Assignment #A: Graph starts

Updated 1830 GMT+8 Apr 22, 2025

2025 spring, Complied by 蔡沐轩 数学科学学院

> **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：

建图存储，再转成矩阵输出。约20min。

代码：

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

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

    def set_vertex(self, key):
        self.vertices[key] = Vertex(key)
    def add_edge(self,u,v):
        if u not in self.vertices:self.set_vertex(u)
        if v not in self.vertices:self.set_vertex(v)
        self.vertices[u].neighbors.append(self.vertices[v])
```

```
        self.vertices[v].neighbors.append(self.vertices[u])

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

for _ in range(m):
    u,v=map(int,input().split())
    g.add_edge(u,v)
for i in range(n):
    if i not in g.vertices:g.set_vertex(i)
l=[[0]*n for _ in range(n)]
for i in range(n):l[i][i]=len(g.vertices[i].neighbors)
for i in range(n):
    for ver in g.vertices[i].neighbors:l[i][ver.key]=-1
print('\n'.join(map(lambda x:' '.join(map(str,x)),l)))
```

这题本质上是邻接矩阵，建类的方式类似邻接表，有点多此一举了。直接用邻接矩阵写要轻松得多。

```
n,m=map(int,input().split())
l=[[0]*n for _ in range(n)]
for _ in range(m):
    u,v=map(int,input().split())
    l[u][v]-=1;l[v][u]-=1;l[u][u]+=1;l[v][v]+=1
print('\n'.join(map(lambda x:' '.join(map(str,x)),l)))
```

代码运行截图 （至少包含有"Accepted"）

#48987729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.neighbors=[]

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

    def set_vertex(self, key):
        self.vertices[key] = Vertex(key)
    def add_edge(self,u,v):
        if u not in self.vertices:self.set_vertex(u)
        if v not in self.vertices:self.set_vertex(v)
        self.vertices[u].neighbors.append(self.vertices[v])
        self.vertices[v].neighbors.append(self.vertices[u])

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

for _ in range(m):
    u,v=map(int,input().split())
    g.add_edge(u,v)
for i in range(n):
    if i not in g.vertices:g.set_vertex(i)
l=[[0]*n for _ in range(n)]
for i in range(n):l[i][i]=len(g.vertices[i].neighbors)
for i in range(n):
    for ver in g.vertices[i].neighbors:l[i][ver.key]=-1
print('\n'.join(map(lambda x:' '.join(map(str,x)),l)))
```

基本信息

#: 48987729
题目: 19943
提交人: 24n2400010617
内存: 3668kB
时间: 21ms
语言: Python3
提交时间: 2025-04-22 21:57:55

LC78.子集

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

思路：

递归判断每个元素是否加入子集即可。约5min。

代码：

```python
class Solution:
    def subsets(self, nums: List[int]) -> List[List[int]]:
        def sets(i:int):
            if i==-1:return [[]]
            temp=sets(i-1)
            return temp+list(map(lambda x:x+[nums[i]],temp))
        return sets(len(nums)-1)
```

代码运行截图（至少包含有"Accepted"）

代码 | Python3

```python
class Solution:
    def subsets(self, nums: List[int]) -> List[List[int]]:
        def sets(i:int):
            if i==-1:return [[]]
            temp=sets(i-1)
            return temp+list(map(lambda x:x+[nums[i]],temp))
        return sets(len(nums)-1)
```

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：

建好字典存储数字对应的字母，然后递归枚举即可。约10min。

代码：

```
class Solution:
    def letterCombinations(self, digits: str) -> List[str]:
        if not digits:return []
        letters=[[],[],['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'h', 'i'], ['j',
'k', 'l'], ['m', 'n', 'o'], ['p', 'q', 'r', 's'], ['t', 'u', 'v'], ['w', 'x',
'y', 'z']]
        def sol(i:int):
            if i==-1:return [""]
            ans=[]
            temp=sol(i-1)
            for letter in letters[int(digits[i])]:
                ans+=list(map(lambda x:x+letter,temp))
            return ans
        return sol(len(digits)-1)
```

代码运行截图 （至少包含有"Accepted"）

代码 | Python3

```
class Solution:
    def letterCombinations(self, digits: str) -> List[str]:
        if not digits:return []
        letters=[[],[],['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'h', 'i'], ['j', '
        def sol(i:int):
            if i==-1:return [""]
            ans=[]
            temp=sol(i-1)
            for letter in letters[int(digits[i])]:
                ans+=list(map(lambda x:x+letter,temp))
            return ans
        return sol(len(digits)-1)
```

收起

M04089:电话号码

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

思路:

用字典套字典的方式建立Trie树，字符串末尾用`'*'`标记，每次输入时一边存储，一边检测其前缀是否为之前字符串，最后检验其能否作为之前字符串的前缀，按照判断结果输出即可。约10min。

代码:

```
for _ in range(int(input())):
    trie={}
    flag=True
    for __ in range(int(input())):
        s=input()
        if flag:
            cur=trie
            for digit in s:
                if '*' in cur:
                    flag=False
```

```
                    break
                if digit not in cur:cur[digit]={}
                cur=cur[digit]
            if cur:flag = False
            cur['*']={}
    print('YES' if flag else '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：

将有三个字母确定的单词存到固定的桶中，进行BFS时只要将同一个桶中未入队的单词入队即可。对每个单词存储其BFS过程中的“父节点”，最后逆序找出路径输出即可。约10min。

代码：

```
from collections import defaultdict,deque

buckets=defaultdict(list)
for _ in range(int(input())):
    word=input()
    for k in range(4):
        buckets[word[:k]+' '+word[k+1:]].append(word)
x,y=input().split()
father={x:x}
q=deque([x])
while q:
    word=q.popleft()
    if word==y:break
    for k in range(4):
        for i in buckets[word[:k]+' '+word[k+1:]]:
            if i not in father:
```

```
                q.append(i)
                father[i]=word
if word==y:
    ans=[y]
    while y!=x:
        y=father[y]
        ans.append(y)
    print(' '.join(reversed(ans)))
else:print('NO')
```

代码运行截图 （至少包含有"Accepted"）

#48987107提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import defaultdict,deque

buckets=defaultdict(list)
for _ in range(int(input())):
    word=input()
    for k in range(4):
        buckets[word[:k]+' '+word[k+1:]].append(word)
x,y=input().split()
father={x:x}
q=deque([x])
while q:
    word=q.popleft()
    if word==y:break
    for k in range(4):
        for i in buckets[word[:k]+' '+word[k+1:]]:
            if i not in father:
                q.append(i)
                father[i]=word
if word==y:
    ans=[y]
    while y!=x:
        y=father[y]
        ans.append(y)
    print(' '.join(reversed(ans)))
else:print('NO')
```

基本信息

#: 48987107
题目: 28046
提交人: 24n2400010617
内存: 5564kB
时间: 46ms
语言: Python3
提交时间: 2025-04-22 21:04:04

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

T51.N皇后

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

思路:

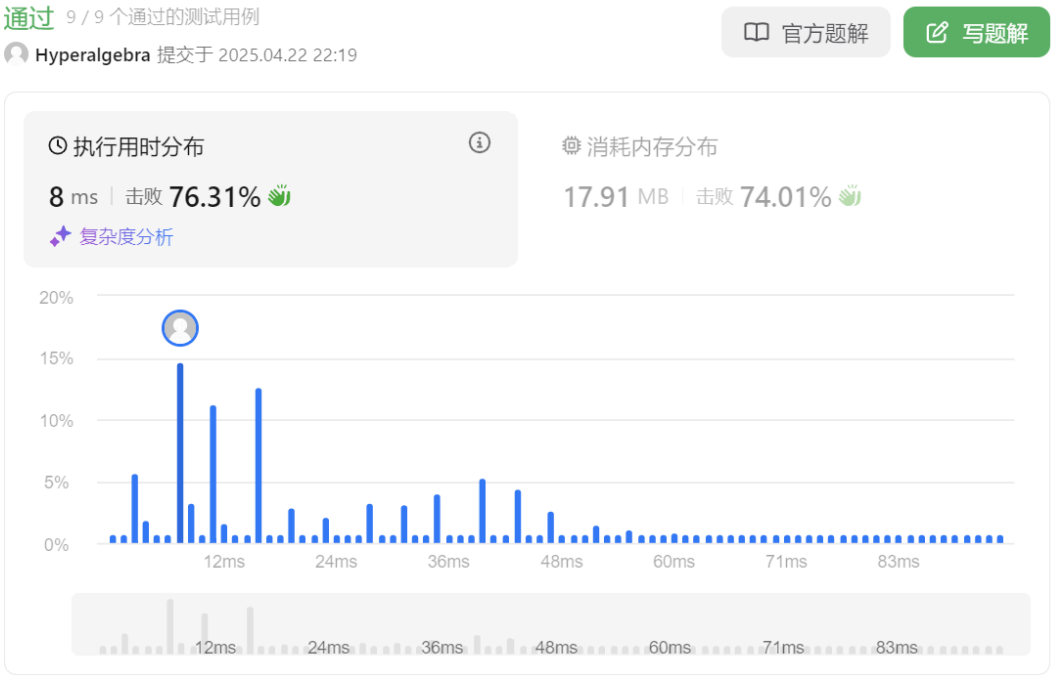
用数组记录每一列、每一条对角线是否摆放棋子，然后按行DFS回溯即可。约15min。

代码:

```
class Solution:
    def solveNQueens(self, n: int) -> List[List[str]]:
        col,diag1,diag2=[False]*n,[False]*(2*n-1),[False]*(2*n-1)
        ans=[]
        grid=[['.']*n for _ in range(n)]
        def dfs(i:int):
            if i==n:
                ans.append(list(map(lambda x:''.join(x),grid)))
                return
```

```
        for j in range(n):
            if not (col[j] or diag1[i-j+n-1] or diag2[i+j]):
                col[j]=diag1[i-j+n-1]=diag2[i+j]=True
                grid[i][j]='Q'
                dfs(i+1)
                grid[i][j]='.'
                col[j]=diag1[i-j+n-1]=diag2[i+j]=False
    dfs(0)
    return ans
```

代码运行截图 （至少包含有"Accepted"）



代码 | Python3

```
class Solution:
    def solveNQueens(self, n: int) -> List[List[str]]:
        col,diag1,diag2=[False]*n,[False]*(2*n-1),[False]*(2*n-1)
        ans=[]
        grid=[['.']*n for _ in range(n)]
        def dfs(i:int):
            if i==n:
                ans.append(list(map(lambda x:''.join(x),grid)))
                return
            for j in range(n):
                if not (col[j] or diag1[i-j+n-1] or diag2[i+j]):
                    col[j]=diag1[i-j+n-1]=diag2[i+j]=True
                    grid[i][j]='Q'
                    dfs(i+1)
                    grid[i][j]='.'
                    col[j]=diag1[i-j+n-1]=diag2[i+j]=False
        dfs(0)
        return ans
```

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。

作业还是比较轻松的。平时就照常完成每日选做和LeetCode每日一题。

上周打了LeetCode周赛，感觉周赛题目模板性都比较强，但是写最后一题的线段树还是经过了反复调试，卡着结束时间才通过。可能最近确实有点手生了，还得继续努力。