Assignment #A: Graph starts

Updated 1830 GMT+8 Apr 22, 2025

2025 spring, Complied by 任宇桐 物理学院

> **说明：**
>
> 1. **解题与记录：**
>
>   对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge， Codeforces，LeetCode等平台上获得Accepted）。请将这些信息连同显示“Accepted”的截图一起填写到下方的作业模板中。（推荐使用Typora https://typoraio.cn 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。
>
> 2. **提交安排：**提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的“作业评论”区。确保你的Canvas账户有一个清晰可见的头像，提交的文件为PDF格式，并且“作业评论”区包含上传的.md或.doc附件。
>
> 3. **延迟提交：**如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。
>
> 请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

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：

先建邻接表，然后逐一排查。

代码：

```
class Vertex:
    def __init__(self, key):
        self.key = key
        self.neighbors = set()

    def set_neighbor(self, other):
        self.neighbors.add(other)

class Graph:
    def __init__(self):
        self.vertices = dict()

    def set_vertex(self, key):
        self.vertices[key] = Vertex(key)
```

```
    def add_edge(self, a, b):
        self.vertices[a].set_neighbor(self.vertices[b])
        self.vertices[b].set_neighbor(self.vertices[a])

    def get_degrees(self, a):
        return len(self.vertices[a].neighbors)

    def check_path(self, a, b):
        return self.vertices[a] in self.vertices[b].neighbors

n, m = map(int, (input().split()))
graph = Graph()
for i in range(n):
    graph.set_vertex(i)
for _ in range(m):
    a, b = map(int, input().split())
    graph.add_edge(a, b)
laplace_matrix = [[0]*n for _ in range(n)]
for i in range(n):
    for j in range(n):
        if i == j:
            laplace_matrix[i][j] = len(graph.vertices[i].neighbors)
        else:
            if graph.check_path(i, j):
                laplace_matrix[i][j] = -1
            else:
                laplace_matrix[i][j] = 0
for i in range(n):
    print(*laplace_matrix[i])
```

代码运行截图 （至少包含有"Accepted"）

LC78.子集

backtracking, https://leetcode.cn/problems/subsets/

思路：

直接回溯即可。

代码：

```
class Solution:
    def subsets(self, nums: List[int]) -> List[List[int]]:
        ans = []
        n = len(nums)
        def dfs(temp, i):
            if i == n:
                ans.append((temp[:]))
                return

            dfs(temp, i+1)
            temp.append(nums[i])
            dfs(temp, i+1)
            temp.pop()
            return
        dfs([], 0)
        return ans
```

代码运行截图 （至少包含有"Accepted"）

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：

感觉最麻烦的是创建对应的映射。

代码：

```
class Solution:
    def __init__(self):
```

```
        self.ans = []
    def letterCombinations(self, digits: str) -> List[str]:
        nums_to_letters = {'1':[], '2':['a','b','c'], '3':['d','e','f'], '4':
['g','h','i'], '5':['j','k','l'],'6':['m','n','o'],'7':['p','q','r','s'],'8':
['t','u','v'],'9':['w','x','y','z']}
        def backtracking(i, temp):
            if i == len(digits):
                if temp:
                    self.ans.append(temp)
                return

            for j in nums_to_letters[digits[i]]:
                backtracking(i+1, temp+j)

        backtracking(0, '')
        return self.ans
```

代码运行截图 （至少包含有"Accepted"）

M04089:电话号码

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

思路：

注意检测谁是谁的前缀，可以对对象进行排列，保证前短后长。

代码：

```
class Node:
    def __init__(self):
        self.children = dict()

class Trie:
    def __init__(self):
        self.root = Node()

    def insert_num(self, num):
        current_node = self.root
        for x in num:
            if x not in current_node.children:
```

```
                current_node.children[x] = Node()
            current_node = current_node.children[x]
            if 'end' in current_node.children:
                return False
        current_node.children['end']= None
        return True

t = int(input())
for _ in range(t):
    trie = Trie()
    n = int(input())
    nums = []
    for _ in range(n):
        s = input()
        nums.append(s)
    nums.sort()
    for s in nums:
        if not trie.insert_num(s):
            print('NO')
            break
    else:
        print('YES')
```

代码运行截图 （至少包含有"Accepted"）

T28046:词梯

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

思路：

感觉这道题目最难的部分在于构建邻接表。

代码：

```
from collections import deque
import string

class Node:
    def __init__(self):
```

```
        self.adjacent = set()

n = int(input())
words = dict()
for _ in range(n):
    s = str(input())
    words[s] = Node()
for word in words:
    if word[0].islower():
        for i in range(4):
            origin_char = word[i]
            for char in list(string.ascii_lowercase):
                if char == origin_char:
                    continue
                if word[:i]+char+word[i+1:] in words:
                    words[word].adjacent.add(word[:i]+char+word[i+1:])
    else:
        for i in range(4):
            origin_char = word[i]
            for char in list(string.ascii_uppercase):
                if char == origin_char:
                    continue
                if word[:i]+char+word[i+1:] in words:
                    words[word].adjacent.add(word[:i]+char+word[i+1:])
s, e = input().split()
def bfs():
    if s not in words:
        return 'NO'
    q = deque([s])
    visited = {s:None}
    while q:
        current = q.popleft()
        for neighbor in words[current].adjacent:
            if neighbor not in visited:
                visited[neighbor] = current
                if neighbor == e:
                    path = []
                    node = neighbor
                    while node is not None:
                        path.append(node)
                        node = visited[node]
                    return ' '.join(reversed(path))
                q.append(neighbor)
    return 'NO'
print(bfs())
```

代码运行截图 （至少包含有"Accepted"）

T51.N皇后

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

思路:

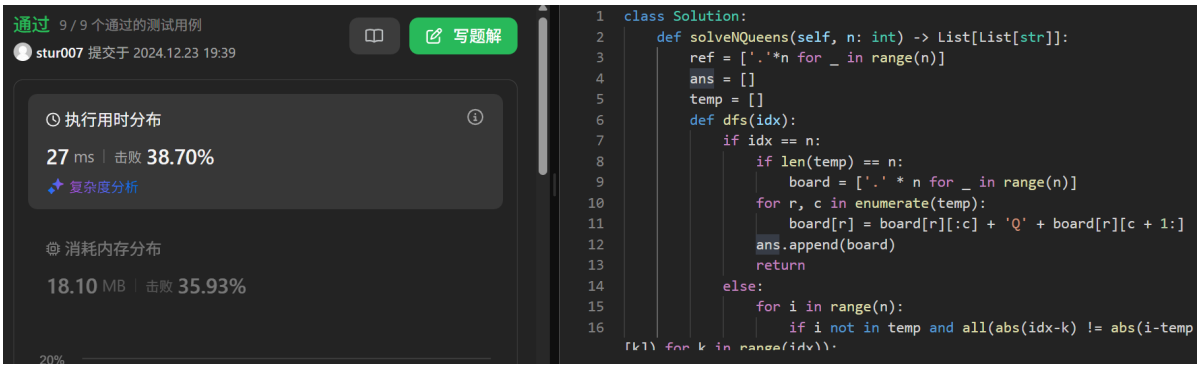
直接回溯即可。

代码:

```
class Solution:
    def solveNQueens(self, n: int) -> List[List[str]]:
        ref = ['.'*n for _ in range(n)]
        ans = []
        temp = []
        def dfs(idx):
            if idx == n:
                if len(temp) == n:
                    board = ['.' * n for _ in range(n)]
                for r, c in enumerate(temp):
                    board[r] = board[r][:c] + 'Q' + board[r][c + 1:]
                ans.append(board)
                return
            else:
                for i in range(n):
                    if i not in temp and all(abs(idx-k) != abs(i-temp[k]) for k
in range(idx)):
                        temp.append(i)
                        dfs(idx+1)
                        temp.pop()

        dfs(0)
        return ans
```

代码运行截图 （至少包含有"Accepted"）



2. 学习总结和收获

如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算2025spring每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。

感觉这一部分的题目并没有比树简单多少。。。