Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Complied by 李佳霖，心理与认知科学学院

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 Sonama 14.3

Python编程环境：VSCode

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

28170: 算鹰

dfs, http://cs101.openjudge.cn/practice/28170/

思路：题干给的表述有点问题，以为是找上下左右都是'.'的棋子，在给的例子里也是8个，但实际上是找连通域的数量，就用dfs就可以解决。

代码

```
#
def count_eagles(board):
    def dfs(x, y):
```

```
        visited[x][y] = True

        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
            nx, ny = x + dx, y + dy
            # 确保新位置在棋盘范围内，并且是己方棋子，并且未被访问
            if 0 <= nx < 10 and 0 <= ny < 10 and board[nx][ny] == '.' and not visited[nx][ny]:
                dfs(nx, ny)

    visited = [[False] * 10 for _ in range(10)]
    eagle_count = 0

    for i in range(10):
        for j in range(10):
            if board[i][j] == '.' and not visited[i][j]:
                dfs(i, j)
                eagle_count += 1

    return eagle_count

grid = []
for i in range(10):
    grid.append(list(input()))
print(count_eagles(grid))
```

代码运行截图 （至少包含有"Accepted"）

#44871002提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def count_eagles(board):
    def dfs(x, y):
        visited[x][y] = True

        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
            nx, ny = x + dx, y + dy
            # 确保新位置在棋盘范围内，并且是己方棋子，并且未被访问
```

基本信息

#:	44871002
题目:	28170
提交人:	李佳霖2000013713
内存:	3936kB
时间:	23ms
语言:	Python3
提交时间:	2024-05-05 16:24:55

02754: 八皇后

dfs, http://cs101.openjudge.cn/practice/02754/

思路：老题新做

代码

```
#
ans = []
def queen_dfs(A, cur=0):
  if cur == len(A):
    ans.append(''.join([str(x+1) for x in A]))
    return
  for col in range(len(A)):
    for row in range(cur):
      # 检查之前的行是否有皇后在同一列或在同一对角线上
      if A[row] == col or abs(col - A[row]) == cur - row:
        break
    else:
      A[cur] = col #对应行的皇后所在的列
      queen_dfs(A, cur+1)

queen_dfs([None]*8)

for _ in range(int(input())):
  print(ans[int(input()) - 1])
```

代码运行截图 （至少包含有"Accepted"）

#44871435提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
ans = []
def queen_dfs(A, cur=0):
    if cur == len(A):
        ans.append(''.join([str(x+1) for x in A]))
        return
    for col in range(len(A)):
        for row in range(cur):
            # 检查之前的行是否有皇后在同一列或在同一对角线上
```

基本信息

#:	44871435
题目:	02754
提交人:	李佳霖2000013713
内存:	3616kB
时间:	30ms
语言:	Python3
提交时间:	2024-05-05 16:48:35

03151: Pots

bfs, http://cs101.openjudge.cn/practice/03151/

思路：和上周的bfs题目有一些异曲同工之处，也是通过优先队列的方式进行bfs。但乍一看这个题目第一次还没有想到该怎么用bfs做，还是需要培养一下敏感性

代码

```
#
from collections import deque

def bfs(a, b, c):
    queue = deque()
    # 初始化
    queue.append(((0, 0), []))

    visited = set()
    visited.add((0, 0))

    while queue:
        (current_a, current_b), operations = queue.popleft()

        if current_a == c or current_b == c:
            return operations

        # 定义可能的操作
        possible_operations = [
            ((a, current_b), operations + ['FILL(1)']),
            ((current_a, b), operations + ['FILL(2)']),
            ((0, current_b), operations + ['DROP(1)']),
            ((current_a, 0), operations + ['DROP(2)']),
            # b-current_b为还能往b里灌多少水，所以current_a - (b - current_b)是灌剩下的
            ((max(0, current_a - (b - current_b)), min(b, current_b + current_a)), operations + ['POUR(1,2)']), # 要么a空了（取max因为容量不能是负数），要么b满了（取min不能超出b的容量）
            ((min(a, current_a + current_b), max(0, current_b - (a - current_a))), operations + ['POUR(2,1)']) # 要么a满了，要么b空了
        ]

        # 将所有可能的操作加入队列
        for new_state, new_operations in possible_operations:
            if new_state not in visited:
                visited.add(new_state)
                queue.append((new_state, new_operations))

    return None

A, B, C = map(int, input().split())
result = bfs(A, B, C)
if result is None:
```

```
    print("impossible")
else:
    print(len(result))
    for operation in result:
        print(operation)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44872074提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

def bfs(a, b, c):
    queue = deque()
    # 初始化
    queue.append(((0, 0), []))
```

基本信息

#: 44872074
题目: 03151
提交人: 李佳霖2000013713
内存: 3672kB
时间: 20ms
语言: Python3
提交时间: 2024-05-05 17:16:39

05907: 二叉树的操作

http://cs101.openjudge.cn/practice/05907/

思路：题目思路比较直接，但是debug了好久，考虑交换节点时要想清楚

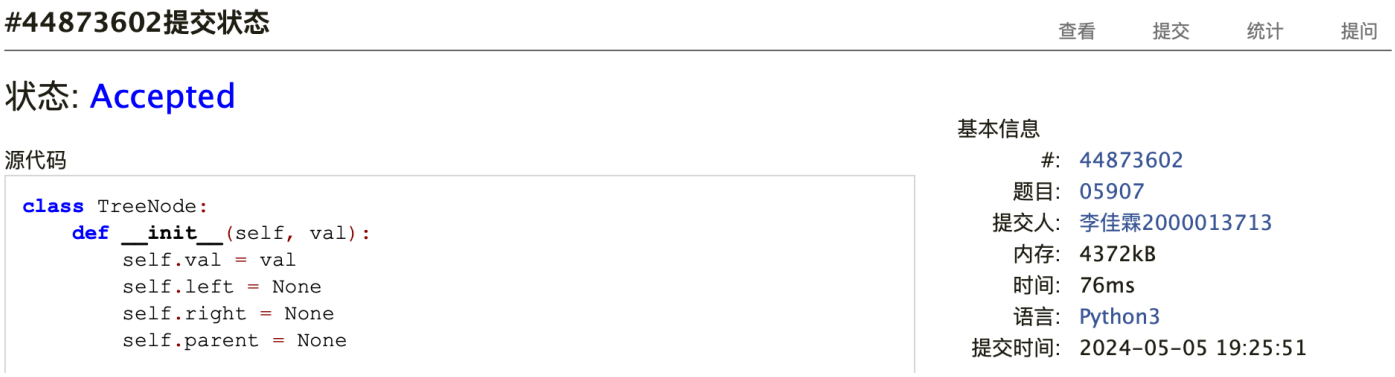
代码

```
#
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None
        self.parent = None

def find_leftmost(node):
    while node.left:
        node = node.left
    return node.val

def swap_nodes(nodes, x, y):
```

```
    node_x, node_y = nodes[x], nodes[y]
    parent_x, parent_y = node_x.parent, node_y.parent

    # 如果两个节点是兄弟节点
    if parent_x == parent_y:
        if parent_x.left == node_x:
            parent_x.left, parent_x.right = node_y, node_x
        else:
            parent_x.left, parent_x.right = node_x, node_y #同样的父节点记得要交换回去
        return

    # 更新父节点的子节点引用
    if parent_x:
        if parent_x.left == node_x:
            parent_x.left = node_y
        elif parent_x.right == node_x:
            parent_x.right = node_y

    if parent_y:
        if parent_y.left == node_y:
            parent_y.left = node_x
        elif parent_y.right == node_y:
            parent_y.right = node_x

    # 更新节点的父节点引用
    node_x.parent, node_y.parent = parent_y, parent_x

cases = int(input())
for case in range(cases):
    n, m = map(int, input().split())
    nodes = {i: TreeNode(i) for i in range(n)}
    root = nodes[0]
    for i in range(n):
        X, Y, Z = map(int, input().split())
        if Y != -1:
            nodes[X].left = nodes[Y]
            nodes[Y].parent = nodes[X]
        if Z != -1:
            nodes[X].right = nodes[Z]
            nodes[Z].parent = nodes[X]
    for j in range(m):
        operations = input().split()
        if int(operations[0]) == 1:
            x, y = int(operations[1]), int(operations[2])
            swap_nodes(nodes, x, y)
        if int(operations[0]) == 2:
            res = find_leftmost(nodes[int(operations[1])])
```

```
        print(res)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44873602提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None
        self.parent = None
```

基本信息

#: 44873602
题目: 05907
提交人: 李佳霖2000013713
内存: 4372kB
时间: 76ms
语言: Python3
提交时间: 2024-05-05 19:25:51

18250: 冰阔落 I

Disjoint set, http://cs101.openjudge.cn/practice/18250/

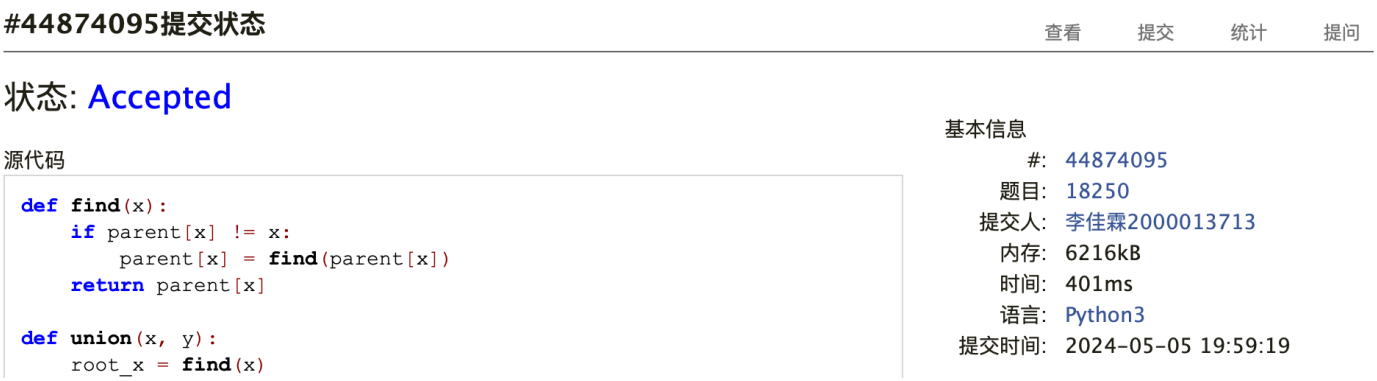
思路：看了题解，对于并查集问题还没有建立起感觉，还需要多加练习

代码

```
#
def find(x):
    if parent[x] != x:
        parent[x] = find(parent[x])
    return parent[x]

def union(x, y):
    root_x = find(x)
    root_y = find(y)
    if root_x != root_y:
        parent[root_y] = root_x

while True:
    try:
        n, m = map(int, input().split())
        parent = list(range(n + 1))
```

```
    for _ in range(m):
        a, b = map(int, input().split())
        if find(a) == find(b):
            print('Yes')
        else:
            print('No')
            union(a, b)

    unique_parents = set(find(x) for x in range(1, n + 1)) # 获取不同集合的根节点
    ans = sorted(unique_parents) # 输出有冰阔落的杯子编号
    print(len(ans))
    print(*ans)

except EOFError:
    break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44874095提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def find(x):
    if parent[x] != x:
        parent[x] = find(parent[x])
    return parent[x]

def union(x, y):
    root_x = find(x)
```

基本信息

#: 44874095
题目: 18250
提交人: 李佳霖2000013713
内存: 6216kB
时间: 401ms
语言: Python3
提交时间: 2024-05-05 19:59:19

05443: 兔子与樱花

http://cs101.openjudge.cn/practice/05443/

思路：和上周的兔子和樱花应该属于姊妹题目，上周需要找最小生成树，而这周需要借助diskstra算法来找出最短路径，还是需要通过字典来存储一个图，然后借助最小堆来实现一个优先队列，最后处理字符串的时候有一些麻烦

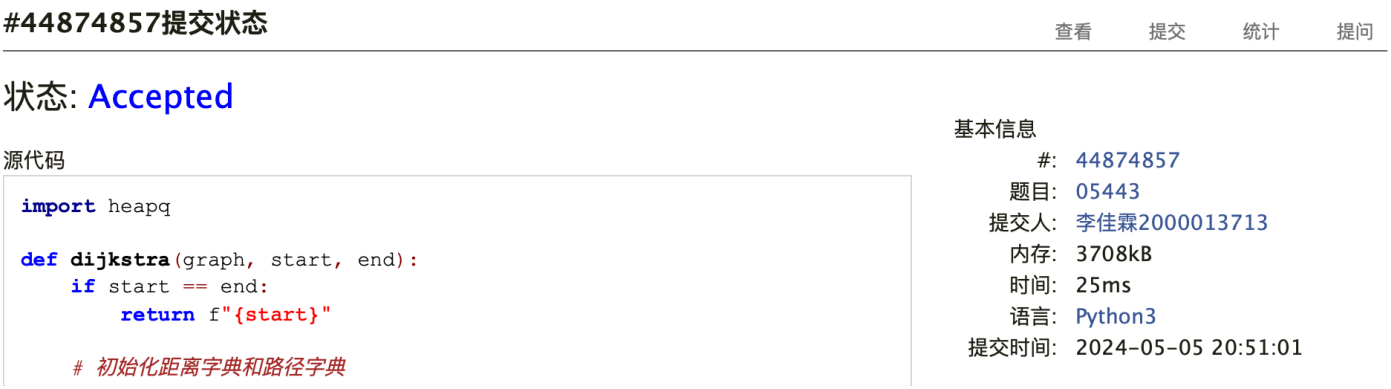
代码

```
#
import heapq
```

```python
def dijkstra(graph, start, end):
    if start == end:
        return f"{start}"

    # 初始化距离字典和路径字典
    dist = {vertex: float('inf') for vertex in graph}
    dist[start] = 0
    path = {vertex: ([]) for vertex in graph}
    path[start] = [(start, 0)] # 起点没有前一个节点，距离为0

    # 优先队列，用于存储待处理的顶点和距离
    priority_queue = []
    heapq.heappush(priority_queue, (0, start))

    while priority_queue:
        current_dist, current_vertex = heapq.heappop(priority_queue)

        # 遍历当前顶点的邻接顶点
        for neighbor, weight in graph[current_vertex].items():
            distance = current_dist + weight

            # 如果找到更短的路径，则更新
            if distance < dist[neighbor]:
                dist[neighbor] = distance
                path[neighbor] = path[current_vertex] + [(neighbor, weight)]
                heapq.heappush(priority_queue, (distance, neighbor))

        # 如果当前顶点是终点，格式化输出路径
        if current_vertex == end:
            return format_path(path[end])

    # 如果终点不可达，返回空字符串
    return ""

def format_path(path):
    if not path:
        return ""
    formatted_path = path[0][0] # 起始节点
    for vertex, weight in path[1:]:
        formatted_path += f"->({weight})->{vertex}"
    return formatted_path

P = int(input())
graph = {}
graph = {input(): {} for _ in range(P)}

Q = int(input())
```

```
for i in range(Q):
    start, end, cost = input().split()
    graph[start][end] = graph[end][start] = int(cost)

R = int(input())
for i in range(R):
    start, end = input().split()
    path = dijkstra(graph, start, end)
    print(path)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44874857提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq

def dijkstra(graph, start, end):
    if start == end:
        return f"{start}"

    # 初始化距离字典和路径字典
```

基本信息

#: 44874857
题目: 05443
提交人: 李佳霖2000013713
内存: 3708kB
时间: 25ms
语言: Python3
提交时间: 2024-05-05 20:51:01

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。

这周因为还在赶工毕设，所以画在数算的时间上比较少，但这周的几个题目都很有价值，让我逐渐找到了一点感觉，特别是之前不熟悉的算法类题目。等忙完这一阵子多花一些时间在数算上。