

# Assignment #A: Graph starts

Updated 1830 GMT+8 Apr 22, 2025

2025 spring, Compiled 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:

```
1 class Vertex:
2     def __init__(self, key):
3         self.key = key
4         self.neighbour = set()
5
6 class Graph:
7     def __init__(self):
8         self.vertices = {}
9     def add_edge(self, start, end):
10        if start not in self.vertices:
11            self.vertices[start] = Vertex(start)
12        if end not in self.vertices:
13            self.vertices[end] = Vertex(end)
14        self.vertices[start].neighbour.add(end)
15        self.vertices[end].neighbour.add(start)
```

```

16
17 graph = Graph()
18 n, m = map(int, input().split())
19
20 for i in range(m):
21     a, b = map(int, input().split())
22     graph.add_edge(a, b)
23
24 for i in range(n):
25     res = [0] * n
26     for j in range(n):
27         if i == j:
28             res[j] = len(graph.vertices.get(i, Vertex(i)).neighbour)
29         elif j in graph.vertices.get(i, Vertex(i)).neighbour:
30             res[j] = -1
31     print(*res, sep=" ")

```

#48997382提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```

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)

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

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

## LC78.子集

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

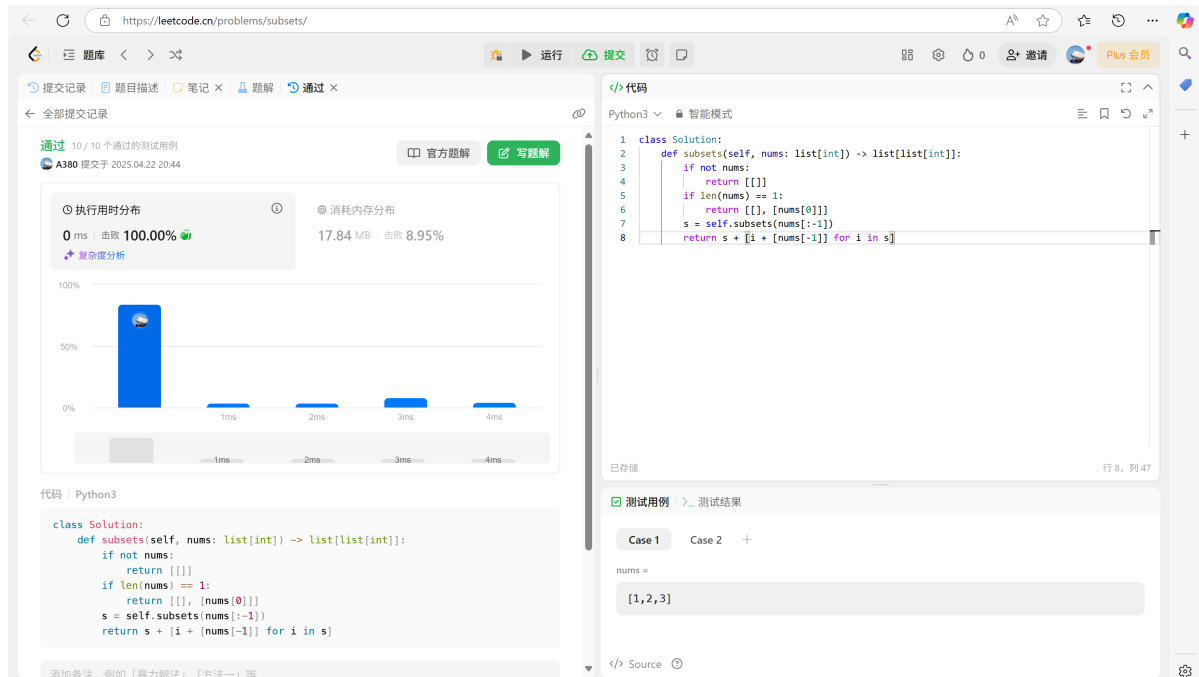
思路:

代码:

```

1 class Solution:
2     def subsets(self, nums: list[int]) -> list[list[int]]:
3         if not nums:
4             return [[]]
5         if len(nums) == 1:
6             return [[], [nums[0]]]
7         s = self.subsets(nums[:-1])
8         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:

```

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

```

通过 25 / 25 个通过的测试用例  
A380 提交于 2025.04.23 20:43

官方题解 写题解

执行用时分布 0 ms 击败 100.00%  
消耗内存分布 17.42 MB 击败 87.84%

复杂度分析

```
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[0]] for j in s]
```

已存储 行 11, 列 1

测试用例 测试结果

Case 1 Case 2 Case 3 +

digits =  
"23"

</> Source

## M04089:电话号码

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

思路:

代码:

```
1 class TreeNode:
2     def __init__(self, val):
3         self.val = val
4         self.children = dict()
5
6 def build(s : str):
7     curr = root
8     isSame = True
9     for c in s:
10         if c in curr.children:
11             curr = curr.children[c]
12         else:
13             curr.children[c] = TreeNode(c)
14             curr = curr.children[c]
15             isSame = False
16     return isSame
17
18 T = int(input())
19 for _ in range(T):
20     root = TreeNode(None)
21     n = int(input())
22     read = []
23     for _ in range(n):
24         read.append(input())
25     read = sorted(read, key=len, reverse=True)
```

```

26     havePrint = False
27     for i in range(n):
28         if build(read[i]):
29             print("NO")
30             havePrint = True
31             break
32     if not havePrint:
33         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

©2002-2022 POJ 京ICP备20010980号-1

[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:

```

1  from collections import deque
2
3  class Vertex:
4      def __init__(self, key):
5          self.key = key
6          self.neighbour = set()
7
8  class Graph:
9      def __init__(self):
10         self.vertices = {}

```

```

11     def add_edge(self, start : str, end : str):
12         if start not in self.vertices:
13             self.vertices[start] = Vertex(start)
14         if end not in self.vertices:
15             self.vertices[end] = Vertex(end)
16         self.vertices[start].neighbour.add(end)
17
18 graph = Graph()
19 n = int(input())
20 buckets = {}
21 for _ in range(n):
22     word = input()
23     for i, _ in enumerate(word):
24         bucket = f"{word[:i]}_{word[i + 1:]}"
25         buckets.setdefault(bucket, set()).add(word)
26
27 for similar_words in buckets.values():
28     for word1 in similar_words:
29         for word2 in similar_words - {word1}:
30             graph.add_edge(word1, word2)
31
32 start, end = input().split()
33
34 def main():
35     que = deque([(graph.vertices.get(start, Vertex(start)), [start])])
36     visited = {start}
37     while que:
38         last, path = que.popleft()
39         if last.key == end:
40             return path
41         for next in last.neighbour:
42             if next not in visited:
43                 que.append((graph.vertices[next], path + [next]))
44                 visited.add(next)
45     return
46
47 res = main()
48 print(*res, sep=" ") if res else print("NO")

```

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:

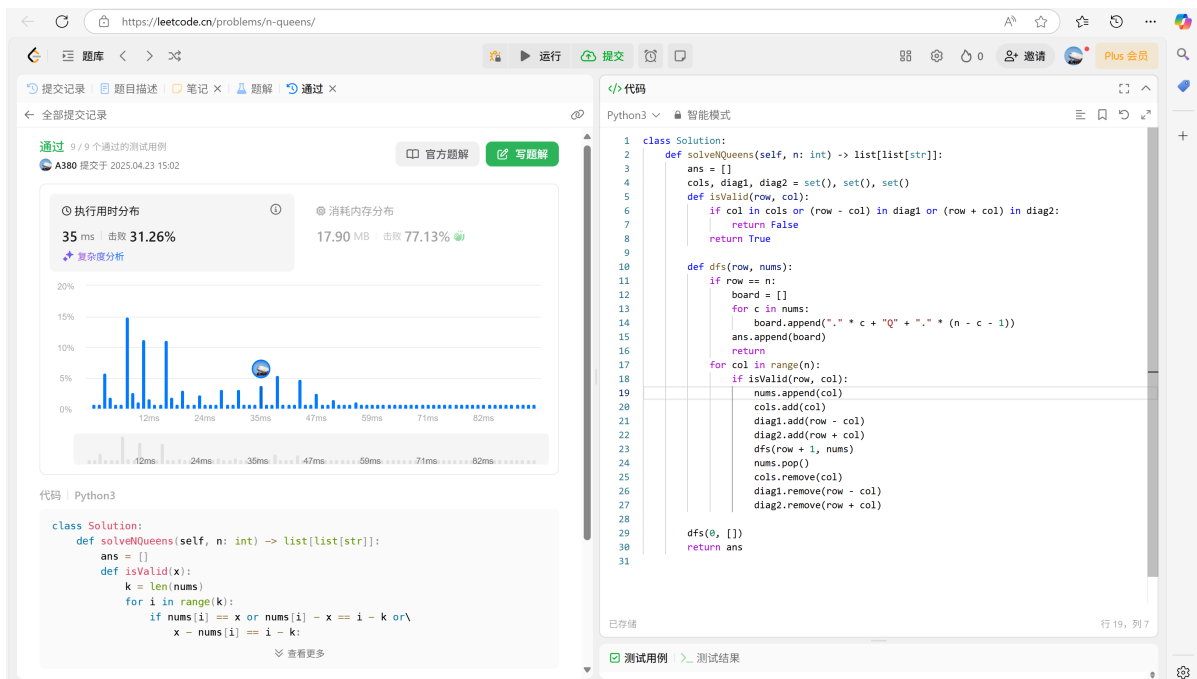
代码:

```
1 class Solution:
2     def solvenQueens(self, n: int) -> list[list[str]]:
3         ans = []
4         def isValid(x):
5             k = len(nums)
6             for i in range(k):
7                 if nums[i] == x or nums[i] - x == i - k or \
8                     x - nums[i] == i - k:
9                     return False
```

```

10         return True
11     nums = []
12     def dfs():
13         if len(nums) == n:
14             ans.append(["." * i + "Q" + "." * (n - i - 1) for i in
nums])
15         return
16     for x in range(n):
17         if isValid(x):
18             nums.append(x)
19             dfs()
20             nums.pop()
21     dfs()
22     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		
<input checked="" type="checkbox"/>	相交链表	简单
<input checked="" type="checkbox"/>	反转链表	简单
<input checked="" type="checkbox"/>	回文链表	简单
<input checked="" type="checkbox"/>	环形链表	简单
<input checked="" type="checkbox"/>	环形链表 II	中等
<input checked="" type="checkbox"/>	合并两个有序链表	简单
<input type="checkbox"/>	两数相加	中等
<input checked="" type="checkbox"/>	删除链表的倒数第 N 个结点	中等
<input checked="" type="checkbox"/>	两两交换链表中的节点	中等
<input checked="" type="checkbox"/>	K 个一组翻转链表	困难
<input type="checkbox"/>	随机链表的复制	中等
<input checked="" type="checkbox"/>	排序链表	中等
<input checked="" type="checkbox"/>	合并 K 个升序链表	困难
<input checked="" type="checkbox"/>	LRU 缓存	中等
二叉树		
<input checked="" type="checkbox"/>	二叉树的中序遍历	简单
<input checked="" type="checkbox"/>	二叉树的最大深度	简单
<input checked="" type="checkbox"/>	翻转二叉树	简单
<input checked="" type="checkbox"/>	对称二叉树	简单
<input checked="" type="checkbox"/>	二叉树的直径	简单
<input checked="" type="checkbox"/>	二叉树的层序遍历	中等
<input checked="" type="checkbox"/>	将有序数组转换为二叉搜索树	简单
<input checked="" type="checkbox"/>	验证二叉搜索树	中等
<input checked="" type="checkbox"/>	二叉搜索树中第 K 小的元素	中等
<input type="checkbox"/>	二叉树的右视图	中等
<input checked="" type="checkbox"/>	二叉树展开为链表	中等
<input checked="" type="checkbox"/>	从前序与中序遍历序列构造二叉树	中等
<input type="checkbox"/>	路径总和 III	中等
<input type="checkbox"/>	二叉树的最近公共祖先	中等
<input checked="" type="checkbox"/>	二叉树中的最大路径和	困难

