Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Complied by 田济维 物理学院

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

（python）

操作系统：macOS Ventura 13.4.1 (c)

Python编程环境：Spyder IDE 5.2.2, PyCharm 2023.1.4 (Professional Edition)

C/C++编程环境：Mac terminal vi (version 9.0.1424), g++/gcc (Apple clang version 14.0.3, clang-1403.0.22.14.1)

1. 题目

19943: 图的拉普拉斯矩阵

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

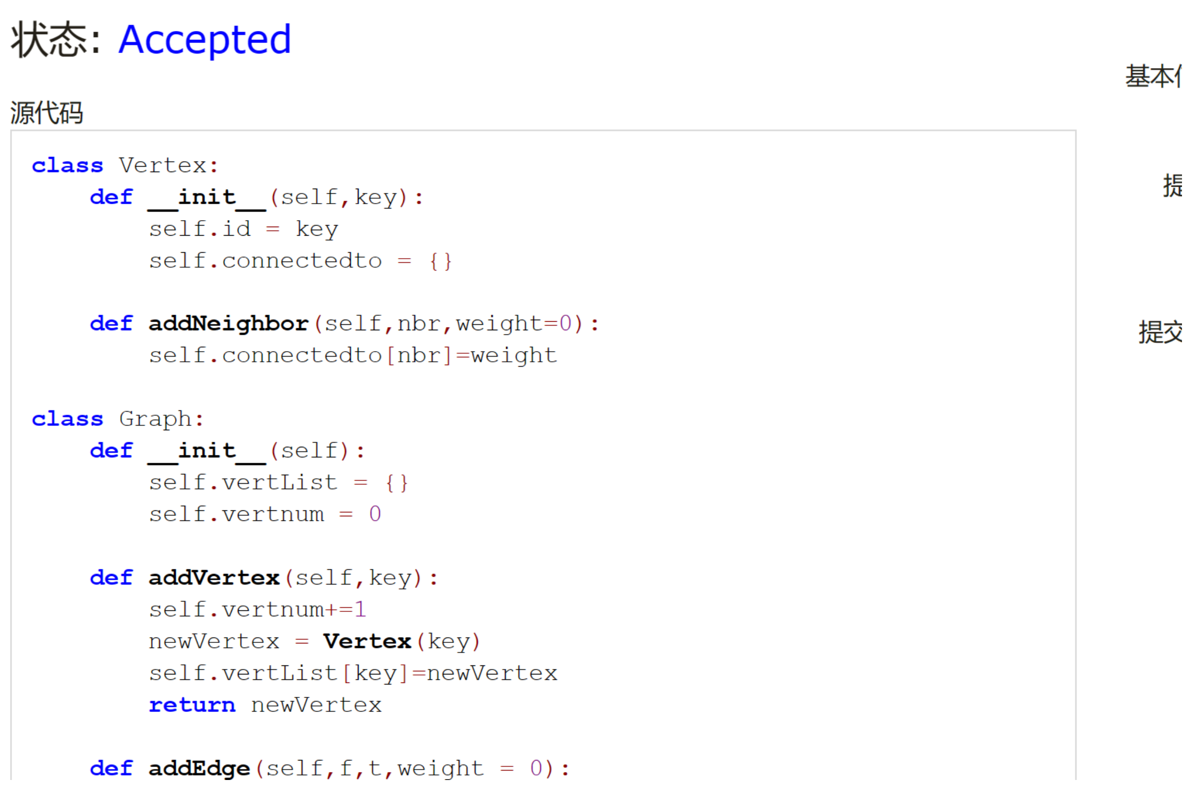
请定义Vertex类，Graph类，然后实现

思路：

代码

```python
#
class Vertex:
    def __init__(self,key):
        self.id = key
        self.connectedto = {}

    def addNeighbor(self,nbr,weight=0):
        self.connectedto[nbr]=weight

class Graph:
    def __init__(self):
        self.vertList = {}
        self.vertnum = 0

    def addVertex(self,key):
        self.vertnum+=1
        newVertex = Vertex(key)
        self.vertList[key]=newVertex
        return newVertex

    def addEdge(self,f,t,weight = 0):
        if f not in self.vertList:
            self.addVertex(f)
        if t not in self.vertList:
            self.addVertex(t)
        self.vertList[f].addNeighbor(t,weight)
        self.vertList[t].addNeighbor(f,weight)

n,m = map(int,input().split())
graph = Graph()
for i in range(n):
    graph.addVertex(i)
for i in range(m):
    f,t = map(int,input().split())
    graph.addEdge(f,t)
Laplace = [[0]*n for i in range(n)]
for i in range(n):
    for j in graph.vertList[i].connectedto:
        Laplace[i][j]-=1
        Laplace[i][i]+=1

for i in range(n):
    print(" ".join(map(str,Laplace[i])))



```

代码运行截图 （至少包含有"Accepted"）


状态: Accepted

基本信

提

源代码

提交

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

    def addNeighbor(self,nbr,weight=0):
        self.connectedto[nbr]=weight

class Graph:
    def __init__(self):
        self.vertList = {}
        self.vertnum = 0

    def addVertex(self,key):
        self.vertnum+=1
        newVertex = Vertex(key)
        self.vertList[key]=newVertex
        return newVertex

    def addEdge(self,f,t,weight = 0):
```

18160: 最大连通域面积

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

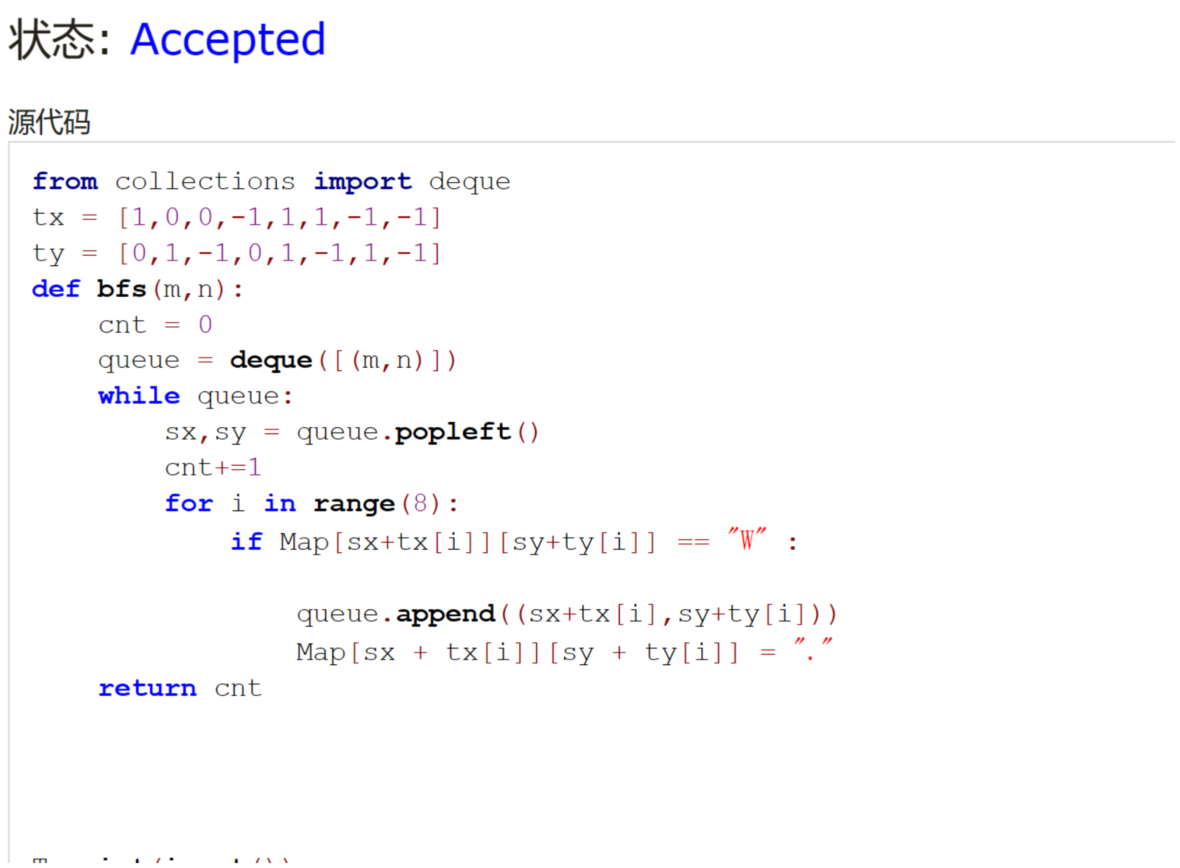
思路:

代码

```python
#
from collections import deque
tx = [1,0,0,-1,1,1,-1,-1]
ty = [0,1,-1,0,1,-1,1,-1]
def bfs(m,n):
    cnt = 0
    queue = deque([(m,n)])
    while queue:
        sx,sy = queue.popleft()
        cnt+=1
        for i in range(8):
            if Map[sx+tx[i]][sy+ty[i]] == "W" :

                queue.append((sx+tx[i],sy+ty[i]))
                Map[sx + tx[i]][sy + ty[i]] = "."
    return cnt




T = int(input())
```

```python
for _ in range(T):
    N,M = map(int,input().split())
    Map = [["."]*(M+2) for i in range(N+2)]
    for i in range(N):
        Map[i+1][1:-1]=input()
    maxn = 0
    for i in range(1,N+1):
        for j in range(1,M+1):
            if Map[i][j]=="W":
                Map[i][j]="."
                maxn = max(maxn,bfs(i,j))
    print(maxn)



```

代码运行截图 （至少包含有"Accepted"）



状态: Accepted

源代码

```python
from collections import deque
tx = [1,0,0,-1,1,1,-1,-1]
ty = [0,1,-1,0,1,-1,1,-1]
def bfs(m,n):
    cnt = 0
    queue = deque([(m,n)])
    while queue:
        sx,sy = queue.popleft()
        cnt+=1
        for i in range(8):
            if Map[sx+tx[i]][sy+ty[i]] == "W" :

                queue.append((sx+tx[i],sy+ty[i]))
                Map[sx + tx[i]][sy + ty[i]] = "."
    return cnt
```

sy383: 最大权值连通块

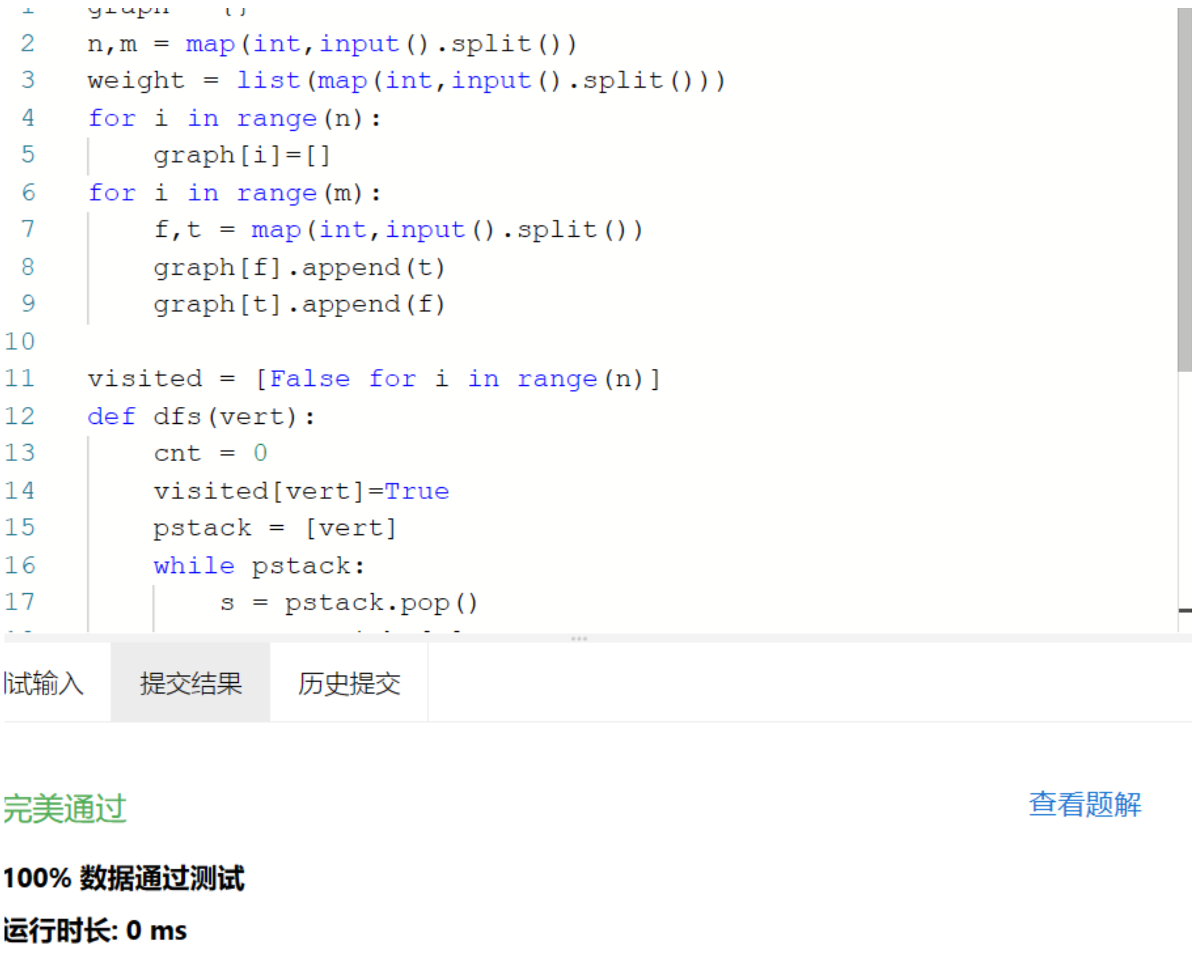
https://sunnywhy.com/sfbj/10/3/383

思路：

代码

```python
#
graph = {}
n,m = map(int,input().split())
weight = list(map(int,input().split()))
for i in range(n):
    graph[i]=[]
for i in range(m):
    f,t = map(int,input().split())
    graph[f].append(t)
    graph[t].append(f)

visited = [False for i in range(n)]
def dfs(vert):
    cnt = 0
    visited[vert]=True
    pstack = [vert]
    while pstack:
        s = pstack.pop()
        cnt +=weight[s]
        for x in graph[s]:
            if visited[x]==False:
                visited[x]=True
                pstack.append(x)
    return cnt
maxn = 0
for i in range(n):
    if visited[i]==False:
        maxn = max(maxn,dfs(i))
print(maxn)


```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



```python
n,m = map(int,input().split())
weight = list(map(int,input().split()))
for i in range(n):
    graph[i]=[]
for i in range(m):
    f,t = map(int,input().split())
    graph[f].append(t)
    graph[t].append(f)

visited = [False for i in range(n)]
def dfs(vert):
    cnt = 0
    visited[vert]=True
    pstack = [vert]
    while pstack:
        s = pstack.pop()
```

完美通过

查看题解

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

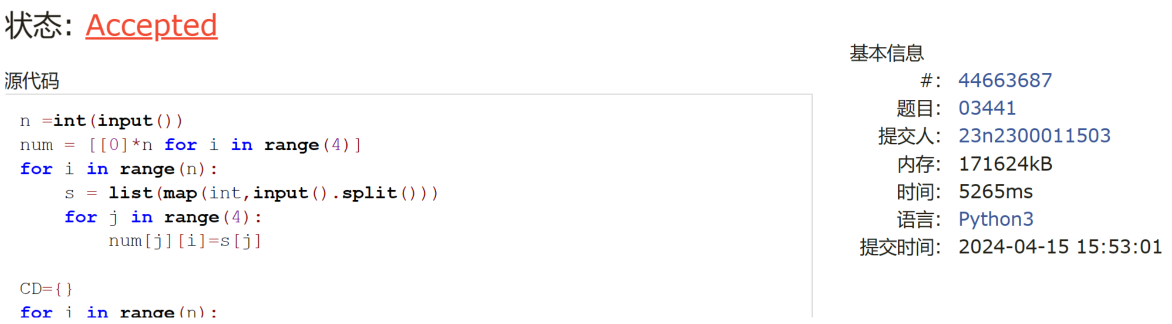
思路：

这题内存空间卡的太死了，一开始用的defaultdict就刚好过不了，用了dict加判断内存空间就恰好过了

代码

```python
#

n =int(input())
num = [[0]*n for i in range(4)]
for i in range(n):
    s = list(map(int,input().split()))
    for j in range(4):
        num[j][i]=s[j]

CD={}
for i in range(n):
    for j in range(n):
        if num[2][i]+num[3][j] not in CD:
            CD[num[2][i] + num[3][j]] =1
        else:
            CD[num[2][i]+num[3][j]]+=1
cnt =0
for i in range(n):
    for j in range(n):
        if -(num[0][i]+num[1][j]) in CD:
            cnt+=CD[-(num[0][i]+num[1][j]) ]
print(cnt)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



状态: Accepted

源代码

```python
n =int(input())
num = [[0]*n for i in range(4)]
for i in range(n):
    s = list(map(int,input().split()))
    for j in range(4):
        num[j][i]=s[j]

CD={}
for i in range(n):
```

基本信息	
#:	44663687
题目:	03441
提交人:	23n2300011503
内存:	171624kB
时间:	5265ms
语言:	Python3
提交时间:	2024-04-15 15:53:01

04089: 电话号码

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

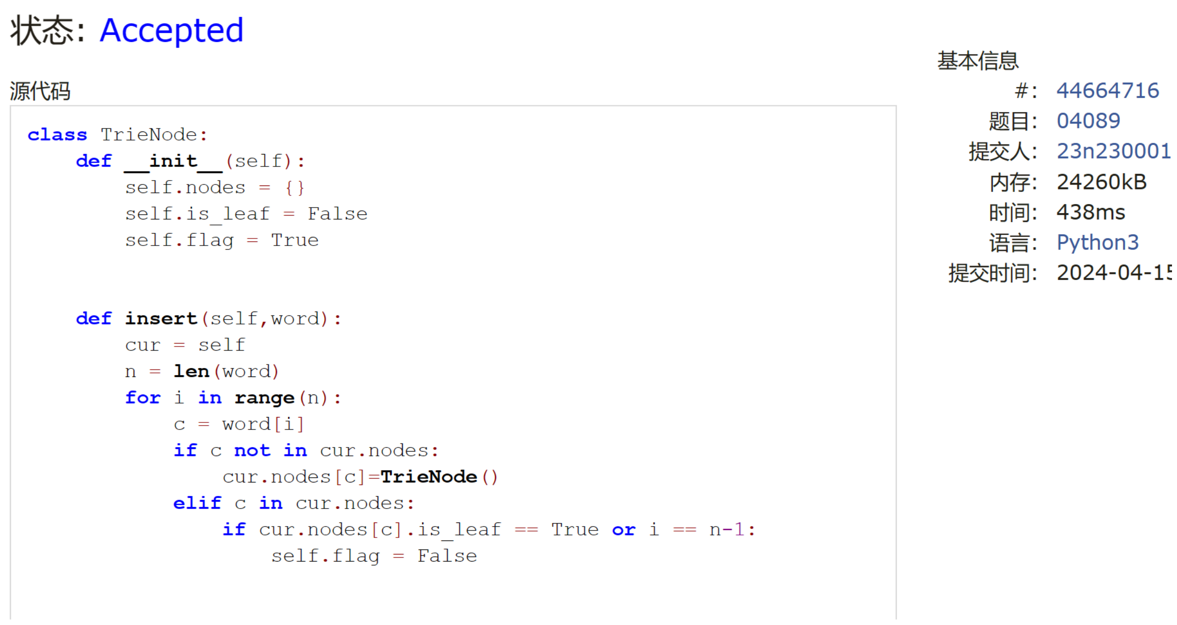
Trie 数据结构可能需要自学下。

思路：

代码

```python
#
class TrieNode:
    def __init__(self):
        self.nodes = {}
        self.is_leaf = False
        self.flag = True


    def insert(self,word):
        cur = self
        n = len(word)
        for i in range(n):
            c = word[i]
            if c not in cur.nodes:
                cur.nodes[c]=TrieNode()
            elif c in cur.nodes:
                if cur.nodes[c].is_leaf == True or i == n-1:
                    self.flag = False




            cur = cur.nodes[c]
        cur.is_leaf=True
t = int(input())
for _ in range(t):
    n = int(input())
    dial = TrieNode()
    for i in range(n):
        dial.insert(input())
    #深度优先搜索
    if dial.flag:
        print("YES")
    else:
        print("NO")

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



状态: Accepted

源代码

```
class TrieNode:
    def __init__(self):
        self.nodes = {}
        self.is_leaf = False
        self.flag = True


    def insert(self,word):
        cur = self
        n = len(word)
        for i in range(n):
            c = word[i]
            if c not in cur.nodes:
                cur.nodes[c]=TrieNode()
            elif c in cur.nodes:
                if cur.nodes[c].is_leaf == True or i == n-1:
                    self.flag = False
```

基本信息

#:	44664716
题目:	04089
提交人:	23n230001
内存:	24260kB
时间:	438ms
语言:	Python3
提交时间:	2024-04-15

04082: 树的镜面映射

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

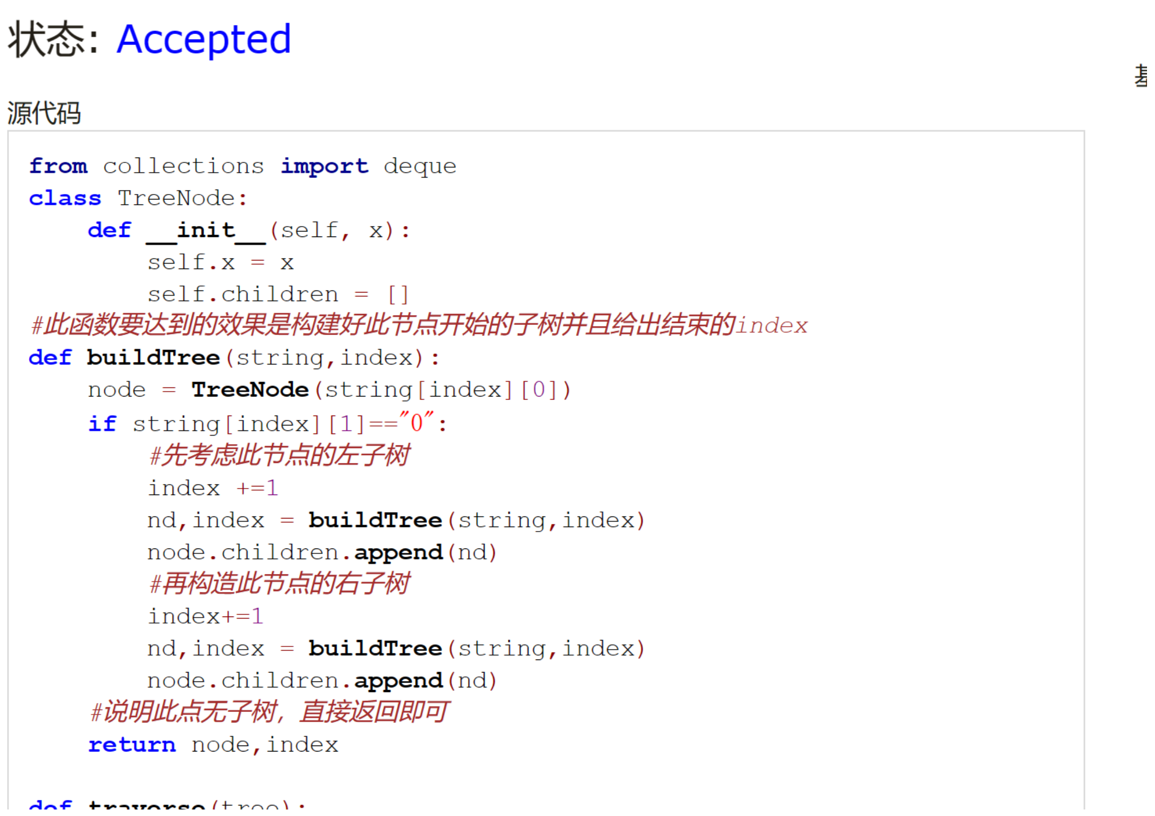
思路:

代码

```
#
from collections import deque
class TreeNode:
    def __init__(self, x):
        self.x = x
```

```python
        self.children = []
#此函数要达到的效果是构建好此节点开始的子树并且给出结束的index
def buildTree(string,index):
    node = TreeNode(string[index][0])
    if string[index][1]=="0":
        #先考虑此节点的左子树
        index +=1
        nd,index = buildTree(string,index)
        node.children.append(nd)
        #再构造此节点的右子树
        index+=1
        nd,index = buildTree(string,index)
        node.children.append(nd)
    #说明此点无子树，直接返回即可
    return node,index

def traverse(tree):
    #由于左儿子右兄弟，先朝右走到底
    queue = deque([tree])
    temp = deque([])
    while queue:
        s = queue.popleft()
        print(s.x,end = " ")
        if len(s.children)>1:
            p = s.children[0]
            while p:
                if p.x !="$":
                    temp.append(p)
                if len(p.children)>1:
                    p = p.children[1]
                else:
                    p = None
            while temp:
                queue.append(temp.pop())

n = int(input())
string = input().split()
tree,i = buildTree(string,0)
traverse(tree)

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


状态: Accepted

源代码

```
from collections import deque
class TreeNode:
    def __init__(self, x):
        self.x = x
        self.children = []
#此函数要达到的效果是构建好此节点开始的子树并且给出结束的index
def buildTree(string,index):
    node = TreeNode(string[index][0])
    if string[index][1]=="0":
        #先考虑此节点的左子树
        index +=1
        nd,index = buildTree(string,index)
        node.children.append(nd)
        #再构造此节点的右子树
        index+=1
        nd,index = buildTree(string,index)
        node.children.append(nd)
    #说明此点无子树，直接返回即可
    return node,index

def traverse(tree):
```

2. 学习总结和收获

如果作业题目简单，有否额外练习题目，比如：OJ“2024spring每日选做”、CF、LeetCode、洛谷等网站题目。

收获1：学习了字典树

收获2：最后一题没想到返回两个量从而实现迭代，最后一题好恶心