Assignment #A: Graph starts

Updated 1830 GMT+8 Apr 22, 2025

2025 spring, Complied by**袁奕 2400010766 数院**

> **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：

代码：

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

class Graph:
    def __init__(self):
        self.vertices = {}
    def add_edge(self, start, end):
        if start not in self.vertices:
            self.vertices[start] = Vertex(start)
        if end not in self.vertices:
            self.vertices[end] = Vertex(end)
        self.vertices[start].neighbour.add(end)
        self.vertices[end].neighbour.add(start)
```

```python

graph = Graph()
n, m = map(int, input().split())

for i in range(m):
    a, b = map(int, input().split())
    graph.add_edge(a, b)

for i in range(n):
    res = [0] * n
    for j in range(n):
        if i == j:
            res[j] = len(graph.vertices.get(i, Vertex(i)).neighbour)
        elif j in graph.vertices.get(i, Vertex(i)).neighbour:
            res[j] = -1
    print(*res, sep=" ")
```

#48997382提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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

class Graph:
    def __init__(self):
        self.vertices = {}
    def add_edge(self, start, end):
        if start not in self.vertices:
            self.vertices[start] = Vertex(start)
        if end not in self.vertices:
            self.vertices[end] = Vertex(end)
        self.vertices[start].neighbour.add(end)
        self.vertices[end].neighbour.add(start)

graph = Graph()
n, m = map(int, input().split())

for i in range(m):
    a, b = map(int, input().split())
    graph.add_edge(a, b)

for i in range(n):
    res = [0] * n
    for j in range(n):
        if i == j:
            res[j] = len(graph.vertices.get(i, Vertex(i)).neighbour)
        elif j in graph.vertices.get(i, Vertex(i)).neighbour:
            res[j] = -1
    print(*res, sep=" ")
```

基本信息

#: 48997382
题目: 19943
提交人: 24n2400010766
内存: 3664kB
时间: 25ms
语言: Python3
提交时间: 2025-04-23 22:08:19

English 帮助 关于

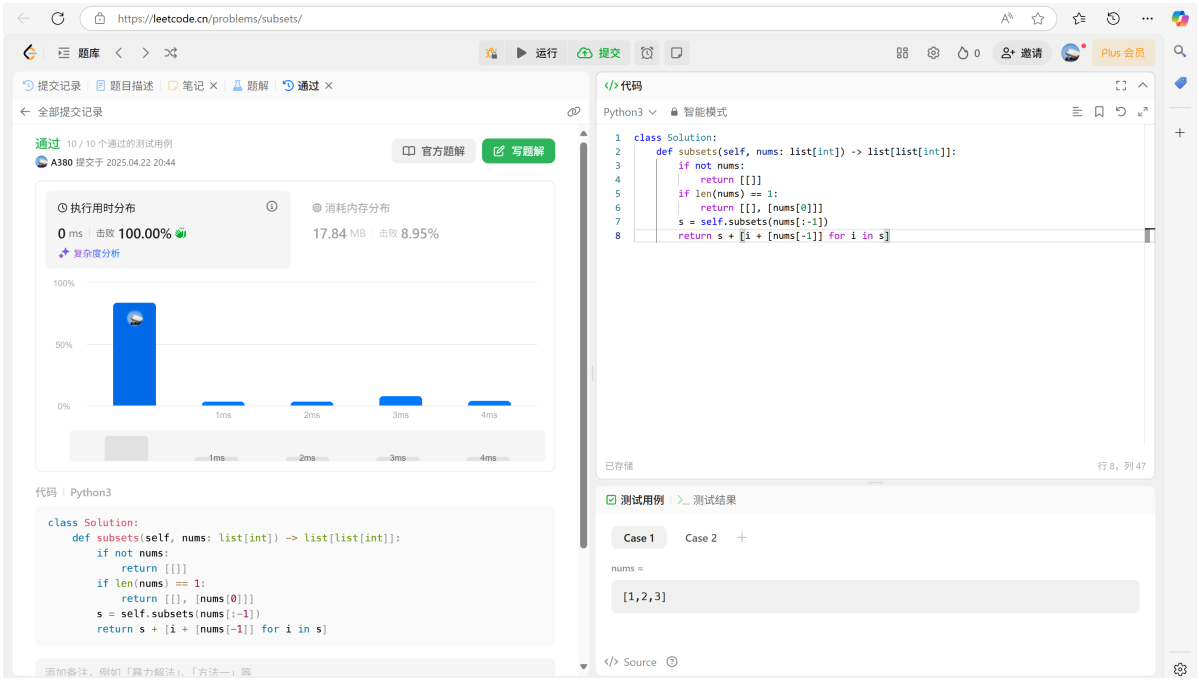
LC78.子集

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

思路:

代码:

```python
class Solution:
    def subsets(self, nums: list[int]) -> list[list[int]]:
        if not nums:
            return [[]]
        if len(nums) == 1:
            return [[], [nums[0]]]
        s = self.subsets(nums[:-1])
        return s + [i + [nums[-1]] for i in s]
```



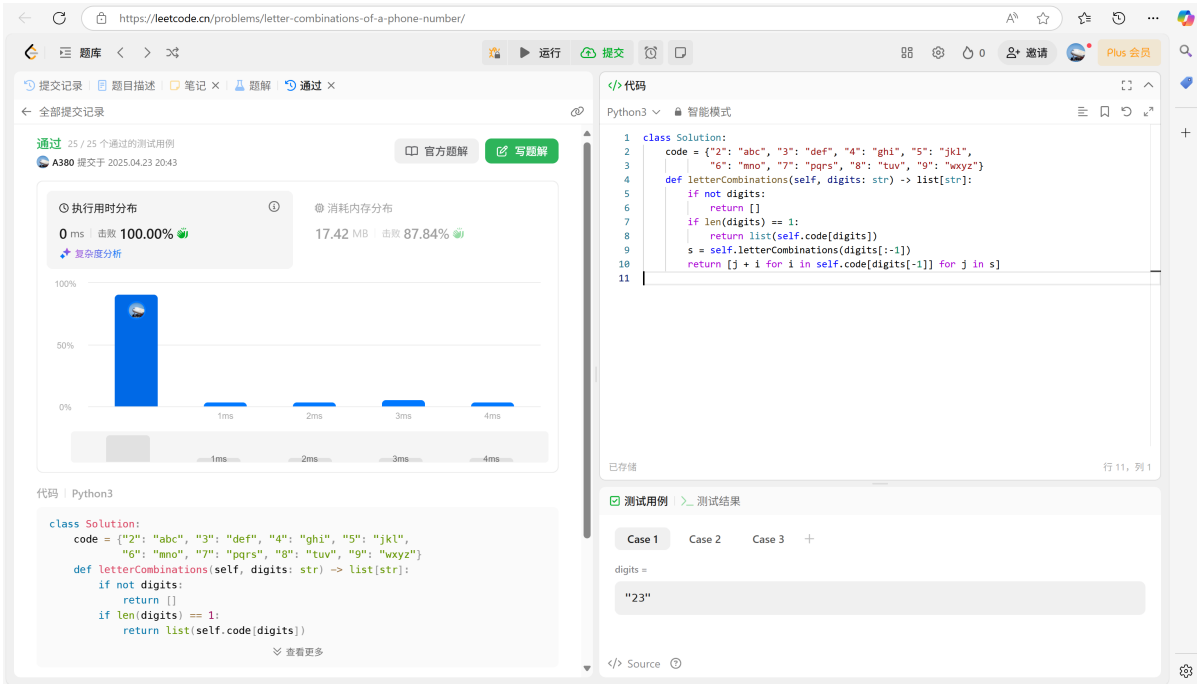
LC17.电话号码的字母组合

hash table, backtracking, https://leetcode.cn/problems/letter-combinations-of-a-phone-number/

思路：

代码：

```python
class Solution:
    code = {"2": "abc", "3": "def", "4": "ghi", "5": "jkl",
            "6": "mno", "7": "pqrs", "8": "tuv", "9": "wxyz"}
    def letterCombinations(self, digits: str) -> list[str]:
        if not digits:
            return []
        if len(digits) == 1:
            return list(self.code[digits])
        s = self.letterCombinations(digits[:-1])
        return [j + i for i in self.code[digits[-1]] for j in s]
```



M04089:电话号码

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

思路：

代码：

```
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.children = dict()

def build(s : str):
    curr = root
    isSame = True
    for c in s:
        if c in curr.children:
            curr = curr.children[c]
        else:
            curr.children[c] = TreeNode(c)
            curr = curr.children[c]
            isSame = False
    return isSame

T = int(input())
for _ in range(T):
    root = TreeNode(None)
    n = int(input())
    read = []
    for _ in range(n):
        read.append(input())
    read = sorted(read, key=len, reverse=True)
```

```
    havePrint = False
    for i in range(n):
        if build(read[i]):
            print("NO")
            havePrint = True
            break
    if not havePrint:
        print("YES")
```

#48987253提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.children = dict()

def build(s : str):
    curr = root
    isSame = True
    for c in s:
        if c in curr.children:
            curr = curr.children[c]
        else:
            curr.children[c] = TreeNode(c)
            curr = curr.children[c]
            isSame = False
    return isSame

T = int(input())
for _ in range(T):
    root = TreeNode(None)
    n = int(input())
    read = []
    for _ in range(n):
        read.append(input())
    read = sorted(read, key=len, reverse=True)
    havePrint = False
    for i in range(n):
        if build(read[i]):
            print("NO")
            havePrint = True
            break
    if not havePrint:
        print("YES")
```

基本信息

#: 48987253
题目: 04089
提交人: 24n2400010766
内存: 26604kB
时间: 350ms
语言: Python3
提交时间: 2025-04-22 21:17:19

English 帮助 关于

T28046:词梯

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

思路：两两逐位比较是 $O(n^2k)$ 复杂度, 会超时. 于是类似筛法, 用`buckets`存储.

代码:

```
from collections import deque

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

class Graph:
    def __init__(self):
        self.vertices = {}
```

```
    def add_edge(self, start : str, end : str):
        if start not in self.vertices:
            self.vertices[start] = Vertex(start)
        if end not in self.vertices:
            self.vertices[end] = Vertex(end)
        self.vertices[start].neighbour.add(end)

graph = Graph()
n = int(input())
buckets = {}
for _ in range(n):
    word = input()
    for i, _ in enumerate(word):
        bucket = f"{word[:i]}_{word[i + 1:]}"
        buckets.setdefault(bucket, set()).add(word)

for similar_words in buckets.values():
    for word1 in similar_words:
        for word2 in similar_words - {word1}:
            graph.add_edge(word1, word2)

start, end = input().split()

def main():
    que = deque([(graph.vertices.get(start, Vertex(start)), [start])])
    visited = {start}
    while que:
        last, path = que.popleft()
        if last.key == end:
            return path
        for next in last.neighbour:
            if next not in visited:
                que.append((graph.vertices[next], path + [next]))
                visited.add(next)
    return

res = main()
print(*res, sep=" ") if res else print("NO")
```

#48997799提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

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

class Graph:
    def __init__(self):
        self.vertices = {}
    def add_edge(self, start : str, end : str):
        if start not in self.vertices:
            self.vertices[start] = Vertex(start)
        if end not in self.vertices:
            self.vertices[end] = Vertex(end)
        self.vertices[start].neighbour.add(end)

graph = Graph()
n = int(input())
buckets = {}
for _ in range(n):
    word = input()
    for i, _ in enumerate(word):
        bucket = f"{word[:i]}_{word[i + 1:]}"
        buckets.setdefault(bucket, set()).add(word)

for similar_words in buckets.values():
    for word1 in similar_words:
        for word2 in similar_words - {word1}:
            graph.add_edge(word1, word2)

start, end = input().split()

def main():
    que = deque([(graph.vertices.get(start, Vertex(start)), [start])])
    visited = {start}
    while que:
        last, path = que.popleft()
        if last.key == end:
            return path
        for next in last.neighbour:
            if next not in visited:
                que.append((graph.vertices[next], path + [next]))
                visited.add(next)
    return

res = main()
print(*res, sep=" ") if res else print("NO")
```

基本信息

#:	48997799
题目:	28046
提交人:	24n2400010766
内存:	11296kB
时间:	77ms
语言:	Python3
提交时间:	2025-04-24 09:07:02

T51.N皇后

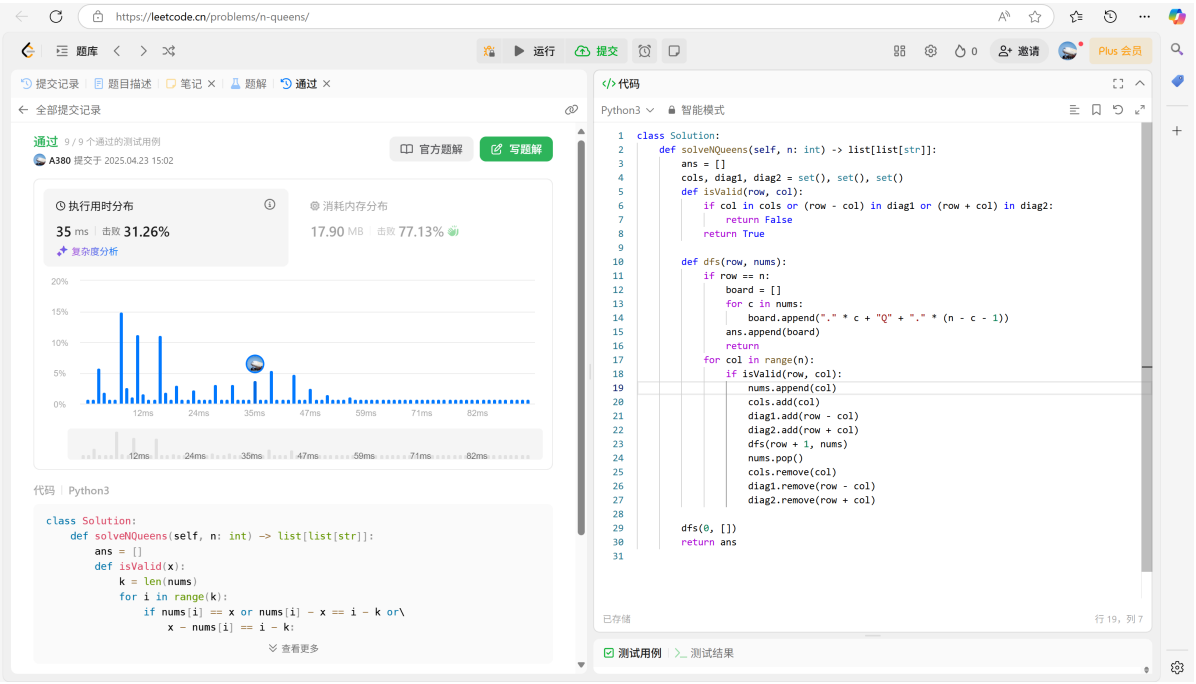
backtracking, https://leetcode.cn/problems/n-queens/

思路:

代码:

```
class Solution:
    def solveNQueens(self, n: int) -> list[list[str]]:
        ans = []
        def isValid(x):
            k = len(nums)
            for i in range(k):
                if nums[i] == x or nums[i] - x == i - k or\
                    x - nums[i] == i - k:
                    return False
```

```
            return True
        nums = []
        def dfs():
            if len(nums) == n:
                ans.append(["." * i + "Q" + "." * (n - i - 1) for i in
nums])
                return
            for x in range(n):
                if isValid(x):
                    nums.append(x)
                    dfs()
                    nums.pop()
        dfs()
        return ans
```



2. 学习总结和收获

学习了 `dict.get()` 和 `dict.setdefault()` 方法. 更安全, 不会有 `KeyError` 报错.

LeetCode 链表和二叉树还差5题, 周末做完

链表

相交链表	简单
反转链表	简单
回文链表	简单
环形链表	简单
环形链表 II	中等
合并两个有序链表	简单
两数相加	中等
删除链表的倒数第 N 个结点	中等
两两交换链表中的节点	中等
K 个一组翻转链表	困难
随机链表的复制	中等
排序链表	中等
合并 K 个升序链表	困难
LRU 缓存	中等

二叉树

二叉树的中序遍历	简单
二叉树的最大深度	简单
翻转二叉树	简单
对称二叉树	简单
二叉树的直径	简单
二叉树的层序遍历	中等
将有序数组转换为二叉搜索树	简单
验证二叉搜索树	中等
二叉搜索树中第 K 小的元素	中等
二叉树的右视图	中等
二叉树展开为链表	中等
从前序与中序遍历序列构造二叉树	中等
路径总和 III	中等
二叉树的最近公共祖先	中等
二叉树中的最大路径和	困难