# 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：

直接dfs然后把走过的标记成0再加入has_vis就可以避免重走连通域同时方便计数

代码

```
'''
2100017810 刘思瑞
'''
m = []
for i in range(10):
    s = input()
    temp = []
    for j in range(10):
        if s[j] == '-':
            temp.append(0)
        else:
```

```
            temp.append(1)
    m.append(temp)
step = [(-1,0),(0,-1),(1,0),(0,1)]
has_vis = set()

def dfs(i,j):
    global m,step,has_vis
    if m[i][j] == 0:
        return
    m[i][j] = 0
    has_vis.add((i,j))
    for _ in step:
        i_,j_ = i+_[0], j+ _[1]
        if i_ < 10 and i_ >=0 and j_ <10 and j_ >= 0 and not((i_,j_) in
has_vis) :
            dfs(i_,j_)

count = 0
for i in range(10):
    for j in range(10):
        if m[i][j] == 1:
            dfs(i,j)
            count += 1
print(count)
```

代码运行截图

## 状态: Accepted

原代码

```
'''
2100017810 刘思瑞
'''
m = []
for i in range(10):
    s = input()
    temp = []
    for j in range(10):
        if s[j] == '-':
            temp.append(0)
        else:
            temp.append(1)
    m.append(temp)
step = [(-1,0),(0,-1),(1,0),(0,1)]
has_vis = set()

def dfs(i,j):
    global m,step,has_vis
    if m[i][j] == 0:
        return
    m[i][j] = 0
    has_vis.add((i,j))
    for _ in step:
        i_,j_ = i+_[0], j+ _[1]
        if i_ < 10 and i_ >=0 and j_ <10 and j_ >= 0 and not((i_,j_) in
            dfs(i_,j_)

count = 0
for i in range(10):
    for j in range(10):
        if m[i][j] == 1:
            dfs(i,j)
            count += 1
print(count)
```

## 02754: 八皇后

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

思路:

直接递归就可以

代码

```
'''
刘思瑞 2100017810
'''
```

```
def search(queen,i):
    global ans
    if i == 8:
        s=''
        for i in queen:
            s += str(i)
        ans.append(int(s))
        return
    rest = [1,2,3,4,5,6,7,8]
    for j in range(i):
        for _ in [queen[j],queen[j]+i-j,queen[j]-i+j]:
            if _ in rest:
                rest.remove(_)
    for j in rest:
        search(queen+[j],i+1)

ans = []
search([],0)
num = int(input())
for i in range(num):
    print(ans[int(input())-1])
```

代码运行截图

---

## 状态：Accepted

基

源代码

```
'''
刘思瑞 2100017810
'''
def search(queen,i):
    global ans
    if i == 8:
        s=''
        for i in queen:
            s += str(i)
        ans.append(int(s))
        return
    rest = [1,2,3,4,5,6,7,8]
    for j in range(i):
        for _ in [queen[j],queen[j]+i-j,queen[j]-i+j]:
            if _ in rest:
                rest.remove(_)
    for j in rest:
        search(queen+[j],i+1)

ans = []
search([],0)
num = int(input())
for i in range(num):
    print(ans[int(input())-1])
```

# 03151: Pots

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

思路:

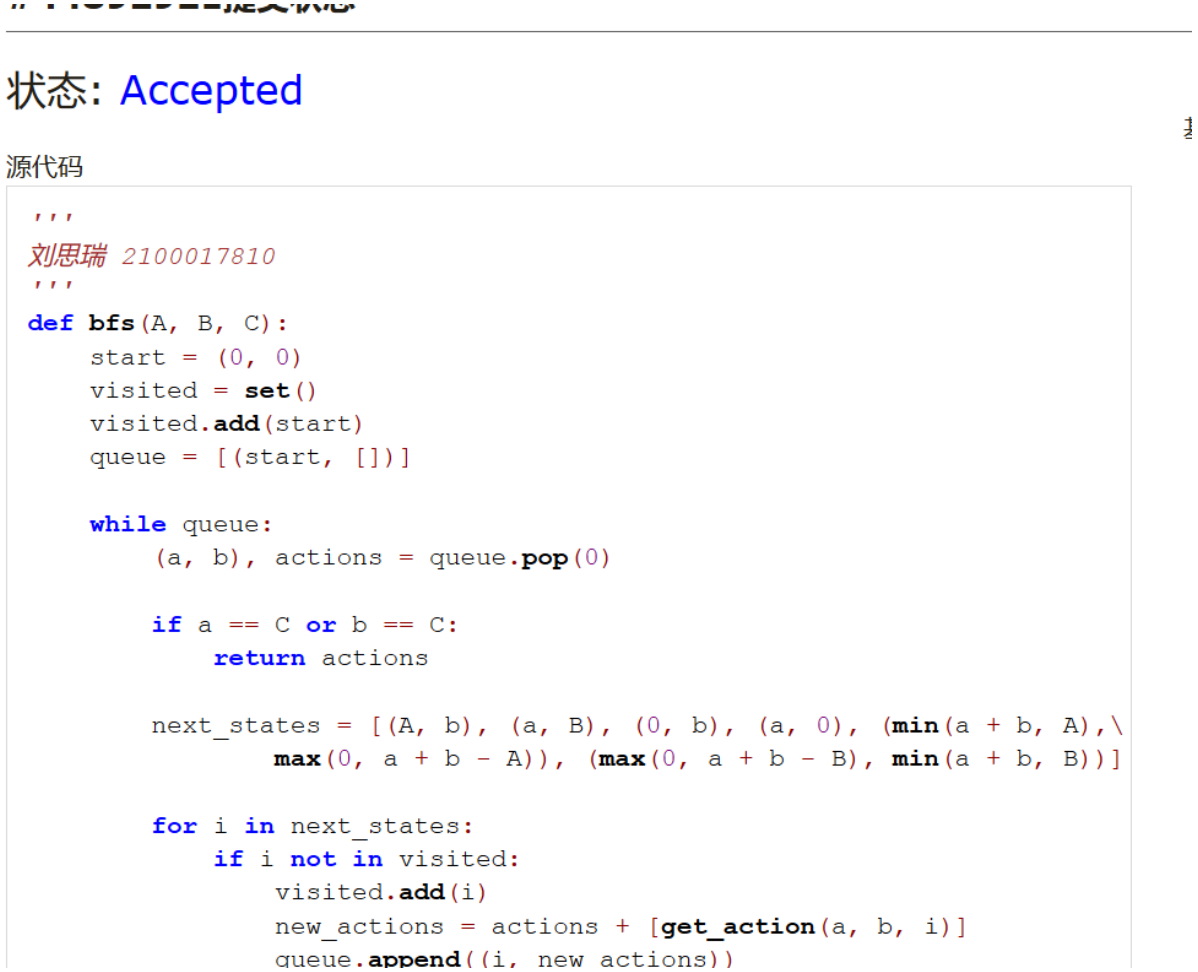
bfs即可

代码

```
'''
刘思瑞 2100017810
'''
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 state in next_states:
            if state not in visited:
                visited.add(state)
                new_actions = actions + [get_action(a, b, state)]
                queue.append((state, new_actions))

    return ["impossible"]


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


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

```
44
45 if solution == ["impossible"]:
46     print(solution[0])
47 else:
48     print(len(solution))
49     for action in solution:
50         print(action)
```

代码运行截图




## 05907: 二叉树的操作

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

思路:

正常操作树

代码

```
1 '''
2 刘思瑞 2100017810
3 '''
```

```
def change(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):
        node, left_child, right_child = map(int, input().split())
        tree[node] = [left_child, right_child]
        loc[left_child], loc[right_child] = [node, 0], [node, 1]

    for _ in range(m):
        op = list(map(int, input().split()))
        if op[0] == 1:
            change(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

---

# 状态：Accepted

源代码

```
'''
刘思瑞 2100017810
'''
def change(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):
        node, left_child, right_child = map(int, input().split())
        tree[node] = [left_child, right_child]
        loc[left_child], loc[right_child] = [node, 0], [node, 1]
```

## 18250: 冰阔落 I

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

思路:

并查集

代码

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

def merge_sets(x, y):
    root_x = find_root(x)
    root_y = find_root(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_root(a) == find_root(b):
                print('Yes')
            else:
                print('No')
                merge_sets(a, b)

        unique_roots = set(find_root(x) for x in range(1, n + 1))
        sorted_roots = sorted(unique_roots)
        print(len(sorted_roots))
        print(*sorted_roots)

    except EOFError:
        break
```

代码运行截图

## 状态: Accepted

源代码

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

def merge_sets(x, y):
    root_x = find_root(x)
    root_y = find_root(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_root(a) == find_root(b):
                print('Yes')
            else:
                print('No')
                merge_sets(a, b)

        unique_roots = set(find_root(x) for x in range(1, n + 1))
        sorted_roots = sorted(unique_roots)
        print(len(sorted_roots))
```

## 05443: 兔子与樱花

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

思路:

是按dijstrack写的但是不知道为什么自己写的一直找不到bug

代码

```
import heapq
import math

def dijkstra(graph, start, end):
    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_node, dist2 in graph[current].items():
            if dist2 + dist1 < dist[next_node][0]:
                dist[next_node] = (dist2 + dist1, path + [next_node])
                heapq.heappush(pos, (dist1 + dist2, next_node, path +
[next_node]))
    return dist[end][1]

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

for _ in range(int(input())):
    place1, place2, dist = input().split()
    dist = int(dist)
    graph[place1][place2] = graph[place2][place1] = dist

for _ in range(int(input())):
    start, end = input().split()
    path = dijkstra(graph, start, end)
    s = start
    current = start
    for i in path:
        s += f'->({graph[current][i]})->{i}'
        current = i
    print(s)
```

代码运行截图

## 状态：Accepted

源代码

```
import heapq
import math

def dijkstra(graph, start, end):
    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_node, dist2 in graph[current].items():
            if dist2 + dist1 < dist[next_node][0]:
                dist[next_node] = (dist2 + dist1, path + [next_node])
                heapq.heappush(pos, (dist1 + dist2, next_node, path +
    return dist[end][1]

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

for _ in range(int(input())):
    place1, place2, dist = input().split()
    dist = int(dist)
    graph[place1][place2] = graph[place2][place1] = dist
```

# 2. 学习总结和收获

bfs和dfs相当于上学期复习了。主要卡在最后一道题，是按dijstrack写的但是不知道为什么自己写的一直找不到bug，最后也是模仿标准答案写了一个。