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==

罗瑞哲 生命科学学院

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：Windows 10

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()))
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 ==（至少包含有"Accepted"）==

#44884293提交状态

状态: Accepted

源代码

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

02754: 八皇后

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

思路:

代码

```
#
answer = []
def Queen(s):
    for col in range(1, 9):
        for j in range(len(s)):
            if (str(col) == s[j] or
                    abs(col - int(s[j])) == abs(len(s) - j)):
                break
        else:
            if len(s) == 7:
                answer.append(s + str(col))
```

```
            else:
                Queen(s + str(col))
Queen('')
n = int(input())
for _ in range(n):
    a = int(input())
    print(answer[a - 1])
```

代码运行截图 ==（至少包含有"Accepted"）==

#44884426提交状态

状态: Accepted

源代码

```
answer = []
def Queen(s):
    for col in range(1, 9):
        for j in range(len(s)):
            if (str(col) == s[j] or
                    abs(col - int(s[j])) == abs(len(s) - j)):
                break
        else:
            if len(s) == 7:
                answer.append(s + str(col))
            else:
                Queen(s + str(col))
Queen('')
n = int(input())
for _ in range(n):
    a = int(input())
    print(answer[a - 1])
```

03151: Pots

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

思路:

代码

```
#
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"]
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"）==

#44884317提交状态

状态: Accepted

源代码

```
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"]
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)
```

05907: 二叉树的操作

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

思路:

代码

```
#
def swap(x, y):
    tree[loc[x][0]][loc[x][1]] = y
    tree[loc[y][0]][loc[y][1]] = x
    loc[x], loc[y] = loc[y], loc[x]
for _ in range(int(input())):
    n, m = map(int, input().split())
    tree = {}
    loc = [[] for _ in range(n)]
    for _ in range(n):
        a, b, c = map(int, input().split())
        tree[a] = [b, c]
        loc[b], loc[c] = [a, 0], [a, 1]
    for _ in range(m):
        op = list(map(int, input().split()))
        if op[0] == 1:
            swap(op[1], op[2])
        else:
            cur = op[1]
            while tree[cur][0] != -1:
                cur = tree[cur][0]
            print(cur)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44884336提交状态

状态：Accepted

源代码

```
def swap(x, y):
    tree[loc[x][0]][loc[x][1]] = y
    tree[loc[y][0]][loc[y][1]] = x
    loc[x], loc[y] = loc[y], loc[x]
for _ in range(int(input())):
    n, m = map(int, input().split())
    tree = {}
    loc = [[] for _ in range(n)]
    for _ in range(n):
        a, b, c = map(int, input().split())
        tree[a] = [b, c]
        loc[b], loc[c] = [a, 0], [a, 1]
    for _ in range(m):
        op = list(map(int, input().split()))
        if op[0] == 1:
            swap(op[1], op[2])
        else:
            cur = op[1]
            while tree[cur][0] != -1:
                cur = tree[cur][0]
            print(cur)
```

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]
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))
        ans = sorted(unique_parents)
        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"）==

#44884380提交状态

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)
    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))
        ans = sorted(unique_parents)
        print(len(ans))
        print(*ans)
    except EOFError:
        break
```

05443: 兔子与樱花

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

思路:

代码

```
#
import heapq
import math
def dijkstra(graph,start,end,P):
    if start == end: return []
    dist = {i:(math.inf,[]) for i in graph}
    dist[start] = (0,[start])
    pos = []
    heapq.heappush(pos,(0,start,[]))
    while pos:
```

```
        dist1,current,path = heapq.heappop(pos)
        for (next,dist2) in graph[current].items():
            if dist2+dist1 < dist[next][0]:
                dist[next] = (dist2+dist1,path+[next])
                heapq.heappush(pos,(dist1+dist2,next,path+[next]))
    return dist[end][1]
P = int(input())
graph = {input():{} for _ in range(P)}
for _ in range(int(input())):
    place1,place2,dist = input().split()
    graph[place1][place2] = graph[place2][place1] = int(dist)
for _ in range(int(input())):
    start,end = input().split()
    path = dijkstra(graph,start,end,P)
    s = start
    current = start
    for i in path:
        s += f'->({graph[current][i]})->{i}'
        current = i
    print(s)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44884393提交状态

状态：Accepted

源代码

```
import heapq
import math
def dijkstra(graph,start,end,P):
    if start == end: return []
    dist = {i:(math.inf,[]) for i in graph}
    dist[start] = (0,[start])
    pos = []
    heapq.heappush(pos,(0,start,[]))
    while pos:
        dist1,current,path = heapq.heappop(pos)
        for (next,dist2) in graph[current].items():
            if dist2+dist1 < dist[next][0]:
                dist[next] = (dist2+dist1,path+[next])
                heapq.heappush(pos,(dist1+dist2,next,path+[next]))
    return dist[end][1]
P = int(input())
graph = {input():{} for _ in range(P)}
for _ in range(int(input())):
    place1,place2,dist = input().split()
    graph[place1][place2] = graph[place2][place1] = int(dist)
for _ in range(int(input())):
    start,end = input().split()
    path = dijkstra(graph,start,end,P)
    s = start
    current = start
    for i in path:
        s += f'->({graph[current][i]})->{i}'
        current = i
    print(s)
```

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。==

关于图、树的相关内容仍需加强。