Assignment #A: Graph starts

Updated 1830 GMT+8 Apr 22, 2025

2025 spring, Complied by 同学的姓名、院系

> **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.neighbors={}
    def get_neighbor(self,other):
        return self.neighbors.get(other)
    def set_neighbor(self,other,weight=0):
        self.neighbors[other]=weight
    def get_neighbors(self):
        return self.neighbors.keys()
    def get_key(self):
        return self.key
class Graph:
```

```python
    def __init__(self):
        self.vertlist={}
    def addvertex(self,key):
        newvert=vertex(key)
        self.vertlist[key]=newvert
        return newvert
    def getvert(self,n):
        return self.vertlist.get(n)
    def addedge(self,a,b,weight=0):
        if a not in self.vertlist:
            new=self.addvertex(a)
        if b not in self.vertlist:
            new=self.addvertex(b)
        self.vertlist[a].set_neighbor(self.vertlist[b],weight)
    def __iter__(self):
        return iter(self.vertlist.values())

n,m=[int(x) for x in input().split()]
edges=[]
for _ in range(m):
    edges.append([int(x) for x in input().split()])
graph=Graph()
for i in range(n):
    graph.addvertex(i)
for edge in edges:
    a,b=edge
    graph.addedge(a,b)
    graph.addedge(b,a)
ans=[]
for vert in graph:
    row=[0]*n
    row[vert.get_key()]=len(vert.get_neighbors())
    for neighbor in vert.get_neighbors():
        row[neighbor.get_key()]=-1
    ans.append(row)
for i in ans:
    print(*i)
```

代码运行截图（至少包含有"Accepted"）

状态: Accepted

源代码

```
class vertex:
    def __init__(self,key):
        self.key=key
        self.neighbors={}
    def get_neighbor(self,other):
        return self.neighbors.get(other)
    def set_neighbor(self,other,weight=0):
        self.neighbors[other]=weight
    def get_neighbors(self):
        return self.neighbors.keys()
    def get_key(self):
        return self.key
class Graph:
    def __init__(self):
        self.vertlist={}
    def addvertex(self,key):
        newvert=vertex(key)
```

LC78.子集

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

思路:

代码:

```
class Solution(object):
    def subsets(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        ans=[]
        def backtrack(st,record):
            ans.append(record)
            for i in range(st,len(nums)):
                backtrack(i+1,record+[nums[i]])
        backtrack(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：

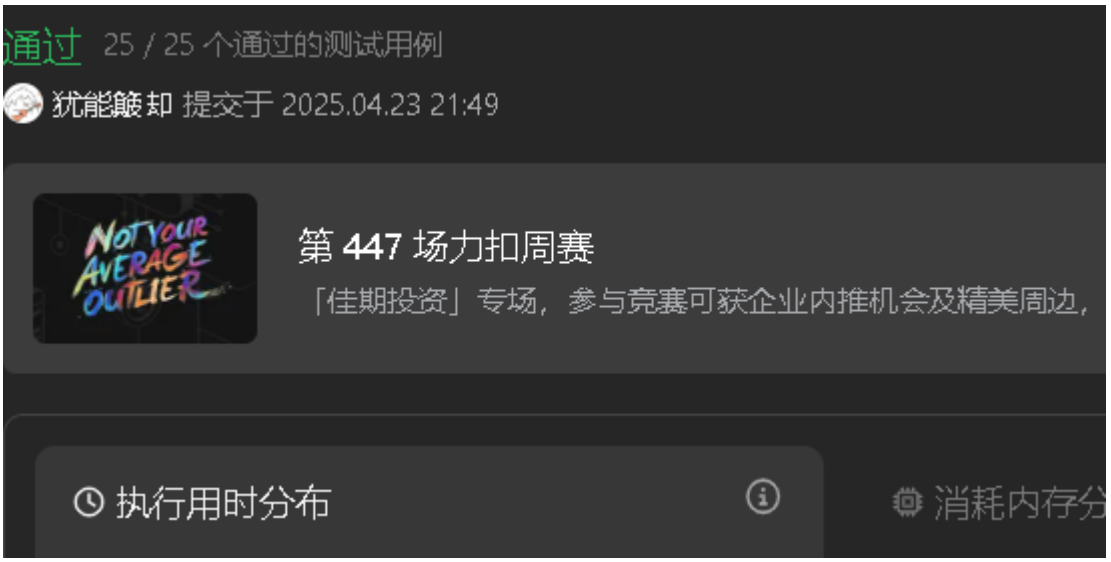
代码：

```
import sys
sys.setrecursionlimit(1<<30)

class Solution(object):
    def letterCombinations(self, digits):
        """
        :type digits: str
        :rtype: List[str]
        """
        dic=
{'1':'','2':'abc','3':'def','4':'ghi','5':'jkl','6':'mno','7':'pqrs','8':'tuv','9':'wxyz'}
        ans=set()
        record=[0]*len(digits)
        def backtrack(record):
            string=''
            for i in range(len(digits)):
                if record[i]<len(dic[digits[i]]):
                    string+=dic[digits[i]][record[i]]
            if string:
                ans.add(string)
            for i in range(len(digits)):
                if record[i]<len(dic[digits[i]])-1:
                    newre=record[:]
                    newre[i]+=1
                    backtrack(newre)
        backtrack(record)
```

```
        return list(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：

代码：

```
class TrieNode:
    def __init__(self):
        self.children={}
        self.is_end=False
class Trie:
    def __init__(self):
        self.root=TrieNode()
    def insert(self,number):
        node=self.root
        for i in number:
            if i not in node.children:
                node.children[i]=TrieNode()
            node=node.children[i]
            if node.is_end:
                return False
        node.is_end=True
        return len(node.children)==0
    def is_consistent(self,numbers):
        for number in numbers:
            if not self.insert(number):
                return False
```

```
        return True

t=int(input())
for _ in range(t):
    n=int(input())
    numbers=[]
    for __ in range(n):
        numbers.append(input())
    numbers.sort(key=lambda x:len(x))
    trie=Trie()
    if trie.is_consistent(numbers):
        print('YES')
    else:
        print('NO')
```

代码运行截图（至少包含有"Accepted"）

T28046:词梯

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

思路:

代码:

```
from collections import deque,defaultdict
def Graph(words):
    graph={i:[] for i in words}
    map=defaultdict(list)
    for word in words:
        for i in range(4):
            record=word[:i]+'*'+word[i+1:]
            map[record].append(word)
    for word in words:
        for i in range(4):
            record=word[:i]+'*'+word[i+1:]
            for a in map[record]:
                if a!=word:
                    graph[word].append(a)
    return graph

def bfs(st,ed,path):
    q=deque()
    inq=set()
    q.append((st,path))
    inq.add(st)
    while q:
```

```
        word,path=q.popleft()
        if word==ed:
            return path
        lst=graph[word]
        for i in lst:
            if i not in inq:
                newpath=path+[i]
                q.append((i,newpath))
                inq.add(i)
    return None

n=int(input())
words=[]
for _ in range(n):
    words.append(input())
st,ed=[x for x in input().split()]
graph=Graph(words)
path=bfs(st,ed,[st])
if path:
    print(*path)
else:
    print('NO')
```

代码运行截图（至少包含有"Accepted"）

状态：Accepted

源代码

```
from collections import deque,defaultdict
def Graph(words):
    graph={i:[] for i in words}
    map=defaultdict(list)
    for word in words:
        for i in range(4):
            record=word[:i]+'*'+word[i+1:]
            map[record].append(word)
    for word in words:
        for i in range(4):
            record=word[:i]+'*'+word[i+1:]
            for a in map[record]:
```

T51.N皇后

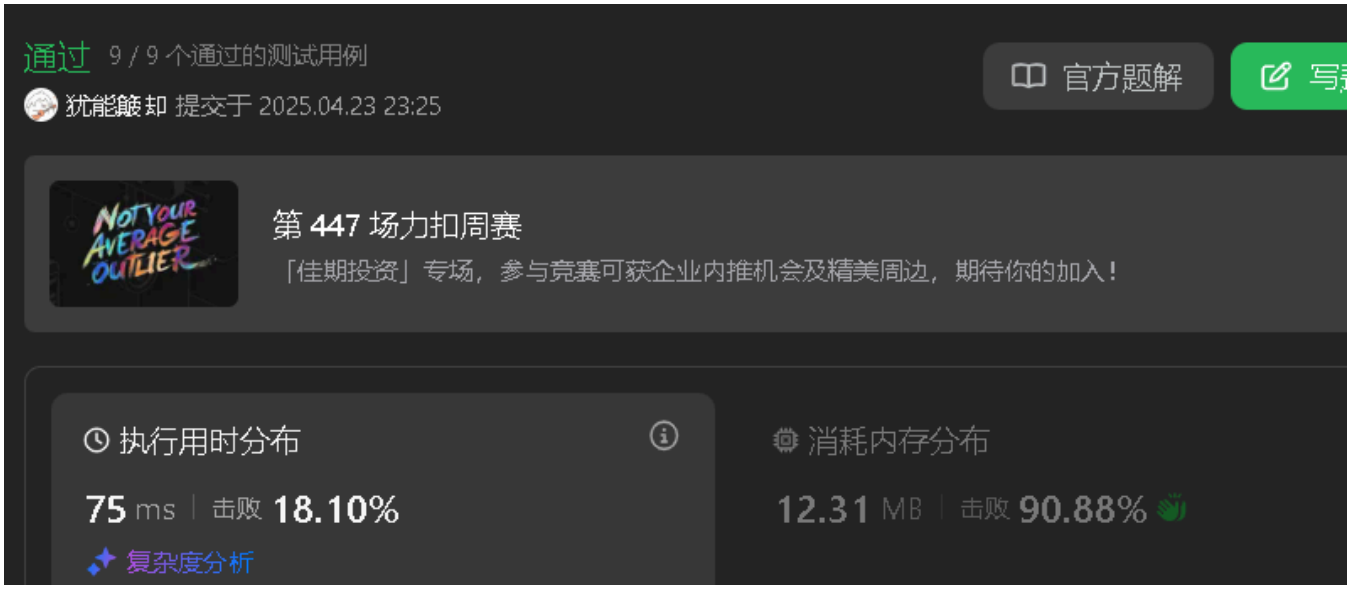
backtracking, https://leetcode.cn/problems/n-queens/

思路:

代码:

```
import sys
sys.setrecursionlimit(1<<30)
class Solution(object):
    def solveNQueens(self, n):
        """
        :type n: int
        :rtype: List[List[str]]
        """
        arr=[]
        def dfs(record,position):
            if len(record)==n:
                #print(record)
                arr.append(record)
                return
            lenth=len(record)
            for i in range(n):
                newrecord,newposition=record[:],position[:]
                newrecord.append('.'*i+'Q'+'.'*(n-1-i))
                newposition.append([lenth,i])
                for j in newposition[:-1]:
                    if j[1]==i or abs(j[0]-lenth)==abs(j[1]-i):
                        break
                else:
                    dfs(newrecord,newposition)
        dfs([],[])
        return arr
```

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。

怎么期中周还没有结束.....

这周的课没有听，打算周末好好恶补一下课件，不知不觉每日选做又落下十几天了啊啊，感觉根本跟不上进度啊