Assignment #A: Graph starts
Updated 1830 GMT+8 Apr 22, 2025
2025 spring, Complied by 李振硕、信息管理系

1. 题目
M19943:图的拉普拉斯矩阵
OOP, implementation,
http://cs101.openjudge.cn/practice/19943/
要求创建 Graph, Vertex 两个类，建图实现。
思路：
代码：

CS101 / 题库（包括计概、数算题目）

题目 排名 状态 提问

#49031237提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
n,m=map(int,input().split())

D={}
dp1=[[0]*n for i in range(n)]
dp2=[[0]*n for i in range(n)]

for i in range(m):
    st,fn=map(int,input().split())
    dp1[st][fn]=1
    dp1[fn][st]=1
    dp2[st][st]+=1
    dp2[fn][fn]+=1



for x in range(n):
    for y in range(n):
        print(dp2[x][y]-dp1[x][y],end=' ')
    print()
```

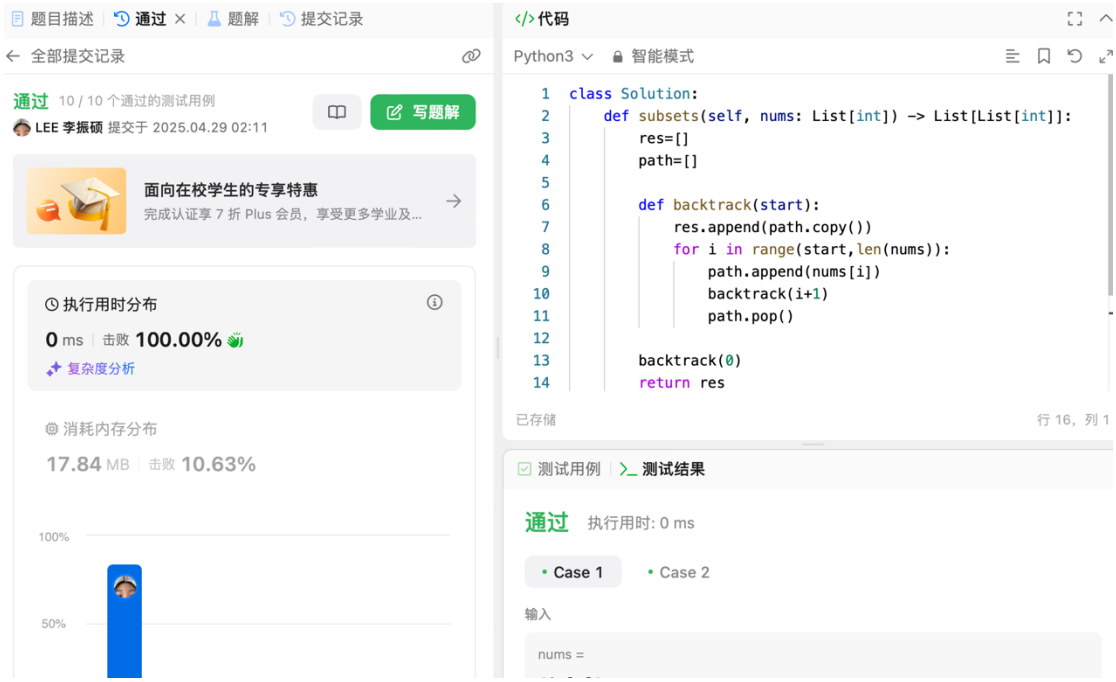
基本信息

#: 49031237
题目: 19943
提交人: 24n2300093007
内存: 3648kB
时间: 21ms
语言: Python3
提交时间: 2025-04-28 19:57:57

English 帮助 关于

LC78.子集
backtracking, https://leetcode.cn/problems/subsets/
思路：

代码：



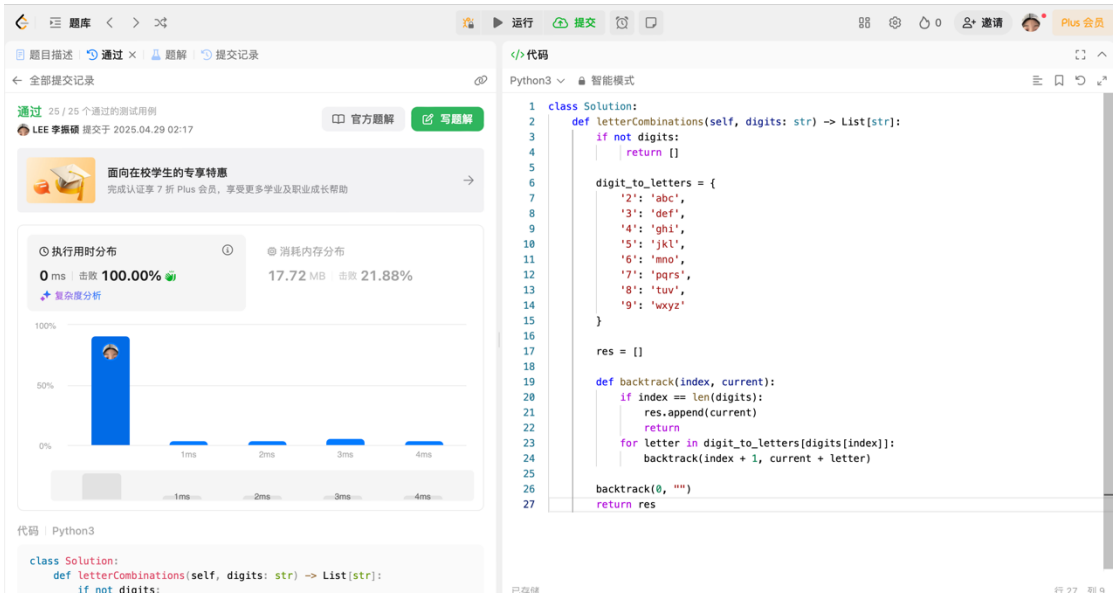
LC17.电话号码的字母组合

hash table, backtracking,

https://leetcode.cn/problems/letter-combinations-of-a-phone-number/

思路：

代码：



M04089:电话号码

trie, http://cs101.openjudge.cn/practice/04089/

思路：

代码：

#49032713提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def is_consistent(numbers):
    numbers.sort()
    for i in range(len(numbers) - 1):
        if numbers[i+1].startswith(numbers[i]):
            return False
    return True

t = int(input())
for _ in range(t):
    n = int(input())
    numbers = [input().strip() for _ in range(n)]
    print("YES" if is_consistent(numbers) else "NO")
```

基本信息

#: 49032713
题目: 04089
提交人: 24n2300093007
内存: 4920kB
时间: 80ms
语言: Python3
提交时间: 2025-04-29 01:35:36

English 帮助 关于

T28046:词梯

bfs, http://cs101.openjudge.cn/practice/28046/

思路：

代码：

#49032919提交状态

查看 提交 统计

状态: Accepted

源代码

```
from collections import deque, defaultdict

def find_word_ladder():
    n = int(input())
    words = [input().strip() for _ in range(n)]
    start, end = input().split()

    if start == end:
        print(start)
        return

    # Preprocess: build a map from wildcard to list of words
    wildcard_map = defaultdict(list)
    for word in words:
        for i in range(4):
            wildcard = word[:i] + '_' + word[i+1:]
            wildcard_map[wildcard].append(word)

    # Build adjacency list
    graph = defaultdict(list)
    for word in words:
        for i in range(4):
            wildcard = word[:i] + '_' + word[i+1:]
            for neighbor in wildcard_map[wildcard]:
                if neighbor != word:
                    graph[word].append(neighbor)

    # BFS setup
    queue = deque()
    queue.append(start)
    visited = {start: None}  # to keep track of the path (stores parent

    found = False
    while queue:
        current = queue.popleft()
        if current == end:
            found = True
            break
        for neighbor in graph.get(current, []):
            if neighbor not in visited:
                visited[neighbor] = current
                queue.append(neighbor)

    if not found:
        print("NO")
    else:
        # Reconstruct the path
        path = []
        node = end
        while node is not None:
            path.append(node)
            node = visited[node]
        path.reverse()
        print(' '.join(path))

find_word_ladder()
```

基本信息

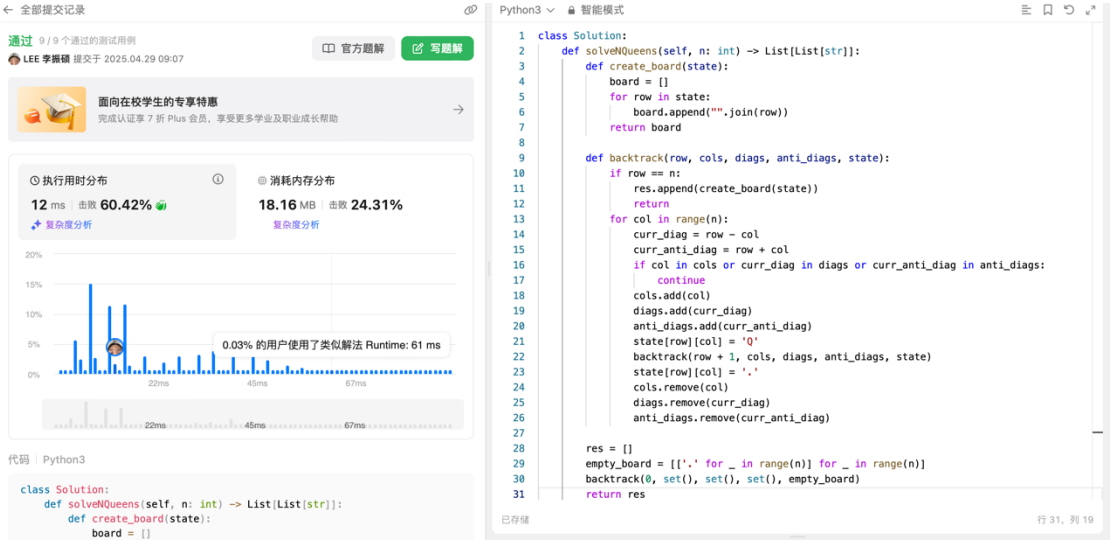
#: 49032919
题目: 28046
提交人: 24n2300093007
内存: 6500kB
时间: 47ms
语言: Python3
提交时间: 2025-04-29 09:04:32

T51.N 皇后

backtracking, https://leetcode.cn/problems/n-queens/

思路：

代码：




```python
class Solution:
    def solveNQueens(self, n: int) -> List[List[str]]:
        def create_board(state):
            board = []
            for row in state:
                board.append("".join(row))
            return board

        def backtrack(row, cols, diags, anti_diags, state):
            if row == n:
                res.append(create_board(state))
                return
            for col in range(n):
                curr_diag = row - col
                curr_anti_diag = row + col
                if col in cols or curr_diag in diags or curr_anti_diag in anti_diags:
                    continue
                cols.add(col)
                diags.add(curr_diag)
                anti_diags.add(curr_anti_diag)
                state[row][col] = 'Q'
                backtrack(row + 1, cols, diags, anti_diags, state)
                state[row][col] = '.'
                cols.remove(col)
                diags.remove(curr_diag)
                anti_diags.remove(curr_anti_diag)

        res = []
        empty_board = [['.' for _ in range(n)] for _ in range(n)]
        backtrack(0, set(), set(), set(), empty_board)
        return res
```

2. 学习总结和收获

感觉我还是不会 backtracking 相关问题，虽然明白了它的运行过程和结构，每次做新的题都不会。。。需要多多做题。