

Assignment #8: 图论：概念、遍历，及树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Compiled by 天幕 化学与分子工程学院

说明：

- 1) 请把每个题目解题思路（可选），源码Python, 或者C++（已经在Codeforces/Openjudge上AC），截图（包含Accepted），填写到下面作业模版中（推荐使用 typora <https://typoraio.cn>，或者用 word）。AC 或者没有AC，都请标上每个题目大致花费时间。
- 2) 提交时候先提交pdf文件，再把md或者doc文件上传到右侧“作业评论”。Canvas需要有同学清晰头像、提交文件有pdf、“作业评论”区有上传的md或者doc附件。
- 3) 如果不能在截止前提交作业，请写明原因。

编程环境**

操作系统：Windows 11 23H2

Python编程环境：Visual Studio Code 1.86.2230.

1. 题目

19943: 图的拉普拉斯矩阵

matrices, <http://cs101.openjudge.cn/practice/19943/>

请定义Vertex类，Graph类，然后实现

思路：按照题目要求构建邻接矩阵。通过遍历输出即可。

代码

```
1 class Vertex:
2     _ID = 0
3     id:int
4     name:str
5     connectedTo:dict
6     connectCount:int
7     def __init__(self, id=False, name=False) -> None:
8         if not id:
9             id = self.__class__._ID
10            self.__class__._ID += 1
11            self.id = id
12            if name:
13                self.name = name
14            self.connectCount = 0
```

```

15         self.connectedTo = {}
16
17     def connectTo(self, other, weight=1):
18         self.connectedTo[other] = weight
19         self.connectCount += 1
20
21 class Graph:
22     vertexNumber:int
23     id_dict:dict
24     def __init__(self):
25         self.vertexNumber = 0
26         self.id_dict = {}
27
28     def addVertex(self, vertex:Vertex):
29         self.id_dict[vertex.id] = vertex
30         self.vertexNumber += 1
31
32     def addVertexSafe(self, vertex:Vertex):
33         if not vertex.id in self.id_dict:
34             self.addVertex(vertex)
35
36     def addEdge(self, v1:Vertex, v2:Vertex, w=1):
37         v1.connectTo(v2, w); v2.connectTo(v1, w)
38
39     def addEdgesafe(self, v1:Vertex, v2:Vertex):
40         self.addVertexSafe(v1); self.addVertexSafe(v2)
41         self.addEdge(v1, v2)
42
43     def createEdgesafe(self, n1:int, n2:int):
44         v1 = self.id_dict[n1]
45         v2 = self.id_dict[n2]
46         self.addEdge(v1, v2)
47
48     def _adjacencyMatrix(self):
49         outL = [[0] * self.vertexNumber for _ in range(self.vertexNumber)]
50         for id1 in self.id_dict:
51             vertex = self.id_dict[id1]
52             for other in vertex.connectedTo:
53                 id2 = other.id
54                 outL[id1][id2] = -vertex.connectedTo[other]
55         return outL
56
57     def _laplaceMatrix(self):
58         outL = self._adjacencyMatrix()
59         for id in self.id_dict:
60             outL[id][id] += self.id_dict[id].connectCount
61         return outL
62
63     def __str__(self):
64         l = self._laplaceMatrix()
65         return '\n'.join([' '.join([str(x) for x in _]) for _ in l])
66 n, m = map(int, input().split())
67 myGraph = Graph()
68 for i in range(n):
69     myGraph.addVertex(Vertex(id=i))
70 for _ in range(m):

```

```
71     n1, n2 = map(int, input().split())
72     myGraph.createEdgeSafe(n1, n2)
73     print(myGraph)
```

代码运行截图

状态: Accepted

源代码

```
class Vertex:
    _ID = 0
    id:int
    name:str
    connectedTo:dict
    connectCount:int
    def __init__(self, id=False, name=False) -> None:
        if not id:
            id = self.__class__._ID
            self.__class__._ID += 1
        self.id = id
        if name:
            self.name = name
        self.connectCount = 0
        self.connectedTo = {}

    def connectTo(self, other, weight=1):
        self.connectedTo[other] = weight
        self.connectCount += 1

class Graph:
    vertexNumber:int
    id_dict:dict
    def __init__(self):
        self.vertexNumber = 0
        self.id_dict = {}

    def addVertex(self, vertex:Vertex):
        self.id_dict[vertex.id] = vertex
        self.vertexNumber += 1

    def addVertexSafe(self, vertex:Vertex):
        if not vertex.id in self.id_dict:
            self.addVertex(vertex)

    def addEdge(self, v1:Vertex, v2:Vertex, w=1):
        v1.connectTo(v2, w); v2.connectTo(v1, w)

    def addEdgeSafe(self, v1:Vertex, v2:Vertex):
        self.addVertexSafe(v1); self.addVertexSafe(v2)
        self.addEdge(v1, v2)

    def createEdgeSafe(self, n1:int, n2:int):
        v1 = self.id_dict[n1]
        v2 = self.id_dict[n2]
        self.addEdge(v1, v2)

    def _adjacencyMatrix(self):
        outL = [[0] * self.vertexNumber for _ in range(self.vertexNumber)]
        for id1 in self.id_dict:
            vertex = self.id_dict[id1]
            for other in vertex.connectedTo:
                id2 = other.id
                outL[id1][id2] = -vertex.connectedTo[other]
        return outL
```

```

def _laplaceMatrix(self):
    outL = self._adjacencyMatrix()
    for id in self.id_dict:
        outL[id][id] += self.id_dict[id].connectCount
    return outL

def __str__(self):
    l = self._laplaceMatrix()
    return '\n'.join([' '.join([str(x) for x in _]) for _ in l])
n, m = map(int, input().split())
myGraph = Graph()
for i in range(n):
    myGraph.addVertex(Vertex(id=i))
for _ in range(m):
    n1, n2 = map(int, input().split())
    myGraph.createEdgeSafe(n1, n2)
print(myGraph)

```

©2002-2022 POJ 京ICP备20010980号-1

18160: 最大连通域面积

matrix/dfs similar, <http://cs101.openjudge.cn/practice/18160>

思路：之前计概的代码，递归完成。

代码

```

1 def trans(x):
2     if x == "W":
3         return(1)
4     else:
5         return(0)
6 def areacount(i,j,list):
7     if list[i][j] != 1:
8         return(0)
9     else:
10        list[i][j] = 0
11        return(1 + (list[i-1][j-1]==1)*(areacount(i-1,j-1,list))+(list[i-1]
12[j]==1)*(areacount(i-1,j,list))+(list[i-1][j+1]==1)*(areacount(i-
131,j+1,list))+(list[i][j-1]==1)*(areacount(i,j-1,list))+(list[i][j+1]==1)*
14(areacount(i,j+1,list))+(list[i+1][j-1]==1)*(areacount(i+1,j-1,list))+
15(list[i+1][j]==1)*(areacount(i+1,j,list))+(list[i+1][j+1]==1)*
16(areacount(i+1,j+1,list)))
17 for _ in range(int(input())):
18     N, M = map(int, input().split())
19     l = [[-1]*(M+2)] + [[[-1] + [trans(x) for x in list(input())] + [-1]]
20 for _ in range(N)] + [[-1]*(M+2)]

```

```

15     ans = 0
16     for i in range(1, N+1):
17         for j in range(1, M+1):
18             if l[i][j] == 1:
19                 ans = max(ans, areacount(i,j,l))
20     print(ans)

```

代码运行截图

状态: Accepted

源代码

```

def trans(x):
    if x == "W":
        return 1
    else:
        return 0
def areacount(i,j,list):
    if list[i][j] != 1:
        return 0
    else:
        list[i][j] = 0
        return 1 + (list[i-1][j-1]==1)*(areacount(i-1,j-1,list))+(list[i-1][j]==1)*(areacount(i-1,j,list))+(list[i][j-1]==1)*(areacount(i,j-1,list))
for _ in range(int(input())):
    N, M = map(int, input().split())
    l = [[-1]*(M+2)] + [[(-1) + [trans(x) for x in list(input())] + [-1]] for i in range(N)]
    ans = 0
    for i in range(1, N+1):
        for j in range(1, M+1):
            if l[i][j] == 1:
                ans = max(ans, areacount(i,j,l))
    print(ans)

```

sy383: 最大权值连通块

<https://sunnywhy.com/sfbj/10/3/383>

思路: 使用递归遍历寻找, 对每个已经访问过的顶点 `markDirty()` 避免重复。

代码

```

1 class Vertex:
2     _ID = 0
3     id:int
4     weight:int
5     connectedTo:dict

```

```

6     connectCount:int
7     isDirty:bool
8     def __init__(self, id, weight) -> None:
9         self.id = id
10        self.weight = weight
11        self.connectCount = 0
12        self.connectedTo = {}
13        self.isDirty = False
14
15    def connectTo(self, other, weight=1):
16        self.connectedTo[other] = weight
17        self.connectCount += 1
18        other.connectedTo[self] = weight
19        other.connectCount += 1
20
21    def markDirty(self):
22        self.isDirty =True
23
24    class Graph:
25        vertexNumber:int
26        id_dict:dict
27        def __init__(self):
28            self.vertexNumber = 0
29            self.id_dict = {}
30
31        def addVertex(self, vertex:Vertex):
32            self.id_dict[vertex.id] = vertex
33            self.vertexNumber += 1
34
35        def addVertexSafe(self, vertex:Vertex):
36            if not vertex.id in self.id_dict:
37                self.addVertex(vertex)
38
39        def addEdge(self, v1:Vertex, v2:Vertex, w=1):
40            v1.connectTo(v2, w)
41
42        def addEdgesafe(self, v1:Vertex, v2:Vertex):
43            self.addVertexSafe(v1); self.addVertexSafe(v2)
44            self.addEdge(v1, v2)
45
46        def createEdgesafe(self, n1:int, n2:int):
47            v1 = self.id_dict[n1]
48            v2 = self.id_dict[n2]
49            self.addEdge(v1, v2)
50
51        def _adjacencyMatrix(self):
52            outL = [[0] * self.vertexNumber for _ in range(self.vertexNumber)]
53            for id1 in self.id_dict:
54                vertex = self.id_dict[id1]
55                for other in vertex.connectedTo:
56                    id2 = other.id
57                    outL[id1][id2] = -vertex.connectedTo[other]
58            return outL
59
60        def _laplaceMatrix(self):
61            outL = self._adjacencyMatrix()

```

```

62         for id in self.id_dict:
63             outL[id][id] += self.id_dict[id].connectCount
64         return outL
65
66     def __str__(self):
67         l = self._laplaceMatrix()
68         return '\n'.join([' '.join([str(x) for x in _]) for _ in l])
69
70     def BFSVertexParser(vertex:Vertex):
71         if vertex.isDirty: return 0
72         vertex.markDirty()
73         ans = vertex.weight
74         for conneted in vertex.connectedTo:
75             ans += BFSVertexParser(conneted)
76         return ans
77
78     def Max(graph:Graph):
79         ans = 0
80         for vertex in graph.id_dict.values():
81             if not vertex.isDirty:
82                 ans = max(ans, BFSVertexParser(vertex))
83         return ans
84
85     n, m = map(int, input().split())
86     myGraph = Graph()
87     lw = list(map(int, input().split()))
88     for i in range(n):
89         myGraph.addVertex(Vertex(i, lw[i]))
90     for _ in range(m):
91         n1, n2 = map(int, input().split())
92         myGraph.createEdgeSafe(n1, n2)
93     print(Max(myGraph))

```

代码运行截图


```
69
70 def BFSVertexParser(vertex:Vertex):
71     if vertex.isDirty: return 0
72     vertex.markDirty()
73     ans = vertex.weight
74     for conneted in vertex.connectedTo:
75         ans += BFSVertexParser(conneted)
76     return ans
77
78 def Max(graph:Graph):
79     ans = 0
80     for vertex in graph.id_dict.values():
81         if not vertex.isDirty:
82             ans = max(ans, BFSVertexParser(vertex))
83     return ans
84
85 n, m = map(int, input().split())
86 myGraph = Graph()
87 lw = list(map(int, input().split()))
88 for i in range(n):
89     myGraph.addVertex(Vertex(i, lw[i]))
90 for _ in range(m):
91     n1, n2 = map(int, input().split())
92     myGraph.createEdgeSafe(n1, n2)
93 print(Max(myGraph))
```

测试输入

提交结果

历史提交

完美通过

[查看题解](#)

100% 数据通过测试

运行时长: 0 ms

[收起面板](#)

运行 ▾

提交

03441: 4 Values whose Sum is 0

data structure/binary search, <http://cs101.openjudge.cn/practice/03441>

思路：使用 `itertools` 与 `bisect` 减少性能开销，不过结果好像不怎么样（python MLE，但是pypy能过），看群里大佬的好像直接遍历第二个列表了，看来bisect是负优化。

代码

```

1 import bisect
2 import itertools
3 count = 0
4 n = int(input())
5 A, B, C, D = zip(*[map(int, input().split()) for _ in range(n)])
6 lf = list(map(sum, itertools.product(A, B)))
7 lf.sort()
8 for x in map(sum, itertools.product(C, D)):
9     count += bisect.bisect_right(lf, -x) - bisect.bisect_left(lf, -x)
10 print(count)

```

代码运行截图

状态: Accepted

源代码

```

import bisect
import itertools
count = 0
n = int(input())
A, B, C, D = zip(*[map(int, input().split()) for _ in range(n)])
lf = list(map(sum, itertools.product(A, B)))
lf.sort()
for x in map(sum, itertools.product(C, D)):
    count += bisect.bisect_right(lf, -x) - bisect.bisect_left(lf, -x)
print(count)

```

04089: 电话号码

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

Trie 数据结构可能需要自学下。

思路：用是否踩到末端节点以及在整个过程中有没有创建新节点判断，可以解决输入顺序问题。

千万不能随意break。

代码

```

1 class PrefixError(IndexError):
2     def __init__(self, *args: object) -> None:
3         super().__init__(*args)
4
5 class Node:
6     _ID=0
7     id:int
8     is_word:bool

```

```

9     name:str
10    sub:list
11
12    def __init__(self, name, sub, is_word=False):
13        self.id = self.__class__.__ID
14        self.__class__.__ID += 1
15        self.name = name
16        self.sub = sub
17        self.is_word = is_word
18    def mark_word(self):
19        self.is_word = True
20
21 class Tree:
22     root:Node
23
24     def __init__(self) -> None:
25         self.root = Node('', [])
26     def push(self, string):
27         flag = self._push(string, self.root, False)
28         if not flag:
29             raise PrefixError
30     def _push(self, string:str, node:Node, flag):
31         if not string:
32             node.mark_word()
33             return flag
34         this, remaining = string[0], string[1:]
35         for subnode in node.sub:
36             if this == subnode.name:
37                 if subnode.is_word: raise PrefixError
38                 return self._push(remaining, subnode, flag)
39         neonode = Node(this, [])
40         node.sub.append(neonode)
41         return self._push(remaining, neonode, True)
42
43 def check(times, tree):
44     flag = True
45     for _ in range(times):
46         x = input()
47         if flag == False: continue
48         try:
49             tree.push(x)
50         except PrefixError:
51             flag = False
52     print(['NO', 'YES'][flag])
53     return
54
55 t = int(input())
56 for _ in range(t):
57     myTree = Tree()
58     check(int(input()), myTree)

```

代码运行截图

状态: Accepted

源代码

```
class PrefixError(IndexError):
    def __init__(self, *args: object) -> None:
        super().__init__(*args)

class Node:
    _ID=0
    id:int
    is_word:bool
    name:str
    sub:list

    def __init__(self, name, sub, is_word=False):
        self.id = self.__class__._ID
        self.__class__._ID += 1
        self.name = name
        self.sub = sub
        self.is_word = is_word
    def mark_word(self):
        self.is_word = True

class Tree:
    root:Node

    def __init__(self) -> None:
        self.root = Node('', [])
    def push(self, string):
        flag = self._push(string, self.root, False)
        if not flag:
            raise PrefixError
    def _push(self, string:str, node:Node, flag):
        if not string:
            node.mark_word()
            return flag
        this, remaining = string[0], string[1:]
        for subnode in node.sub:
            if this == subnode.name:
                if subnode.is_word: raise PrefixError
                return self._push(remaining, subnode, flag)
        neonode = Node(this, [])
        node.sub.append(neonode)
        return self._push(remaining, neonode, True)

    def check(times, tree):
        flag = True
        for _ in range(times):
            x = input()
            if flag == False: continue
            try:
                tree.push(x)
            except PrefixError:
                flag = False
        print(['NO', 'YES'][flag])
        return

t = int(input())
```

```
for _ in range(t):
    myTree = Tree()
    check(int(input()), myTree)
```

©2002-2022 POJ 京ICP备20010980号-1

04082: 树的镜面映射

<http://cs101.openjudge.cn/practice/04082/>

思路：树相关的经典缝合题。倒序需要理清思路，感谢老师的测试数据！

代码

```
1 class Node:
2     __ID=0
3     id:int
4     name:str
5     sub:list
6
7     def __init__(self, name, sub):
8         self.id = self.__class__.__ID
9         self.__class__.__ID += 1
10        self.name = name
11        self.sub = sub
12    def isFake(self):
13        return self.name == '$'
14
15    def levelOrder(root:Node):
16        l = [[]]
17        def pseudoLeverParser(node:Node, l:list, level:int):
18            if node is None or node.isFake(): return
19            try:
20                l[level].append(node)
21            except IndexError:
22                l.append([node])
23            pseudoLeverParser(node.sub[0], l, level + 1)
24            pseudoLeverParser(node.sub[1], l, level)
25        pseudoLeverParser(root, l, 0)
26        return ' '.join(' '.join([' '.join([y.name for y in reversed(x)]) for x in
27        l]))
28
29    n = int(input())
30    l = input().split()
31    stack = []
32    root = None
33    for x in l[::-1]:
34        name, nodetype = x[0], int(x[1])
```

```
35         stack.append(Node(name, [None, None]))
36     else:
37         node1 = stack.pop()
38         node2 = stack.pop()
39         neonode = Node(name, [node1, node2])
40         stack.append(neonode)
41         root = neonode
42 ans = levelOrder(root)
43 print(ans)
```

代码运行截图

状态: Accepted

源代码

```
class Node:
    _ID=0
    id:int
    name:str
    sub:list

    def __init__(self, name, sub):
        self.id = self.__class__._ID
        self.__class__._ID += 1
        self.name = name
        self.sub = sub

    def isFake(self):
        return self.name == '$'

def levelOrder(root:Node):
    l = [[]]
    def pseudoLevelParser(node:Node, l:list, level:int):
        if node is None or node.isFake(): return
        try:
            l[level].append(node)
        except IndexError:
            l.append([node])
            pseudoLevelParser(node.sub[0], l, level + 1)
            pseudoLevelParser(node.sub[1], l, level)
    pseudoLevelParser(root, l, 0)
    return ' '.join(' '.join([' '.join([y.name for y in reversed(x)]) for x in l])

n = int(input())
l = input().split()
stack = []
root = None
for x in l[::-1]:
    name, nodetype = x[0], int(x[1])
    if nodetype:
        stack.append(Node(name, [None, None]))
    else:
        node1 = stack.pop()
        node2 = stack.pop()
        neonode = Node(name, [node1, node2])
        stack.append(neonode)
        root = neonode
ans = levelOrder(root)
print(ans)
```

©2002-2022 POJ 京ICP备20010980号-1

2. 学习总结和收获

感觉比前两周的题简单一些，也可能是写手熟了，代码也更加简洁。

