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：根据课上的标准做法处理

代码：

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

    def get_neighbor(self, other):
        return self.neighbors.get(other, None)

    def set_neighbor(self, other, weight=0):
        self.neighbors[other] = weight

    def __repr__(self):  # 为开发者提供调试信息
        return f"Vertex({self.key})"
```

```python
    def __str__(self):  # 面向用户的输出
        return (
                str(self.key)
                + " connected to: "
                + str([x.key for x in self.neighbors])
        )

    def get_neighbors(self):
        return self.neighbors.keys()

    def get_key(self):
        return self.key
class Graph:
    def __init__(self):
        self.vertices = {}

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

    def get_vertex(self, key):
        return self.vertices.get(key, None)

    def __contains__(self, key):
        return key in self.vertices

    def add_edge(self, from_vert, to_vert, weight=0):
        if from_vert not in self.vertices:
            self.set_vertex(from_vert)
        if to_vert not in self.vertices:
            self.set_vertex(to_vert)
        self.vertices[from_vert].set_neighbor(self.vertices[to_vert], weight)

    def get_vertices(self):
        return self.vertices.keys()

    def __iter__(self):
        return iter(self.vertices.values())
n,m=map(int,input().split())
g = Graph()
for i in range(n):
    g.set_vertex(i)
for _ in range(m):
    x,y = map(int,input().split())
    g.add_edge(x,y,1)
    g.add_edge(y,x,1)
for v in g:
    row = [0]*n
    row[v.get_key()]=len(v.get_neighbors())
    for index in v.get_neighbors():
        row[index.get_key()]=-1
    print(*row)
```

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, None)
```

基本信息

#:	49007972
题目:	19943
提交人:	24n2400011481
内存:	3684kB
时间:	20ms
语言:	Python3
提交时间:	2025-04-25 12:09:08

LC78.子集

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

思路：用位运算枚举处理

代码：

```
class Solution:
    def subsets(self, nums: List[int]) -> List[List[int]]:
        n = len(nums)
        a = []
        for i in range(2**n):
            a.append([])
            for j in range(n):
                if i>>j & 1:
                    a[-1].append(nums[j])
        return a
```

代码运行截图（至少包含有"Accepted"）

通过 10 / 10 个通过的测试用例

Hungry NorthcuttqqE 提交于 2025.04.25 12:20

官方题解 写题解

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：dfs回溯处理

代码：

```
class Solution:
    def letterCombinations(self, digits: str) -> List[str]:
        if not digits:
            return []
        dic = {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']}
        a = []
        n = len(digits)
        c = [int(index) for index in digits]
        def f(temp,i):
            if i==n:
                a.append(temp)
                return
            b = dic[c[i]]
            for index in b:
                f(temp+index,i+1)
        f('',0)
        return a
```

代码运行截图（至少包含有"Accepted"）

通过 25 / 25 个通过的测试用例

Hungry NorthcuttqqE 提交于 2025.04.25 12:31

官方题解 写题解

M04089:电话号码

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

思路：按照讲义上的方式处理即可

代码：

```
class Node:
    def __init__(self):
        self.child = {}


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

    def insert(self,nums):
        new_node = self.root
        for index in nums:
            if index not in new_node.child:
                new_node.child[index]=Node()
            new_node = new_node.child[index]
```

```python
    def search(self,nums):
        new_node = self.root
        for index in nums:
            if index not in new_node.child:
                return 0
            new_node = new_node.child[index]
        return 1
t = int(input())
for _ in range(t):
    n = int(input())
    nums = [input() for _ in range(n)]
    nums.sort(reverse=True)
    s = 0
    tri = Tri()
    for index in nums:
        s += tri.search(index)
        tri.insert(index)
    if s>0:
        print('NO')
    else:
        print('YES')
```

代码运行截图（至少包含有"Accepted"）

状态：Accepted

源代码

```python
class Node:
    def __init__(self):
        self.child = {}


class Tri:
    def __init__(self):
        self.root = Node()
```

基本信息

#:	49007761
题目:	04089
提交人:	24n2400011481
内存:	24752kB
时间:	379ms
语言:	Python3
提交时间:	2025-04-25 11:10:42

T28046:词梯

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

思路：按照课上讲的思路处理即可

代码：

```python
class Graph:
    def __init__(self):
        self.vertices = {}
        self.num_vertices = 0

    def add_vertex(self, key):
```

```
        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(key)
        self.vertices[key] = new_vertex
        return new_vertex

    def get_vertex(self, n):
        if n in self.vertices:
            return self.vertices[n]
        else:
            return None

    def __len__(self):
        return self.num_vertices

    def __contains__(self, n):
        return n in self.vertices

    def add_edge(self, f, t, cost=0):
        if f not in self.vertices:
            nv = self.add_vertex(f)
        if t not in self.vertices:
            nv = self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t], cost)

    def get_vertices(self):
        return list(self.vertices.keys())

    def __iter__(self):
        return iter(self.vertices.values())
class Vertex:
    def __init__(self, num):
        self.key = num
        self.connectedTo = {}
        self.color = 'white'
        self.distance = sys.maxsize
        self.previous = None

    def add_neighbor(self, nbr, weight=0):
        self.connectedTo[nbr] = weight

    def get_neighbors(self):
        return self.connectedTo.keys()

from collections import deque
import sys
n = int(input())
s = [input().strip() for _ in range(n)]
x,y  = input().split()
d  = dict()
g = Graph()
for index in s:
    for j in range(4):
        temp = f'{index[:j]}_{index[j+1:]}'
```

```
        d.setdefault(temp,set([]))
        d[temp].add(index)
for j in d.values():
    for word1 in j:
        for word2 in j-{word1}:
            g.add_edge(word1,word2,1)
if x not in g.vertices or y not in g.vertices:
    print('NO')
    exit()
a = deque([g.vertices[x]])
ans = None
while a:
    v = a.popleft()
    if v.key ==y:
        ans = v
        break
    b = list(v.get_neighbors())
    for index in b:
        if index.color=='white':
          index.color = 'black'
          index.pre = v
          a.append(index)
if not ans:
    print('NO')
    exit()
cur = ans
l = []
while cur.key!=x:
    l.append(cur.key)
    cur = cur.pre
else:
    l.append(x)
print(*l[::-1])
```

代码运行截图 （至少包含有"Accepted"）

状态：Accepted

源代码

```
class Graph:
    def __init__(self):
        self.vertices = {}
        self.num_vertices = 0

    def add_vertex(self, key):
        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(key)
        self.vertices[key] = new_vertex
        return new_vertex
```

基本信息

#:	49029414
题目:	28046
提交人:	24n2400011481
内存:	10580kB
时间:	86ms
语言:	Python3
提交时间:	2025-04-28 16:03:55

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]]:
        a = []
        b = set([])
        c = set([])
        d = set([])
        temp = []
        def f(i):
            if i==n:
                a.append(temp[:])
                return
            for j in range(n):
                if j not in b and j-i not in c and j+i not in d:
                    l,r = j-i,j+i
                    b.add(j)
                    c.add(l)
                    d.add(r)
                    temp.append(j)
                    f(i+1)
                    temp.pop()
                    b.remove(j)
                    c.remove(l)
                    d.remove(r)
        f(0)
        b = []
        for index in a:
            s = []
            for j in index:
                s.append('.'*j+'Q'+'.'*(n-j-1))
            b.append(s)
        return b
```

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。

之前缺的部分都补上了，每日选做在跟进，感觉还是不够熟悉图，每次都要看原始定义