

Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Compiled 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

```
1  '''
2  2100017810 刘思瑞
3  '''
4  m = []
5  for i in range(10):
6      s = input()
7      temp = []
8      for j in range(10):
9          if s[j] == '-':
10             temp.append(0)
11         else:
```

```
12         temp.append(1)
13     m.append(temp)
14     step = [(-1,0),(0,-1),(1,0),(0,1)]
15     has_vis = set()
16
17     def dfs(i,j):
18         global m,step,has_vis
19         if m[i][j] == 0:
20             return
21         m[i][j] = 0
22         has_vis.add((i,j))
23         for _ in step:
24             i_,j_ = i+_ [0], j+_ [1]
25             if i_ < 10 and i_ >=0 and j_ <10 and j_ >= 0 and not((i_,j_) in
has_vis) :
26                 dfs(i_,j_)
27
28     count = 0
29     for i in range(10):
30         for j in range(10):
31             if m[i][j] == 1:
32                 dfs(i,j)
33             count += 1
34     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

```
1 '''
2 刘思瑞 2100017810
3 '''
```

```

4 def search(queen,i):
5     global ans
6     if i == 8:
7         s=''
8         for i in queen:
9             s += str(i)
10            ans.append(int(s))
11            return
12    rest = [1,2,3,4,5,6,7,8]
13    for j in range(i):
14        for _ in [queen[j],queen[j]+i-j,queen[j]-i+j]:
15            if _ in rest:
16                rest.remove(_)
17    for j in rest:
18        search(queen+[j],i+1)
19
20 ans = []
21 search([],0)
22 num = int(input())
23 for i in range(num):
24     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])

```

1

03151: Pots

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

思路:

bfs即可

代码

```
1  '''
2  刘思瑞 2100017810
3  '''
4  def bfs(A, B, C):
5      start = (0, 0)
6      visited = set()
7      visited.add(start)
8      queue = [(start, [])]
9
10     while queue:
11         (a, b), actions = queue.pop(0)
12
13         if a == C or b == C:
14             return actions
15
16         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))]
17
18         for state in next_states:
19             if state not in visited:
20                 visited.add(state)
21                 new_actions = actions + [get_action(a, b, state)]
22                 queue.append((state, new_actions))
23
24     return ["impossible"]
25
26
27 def get_action(a, b, next_state):
28     if next_state == (A, b):
29         return "FILL(A)"
30     elif next_state == (a, B):
31         return "FILL(B)"
32     elif next_state == (0, b):
33         return "EMPTY(A)"
34     elif next_state == (a, 0):
35         return "EMPTY(B)"
36     elif next_state == (min(a + b, A), max(0, a + b - A)):
37         return "POUR(B->A)"
38     else:
39         return "POUR(A->B)"
40
41
42 A, B, C = map(int, input().split())
43 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

刘思瑞 2100017810

状态: Accepted

源代码

```

'''
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 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))

```

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 '''

```

```

4  def change(x, y):
5      tree[loc[x][0]][loc[x][1]] = y
6      tree[loc[y][0]][loc[y][1]] = x
7      loc[x], loc[y] = loc[y], loc[x]
8
9
10 for _ in range(int(input())):
11     n, m = map(int, input().split())
12     tree = {}
13     loc = [[] for _ in range(n)]
14
15     for _ in range(n):
16         node, left_child, right_child = map(int, input().split())
17         tree[node] = [left_child, right_child]
18         loc[left_child], loc[right_child] = [node, 0], [node, 1]
19
20     for _ in range(m):
21         op = list(map(int, input().split()))
22         if op[0] == 1:
23             change(op[1], op[2])
24         else:
25             cur = op[1]
26             while tree[cur][0] != -1:
27                 cur = tree[cur][0]
28             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

```
1 def find_root(x):
2     if parent[x] != x:
3         parent[x] = find_root(parent[x])
4     return parent[x]
5
6 def merge_sets(x, y):
7     root_x = find_root(x)
8     root_y = find_root(y)
9     if root_x != root_y:
10        parent[root_y] = root_x
11
12 while True:
13     try:
14         n, m = map(int, input().split())
15         parent = list(range(n + 1))
16
17         for _ in range(m):
18             a, b = map(int, input().split())
19             if find_root(a) == find_root(b):
20                 print('Yes')
21             else:
22                 print('No')
23                 merge_sets(a, b)
24
25         unique_roots = set(find_root(x) for x in range(1, n + 1))
26         sorted_roots = sorted(unique_roots)
27         print(len(sorted_roots))
28         print(*sorted_roots)
29
30 except EOFError:
31     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

```
1 import heapq
2 import math
3
4 def dijkstra(graph, start, end):
5     if start == end:
6         return []
7     dist = {i: (math.inf, []) for i in graph}
8     dist[start] = (0, [start])
9     pos = []
```

```

10     heapq.heappush(pos, (0, start, []))
11     while pos:
12         dist1, current, path = heapq.heappop(pos)
13         for next_node, dist2 in graph[current].items():
14             if dist2 + dist1 < dist[next_node][0]:
15                 dist[next_node] = (dist2 + dist1, path + [next_node])
16                 heapq.heappush(pos, (dist1 + dist2, next_node, path +
[next_node]))
17     return dist[end][1]
18
19 P = int(input())
20 graph = {input(): {} for _ in range(P)}
21
22 for _ in range(int(input())):
23     place1, place2, dist = input().split()
24     dist = int(dist)
25     graph[place1][place2] = graph[place2][place1] = dist
26
27 for _ in range(int(input())):
28     start, end = input().split()
29     path = dijkstra(graph, start, end)
30     s = start
31     current = start
32     for i in path:
33         s += f'->({graph[current][i]})->{i}'
34         current = i
35     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 +
                                     [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
```

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。