left + char + right
                if next_word in dict:
                    words.append(next_word)

        return words
```

获赞 10　　1 条评论

令狐冲
更新于 9/28/2020, 4:11:58 PM

Given two words (start and end), and a dictionary, find the length of shortest transformation sequence from start to end, such that:

Only one letter can be changed at a time Each intermediate word must exist in the dictionary For example,

Given: start = "hit" end = "cog" dict = "hot","dot","dog","lot","log" () As one shortest transformation is "hit" -> "hot" -> "dot" -> "dog" -> "cog", return its length 5.

Note: Return 0 if there is no such transformation sequence. All words have the same length. All words contain only lowercase alphabetic characters.

考点：

- bfs

题解： 从头部开始枚举替换字母，每次替换一个字母，即为一步，每次得到新单词如果在dict中就继续搜索。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
// version: LintCode ( Set<String> )
public class Solution {
    public int ladderLength(String start, String end, Set<String> dict) {
        if (dict == null) {
            return 0;
        }

        if (start.equals(end)) {
            return 1;
        }

        dict.add(start);
        dict.add(end);

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

        int length = 1;
        while(!queue.isEmpty()) {                       //开始bfs
            length++;
            int size = queue.size();            //当前步数的队列大小
            for (int i = 0; i < size; i++) {
                String word = queue.poll();
                for (String nextWord: getNextWords(word, dict)) {      //得到新单词
                    if (hash.contains(nextWord)) {
                        continue;
                    }
                    if (nextWord.equals(end)) {
                        return length;
                    }

                    hash.add(nextWord);
                    queue.offer(nextWord);                      //存入队列继续搜索
                }
            }
        }
        return 0;
    }

    // replace character of a string at given index to a given character
    // return a new string
    private String replace(String s, int index, char c) {
        char[] chars = s.toCharArray();
        chars[index] = c;
        return new String(chars);
    }

    // get connections with given word.
    // for example, given word = 'hot', dict = {'hot', 'hit', 'hog'}
    // it will return ['hit', 'hog']
    private ArrayList<String> getNextWords(String word, Set<String> dict) {
        ArrayList<String> nextWords = new ArrayList<String>();
        for (char c = 'a'; c <= 'z'; c++) {                                     //枚举当前替换字母
            for (int i = 0; i < word.length(); i++) {          //枚举替换位置
```

```
                if (c == word.charAt(i)) {
                    continue;
                }
                String nextWord = replace(word, i, c);
                if (dict.contains(nextWord)) {         //如果dict中包含新单词，存入nextWords
                    nextWords.add(nextWord);
                }
            }
        }
        return nextWords;                                                    //构造当前单词的全部下一步方案

    }
}

// version: LeetCode
public class Solution {
    public int ladderLength(String start, String end, List<String> wordList) {
        Set<String> dict = new HashSet<>();
        for (String word : wordList) {    //将wordList中的单词加入dict
            dict.add(word);
        }

        if (start.equals(end)) {
            return 1;
        }

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

        int length = 1;
        while (!queue.isEmpty()) {                    //开始bfs
            length++;
            int size = queue.size();
            for (int i = 0; i < size; i++) {          //枚举当前步数队列的情况
                String word = queue.poll();
                for (String nextWord: getNextWords(word, dict)) {
                    if (hash.contains(nextWord)) {
                        continue;
                    }
                    if (nextWord.equals(end)) {
                        return length;
                    }

                    hash.add(nextWord);                                  //存入新单词
                    queue.offer(nextWord);
                }
            }
        }

        return 0;
    }

    // replace character of a string at given index to a given character
    // return a new string
    private String replace(String s, int index, char c) {
        char[] chars = s.toCharArray();
        chars[index] = c;
        return new String(chars);
    }

    // get connections with given word.
    // for example, given word = 'hot', dict = {'hot', 'hit', 'hog'}
    // it will return ['hit', 'hog']
    private ArrayList<String> getNextWords(String word, Set<String> dict) {
        ArrayList<String> nextWords = new ArrayList<String>();
        for (char c = 'a'; c <= 'z'; c++) {                                  //枚举替换字母
```

```
            for (int i = 0; i < word.length(); i++) {          //枚举替换位置
                if (c == word.charAt(i)) {
                    continue;
                }
                String nextWord = replace(word, i, c);
                if (dict.contains(nextWord)) {                        //如果dict中包含新单词，存入nextWords
                    nextWords.add(nextWord);
                }
            }
        }
        return nextWords;                                                                    //构造当前单词的全部下一步方案
    }
}
```

获赞 7　　11 条评论

九章-小原
更新于 12/23/2020, 1:18:37 AM

题目理解：

给定起始单词start和结尾单词end，当前单词和字典中某个单词（或是end单词）有且只有一个字母不相同时，可以转换为该单词，问最少变换多少次可以得到end单词。

由此可以联想到，我们可以将每个单词视为一个节点，可以互相转换的两个单词之间即存在边，那么题目就转换为，从起始点start走到终点end的最短路径（每条边长度相等，均为1），可以想到利用bfs求解。题目存在一个需要转换题意的部分，在判断两个点之间是否存在边时，需要判断两个字符串是否只有一个字母不相同。

解题思路：

- 新建队列，将当前队列中所有节点遍历并取出，将这些节点能走到的所有节点均推入队列，当遍历到end节点时退出bfs，否则将路径长度+1，然后继续遍历直到队列为空
- 当判断当前节点的可到达节点时，可以循环本节点单词的所有字符，用'a' - 'z'中与原来不相等的字符替换后判断dict内是否存在该字符，若存在，则可到达该节点
    - 伪代码

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
for i 0:len_word //遍历整个单词
    oldchar = word[i]
    for j 'a':'z' //遍历26个字母
        if (oldchar == j)
            continue
        word[i] = j //得到新单词
        if (dict.find(word) or word == end) //若该单词在dict中或等于end
            queue.push(word)
            dict.erase(word)
     word[i] = oldchar //回溯，恢复当前单词
```

- 题意所给dict是一个set类型，每当走过dict中的一个节点，即可以从dict中删去该节点，因为重复经过一定比第一次经过的路径要长，可以不用去遍历

复杂度分析：

- 时间复杂度：$O(snm^2)$

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
n为dict中单词个数，s代表字符集大小(这题中是26)，m为单词长度。因为bfs所有节点最多遍历一次，每次遍历到之后，需要扫遍单词的每个字符，每个字符均可以变化为其他25个不同字母
```

- 空间复杂度：$O(nm)$

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
n为dict中单词个数，m为单词长度。用于bfs的队列最大需存下所有节点。
```

代码：

c++ | java | python

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    /**
      * @param start, a string
      * @param end, a string
      * @param dict, a set of string
      * @return an integer
      */
    int ladderLength(string start, string end, unordered_set<string> &dict) {
        int n = start.size();
        queue<string> queue;
        int length = 2;

        if (start == end) {  //若起始单词等于终点单词，直接返回1
            return 1;
        }

        queue.push(start);       //将起点推进队列
        dict.erase(start);   //删除dict中的起点单词（若存在的话）
        while (!queue.empty()) {
            int size = queue.size();
            for (int i = 0; i < size; i++) {  //遍历当前队列中所有节点
                string word = queue.front();
                queue.pop();
                for (int i = 0; i < n; i++) {
                    char oldChar = word[i];
                    for (char c = 'a'; c <= 'z'; c++) {
                        if (c == oldChar) continue;
                        word[i] = c;           //得到本节点能变换到达的单词
                        if (word == end) {     //判断是否到达终点，若到达返回路径长度
                            return length;
                        }
                        if (dict.find(word) != dict.end()) {  //判断可到的下一个单词是否在dict中
                            queue.push(word);
                            dict.erase(word);
                        }
                    }
                    word[i] = oldChar;
                }
            } // for size
            length++;   //当遍历完当前队列所有节点，那么所有情况下的点都已经走到下一步，路径长度+1
        }
        return 0;
    }
};
```

获赞 5　　3 条评论

令狐冲
更新于 12/18/2020, 1:56:47 AM

使用双向宽度优先搜索算法

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: start: a string
    @param: end: a string
    @param: dict: a set of string
    @return: An integer
    """
    def ladderLength(self, start, end, wordSet):
        if start == end:
            return 1

        wordSet.add(start)
        wordSet.add(end)
        graph = self.construct_graph(wordSet)

        forward_queue = collections.deque([start])
        forward_set = set([start])
        backward_queue = collections.deque([end])
        backward_set = set([end])

        distance = 1
        while forward_queue and backward_queue:
            distance += 1
            if self.extend_queue(graph, forward_queue, forward_set, backward_set):
                return distance
            distance += 1
            if self.extend_queue(graph, backward_queue, backward_set, forward_set):
                return distance

        return -1

    def extend_queue(self, graph, queue, visited, opposite_visited):
        for _ in range(len(queue)):
            word = queue.popleft()
            for next_word in graph[word]:
                if next_word in visited:
                    continue
                if next_word in opposite_visited:
                    return True
                queue.append(next_word)
                visited.add(next_word)
        return False

    def construct_graph(self, wordSet):
        graph = {}
        for word in wordSet:
            graph[word] = self.get_next_words(word, wordSet)
        return graph

    def get_next_words(self, word, wordSet):
        next_word_set = set()
        for i in range(len(word)):
            prefix = word[:i]
            suffix = word[i + 1:]
            chars = list('abcdefghijklmnopqrstuvwxyz')
            chars.remove(word[i])
            for char in chars:
                next_word = prefix + char + suffix
```

```
                if next_word in wordSet:
                    next_word_set.add(next_word)
        return next_word_set
```

获赞 1　　添加评论

九章-加贺
更新于 8/26/2020, 11:18:30 AM

使用双向宽度优先搜索算法

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /*
     * @param start: a string
     * @param end: a string
     * @param dict: a set of string
     * @return: An integer
     */
    public int ladderLength(String start,
                            String end,
                            Set<String> wordSet) {
        if (start.equals(end)) {
            return 1;
        }

        HashMap<String, Set<String>> graph;
        Queue<String> forwardQueue = new LinkedList<String>();
        Queue<String> backwardQueue = new LinkedList<String>();
        Set<String> forwardSet = new HashSet<String>();
        Set<String> backwardSet = new HashSet<String>();

        wordSet.add(start);
        wordSet.add(end);
        graph = constructGraph(wordSet);
        forwardQueue.offer(start);
        backwardQueue.offer(end);
        forwardSet.add(start);
        backwardSet.add(end);

        int distance = 1;
        while (!forwardQueue.isEmpty() && !backwardQueue.isEmpty()) {
            distance++;
            if (extendQueue(graph, forwardQueue, forwardSet, backwardSet)) {
                return distance;
            }
            distance++;
            if (extendQueue(graph, backwardQueue, backwardSet, forwardSet)) {
                return distance;
            }
        }

        return -1;
    }

    boolean extendQueue(HashMap<String, Set<String>> graph,
```

```
                        Queue<String> queue,
                        Set<String> visited,
                        Set<String> oppositeVisited) {
        int queueLength = queue.size();
        for (int i = 0; i < queueLength; i++) {
            String word = queue.poll();
            Set<String> nextWordSet = graph.get(word);
            for (String nextWord : nextWordSet) {
                if (visited.contains(nextWord)) {
                    continue;
                }
                if (oppositeVisited.contains(nextWord)) {
                    return true;
                }

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

        return false;
    }

    HashMap<String, Set<String>> constructGraph(Set<String> wordSet) {
        HashMap<String, Set<String>> graph = new HashMap<String, Set<String>>();
        for (String word : wordSet) {
            graph.put(word, getNextWords(word, wordSet));
        }

        return graph;
    }

    Set<String> getNextWords(String word, Set<String> wordSet) {
        Set<String> nextWordSet = new HashSet<String>();
        int wordLength = word.length();
        for (int i = 0; i < wordLength; i++) {
            String prefix = word.substring(0, i);
            String suffix = word.substring(i + 1);
            char[] chars = ("abcdefghijklmnopqrstuvwxyz").toCharArray();
            for (int j = 0; j < 26; j++) {
                if (word.charAt(i) == chars[j]) {
                    continue;
                }
                String nextWord = prefix + chars[j] + suffix;
                if (wordSet.contains(nextWord)) {
                    nextWordSet.add(nextWord);
                }
            }
        }

        return nextWordSet;
    }

}
```

获赞 1　　添加评论

令狐冲
更新于 6/9/2020, 7:04:28 AM

Given two words (start and end), and a dictionary, find the length of shortest transformation sequence from start to end, such that:

Only one letter can be changed at a time Each intermediate word must exist in the dictionary For example,

Given: start = "hit" end = "cog" dict = "hot","dot","dog","lot","log" () As one shortest transformation is "hit" -> "hot" -> "dot" -> "dog" -> "cog", return its length 5.

Note: Return 0 if there is no such transformation sequence. All words have the same length. All words contain only lowercase alphabetic characters.

考点：

- bfs

题解： 从头部开始搜索，枚举队列头部单词替换字母的位置，然后枚举替换字母，新的单词在wordlist中存在时，先压入队列，以后进行下一步数的搜索。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
// version LintCode
class Solution {
public:
    /**
      * @param start, a string
      * @param end, a string
      * @param dict, a set of string
      * @return an integer
      */
    int ladderLength(string start, string end, unordered_set<string> &dict) {
        if (start == end) {
            return 1;
        }
        int n = start.size();
        if (n < 1 || n != end.size()) {
            return 0;
        }

        queue<string> Q;
        Q.push(start);
        dict.erase(start);
        int length = 2;

        while (!Q.empty()) {                    //开始bfs
            int size = Q.size();
            for (int i = 0; i < size; i++) {           //获取当前队列大小
                string word = Q.front(); Q.pop();
                for (int i = 0; i < n; i++) {          //枚举替换位置
                    char oldChar = word[i];
                    for (char c = 'a'; c <= 'z'; c++) {                //枚举替换字母
                        if (c == oldChar) continue;
                        word[i] = c;
                        if (word == end) {
                            return length;
                        }
                        if (dict.find(word) != dict.end()) {   //如果新单词存在于dict中
                            Q.push(word);                                           //存入队列继续搜索
                            dict.erase(word);
                        }
                    }
                    word[i] = oldChar;
                }
            } // for size
            length++;                                          //外层每循环一次就步数+1
        }
        return 0;
    }
};
```

```
// version LeetCode
class Solution {
public:
    int ladderLength(string start, string end, vector<string>& wordList) {
        if (start == end) {
            return 1;
        }

        int n = start.size();
        if (n < 1 || n != end.size()) {
            return 0;
        }

        unordered_set<string> dict;
        for (int i = 0; i < wordList.size(); i++) {
            dict.insert(wordList[i]);
        }

        queue<string> Q;
        Q.push(start);
        dict.erase(start);
        int length = 2;

        while (!Q.empty()) {           //bfs
            int size = Q.size();
            for (int i = 0; i < size; i++) {
                string word = Q.front(); Q.pop();
                for (int i = 0; i < n; i++) {   //枚举替换的每个位置
                    char oldChar = word[i];
                    for (char c = 'a'; c <= 'z'; c++) {     //枚举替换的字母
                        if (c == oldChar) continue;
                        word[i] = c;
                        if (dict.find(word) != dict.end()) {   //如果当前字符串在dict
                            if (word == end) {
                                return length;
                            }
                            Q.push(word);                               //存入队列中，以后进入下一步数单词的搜索
                            dict.erase(word);                   //从dict中除去当前单词
                        }
                    }
                    word[i] = oldChar;
                }
            } // for size
            length++;                                                                   //每循环一次，步数+1
        }
        return 0;
    }
};
```

获赞 0　　添加评论

令狐冲
更新于 6/9/2020, 7:04:28 AM

LintCode 的版本

考点：

- bfs

题解： 从头开始进行bfs，每次改变一个字母，构造新的单词，然后存入队列，继续搜索，直至得到最终单词。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
import collections
class Solution:
    # @param start, a string
    # @param end, a string
    # @param dict, a set of string
    # @return an integer
    def ladderLength(self, start, end, wordList):
        wordSet = set([])
        for word in wordList:  #将wordList中的word加入WordSet
            wordSet.add(word)

        wordLen = len(start)
        queue = collections.deque([(start, 1)])
        while queue:                                                    #bfs
            currWord, currLen = queue.popleft()
            if currWord == end:
                return currLen
            for i in xrange(wordLen):
                part1 = currWord[:i]    #取出当前单词左部分
                part2 = currWord[i+1:]  #取出当前单词右部分
                for j in 'abcdefghijklmnopqrstuvwxyz':   #枚举替换字母
                    if currWord[i] != j:
                        nextWord = part1 + j + part2     #构成新单词
                        if nextWord in wordSet:                         #如果新单词在Set中
                            queue.append((nextWord, currLen + 1))       #加入队列，继续搜索
                            wordSet.remove(nextWord)
        return 0
```

获赞 0　　3 条评论

令狐冲
更新于 6/9/2020, 7:04:21 AM

LintCode 版本。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
import collections

class Solution:
    # @param start, a string
    # @param end, a string
    # @param dict, a set of string
    # @return an integer
    def ladderLength(self, start, end, dict):
        # write your code here
        dict.add(end)
        wordLen = len(start)
        queue = collections.deque([(start, 1)])
        while queue:
            curr = queue.popleft()
            currWord = curr[0]; currLen = curr[1]
            if currWord == end: return currLen
            for i in range(wordLen):
                part1 = currWord[:i]; part2 = currWord[i+1:]
                for j in 'abcdefghijklmnopqrstuvwxyz':
                    if currWord[i] != j:
                        nextWord = part1 + j + part2
                        if nextWord in dict:
                            queue.append((nextWord, currLen + 1))
                            dict.remove(nextWord)
        return 0
```

获赞 0 添加评论

令狐冲
更新于 6/9/2020, 7:04:09 AM

省去一重循环的 BFS 写法

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: start: a string
    @param: end: a string
    @param: dict: a set of string
    @return: An integer
    """
    def ladderLength(self, start, end, dict):
        dict.add(end)
        queue = collections.deque([start])
        visited = set([start])

        distance = {start: 1}
        while queue:
            word = queue.popleft()
            if word == end:
                return distance[word]

            for next_word in self.get_next_words(word):
                if next_word not in dict or next_word in visited:
                    continue
                queue.append(next_word)
                visited.add(next_word)
                distance[next_word] = distance[word] + 1

        return 0

    # O(26 * L^2)
    # L is the length of word
    def get_next_words(self, word):
        words = []
        for i in range(len(word)):
            left, right = word[:i], word[i + 1:]
            for char in 'abcdefghijklmnopqrstuvwxyz':
                if word[i] == char:
                    continue
                words.append(left + char + right)
        return words
```

获赞 0 添加评论

九章用户LRVWUP
更新于 6/9/2020, 7:03:45 AM

经典的BFS，为了凑够30个字符，多说几句吧，getNextAndRemove方法简单方便。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
```

```
*/
public class Solution {
    /*
     * @param start: a string
     * @param end: a string
     * @param dict: a set of string
     * @return: An integer
     */
    public int ladderLength(String start, String end, Set<String> dict) {
        // write your code here
        if (start.equals(end)) {
            return 1;
        }
        if (dict.size() == 0) {
            return 0;
        }

        int change = 0;
        dict.add(end);
        Queue<String> queue = new LinkedList<>();
        queue.offer(start);

        while (!queue.isEmpty()) {
            change++;
            int size = queue.size();
            for (int i = 0; i < size; i++) {
                String current = queue.poll();
                if (current.equals(end)) {
                    return change;
                }
                List<String> next = findNextAndRemove(current, dict);
                for (String n : next) {
                    if (n.equals(end)) {
                        return change + 1;
                    }
                    queue.offer(n);
                }
            }
        }
        return change;
    }

    // 找到所有和a只差一个字母的单词 并从dict里面删除
    public List<String> findNextAndRemove(String a, Set<String> dict) {
        List<String> next = new ArrayList<>();

        for (String d : dict) {
            int diff = 0;
            for (int i = 0; i < a.length(); i++) {
                if (a.charAt(i) == d.charAt(i)) {
                    continue;
                }
                diff++;
            }
            if (diff == 1) {
                next.add(d);
            }
        }
        dict.removeAll(next); // 每次找到已经加入queue的单词，从dict里面remove掉
        return next;
    }
}
```

九章用户G1LOUJ
更新于 6/9/2020, 7:03:46 AM

硅谷求职算法集训营版本，使用双向BFS优化搜索过程，降低时间复杂度。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /*
     * @param start: a string
     * @param end: a string
     * @param dict: a set of string
     * @return: An integer
     */
    public int ladderLength(String start, String end, Set<String> dict) {
        if (start.equals(end)) {
            return 1;
        }

        Queue<String> queA = new LinkedList<>();
        Set<String> setA = new HashSet<>();
        queA.offer(start);
        setA.add(start);

        Queue<String> queB = new LinkedList<>();
        Set<String> setB = new HashSet<>();
        queB.offer(end);
        setB.add(end);

        Queue<String> que = null;
        Set<String> setCur = null, setOp = null;

        int res = 1;
        while (!queA.isEmpty() && !queB.isEmpty()) {
            if (queA.size() <= queB.size()) {
                que = queA;
                setCur = setA;
                setOp = setB;
            } else {
                que = queB;
                setCur = setB;
                setOp = setA;
            }

            res++;
            int n = que.size();
            while (n-- != 0) {
                String cur = que.poll();
                ArrayList<String> neighbors = getNext(cur, dict);

                for (String nei : neighbors) {
                    if (setOp.contains(nei)) {
                        return res;
                    }
                    if (!setCur.contains(nei)) {
                        que.offer(nei);
                        setCur.add(nei);
                    }
```

```
                }
            }
        }

        return 0;
    }

    private ArrayList<String> getNext(String cur, Set<String> dict) {
        ArrayList<String> res = new ArrayList<>();
        int n = cur.length();
        StringBuilder sb = new StringBuilder(cur);

        for (int i = 0; i < n; ++i) {
            char old = sb.charAt(i);
            for (char c = 'a'; c <= 'z'; ++c) {
                if (c != old) {
                    sb.setCharAt(i, c);
                    if (dict.contains(sb.toString())) {
                        res.add(sb.toString());
                    }
                }
            }

            sb.setCharAt(i, old);
        }

        return res;
    }
}
```

获赞 5　　3 条评论

社会我喵哥
更新于 6/9/2020, 7:03:50 AM

1. BFS遍历搜索字典，由于需要返回层数，需要分层
2. 搜索字典的时候需要对每个单词的字母通过26个字母变换，否则如果字典单词很多会超时

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: start: a string
    @param: end: a string
    @param: dict: a set of string
    @return: An integer
    """
    def ladderLength(self, start, end, dict):
        # write your code here
        if len(start) != len(end):
            return -1
        if start == end:
            return 1
        queue = []
        nodes = set()
        count = 1
        dict.add(end)
        queue.append(start)
        nodes.add(start)
        while queue:
            count += 1
            size = len(queue)
            for i in range(size):
                word = queue.pop(0)
                nextWords = self.findNextWord(word, dict)
                for nw in nextWords:
                    if nw in nodes:
                        continue
                    if nw == end:
                        return count
                    queue.append(nw)
                    nodes.add(nw)

    def findNextWord(self, word, dict):
        nextWords = []
        for i in range(len(word)):
            for ch in "abcdefghijklmnopqrstuvwxyz":
                if word[i] == ch:
                    continue
                newWord = word[:i] + ch + word[i+1:]
                if newWord in dict:
                    nextWords.append(newWord)
        return nextWords
```

获赞 2　　添加评论

Leon
更新于 6/9/2020, 7:03:50 AM

提供两种解法，BFS和双向BFS。 时间复杂度大概BFS是双向BFS的平方量级。 1. BFS: 没有做任何优化的原装BFS。 需要注意的点1）对每一位字母完成a-z的常识之后要换回初始字母，再进行下一位；2）如果一个单词已经在dict中出现过，那么可以从dict中删除这个单词，相当于BFS中的visited 2. 双向BFS：有点类似于用两个queue的BFS，从start开始一个queue1，从end 开始一个queue2， 然后两边寻找（我的code里每次处理短的queue，然后search 长的queue）。 为了便于search，使用了set而不是list。实现起来也没有特别复杂。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: start: a string
    @param: end: a string
    @param: dict: a set of string
    @return: An integer
    """


    def ladderLength1(self, start, end, dict):
        # write your code here
        if start == end:
            return 1
        ans = 1
        q = [start]
        abc = 'abcdefghijklmnopqrstuvwxyz'
        while q:
            ans += 1
            size = len(q)
            for _ in range(size):
                temp = list(q.pop(0))
                for i in range(len(temp)):
                    ch = temp[i]
                    for j in range(26):
                        temp[i] = abc[j]
                        new_word = ''.join(temp)
                        if new_word == end:
                            return ans
                        if new_word in dict:
                            q.append(new_word)
                            dict.remove(new_word)
                    temp[i] = ch
        return 0

        def ladderLength2(self,start,end,dict):
        if start == end:
            return 1
        ans = 1
        q1 = {start}
        q2 = {end}
        abc = 'abcdefghijklmnopqrstuvwxyz'
        while q1 and q2:
            ans += 1
            if len(q1) > len(q2):
                q1,q2 = q2,q1
            new_q = set()
            for word in q1:
                temp = list(word)
                for i in range(len(temp)):
                    ch = temp[i]
                    for j in range(26):
                        temp[i] = abc[j]
                        new_word = ''.join(temp)
                        if new_word in q2:
                            return ans
                        if new_word in dict:
                            new_q.add(new_word)
                            dict.remove(new_word)
```

```
                temp[i] = ch
        q1 = new_q
    return -1
```

获赞 2　　添加评论

Jimmy
更新于 6/9/2020, 7:03:50 AM

感覺目前Python Bidirectional BFS交換寫的有點難 所以用隨課教程的模板再寫一次

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
from collections import deque
class Solution:
    """
    @param: start: a string
    @param: end: a string
    @param: dict: a set of string
    @return: An integer
    """
    def ladderLength(self, start, end, dict):
        # write your code here
        # Bidirectional BFS
        if start == end:
            return 1
        q1, q2 = deque([start]), deque([end])
        steps = 1
        while q1 and q2:
            steps += 1
            len_q1 = len(q1)
            for _ in range(len_q1):
                word = q1.popleft()
                n = len(word)
                for i in range(n):
                    prefix, suffix = word[ : i], word[i + 1 : ]
                    for j in range(26):
                        middle = chr(j + ord('a'))
                        new_word = prefix + middle + suffix
                        if new_word in q2:
                            return steps
                        if new_word in dict:
                            q1.append(new_word)
                            dict.remove(new_word)
            steps += 1
            len_q2 = len(q2)
            for _ in range(len_q2):
                word = q2.popleft()
                m = len(word)
                for i in range(m):
                    prefix, suffix = word[ : i], word[i + 1 : ]
                    for j in range(26):
                        middle = chr(j + ord('a'))
                        new_word = prefix + middle + suffix
                        if new_word in q1:
                            return steps
                        if new_word in dict:
                            q2.append(new_word)
                            dict.remove(new_word)
        return -1
```

获赞 2　　添加评论

九章用户GIKHOL
更新于 6/9/2020, 7:03:49 AM

按照九章算法 Unit 4: BFS & Topological sort的随课教程的"双向BFS"模板, 从Java port到Python3. 这里是分层BFS的写法. 终于理解了双向BFS说的同时从首尾BFS的意思了.

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
from collections import deque


class Solution:
    """
    @param: start: a string
    @param: end: a string
    @param: dict: a set of string
    @return: An integer
    """

    def ladderLength(self, start, end, dict):
        # write your code here
        # Bidirectional BFS

        if start == end:
            return 1
        q1, q2 = deque([start]), deque([end])
        s1, s2 = set([start]), set([end])
        step = 1
        while q1 or q2:
            step += 1
            for _ in range(len(q1)):
                word = q1.popleft()
                for next_word in self.get_next_words(word):

                    if next_word not in dict or next_word in s1:  # same as in normal BFS
                        continue
                    if next_word in s2: # start BFS connected end BFS
                        return step
                    s1.add(next_word)
                    q1.append(next_word)
                    # print("q1", q1)

            step += 1
            for _ in range(len(q2)):
                word = q2.popleft()
                for next_word in self.get_next_words(word):
                    if next_word not in dict or next_word in s2:  # same as in normal BFS
                        continue
                    if next_word in s1:  # start BFS connected end BFS
                        return step
                    s2.add(next_word)
                    q2.append(next_word)
                    # print("q2", q2)
        return -1

    def get_next_words(self, word):
        words = []
        for i in range(len(word)):
            left, right = word[:i], word[i+1:]
            for char in 'abcdefghijklmnopqrstuvwxyz':
                if word[i] == char:
                    continue
                words.append(left + char + right)
        return words
```

获赞 2 添加评论

九章用户U6HP8S
更新于 6/9/2020, 7:03:49 AM

双向BFS 每次总是选择小的一段来拓展 紫薯紫薯紫薯紫薯紫薯紫薯紫薯紫薯紫薯紫薯

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /*
     * @param start: a string
     * @param end: a string
     * @param dict: a set of string
     * @return: An integer
     */
    public int ladderLength(String start, String end, Set<String> dict) {
        if (start.equals(end)) return 1;
        Set<String> startSet = new HashSet<>(), endSet = new HashSet<>(), temp;
        int steps = 1, len = start.length();
        startSet.add(start);
        endSet.add(end);
        if (dict.contains(start)) dict.remove(start);
        if (dict.contains(end)) dict.remove(end);
        while (!startSet.isEmpty() && !endSet.isEmpty()) {
            steps++;
            if (startSet.size() > endSet.size()) {
                temp = startSet;
                startSet = endSet;
                endSet = temp;
            }
            temp = new HashSet<>();
            for (String curr : startSet) {
                char[] chs = curr.toCharArray();
                for (int i = 0; i < len; i++) {
                    char c = chs[i];
                    for (int j = +'a'; j < +'z'; j++) {
                        chs[i] = (char) j;
                        String next = String.valueOf(chs);
                        if (endSet.contains(next)) return steps;
                        if (!dict.contains(next)) continue;
                        dict.remove(next);
                        temp.add(next);
                    }
                    chs[i] = c;
                }
            }
            startSet = temp;
        }
        return -1;
    }
}
```

获赞 2 添加评论

kengran
更新于 8/12/2020, 3:47:19 PM

想了个优化单词升成的算法，选单词部分最优时间复杂度大概是O（L^2（L为单词长度）），最坏还是O（26*L^2）。beat 96%。如有错漏请指正

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: start: a string
    @param: end: a string
    @param: dict: a set of string
    @return: An integer
    """
    def ladderLength(self, start, end, dict):
        # write your code here
        if not dict:
            return 0
        if start == end:
            return 1

        # dict transformation
        dictList = [{} for _ in range (len(start))]

        for i in range (len(start)):
            for word in dict:
                key = word[:i] + word[i + 1:]
                if key in dictList[i]:
                    dictList[i][key].append(word)
                else:
                    dictList[i][key] = [word]
        # print (dictList)

        endList = []
        for i in range (len(end)):
            partialEnd = end[:i] + end[i + 1:]
            endList.append(partialEnd)

        from collections import deque
        q = deque([start])
        seen = set([start])
        ans = 2
        while q:
            # print (q)
            for _ in range (len(q)):
                node = q.popleft()
                for i in range (len(node)):
                    partialWord = node[:i] + node[i + 1:]
                    if partialWord == endList[i]:
                        print ("here")
                        return ans
                    if partialWord in dictList[i]:
                        for w in dictList[i][partialWord]:
                            if w not in seen:
                                q.append(w)
                                seen.add(w)
            if len(q) > 0:
                ans += 1

        return 0
```

九章用户BHJVP5
更新于 6/9/2020, 7:03:56 AM

第一次遍历了一遍dict， 找合适的String，结果超时了， 只好按照老师说的构造所有可能的String， 去dict里边match

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /*
     * @param start: a string
     * @param end: a string
     * @param dict: a set of string
     * @return: An integer
     */
    public int ladderLength(String start, String end, Set<String> dict) {
        // write your code here
        if (dict == null) {
            return 0;
        }

        if (start.equals(end)) {
            return 1;
        }

        dict.add(start);
        dict.add(end);

        Queue<String> queue = new LinkedList<String>();
        Set<String> set = new HashSet<String>();
        queue.offer(start);
        set.add(start);

        int length = 1;
        while (!queue.isEmpty()) {
            length++;
            int size = queue.size();
            for (int i = 0; i < size; i++) {
                String current = queue.poll();
                for (String word : getNext(current, dict)) {
                    if (set.contains(word)) {
                        continue;
                    }
                    if (word.equals(end)) {
                        return length;
                    }
                    queue.add(word);
                    set.add(word);
                }
            }
        }
        return 0;
    }

    private ArrayList<String> getNext(String current, Set<String> dict) {
        ArrayList<String> rel = new ArrayList<String>();

        for (char c = 'a'; c <= 'z'; c++) {
            for (int i = 0; i < current.length(); i++) {
```

```
                char[] chars = current.toCharArray(); //26个字母只换一个
                chars[i] = c;
                String temp = new String(chars);//生成新的String

                if (dict.contains(temp)) {
                     rel.add(temp);
                }
            }
        }
        return rel;
    }
}
```

获赞 1　　添加评论

九章用户51HVT8
更新于 6/9/2020, 7:03:56 AM

举例hot 分三步 *ot h_t ho* 用一个queue track 并且每次把获得的word存进queue， 因为每次都是在当前queue的size底下，所以不用担心length的问题

这个题和leetcode 127 稍微有一点不同

个人感觉不要continue那一个check是否本身，一样可以ac。因为一开始在进入 while（）循环的之前，首先在set里面清除了第一个startword，之后只要在set查到下一个 单词就会立即清除这个词的记录，所以set里面不会重复出现之前的那个原始词。 表述不太到位，大家仔细走一遍程序看看

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    int ladderLength(string &start, string &end, unordered_set<string> &dict) {
        if(start == end)
            return 1;
        int n = start.size();
        if(n < 1 || n != end.size())
            return 0;
        std::queue<string> Q;
        Q.push(start);
        dict.erase(start);
        int length = 2;
        while(!Q.empty())
        {
            int size = Q.size();
            for(int i = 0; i < size; i++)
            {
                string word = Q.front();
                Q.pop();
                for(int i = 0; i < n; i++)
                {
                    char oldchar = word[i];
                    for(char c = 'a'; c <= 'z'; c++)
                    {
                        // if(c == oldchar)
                        //     continue;
                        word[i] = c;
                        if (word == end)
                            return length;
                        if(dict.find(word) != dict.end())
                        {
                            Q.push(word);
                            dict.erase(word);
                        }
                    }
                    word[i] = oldchar;
                }
            }
            length++;
        }
        return 0;
    }
};
```

获赞 1　　1 条评论

九章用户FSUHLT
更新于 6/9/2020, 7:03:53 AM

算法：　BFS or backtrack 剪枝：　如果已经使用过，就从字典删除，因为不需要再经过 note: expand比比较长度差1快 这题就是backtrack 和剪枝，　尝试所有可能性并且剪掉重复路径

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /*
     * @param start: a string
     * @param end: a string
     * @param dict: a set of string
     * @return: An integer
     */
    public int ladderLength(String start, String end, Set<String> dict) {
        dict.add(end);
        Queue<String> queue = new LinkedList<>();
        queue.offer(start);
        int step = 2;
        while (!queue.isEmpty()) {
            int size = queue.size();
            for (int i=0; i<size; i++) {
                String word = queue.poll();
                List<String> neightbors = expand(word, dict);
                if (neightbors.contains(end)) {
                    return step;
                }
                queue.addAll(neightbors);
                dict.removeAll(neightbors);
            }
            step++;
        }
        return -1;
    }

    public List<String> expand(String a, Set<String> dict) {
        List<String> expansion = new ArrayList<>();
        for (int i=0; i<a.length(); i++) {
            for (char c='a'; c<='z'; c++) {
                if (a.charAt(i) != c){
                     String t = a.substring(0, i) + c + a.substring(i+1);
                     if (dict.contains(t)){
                         expansion.add(t);
                     }
                }
            }
        }
        return expansion;
    }
}
```

获赞 1　　添加评论

Asterisk
更新于 6/9/2020, 7:03:53 AM

BFS, 多用一个vistedMap来剪枝，每次找到和上一个只差一个字符的全部list

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
var ladderLength = function(beginWord, endWord, wordList) {
    let queue = [beginWord];
    let visitedMap  = {};
    let count = 0;
    while(queue.length != 0){
        count ++;
        const length = queue.length;
        for(let i = 0; i < length; i ++){
            let node = queue.pop();
            if(node == endWord){
                return count;
            }
            const nextList = findNext(node, wordList, visitedMap);
            for(let j = 0; j < nextList.length; j ++){
                if(nextList[j] == endWord){
                    return count + 1;
                }
                visitedMap[nextList[j]] = true;
                queue.unshift(nextList[j]);
            }
        }
    }
    return 0;
};
function findNext(word, arr, visitedMap){
    let result = [];
    for(let i = 0;i < arr.length; i ++){
        let ele = arr[i];
        if(visitedMap[ele]){
            continue;
        }
        let diff = 0;
        for(let j = 0;j < word.length; j ++){
            if(word.charAt(j) != ele.charAt(j)){
                diff ++;
            }
        }
        if(diff == 1){
            result.push(ele);
        }
    }
    return result;
}
```

获赞 1　　添加评论

Jet
更新于 6/9/2020, 7:03:53 AM

法一：BFS分层，用k代表深度，最后return k.通过检查string的每一位置总共24个字母来确定是否在dict内，最后判断是否加入过visited数组 复杂度O(L*24*L)

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:

    int ladderLength(string &start, string &end, unordered_set<string> &dict) {

        if(dict.count(end)==0){
            dict.insert(end);
        }
        std::queue<string> queue;
        unordered_set<string> seen;
        queue.push(start);

        seen.insert(start);

        int depth=0;
        while(!queue.empty()){
            size_t k=queue.size();
            depth++;
          while(k>0){
            string word=queue.front();
            queue.pop();
            k--;
            if(word==end){
                return depth;
            }

         for(int i=0;i<word.length();i++){
            char old = word[i];
            for(char c ='a';c<='z';c++){

                word[i]=c;

                if(dict.count(word)){
                    if(!seen.count(word)){
                    queue.push(word);
                    seen.insert(word);

                    }
                }
            }
            word[i]=old;
          }
            }
    }
    return 0;
    }
};
```

获赞 1　　添加评论

Jet
更新于 6/9/2020, 7:03:53 AM

法二：令狐冲老师上课说的用一个hashTable来记录每个点到起点start的距离，到终点停止。该题注意点1）不要忘记将end加入dict 2)最后check hashtable是否含有target string(end)。若无return 0,若有return hashtabletarget ()+1 3）通过对dict一个预处理，用模板pattern方法，提高运行速度

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
  /**
   * @param grid: a chessboard included 0 (false) and 1 (true)
   * @param source: a point
   * @param destination: a point
   * @return: the shortest path
   */
  int ladderLength(string& start, string& end, unordered_set<string>& dict) {
    if (dict.count(end) == 0) {
      dict.insert(end);
    }
    unordered_map<string,vector<string>> mapping = preprocess(dict);
    queue<string> Queue;
    unordered_map<string, int> distance;
    distance[start] = 0;
    Queue.push(start);
    while (!Queue.empty()) {

        string first = Queue.front();
        Queue.pop();
        if (first == end) {
          break;
        }
        vector<string> neighbors = index(first, mapping);
        for (int i = 0; i < neighbors.size(); i++) {
          if (distance.count(neighbors[i]) != 0) {
            continue;
          }
          Queue.push(neighbors[i]);
          distance[neighbors[i]] = distance[first] + 1;
        }
    }
    if (distance.count(end) == 0) {
      return 0;
    }
    return distance[end]+1;
  }

  vector<string> index(string str, unordered_map<string,vector<string> >& mapping) {
    vector<string> result;
    for (int i = 0; i < str.length(); i++) {
      string p = str;
      p[i] = '*';
      if (mapping.count(p) == 0) {
        continue;
      }
      for (int j = 0; j < mapping[p].size(); j++) {
        result.push_back(mapping[p][j]);//repeat?
      }
    }
    return result;
  }

  unordered_map<string, vector<string>> preprocess(unordered_set<string>& dict) {
    unordered_map<string, vector<string>> result;
    for (auto it = dict.begin(); it != dict.end(); it++) {
      for (int i = 0; i < (*it).length(); i++) {
```

```
        string p = *it;
        p[i] = '*';
        if (result.count(p) == 0) {
          vector<string> temp;
          temp.push_back(*it);
          result[p] = temp;
          continue;
        }
        result[p].push_back(*it);
      }
    }
    return result;
  }


};
```

获赞 1　　添加评论

HUE
更新于 8/15/2020, 10:05:11 AM

雙向 BFS，用 getNextWord 降低搜尋字典時的時間複雜度

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    /*
     * @param start: a string
     * @param end: a string
     * @param dict: a set of string
     * @return: An integer
     */
    int ladderLength(string &start, string &end, unordered_set<string> &dict) {
        dict.insert(start);
        dict.insert(end);

        queue<string> forwardQueue;
        unordered_set<string> forwardSet;
        queue<string> backwardQueue;
        unordered_set<string> backwardSet;

        forwardQueue.push(start);
        forwardSet.insert(start);
        backwardQueue.push(end);
        backwardSet.insert(end);

        int distance = 0;
        while (!forwardQueue.empty() && !backwardQueue.empty()) {
            distance++;
            if (extendQueue(dict, forwardQueue, forwardSet, backwardSet)) {
                return distance;
            }

            distance++;
            if (extendQueue(dict, backwardQueue, backwardSet, forwardSet)) {
```

```
                return distance;
            }
        }

        return 0;
    }

    bool extendQueue(unordered_set<string> &dict,
                     queue<string> &q,
                     unordered_set<string> &visited,
                     unordered_set<string> &oppositeVisited) {
        int queueSize = q.size();
        for (int i = 0; i < queueSize; i++) {
            string curStr = q.front();
            q.pop();
            if (oppositeVisited.count(curStr)) {
                return true;
            }
            for (string nextStr : getNextStr(dict, visited, curStr)) {
                q.push(nextStr);
                visited.insert(nextStr);
            }
        }
        return false;
    }

    vector<string> getNextStr(unordered_set<string> &dict,
                              unordered_set<string> &visited,
                              string &str) {
        vector<string> res;
        for (int i = 0; i < str.size(); i++) {
            for (char ch : "abcdefghijklmnopqrstwxyz") {
                if (ch == str[i]) {
                    continue;
                }
                char tmp = str[i];
                str[i] = ch;
                if (dict.count(str) && !visited.count(str)) {
                    res.push_back(str);
                }
                str[i] = tmp;
            }
        }
        return res;
    }
};
```

获赞 0　　添加评论

九章用户X4EYT8
更新于 6/18/2020, 2:42:35 PM

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    public int ladderLength(String start, String end, Set<String> dict) {
        // Assumptions: start and end are non-empty, and are not the same
        Queue<String> queue = new LinkedList<>();
        Set<String> visited = new HashSet<>();
        queue.offer(start);
        visited.add(start);
        int step = 1;

        while (!queue.isEmpty()) {
            int size = queue.size();
            step++;
            for (int i = 0; i < size; i++) {
                String cur = queue.poll();
                if (canTransform(cur, end)) {
                    return step;
                }
                for (String s : dict) {
                    if (canTransform(cur, s) && !visited.contains(s)) {
                        queue.offer(s);
                        visited.add(s);
                    }
                }
            }
        }

        // return 0 if there is no such transformation sequence
        return 0;
    }

    // s1 can transform to s2 if s1 needs to change exactly one letter to become s2
    boolean canTransform(String s1, String s2) {
        // Assumption: all words have a same length
        int diff = 0;
        for (int i = 0; i < s1.length(); i++) {
            if (s1.charAt(i) != s2.charAt(i)) {
                diff++;
            }
        }
        return diff == 1;
    }
}
```


获赞 0 添加评论

你是疯儿我是傻
更新于 6/14/2020, 2:29:13 PM

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
```

```
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {

    /*
     * @param start: a string
     * @param end: a string
     * @param dict: a set of string
     * @return: An integer
     */
    public int ladderLength(String start, String end, Set<String> dict) {
        // write your code here
        if (dict == null || dict.isEmpty()) return -1;

        //step 1: for each word we generate its neighber list
        HashMap<String, Set<String>> subWords = new HashMap<>();//saves all (m-1) sized sub words from the word in size of
m
        HashMap<String, Set<String>> connectedWords = new HashMap<>(); //maps all known sub words to the words that has it
as subword

        HashSet<String> dictWithStartandEnd = new HashSet<>(dict);
        dictWithStartandEnd.add(start);
        dictWithStartandEnd.add(end);
        for (String word : dictWithStartandEnd) {
            Set<String> itsPath = new HashSet<>();
            for (int i = 0; i < word.length(); i++) {
                StringBuilder sb = new StringBuilder(word);
                sb.setCharAt(i, '#');
                String subWord = sb.toString();

                itsPath.add(sb.toString());
                if (!connectedWords.containsKey(subWord)) {
                    Set<String> pathToWords = new HashSet<>();
                    connectedWords.put(subWord, pathToWords);
                }
                connectedWords.get(subWord).add(word);
            }

            subWords.put(word, itsPath);
        }

        //step2: bfs each word that from start can reach
        Queue<String> bfsQueue = new LinkedList<>();
        Set<String> visistedWord = new HashSet<>();
        bfsQueue.offer(start);
        visistedWord.add(start);
        Map<String, Integer> shortestPath = init(start, dictWithStartandEnd);

        while (!bfsQueue.isEmpty()) {
            String cur = bfsQueue.poll();
            if (cur.equals(end)) break;

            for (String path : subWords.get(cur)) {
                Set<String> nextWordCand = connectedWords.get(path);
                for (String nextWord : nextWordCand) {
                    int nextLength = shortestPath.get(cur) + 1;
                    if (nextWord.equals(cur) || visistedWord.contains(nextWord)) continue;

                    shortestPath.put(nextWord, nextLength);
                    bfsQueue.offer(nextWord);
                    visistedWord.add(nextWord);
                }
            }
        }

        int shortestToEnd = shortestPath.get(end);
        return (shortestToEnd == Integer.MAX_VALUE) ? -1 : shortestToEnd;
```

```
    }

    private Map<String, Integer> init(String start, Set<String> dict) {
        Map<String, Integer> res = new HashMap<>();
        for (String word : dict) {
            if(word.equals(start)) res.put(word, 1);
            else res.put(word, Integer.MAX_VALUE);
        }
        return res;
    }

}
```

获赞 0 添加评论

九章用户FB92NX
更新于 6/9/2020, 7:04:27 AM

参照Java版答案思想，每次将dict中取出到queue中的字符串从dict中移除掉，这样可以将getNextWords函数中hashtable中的查找时间再降一降。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    /*
     * @param start: a string
     * @param end: a string
     * @param dict: a set of string
     * @return: An integer
     */
    vector<string> getNextWords(string str, unordered_set<string> &dict){
        vector<string> results;
        for (int i = 0; i < str.size(); i ++) {
            char t = str[i];
            for (char c = 'a'; c <= 'z'; c++) {
                if (c == str[i]) {
                    continue;
                }
                str[i] = c;
                auto tempptr = dict.find(str);
                if (tempptr != dict.end()) {
                    results.push_back(str);
                    //从dict中删除string
                    dict.erase(tempptr);
                }
            }
            str[i] = t;
        }
        return results;
    }
    int ladderLength(string &start, string &end, unordered_set<string> &dict) {
        if (start == end) {
            return 1;
        }
        dict.insert(end);
        queue<string> q;
        q.push(start);
        int length = 2;
        while (!q.empty()) {
            int s = q.size();
            for (int i = 0; i < s; i ++) {
                string tempstr = q.front();
                q.pop();
                for (auto wordstr: getNextWords(tempstr, dict)) {
                    if (wordstr == end) {
                        return length;
                    }
                    q.push(wordstr);
                }
            }
            length++;
        }
        return 0;
    }
};
```

获赞 0 添加评论

Frederic
更新于 6/9/2020, 7:04:25 AM

应用Breath first search，把getNewWords()单独写出来的版本。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: start: a string
    @param: end: a string
    @param: dict: a set of string
    @return: An integer
    """
    def ladderLength(self, start, end, dict):
        # write your code here
        if dict == None:
            return 0

        if start == end:
            return 1

        dict.add(start)
        dict.add(end)

        queue = collections.deque([start])
        hash = {start:True}

        length = 1
        while queue:
            length += 1
            for i in range(len(queue)):
                word = queue.popleft()
                for nextWord in self.getNextWords(word, dict):
                    if nextWord in hash:
                        continue
                    if nextWord == end:
                        return length

                    hash[nextWord] = True
                    queue.append(nextWord)

    def replace(self, string, index, char):
        return string[:index] + char + string[index+1:]

    def getNextWords(self, word, dict):
        nextWord = ''
        nextWords = []
        wordLen = len(word)
        for c in 'abcdefghijklmnopqrstuvwxyz':
            for i in xrange(wordLen):
                if c != word[i]:
                    nextWord = self.replace(word, i, c)
                if nextWord in dict:
                    nextWords.append(nextWord)
        return nextWords
```

获赞 0　　添加评论

九章用户51HVT8
更新于 6/9/2020, 7:04:23 AM

举例hot 分三步 *ot h_t ho* 用一个queue track 并且每次把获得的word存进queue， 因为每次都是在当前queue的size底下，所以不用担心length的问题

这个题和leetcode 127 稍微有一点不同

个人感觉不要continue那一个check是否本身，一样可以ac。因为一开始在进入 while（）循环的之前，首先在set里面清除了第一个startword，之后只要在set查到下一个 单词就会立即清除这个词的记录，所以set里面不会重复出现之前的那个原始词。 表述不太到位，大家仔细走一遍程序看看

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    int ladderLength(string &start, string &end, unordered_set<string> &dict) {
        if(start == end)
            return 1;
        int n = start.size();
        if(n < 1 || n != end.size())
            return 0;
        std::queue<string> Q;
        Q.push(start);
        dict.erase(start);
        int length = 2;
        while(!Q.empty())
        {
            int size = Q.size();
            for(int i = 0; i < size; i++)
            {
                string word = Q.front();
                Q.pop();
                for(int i = 0; i < n; i++)
                {
                    char oldchar = word[i];
                    for(char c = 'a'; c <= 'z'; c++)
                    {
                        // if(c == oldchar)
                        //     continue;
                        word[i] = c;
                        if (word == end)
                            return length;
                        if(dict.find(word) != dict.end())
                        {
                            Q.push(word);
                            dict.erase(word);
                        }
                    }
                    word[i] = oldchar;
                }
            }
            length++;
        }
        return 0;
    }
};
```

获赞 0 添加评论

九章用户51HVT8
更新于 6/9/2020, 7:04:23 AM

举例hot 分三步 *ot h_t ho* 用一个queue track 并且每次把获得的word存进queue， 因为每次都是在当前queue的size底下，所以不用担心length的问题

这个题和leetcode 127 稍微有一点不同

个人感觉不要continue那一个check是否本身，一样可以ac。因为一开始在进入 while（）循环的之前，首先在set里面清除了第一个startword，之后只要在set查到下一个 单词就会立即清除这个词的记录，所以set里面不会重复出现之前的那个原始词。 表述不太到位，大家仔细走一遍程序看看

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    int ladderLength(string &start, string &end, unordered_set<string> &dict) {
        if(start == end)
            return 1;
        int n = start.size();
        if(n < 1 || n != end.size())
            return 0;
        std::queue<string> Q;
        Q.push(start);
        dict.erase(start);
        int length = 2;
        while(!Q.empty())
        {
            int size = Q.size();
            for(int i = 0; i < size; i++)
            {
                string word = Q.front();
                Q.pop();
                for(int i = 0; i < n; i++)
                {
                    char oldchar = word[i];
                    for(char c = 'a'; c <= 'z'; c++)
                    {
                        // if(c == oldchar)
                        //     continue;
                        word[i] = c;
                        if (word == end)
                            return length;
                        if(dict.find(word) != dict.end())
                        {
                            Q.push(word);
                            dict.erase(word);
                        }
                    }
                    word[i] = oldchar;
                }
            }
            length++;
        }
        return 0;
    }
};
```

获赞 0 添加评论

九章用户51HVT8
更新于 6/9/2020, 7:04:23 AM

举例hot 分三步 *ot h_t ho* 用一个queue track 并且每次把获得的word存进queue， 因为每次都是在当前queue的size底下，所以不用担心length的问题

这个题和leetcode 127 稍微有一点不同

个人感觉不要continue那一个check是否本身，一样可以ac。因为一开始在进入 while（）循环的之前，首先在set里面清除了第一个startword，之后只要在set查到下一个 单词就会立即清除这个词的记录，所以set里面不会重复出现之前的那个原始词。 表述不太到位，大家仔细走一遍程序看看

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    int ladderLength(string &start, string &end, unordered_set<string> &dict) {
        if(start == end)
            return 1;
        int n = start.size();
        if(n < 1 || n != end.size())
            return 0;
        std::queue<string> Q;
        Q.push(start);
        dict.erase(start);
        int length = 2;
        while(!Q.empty())
        {
            int size = Q.size();
            for(int i = 0; i < size; i++)
            {
                string word = Q.front();
                Q.pop();
                for(int i = 0; i < n; i++)
                {
                    char oldchar = word[i];
                    for(char c = 'a'; c <= 'z'; c++)
                    {
                        // if(c == oldchar)
                        //     continue;
                        word[i] = c;
                        if (word == end)
                            return length;
                        if(dict.find(word) != dict.end())
                        {
                            Q.push(word);
                            dict.erase(word);
                        }
                    }
                    word[i] = oldchar;
                }
            }
            length++;
        }
        return 0;
    }
};
```

获赞 0 添加评论

九章用户51HVT8
更新于 6/9/2020, 7:04:23 AM

举例hot 分三步 *ot h_t ho* 用一个queue track 并且每次把获得的word存进queue， 因为每次都是在当前queue的size底下，所以不用担心length的问题

这个题和leetcode 127 稍微有一点不同

个人感觉不要continue那一个check是否本身，一样可以ac。因为一开始在进入 while（）循环的之前，首先在set里面清除了第一个startword，之后只要在set查到下一个 单词就会立即清除这个词的记录，所以set里面不会重复出现之前的那个原始词。 表述不太到位，大家仔细走一遍程序看看

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    int ladderLength(string &start, string &end, unordered_set<string> &dict) {
        if(start == end)
            return 1;
        int n = start.size();
        if(n < 1 || n != end.size())
            return 0;
        std::queue<string> Q;
        Q.push(start);
        dict.erase(start);
        int length = 2;
        while(!Q.empty())
        {
            int size = Q.size();
            for(int i = 0; i < size; i++)
            {
                string word = Q.front();
                Q.pop();
                for(int i = 0; i < n; i++)
                {
                    char oldchar = word[i];
                    for(char c = 'a'; c <= 'z'; c++)
                    {
                        // if(c == oldchar)
                        //     continue;
                        word[i] = c;
                        if (word == end)
                            return length;
                        if(dict.find(word) != dict.end())
                        {
                            Q.push(word);
                            dict.erase(word);
                        }
                    }
                    word[i] = oldchar;
                }
            }
            length++;
        }
        return 0;
    }
};
```

获赞 0 添加评论

九章用户51HVT8
更新于 6/9/2020, 7:04:23 AM

举例hot 分三步 *ot h_t ho* 用一个queue track 并且每次把获得的word存进queue， 因为每次都是在当前queue的size底下，所以不用担心length的问题

这个题和leetcode 127 稍微有一点不同

个人感觉不要continue那一个check是否本身，一样可以ac。因为一开始在进入 while（）循环的之前，首先在set里面清除了第一个startword，之后只要在set查到下一个 单词就会立即清除这个词的记录，所以set里面不会重复出现之前的那个原始词。 表述不太到位，大家仔细走一遍程序看看

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    int ladderLength(string &start, string &end, unordered_set<string> &dict) {
        if(start == end)
            return 1;
        int n = start.size();
        if(n < 1 || n != end.size())
            return 0;
        std::queue<string> Q;
        Q.push(start);
        dict.erase(start);
        int length = 2;
        while(!Q.empty())
        {
            int size = Q.size();
            for(int i = 0; i < size; i++)
            {
                string word = Q.front();
                Q.pop();
                for(int i = 0; i < n; i++)
                {
                    char oldchar = word[i];
                    for(char c = 'a'; c <= 'z'; c++)
                    {
                        // if(c == oldchar)
                        //     continue;
                        word[i] = c;
                        if (word == end)
                            return length;
                        if(dict.find(word) != dict.end())
                        {
                            Q.push(word);
                            dict.erase(word);
                        }
                    }
                    word[i] = oldchar;
                }
            }
            length++;
        }
        return 0;
    }
};
```

获赞 0 添加评论

九章用户51HVT8
更新于 6/9/2020, 7:04:23 AM

举例hot 分三步 *ot h_t ho* 用一个queue track 并且每次把获得的word存进queue， 因为每次都是在当前queue的size底下，所以不用担心length的问题

这个题和leetcode 127 稍微有一点不同

个人感觉不要continue那一个check是否本身，一样可以ac。因为一开始在进入 while（）循环的之前，首先在set里面清除了第一个startword，之后只要在set查到下一个 单词就会立即清除这个词的记录，所以set里面不会重复出现之前的那个原始词。 表述不太到位，大家仔细走一遍程序看看

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    int ladderLength(string &start, string &end, unordered_set<string> &dict) {
        if(start == end)
            return 1;
        int n = start.size();
        if(n < 1 || n != end.size())
            return 0;
        std::queue<string> Q;
        Q.push(start);
        dict.erase(start);
        int length = 2;
        while(!Q.empty())
        {
            int size = Q.size();
            for(int i = 0; i < size; i++)
            {
                string word = Q.front();
                Q.pop();
                for(int i = 0; i < n; i++)
                {
                    char oldchar = word[i];
                    for(char c = 'a'; c <= 'z'; c++)
                    {
                        // if(c == oldchar)
                        //     continue;
                        word[i] = c;
                        if (word == end)
                            return length;
                        if(dict.find(word) != dict.end())
                        {
                            Q.push(word);
                            dict.erase(word);
                        }
                    }
                    word[i] = oldchar;
                }
            }
            length++;
        }
        return 0;
    }
};
```

获赞 0 添加评论

九章用户51HVT8
更新于 6/9/2020, 7:04:23 AM

举例hot 分三步 *ot h_t ho* 用一个queue track 并且每次把获得的word存进queue， 因为每次都是在当前queue的size底下，所以不用担心length的问题

这个题和leetcode 127 稍微有一点不同

个人感觉不要continue那一个check是否本身，一样可以ac。因为一开始在进入 while（）循环的之前，首先在set里面清除了第一个startword，之后只要在set查到下一个 单词就会立即清除这个词的记录，所以set里面不会重复出现之前的那个原始词。 表述不太到位，大家仔细走一遍程序看看

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    int ladderLength(string &start, string &end, unordered_set<string> &dict) {
        if(start == end)
            return 1;
        int n = start.size();
        if(n < 1 || n != end.size())
            return 0;
        std::queue<string> Q;
        Q.push(start);
        dict.erase(start);
        int length = 2;
        while(!Q.empty())
        {
            int size = Q.size();
            for(int i = 0; i < size; i++)
            {
                string word = Q.front();
                Q.pop();
                for(int i = 0; i < n; i++)
                {
                    char oldchar = word[i];
                    for(char c = 'a'; c <= 'z'; c++)
                    {
                        // if(c == oldchar)
                        //     continue;
                        word[i] = c;
                        if (word == end)
                            return length;
                        if(dict.find(word) != dict.end())
                        {
                            Q.push(word);
                            dict.erase(word);
                        }
                    }
                    word[i] = oldchar;
                }
            }
            length++;
        }
        return 0;
    }
};
```

获赞 0 添加评论

加载更多题解

进阶课程

视频+互动

九章算法班 2021 版

8周时间精通 57 个核心高频考点，9招击破 FLAG、BATJ 算法面试。22....

直播+互动

系统架构设计 System Design 2021 版

成为百万架构师必上。30 课时带你快速掌握18大系统架构设计知识点与面...

直播+互动

九章算法面试高频题冲刺班

每期更新 15% 题目，考前押题，一举拿下FLAG & BATJ Offer

互动课

面向对象设计 OOD

应届生及亚马逊面试必考，IT求职必备基础

(/)