Leetcode(python) 广度优先遍历 中等题

127单词接龙

题目：

给定两个单词（beginWord 和 endWord）和一个字典，找到从 beginWord 到 endWord 的最短转换序列的长度。转换需遵循如下规则：

每次转换只能改变一个字母。

转换过程中的中间单词必须是字典中的单词。

说明:

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。

示例 1:

输入:

beginWord = "hit",

endWord = "cog",

wordList = ["hot","dot","dog","lot","log","cog"]

输出: 5

解释: 一个最短转换序列是 "hit" -> "hot" -> "dot" -> "dog" -> "cog",

返回它的长度 5。

示例 2:

输入:

beginWord = "hit"

endWord = "cog"

wordList = ["hot","dot","dog","lot","log"]

输出: 0

解释: endWord "cog" 不在字典中，所以无法进行转换。

思路：

代码：

1. **class** Solution(object):
2. **def** ladderLength(self, beginWord, endWord, wordList):
3. """
4. :type beginWord: str
5. :type endWord: str
6. :type wordList: List[str]
7. :rtype: int
8. """
9. wordDict = set(wordList)
10. **if** endWord **not** **in** wordDict:
11. **return** 0
12. **if** beginWord **in** wordDict:
13. wordDict.remove(beginWord)
15. stack, visited = [(beginWord, 1)], set()
16. **while** stack:
17. word, step = stack.pop(0)
18. **if** word **not** **in** visited:
19. visited.add(word)
20. **if** word == endWord:
21. **return** step
22. **for** i **in** range(len(word)):
23. **for** j **in** 'abcdefghijklmnopqrstuvwxyz':
24. tmp = word[:i] + j + word[i+1:]
25. **if** tmp **in** wordDict **and** tmp **not** **in** visited:
26. stack.append((tmp, step + 1))
28. **return** 0

279完全平方数

题目：

给定正整数 n，找到若干个完全平方数（比如 1, 4, 9, 16, ...）使得它们的和等于 n。你需要让组成和的完全平方数的个数最少。

示例 1:

输入: n = 12

输出: 3

解释: 12 = 4 + 4 + 4.

示例 2:

输入: n = 13

输出: 2

解释: 13 = 4 + 9.

思路：

代码：

1. **class** Solution(object):
2. **def** numSquares(self, n):
3. """
4. :type n: int
5. :rtype: int
6. """
7. dp = [float('inf') **for** i **in** range(n + 1)]
8. dp[0] = 0
9. **for** i **in** range(n+1):
10. j = 1
11. **while** i + j\*\*2 <= n:
12. dp[i + j\*\*2] = min(dp[i + j\*\*2], dp[i] + 1)
13. j += 1
14. **return** dp[n]

529扫雷游戏

题目：

让我们一起来玩扫雷游戏！

给定一个代表游戏板的二维字符矩阵。 'M' 代表一个未挖出的地雷，'E' 代表一个未挖出的空方块，'B' 代表没有相邻（上，下，左，右，和所有4个对角线）地雷的已挖出的空白方块，数字（'1' 到 '8'）表示有多少地雷与这块已挖出的方块相邻，'X' 则表示一个已挖出的地雷。

现在给出在所有未挖出的方块中（'M'或者'E'）的下一个点击位置（行和列索引），根据以下规则，返回相应位置被点击后对应的面板：

如果一个地雷（'M'）被挖出，游戏就结束了- 把它改为 'X'。

如果一个没有相邻地雷的空方块（'E'）被挖出，修改它为（'B'），并且所有和其相邻的方块都应该被递归地揭露。

如果一个至少与一个地雷相邻的空方块（'E'）被挖出，修改它为数字（'1'到'8'），表示相邻地雷的数量。

如果在此次点击中，若无更多方块可被揭露，则返回面板。

思路：

代码：

1. **class** Solution(object):
2. **def** updateBoard(self, board, click):
3. """
4. :type board: List[List[str]]
5. :type click: List[int]
6. :rtype: List[List[str]]
7. """
8. (row, col), directions = click, ((-1, 0), (1, 0), (0, 1), (0, -1), (-1, 1), (-1, -1), (1, 1), (1, -1))
9. **if** 0 <= row < len(board) **and** 0 <= col < len(board[0]):
10. **if** board[row][col] == 'M':
11. board[row][col] = 'X'
12. **elif** board[row][col] == 'E':
13. n = sum([board[row + r][col + c] == 'M' **for** r, c **in** directions **if** 0 <= row + r < len(board) **and** 0 <= col +c < len(board[0])])
14. board[row][col] = str(n **if** n **else** 'B')
15. **for** r, c **in** directions \* (**not** n):
16. self.updateBoard(board, [row + r, col + c])
17. **return** board