Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Complied by 陈奕好 工学院

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

（请改为同学的操作系统、编程环境等）

操作系统：macOS Sonoma 14.4 (23E214)

Python编程环境：PyCharm 2023.3.1 (Professional Edition)

1. 题目

19943: 图的拉普拉斯矩阵

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

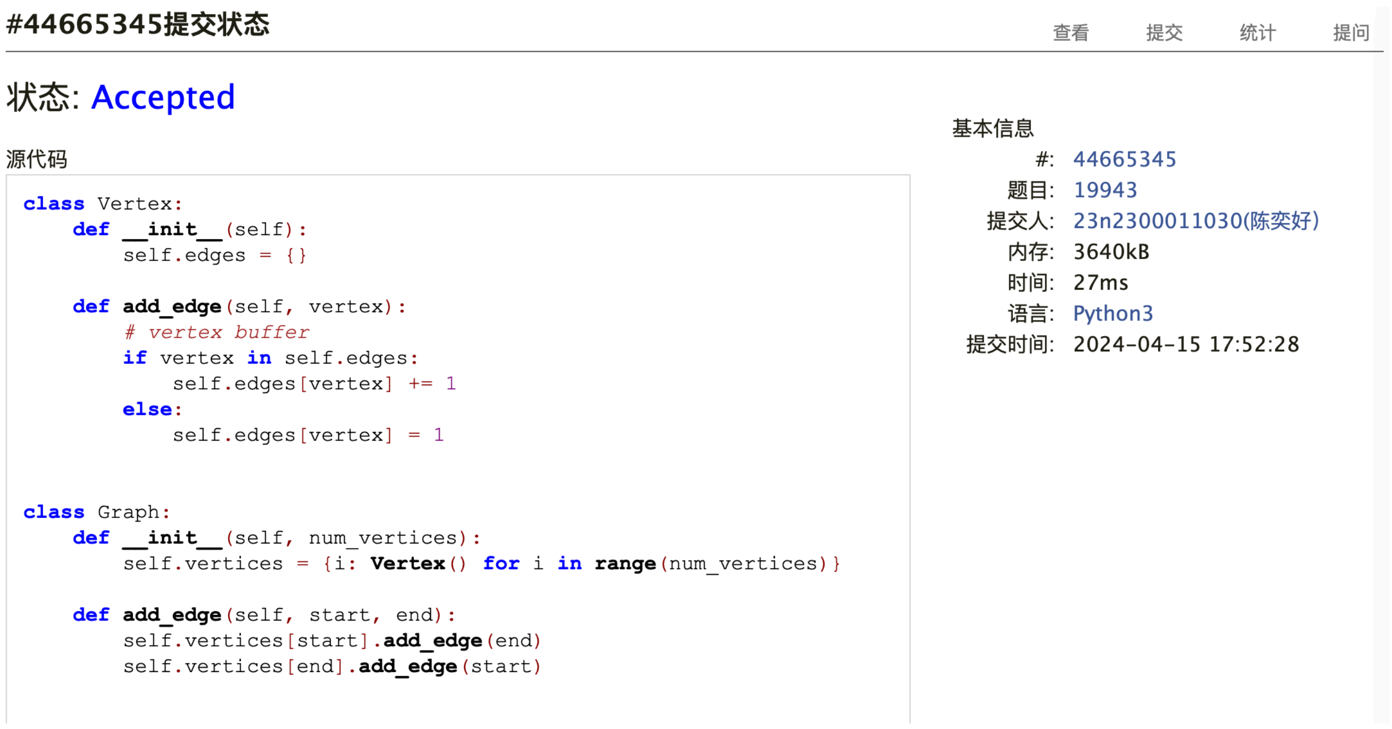
请定义Vertex类，Graph类，然后实现

思路：先定义Vertex类存储点，这个点存储使用了buffer。再定义Graph类初始化每行。
　　main还是初始化一个0矩阵，在零矩阵里操作更加便捷。

代码

```python
class Vertex:
    def __init__(self):
        self.edges = {}

    def add_edge(self, vertex):
        # vertex buffer
        if vertex in self.edges:
                self.edges[vertex] += 1
            else:
                self.edges[vertex] = 1


class Graph:
    def __init__(self, num_vertices):
        self.vertices = {i: Vertex() for i in range(num_vertices)}

    def add_edge(self, start, end):
        self.vertices[start].add_edge(end)
        self.vertices[end].add_edge(start)


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

for i in range(m):
    start, end = map(int, input().split())
    graph.add_edge(start, end)

for vertex, data in graph.vertices.items():
    line = [0]*n
    for connected_vertex, weight in data.edges.items():
        line[vertex] += weight
        line[connected_vertex] -= weight
    print(*line)


```

代码运行截图 （至少包含有"Accepted"）



#44665345提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
class Vertex:
    def __init__(self):
        self.edges = {}

    def add_edge(self, vertex):
        # vertex buffer
        if vertex in self.edges:
            self.edges[vertex] += 1
        else:
            self.edges[vertex] = 1


class Graph:
    def __init__(self, num_vertices):
        self.vertices = {i: Vertex() for i in range(num_vertices)}

    def add_edge(self, start, end):
        self.vertices[start].add_edge(end)
        self.vertices[end].add_edge(start)
```

基本信息

#:	44665345
题目:	19943
提交人:	23n2300011030(陈奕好)
内存:	3640kB
时间:	27ms
语言:	Python3
提交时间:	2024-04-15 17:52:28

18160: 最大连通域面积

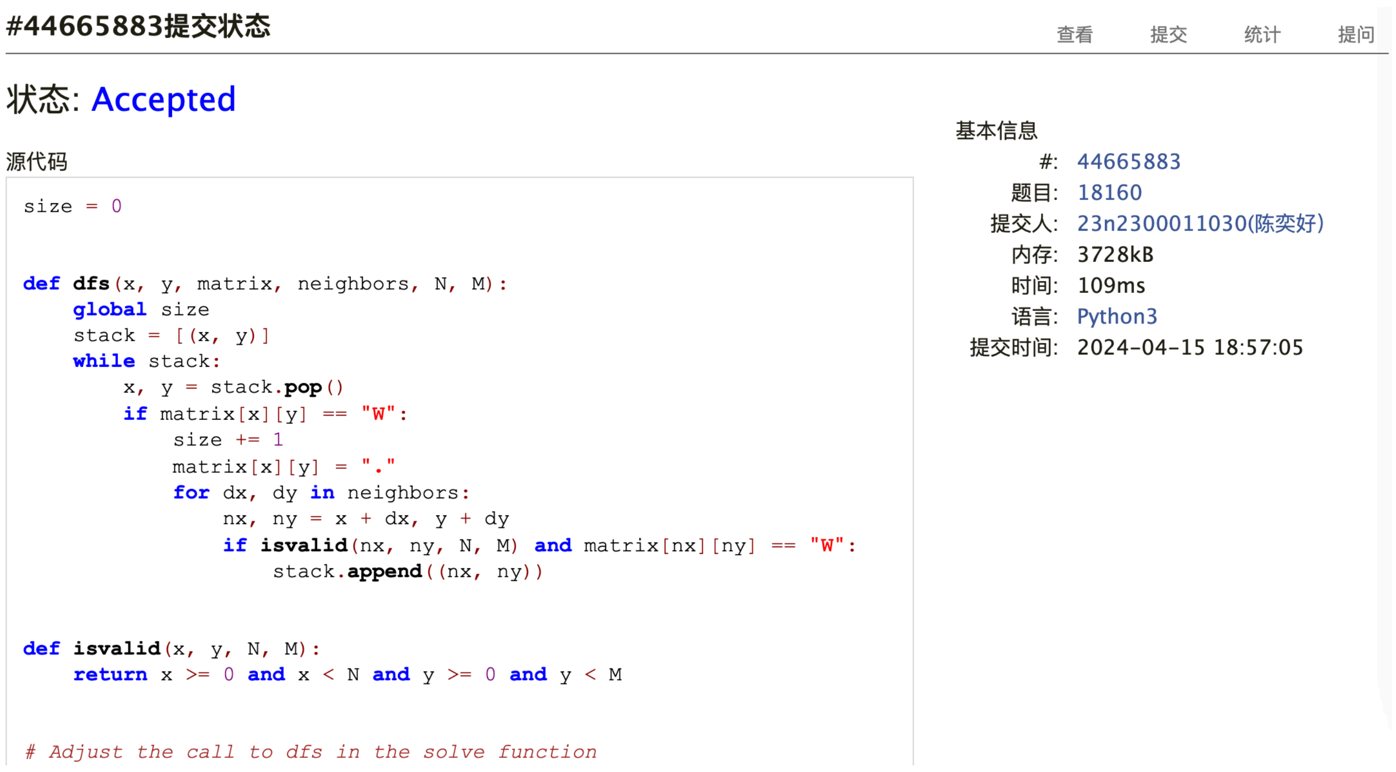
matrix/dfs similar, http://cs101.openjudge.cn/practice/18160

思路：N,M又看反了，而且python3.8卡我变量。

代码

```python
size = 0
def dfs(x, y, matrix, neighbors, N, M):
    global size
    stack = [(x, y)]
    while stack:
        x, y = stack.pop()
        if matrix[x][y] == "W":
            size += 1
            matrix[x][y] = "."
            for dx, dy in neighbors:
                nx, ny = x + dx, y + dy
                if isvalid(nx, ny, N, M) and matrix[nx][ny] == "W":
                    stack.append((nx, ny))


def isvalid(x, y, N, M):
    return x >= 0 and x < N and y >= 0 and y < M


# Adjust the call to dfs in the solve function
def solve(N, M, matrix):
    global size
    largest_size = 0
    neighbors = [(0, 1), (0, -1), (1, 0), (-1, 0), (1, 1), (-1, -1), (1, -1), (-1, 1)]
    for i in range(N):
        for j in range(M):
            if matrix[i][j] == "W":
                dfs(i, j, matrix, neighbors, N, M)
                largest_size = max(largest_size, size)
                size = 0
    return largest_size


T = int(input())
for turn in range(T):
    size = 0
    N, M = map(int, input().split())
    graph = [list(map(str, input())) for _ in range(N)]
    print(solve(N, M, graph))

"""

temp = 0
def search(i,j):
    global temp
    temp += 1
    matrix[i][j] = "."
    for p in dfs:
        if matrix[i+p[0]][j+p[1]] == "W":
            search(i+p[0],j+p[1])


dfs = [(-1,-1),(-1,0),(-1,1),(0,-1),(0,1),(1,-1),(1,0),(1,1)]

T = int(input())
for _ in range(T):
    maxium = 0
    N,M = map(int,input().split())
    matrix = [["."]*(M+2)]
    for i in range(N):
        matrix.append(["."]+list(input())+["."])
    matrix.append(["."]*(M+2))
    #print(matrix)
    for i in range(1,N+1):
        for j in range(1,M+1):
            if matrix[i][j] == "W":
                temp = 0
                search(i,j)
                maxium = max(maxium,temp)
    print(maxium)
"""

```

代码运行截图 （至少包含有"Accepted"）



#44665883提交状态

状态: Accepted

源代码

```python
size = 0


def dfs(x, y, matrix, neighbors, N, M):
    global size
    stack = [(x, y)]
    while stack:
        x, y = stack.pop()
        if matrix[x][y] == "W":
            size += 1
            matrix[x][y] = "."
            for dx, dy in neighbors:
                nx, ny = x + dx, y + dy
                if isvalid(nx, ny, N, M) and matrix[nx][ny] == "W":
                    stack.append((nx, ny))


def isvalid(x, y, N, M):
    return x >= 0 and x < N and y >= 0 and y < M


# Adjust the call to dfs in the solve function
```

基本信息

#:	44665883
题目:	18160
提交人:	23n2300011030(陈奕好)
内存:	3728kB
时间:	109ms
语言:	Python3
提交时间:	2024-04-15 18:57:05

sy383: 最大权值连通块

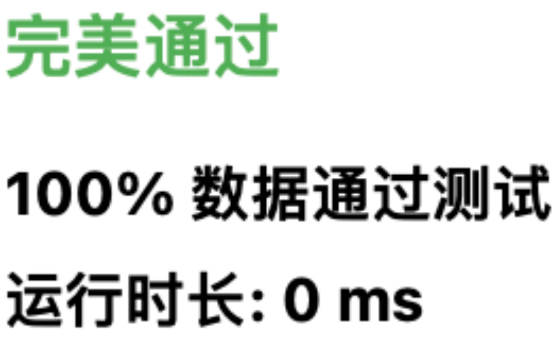
https://sunnywhy.com/sfbj/10/3/383

思路：disjset的运用

代码

```python
class DisjSet:
    def __init__(self, n):
        # Constructor to create and
        # initialize sets of n items
        self.rank = [1] * n
        self.parent = [i for i in range(n)]
        self.weights = [0] * n

    def find(self, x):
        # Find the root of the set in which element x belongs
        if self.parent[x] != x:
            # Path compression: Make the parent of x the root of its set
            self.parent[x] = self.find(self.parent[x])
        return self.parent[x]

    def union(self, x, y):
        # Perform union of two sets
        x_root, y_root = self.find(x), self.find(y)

        if x_root == y_root:
            return
        # Attach smaller rank tree under root of higher rank tree
        if self.rank[x_root] < self.rank[y_root]:
            self.parent[x_root] = y_root
            self.weights[y_root] += self.weights[x_root]
        else:
            self.parent[y_root] = x_root
            self.weights[x_root] += self.weights[y_root]
            if self.rank[x_root] == self.rank[y_root]:
                self.rank[x_root] += 1


n, m = map(int, input().split())
ds = DisjSet(n)
weights = list(map(int, input().split()))
ds.weights = weights.copy()
for i in range(m):
    u, v = map(int, input().split())
    ds.union(u, v)

max_weight = max(ds.weights)
print(max_weight)

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


完美通过

100% 数据通过测试

运行时长: 0 ms

03441: 4 Values whose Sum is 0

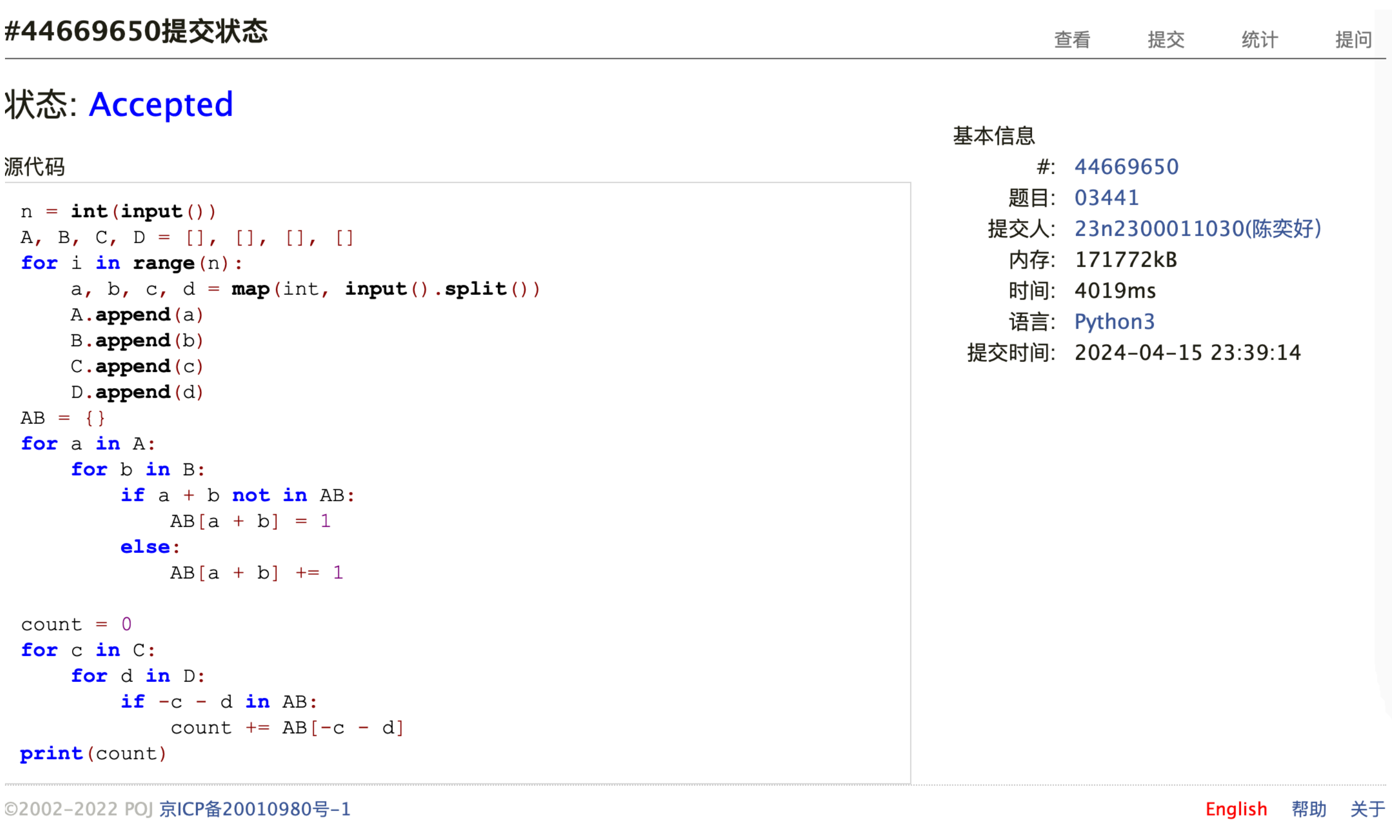
data structure/binary search, http://cs101.openjudge.cn/practice/03441

思路：这里思路比较淳朴，就是字典。

代码

```python
n = int(input())
A, B, C, D = [], [], [], []
for i in range(n):
    a, b, c, d = map(int, input().split())
    A.append(a)
    B.append(b)
    C.append(c)
    D.append(d)
AB = {}
for a in A:
    for b in B:
        if a + b not in AB:
            AB[a + b] = 1
        else:
            AB[a + b] += 1

count = 0
for c in C:
    for d in D:
        if -c - d in AB:
            count += AB[-c - d]
print(count)

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44669650提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
n = int(input())
A, B, C, D = [], [], [], []
for i in range(n):
    a, b, c, d = map(int, input().split())
    A.append(a)
    B.append(b)
    C.append(c)
    D.append(d)
AB = {}
for a in A:
    for b in B:
        if a + b not in AB:
            AB[a + b] = 1
        else:
            AB[a + b] += 1

count = 0
for c in C:
    for d in D:
        if -c - d in AB:
            count += AB[-c - d]
print(count)
```

基本信息

#:	44669650
题目:	03441
提交人:	23n2300011030(陈奕好)
内存:	171772kB
时间:	4019ms
语言:	Python3
提交时间:	2024-04-15 23:39:14

©2002-2022 POJ 京ICP备20010980号-1 English 帮助 关于

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：

字典树（前缀树，Trie）：字典树是一种树形数据结构，用于高效地存储和检索字符串数据集中的键。如果你使用嵌套的字典来表示字典树，其中每个字典代表一个节点，键表示路径上的字符，而值表示子节点，那么就构成了字典树。例如：

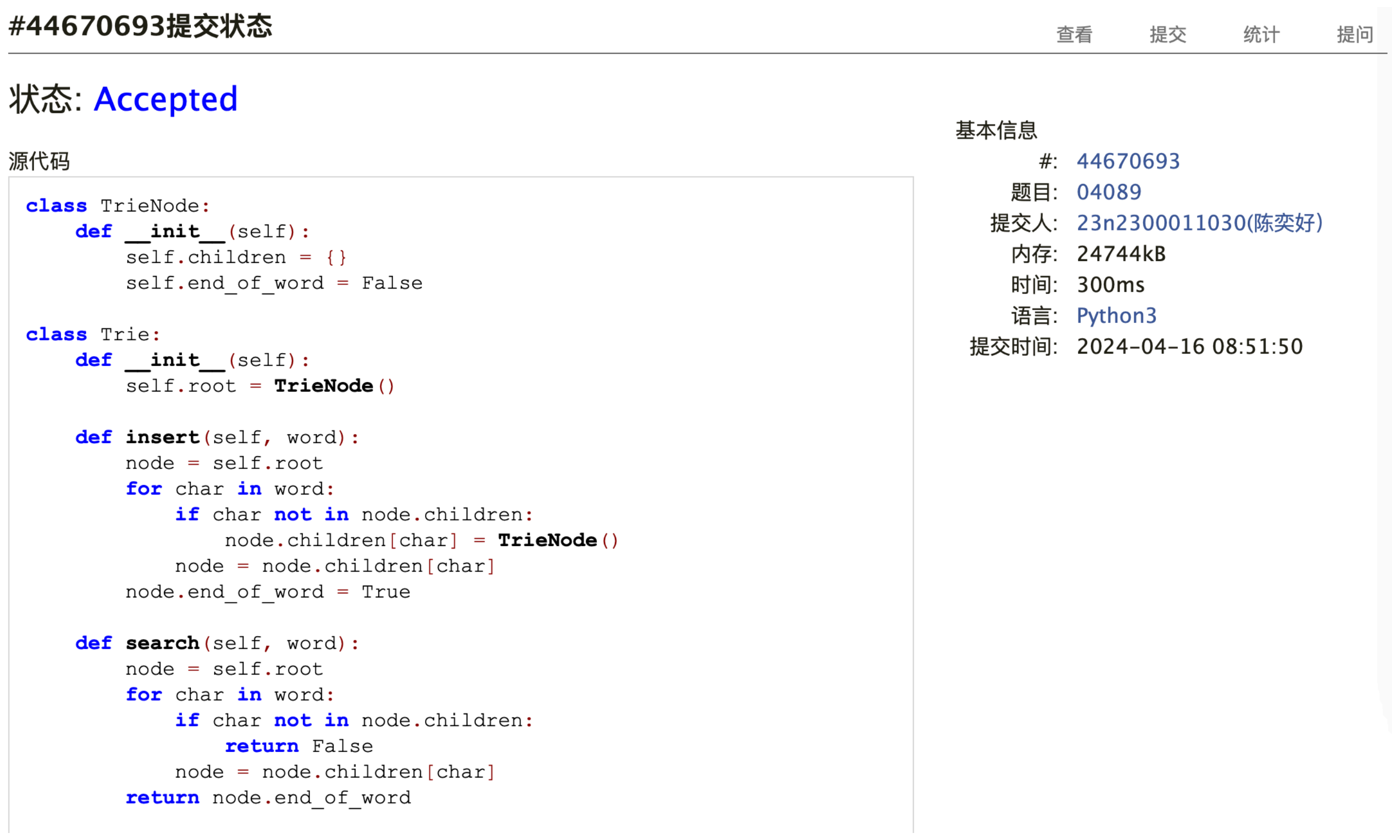
```python
trie = {
    'a': {
        'p': {
            'p': {
                'l': {
                    'e': {'is_end': True}
                }
            }
        }
    },
    'b': {
        'a': {
            'l': {
                'l': {'is_end': True}
            }
        }
    },
    'c': {
        'a': {
            't': {'is_end': True}
        }
    }
}
```

这样的表示方式使得我们可以非常高效地搜索和插入字符串，特别是在大型数据集上。

代码

```python
class TrieNode:
    def __init__(self):
        self.children = {}
        self.end_of_word = False

class Trie:
    def __init__(self):
        self.root = TrieNode()

    def insert(self, word):
        node = self.root
        for char in word:
            if char not in node.children:
                node.children[char] = TrieNode()
            node = node.children[char]
        node.end_of_word = True

    def search(self, word):
        node = self.root
        for char in word:
            if char not in node.children:
                return False
            node = node.children[char]
        return node.end_of_word

    def is_prefix(self, word):
        node = self.root
        for char in word:
            if char not in node.children:
                return False
            elif node.children[char].end_of_word:
                return True
            node = node.children[char]
        return False


t = int(input())
for _ in range(t):
    n = int(input())
    phone_numbers = [input() for _ in range(n)]
    phone_numbers.sort()
    trie = Trie()
    consistent = True

    for phone in phone_numbers:
        if trie.is_prefix(phone):
            consistent = False
            break
        trie.insert(phone)
    if consistent:
        print("YES")
    else:
        print("NO")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44670693提交状态

状态: Accepted

源代码

```
class TrieNode:
    def __init__(self):
        self.children = {}
        self.end_of_word = False

class Trie:
    def __init__(self):
        self.root = TrieNode()

    def insert(self, word):
        node = self.root
        for char in word:
            if char not in node.children:
                node.children[char] = TrieNode()
            node = node.children[char]
        node.end_of_word = True

    def search(self, word):
        node = self.root
        for char in word:
            if char not in node.children:
                return False
            node = node.children[char]
        return node.end_of_word
```

基本信息

#:	44670693
题目:	04089
提交人:	23n2300011030(陈奕好)
内存:	24744kB
时间:	300ms
语言:	Python3
提交时间:	2024-04-16 08:51:50

04082: 树的镜面映射

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

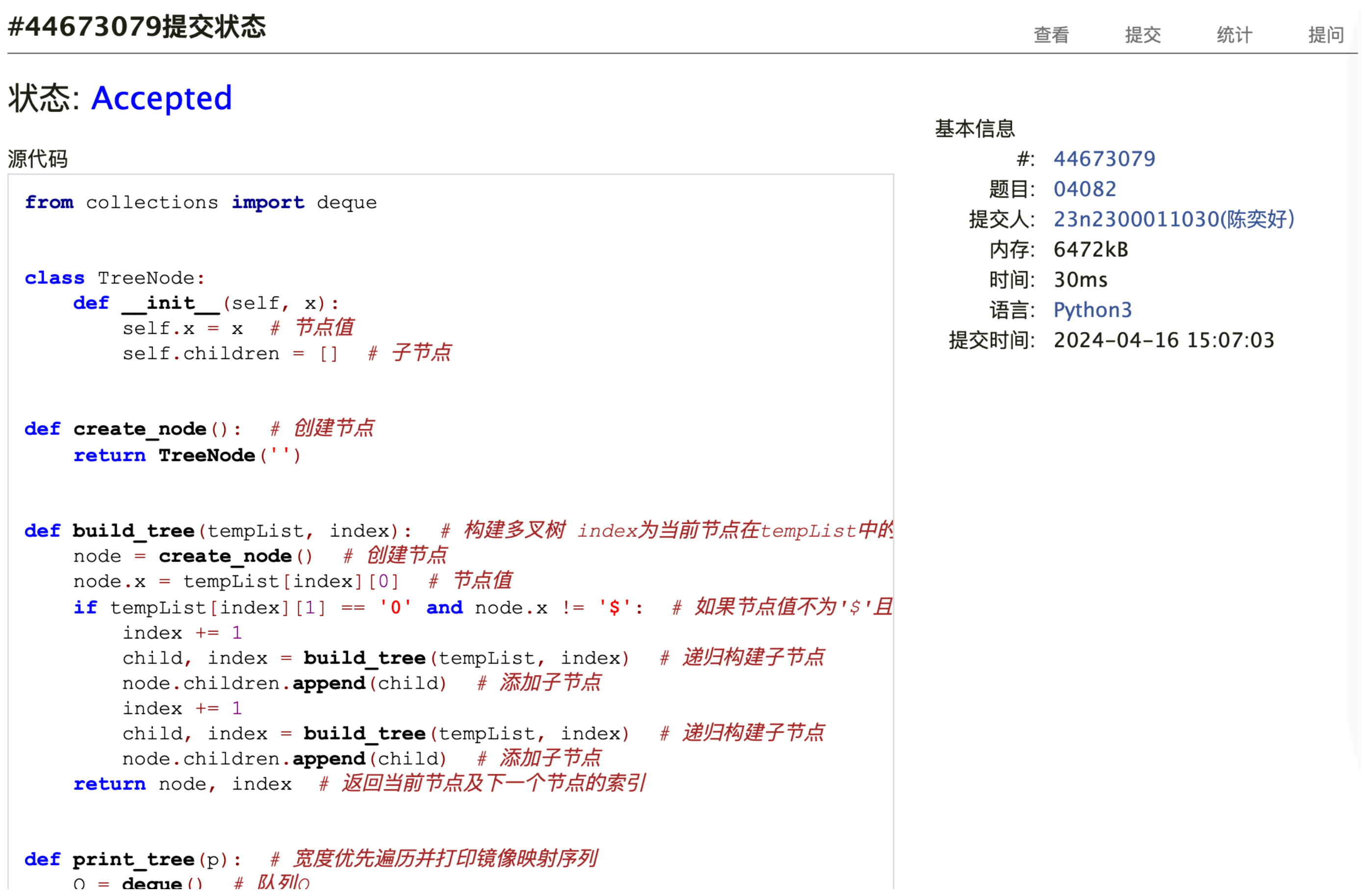
思路：做了很久，表示不出来。呼之欲出，求之不得。

代码

```python
from collections import deque


class TreeNode:
    def __init__(self, x):
        self.x = x  # 节点值
        self.children = []  # 子节点


def create_node():  # 创建节点
    return TreeNode('')


def build_tree(tempList, index):  # 构建多叉树 index为当前节点在tempList中的索引
    node = create_node()  # 创建节点
    node.x = tempList[index][0]  # 节点值
    if tempList[index][1] == '0' and node.x != '$':  # 如果节点值不为'$'且有子节点
```

```python
        index += 1
        child, index = build_tree(tempList, index)  # 递归构建子节点
        node.children.append(child)  # 添加子节点
        index += 1
        child, index = build_tree(tempList, index)  # 递归构建子节点
        node.children.append(child)  # 添加子节点
    return node, index  # 返回当前节点及下一个节点的索引


def print_tree(p):  # 宽度优先遍历并打印镜像映射序列
    Q = deque()  # 队列Q
    s = deque()  # 栈s

    # 遍历右子节点并将非虚节点加入栈s
    while p is not None:
        if p.x != '$':
            s.append(p)
        p = p.children[1] if len(p.children) > 1 else None  # 右子节点

    # 将栈s中的节点逆序放入队列Q
    while s:
        Q.append(s.pop())

    # 宽度优先遍历队列Q并打印节点值
    while Q:
        p = Q.popleft()
        print(p.x, end=' ')

        # 如果节点有左子节点，将左子节点及其右子节点加入栈s
        if p.children:
            p = p.children[0]
            while p is not None:
                if p.x != '$':
                    s.append(p)
                p = p.children[1] if len(p.children) > 1 else None

            # 将栈s中的节点逆序放入队列Q
            while s:
                Q.append(s.pop())

# 读取输入
n = int(input())
tempList = input().split(' ')

# 构建多叉树
root, _ = build_tree(tempList, 0)

# 执行宽度优先遍历并打印镜像映射序列
print_tree(root)

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


#44673079提交状态

状态: Accepted

源代码

```
from collections import deque


class TreeNode:
    def __init__(self, x):
        self.x = x  # 节点值
        self.children = []  # 子节点


def create_node():  # 创建节点
    return TreeNode('')


def build_tree(tempList, index):  # 构建多叉树 index为当前节点在tempList中的
    node = create_node()  # 创建节点
    node.x = tempList[index][0]  # 节点值
    if tempList[index][1] == '0' and node.x != '$':  # 如果节点值不为'$'且
        index += 1
        child, index = build_tree(tempList, index)  # 递归构建子节点
        node.children.append(child)  # 添加子节点
        index += 1
        child, index = build_tree(tempList, index)  # 递归构建子节点
        node.children.append(child)  # 添加子节点
    return node, index  # 返回当前节点及下一个节点的索引


def print_tree(p):  # 宽度优先遍历并打印镜像映射序列
    Q = deque()  # 队列Q
```

基本信息

#:	44673079
题目:	04082
提交人:	23n2300011030(陈奕好)
内存:	6472kB
时间:	30ms
语言:	Python3
提交时间:	2024-04-16 15:07:03

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。

期中寄也结束了，回归刷题模式 \(0u0)/