Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Complied by ==同学的姓名、院系==

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 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

28170: 算鹰

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

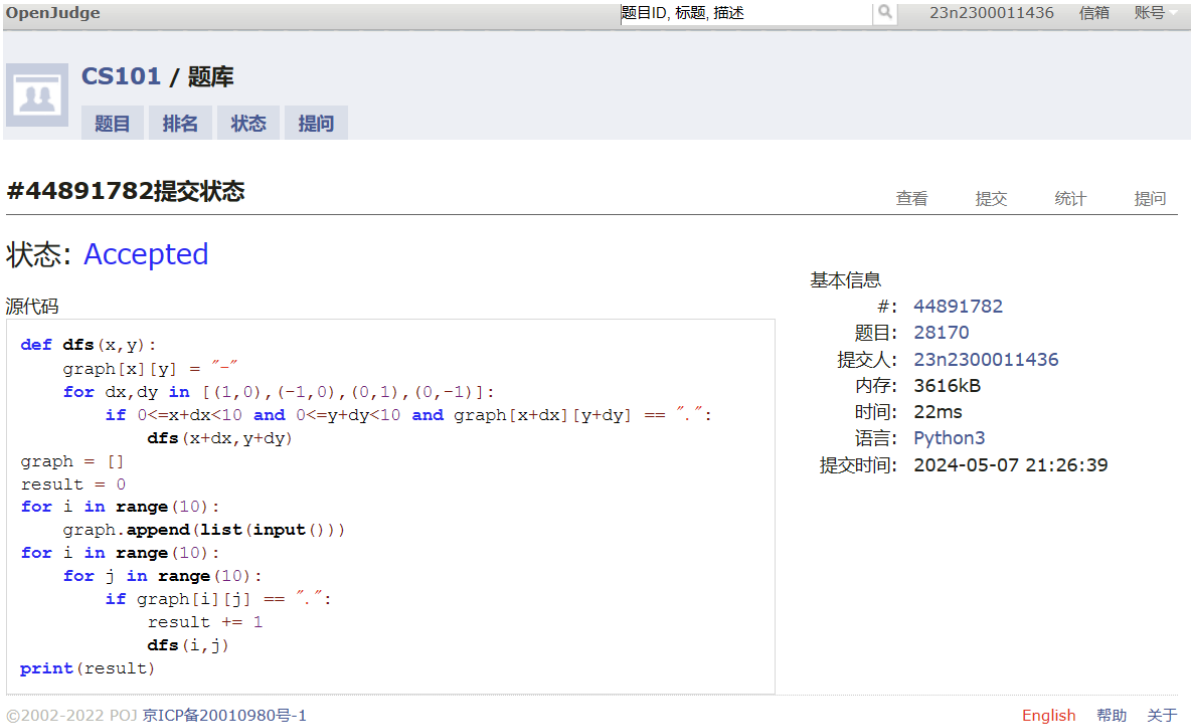
思路：

代码

```
def dfs(x,y):
    graph[x][y] = "-"
    for dx,dy in [(1,0),(-1,0),(0,1),(0,-1)]:
        if 0<=x+dx<10 and 0<=y+dy<10 and graph[x+dx][y+dy] == ".":
            dfs(x+dx,y+dy)
graph = []
result = 0
for i in range(10):
    graph.append(list(input()))
```

```python
for i in range(10):
    for j in range(10):
        if graph[i][j] == ".":
            result += 1
            dfs(i,j)
print(result)
```

代码运行截图



02754: 八皇后

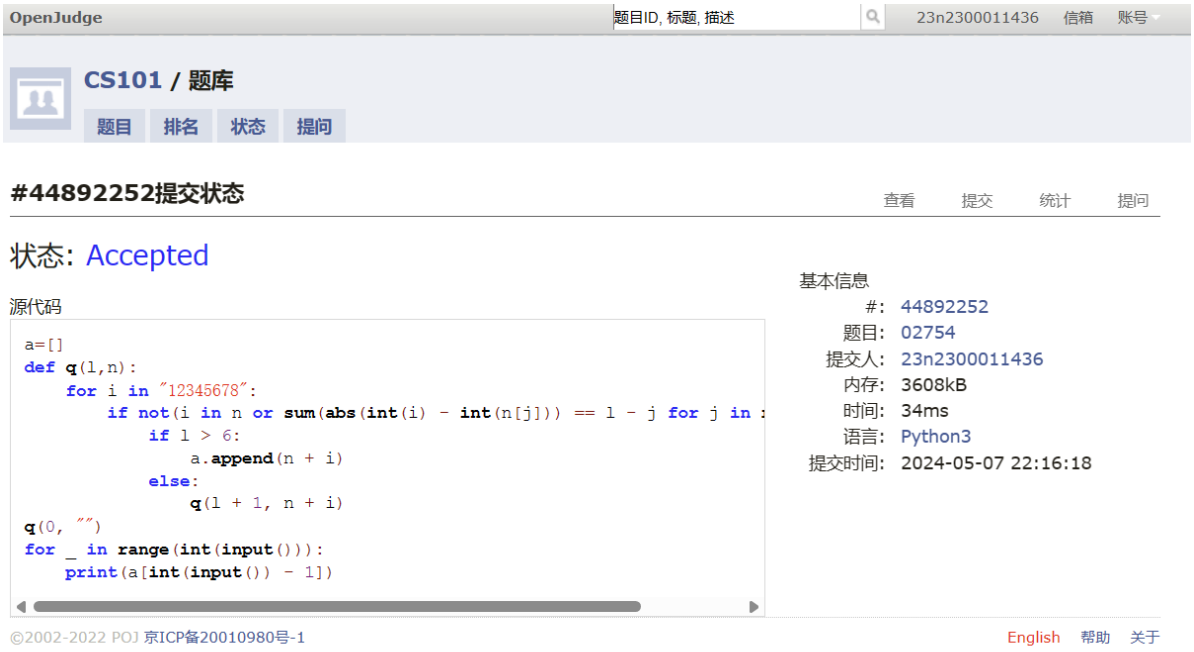
dfs, http://cs101.openjudge.cn/practice/02754/

思路:

代码

```
a=[]
def q(l,n):
    for i in "12345678":
        if not(i in n or sum(abs(int(i) - int(n[j])) == l - j for j in
range(l))):
            if l > 6:
                a.append(n + i)
            else:
                q(l + 1, n + i)
q(0, "")
for _ in range(int(input())):
    print(a[int(input()) - 1])
```

代码运行截图



03151: Pots

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

思路:

代码

```
# 23生科崔灏梵
def bfs(A, B, C):
    start = (0, 0)
    visited = set()
    visited.add(start)
```

```python
    queue = [(start, [])]

    while queue:
        (a, b), actions = queue.pop(0)

        if a == C or b == C:
            return actions

        next_states = [(A, b), (a, B), (0, b), (a, 0), (min(a + b, A),\
                max(0, a + b - A)), (max(0, a + b - B), min(a + b, B))]

        for i in next_states:
            if i not in visited:
                visited.add(i)
                new_actions = actions + [get_action(a, b, i)]
                queue.append((i, new_actions))

    return ["impossible"]


def get_action(a, b, next_state):
    if next_state == (A, b):
        return "FILL(1)"
    elif next_state == (a, B):
        return "FILL(2)"
    elif next_state == (0, b):
        return "DROP(1)"
    elif next_state == (a, 0):
        return "DROP(2)"
    elif next_state == (min(a + b, A), max(0, a + b - A)):
        return "POUR(2,1)"
    else:
        return "POUR(1,2)"


A, B, C = map(int, input().split())
solution = bfs(A, B, C)

if solution == ["impossible"]:
    print(solution[0])
else:
    print(len(solution))
    for i in solution:
        print(i)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

OpenJudge 题目ID, 标题, 描述 23n2300011436 信箱 账号

CS101 / 题库

题目 排名 状态 提问

#44892270提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
# 23生科崔灏梵
def bfs(A, B, C):
    start = (0, 0)
    visited = set()
    visited.add(start)
    queue = [(start, [])]

    while queue:
        (a, b), actions = queue.pop(0)

        if a == C or b == C:
            return actions

        next_states = [(A, b), (a, B), (0, b), (a, 0), (min(a + b, A),\
                max(0, a + b - A)), (max(0, a + b - B), min(a + b, B))]

        for i in next_states:
            if i not in visited:
                visited.add(i)
                new_actions = actions + [get_action(a, b, i)]
                queue.append((i, new_actions))

    return ["impossible"]
```

基本信息

#:	44892270
题目:	03151
提交人:	23n2300011436
内存:	3712kB
时间:	22ms
语言:	Python3
提交时间:	2024-05-07 22:18:13

05907: 二叉树的操作

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

思路:

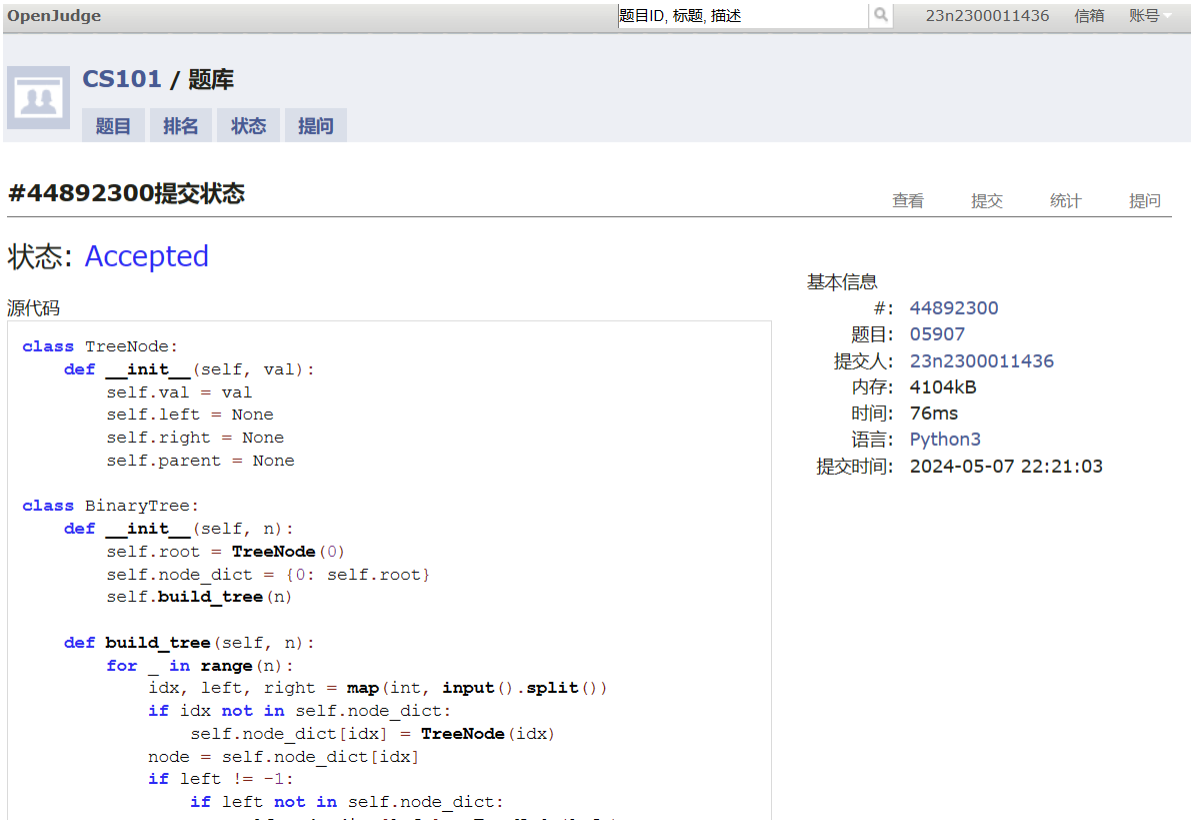
代码

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

class BinaryTree:
    def __init__(self, n):
        self.root = TreeNode(0)
        self.node_dict = {0: self.root}
        self.build_tree(n)

    def build_tree(self, n):
        for _ in range(n):
```

```python
            idx, left, right = map(int, input().split())
            if idx not in self.node_dict:
                self.node_dict[idx] = TreeNode(idx)
            node = self.node_dict[idx]
            if left != -1:
                if left not in self.node_dict:
                    self.node_dict[left] = TreeNode(left)
                left_node = self.node_dict[left]
                node.left = left_node
                left_node.parent = node
            if right != -1:
                if right not in self.node_dict:
                    self.node_dict[right] = TreeNode(right)
                right_node = self.node_dict[right]
                node.right = right_node
                right_node.parent = node

    def swap_nodes(self, x, y):
        node_x = self.node_dict[x]
        node_y = self.node_dict[y]
        px, py = node_x.parent, node_y.parent

        if px == py:
            px.left, px.right = px.right, px.left
            return

        # Swap in the parent's children references
        if px.left == node_x:
            px.left = node_y
        else:
            px.right = node_y

        if py.left == node_y:
            py.left = node_x
        else:
            py.right = node_x

        # Swap their parent references
        node_x.parent, node_y.parent = py, px

    def find_leftmost_child(self, x):
        node = self.node_dict[x]
        while node.left:
            node = node.left
        return node.val

def main():
    t = int(input())
    for _ in range(t):
        n, m = map(int, input().split())
        tree = BinaryTree(n)
        for _ in range(m):
            op, *args = map(int, input().split())
            if op == 1:
                x, y = args
```

```
                tree.swap_nodes(x, y)
            elif op == 2:
                x, = args
                print(tree.find_leftmost_child(x))

if __name__ == "__main__":
    main()
```

代码运行截图 == （AC代码截图，至少包含有"Accepted"） ==



18250: 冰阔落 I

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

思路:

代码

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

```python
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"） ==

OpenJudge 题目ID, 标题, 描述 23n2300011436 信箱 账号

CS101 / 题库

题目 排名 状态 提问

#44892326提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```python
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)
```

基本信息

#:	44892326
题目:	18250
提交人:	23n2300011436
内存:	6092kB
时间:	375ms
语言:	Python3
提交时间:	2024-05-07 22:25:16

05443: 兔子与樱花

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

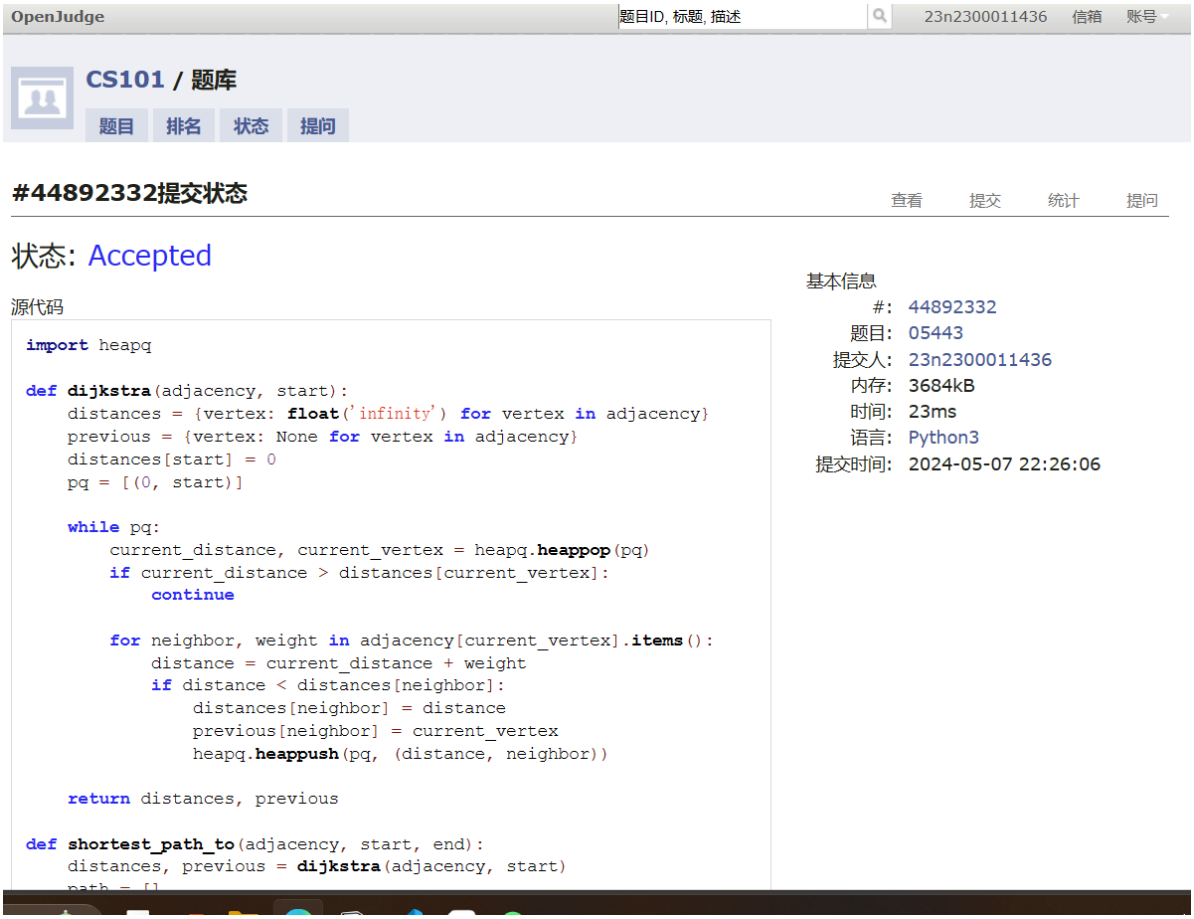
思路:

代码

```
import heapq

def dijkstra(adjacency, start):
    distances = {vertex: float('infinity') for vertex in adjacency}
    previous = {vertex: None for vertex in adjacency}
    distances[start] = 0
    pq = [(0, start)]

    while pq:
        current_distance, current_vertex = heapq.heappop(pq)
        if current_distance > distances[current_vertex]:
            continue

        for neighbor, weight in adjacency[current_vertex].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous[neighbor] = current_vertex
                heapq.heappush(pq, (distance, neighbor))

    return distances, previous

def shortest_path_to(adjacency, start, end):
    distances, previous = dijkstra(adjacency, start)
    path = []
    current = end
    while previous[current] is not None:
        path.insert(0, current)
        current = previous[current]
    path.insert(0, start)
    return path, distances[end]

# Read the input data
P = int(input())
places = {input().strip() for _ in range(P)}

Q = int(input())
graph = {place: {} for place in places}
for _ in range(Q):
    src, dest, dist = input().split()
    dist = int(dist)
    graph[src][dest] = dist
```

```python
    graph[dest][src] = dist  # Assuming the graph is bidirectional

R = int(input())
requests = [input().split() for _ in range(R)]

# Process each request
for start, end in requests:
    if start == end:
        print(start)
        continue

    path, total_dist = shortest_path_to(graph, start, end)
    output = ""
    for i in range(len(path) - 1):
        output += f"{path[i]}->({graph[path[i]][path[i+1]]})->"
    output += f"{end}"
    print(output)
```

代码运行截图 == （AC代码截图，至少包含有"Accepted"） ==



2. 学习总结和收获

熟悉了dfs和bfs，学习了dijkstra算法