Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Complied by ~~天幂~~ 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

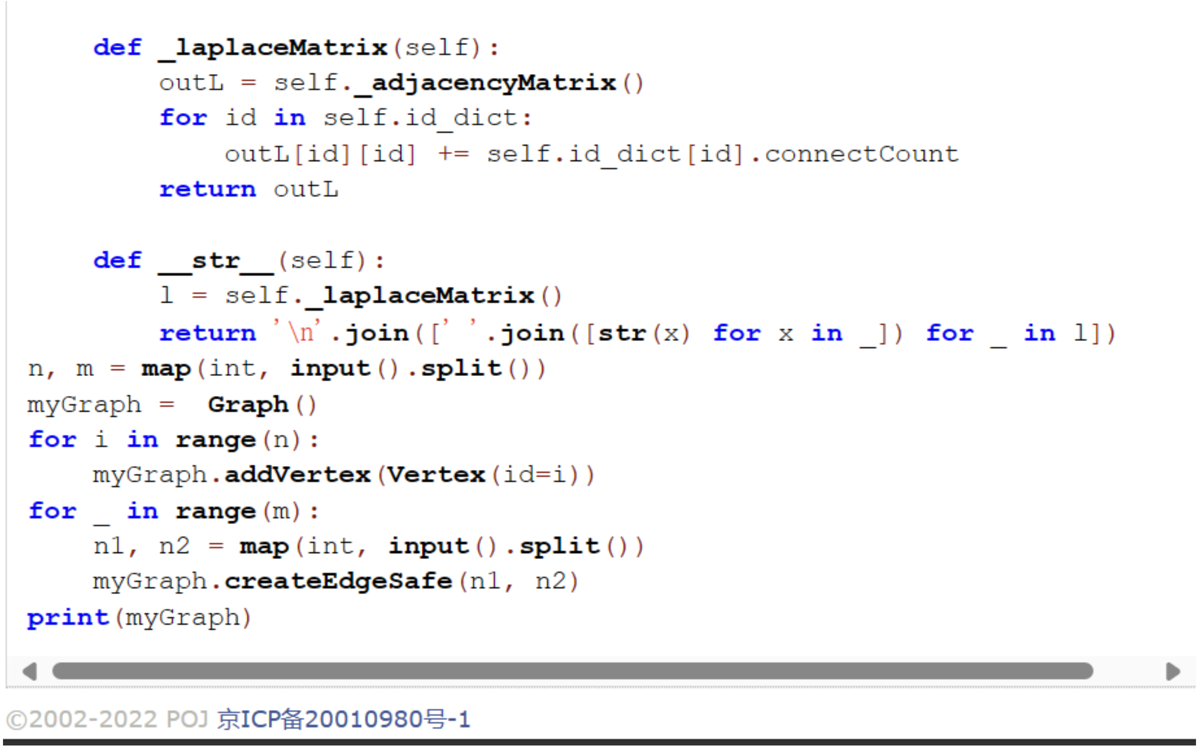
```python
class Vertex:
    _ID = 0
    id:int
    name:str
    connectedTo:dict
    connectCount:int
    def __init__(self, id=False, name=False) -> None:
        if not id:
            id = self.__class__._ID
            self.__class__._ID += 1
        self.id = id
        if name:
            self.name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

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.vertexNumbe
        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

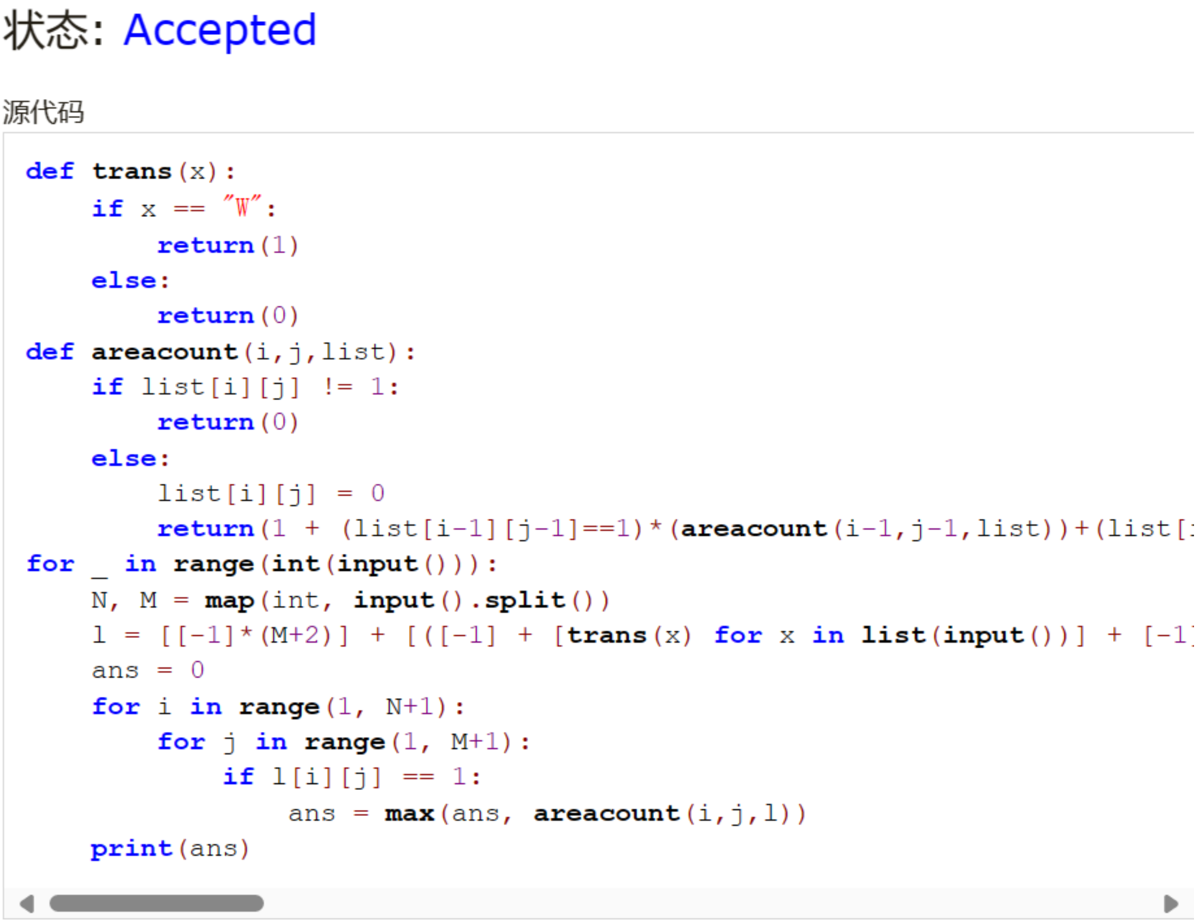
思路：之前计概的代码，递归完成。

代码

```
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-1][j+1]==1)*(areacount(i-1,j+1,list))+(list[i][j-1]==1)*(areacount(i,j-1,list))+(list[i][j+1]==1)*(areacount(i,j+1,list))+(list[i+1][j-1]==1)*(areacount(i+1,j-1,list))+(list[i+1][j]==1)*(areacount(i+1,j,list))+(list[i+1][j+1]==1)*(areacount(i+1,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 _ in range(N)] + [[-1]*(M+2)]
```

```python
    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)
```

代码运行截图



状态: Accepted

源代码

```python
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[
for _ in range(int(input())):
    N, M = map(int, input().split())
    l = [[-1]*(M+2)] + [([-1] + [trans(x) for x in list(input())] + [-1
    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

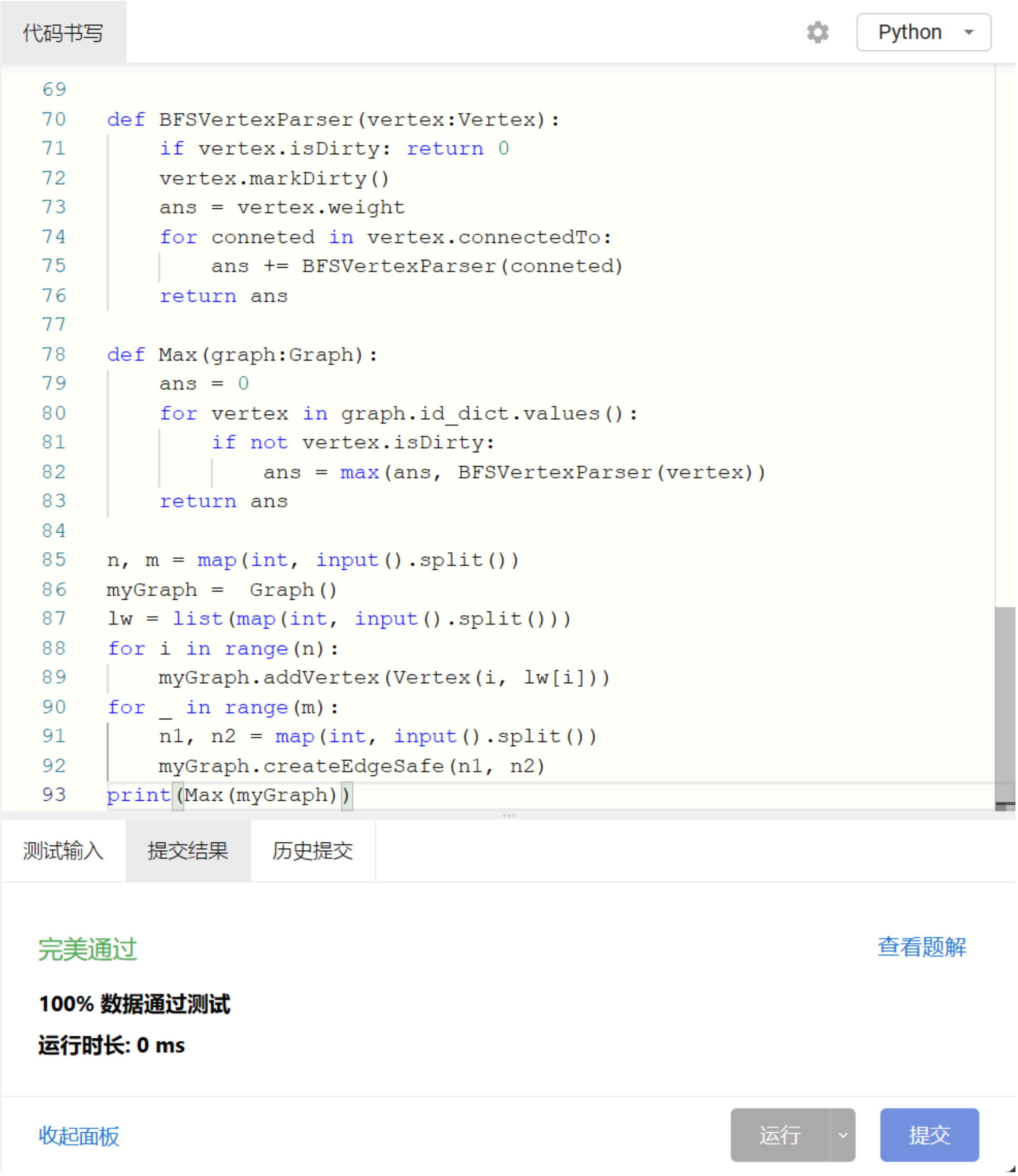
https://sunnywhy.com/sfbj/10/3/383

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

```python
class Vertex:
    _ID = 0
    id:int
    weight:int
    connectedTo:dict
    connectCount:int
    isDirty:bool
    def __init__(self, id, weight) -> None:
        self.id = id
        self.weight = weight
        self.connectCount = 0
        self.connectedTo = {}
        self.isDirty = False

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

    def markDirty(self):
        self.isDirty =True

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)

    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])

def BFSVertexParser(vertex:Vertex):
    if vertex.isDirty: return 0
    vertex.markDirty()
    ans = vertex.weight
    for conneted in vertex.connectedTo:
        ans += BFSVertexParser(conneted)
    return ans

def Max(graph:Graph):
    ans = 0
    for vertex in graph.id_dict.values():
        if not vertex.isDirty:
            ans = max(ans, BFSVertexParser(vertex))
    return ans

n, m = map(int, input().split())
myGraph =  Graph()
lw = list(map(int, input().split()))
for i in range(n):
    myGraph.addVertex(Vertex(i, lw[i]))
for _ in range(m):
    n1, n2 = map(int, input().split())
    myGraph.createEdgeSafe(n1, n2)
print(Max(myGraph))
```

代码运行截图



```python

def BFSVertexParser(vertex:Vertex):
    if vertex.isDirty: return 0
    vertex.markDirty()
    ans = vertex.weight
    for conneted in vertex.connectedTo:
        ans += BFSVertexParser(conneted)
    return ans

def Max(graph:Graph):
    ans = 0
    for vertex in graph.id_dict.values():
        if not vertex.isDirty:
            ans = max(ans, BFSVertexParser(vertex))
    return ans

n, m = map(int, input().split())
myGraph =  Graph()
lw = list(map(int, input().split()))
for i in range(n):
    myGraph.addVertex(Vertex(i, lw[i]))
for _ in range(m):
    n1, n2 = map(int, input().split())
    myGraph.createEdgeSafe(n1, n2)
print(Max(myGraph))
```

完美通过

100% 数据通过测试

运行时长: 0 ms

03441: 4 Values whose Sum is 0

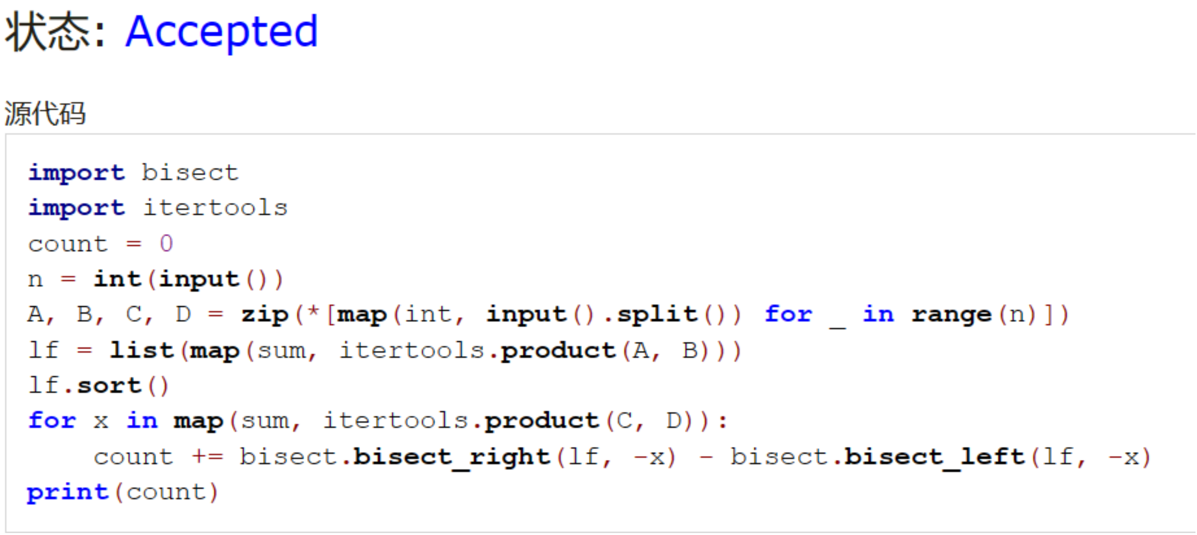
data structure/binary search, http://cs101.openjudge.cn/practice/03441

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

```python
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)
```

代码运行截图



状态: Accepted

源代码

```python
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。

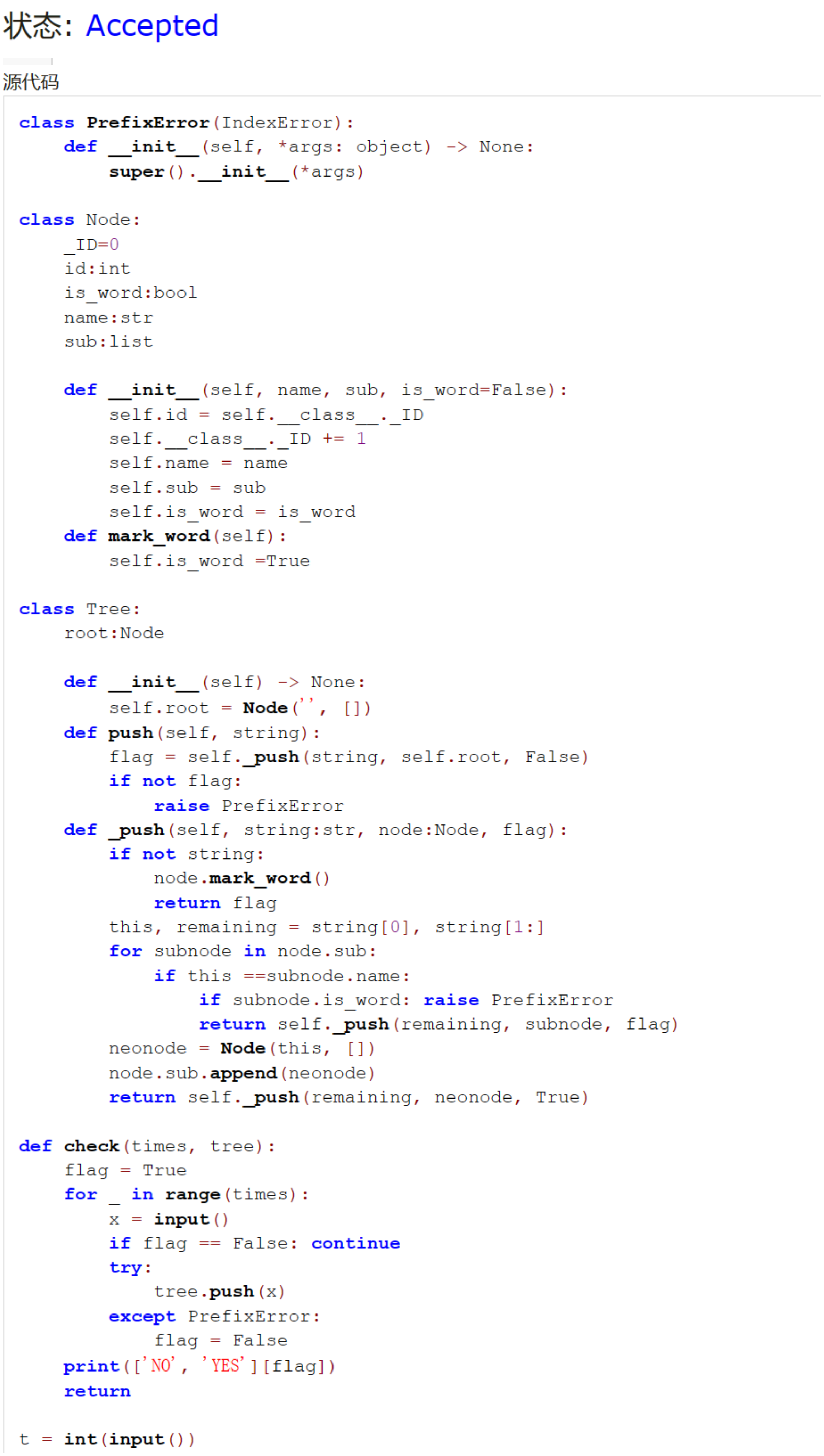
思路：用是否踩到末端节点以及在整个过程中有没有创建新节点判断，可以解决输入顺序问题。、

千万不能随意break。

代码

```python
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)
```

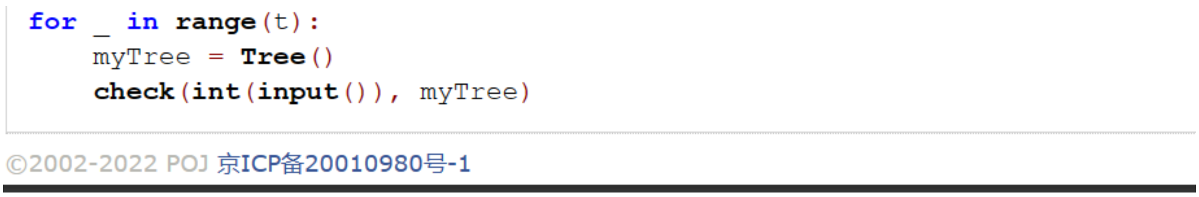
代码运行截图



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

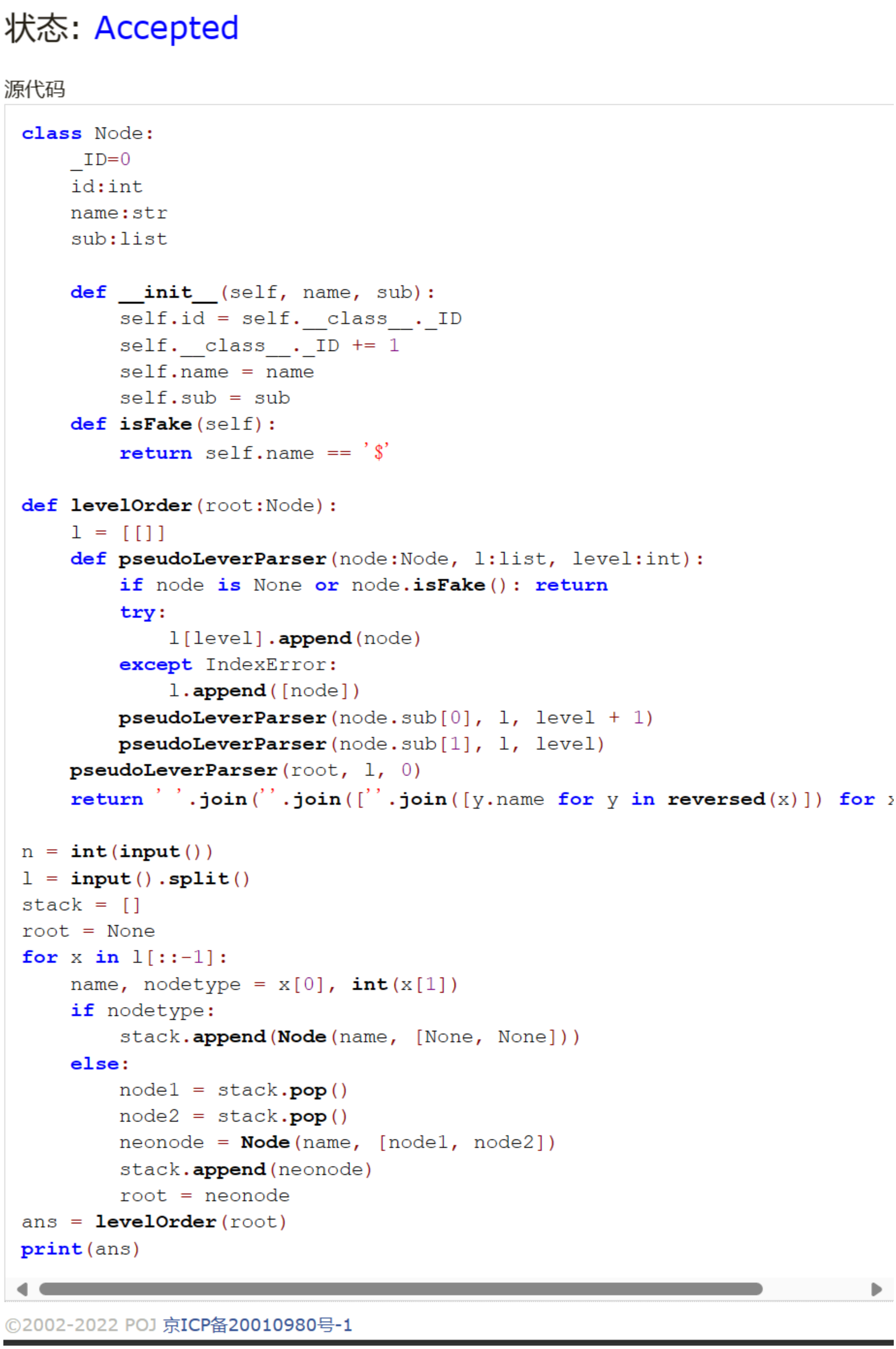
http://cs101.openjudge.cn/practice/04082/

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

```python
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 pseudoLeverParser(node:Node, l:list, level:int):
        if node is None or node.isFake(): return
        try:
            l[level].append(node)
        except IndexError:
            l.append([node])
        pseudoLeverParser(node.sub[0], l, level + 1)
        pseudoLeverParser(node.sub[1], l, level)
    pseudoLeverParser(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)
```

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

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。