Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Complied by ==狄晨阳 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：Windows11

Python编程环境：Spyder IDE 5.4.3

C/C++编程环境：无

1. 题目

28170: 算鹰

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

思路：一个比较模版的dfs题目

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"）==

状态: Accepted

源代码

```
# -*- coding: utf-8 -*-
"""
Created on Tue May  7 23:02:09 2024

@author: 20311
"""

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)
```

基本信息

#:	44892592
题目:	28170
提交人:	23n2300012138(yukino)
内存:	3616kB
时间:	22ms
语言:	Python3
提交时间:	2024-05-07 23:02:57

02754: 八皇后

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

思路：上学期的时候写过，利用回溯法进行遍历

代码

```
#
ans=[]
q=[None]*8

def operate(q,x=0):
    if x==8:
        ans.append(''.join([str(it+1) for it in q]))
        return

    for y in range(8):
        for t in range(x):
            if y==q[t] or abs(x-t)==abs(y-q[t]):
                break
        else:
            q[x]=y
            operate(q,x+1)
```

```
operate(q)

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

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```
ans=[]
q=[None]*8

def operate(q,x=0):
    if x==8:
        ans.append(''.join([str(it+1) for it in q]))
        return

    for y in range(8):
        for t in range(x):
            if y==q[t] or abs(x-t)==abs(y-q[t]):
                break
        else:
            q[x]=y
            operate(q,x+1)

operate(q)

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

基本信息

#:	44892603
题目:	02754
提交人:	23n2300012138(yukino)
内存:	3632kB
时间:	30ms
语言:	Python3
提交时间:	2024-05-07 23:04:14

03151: Pots

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

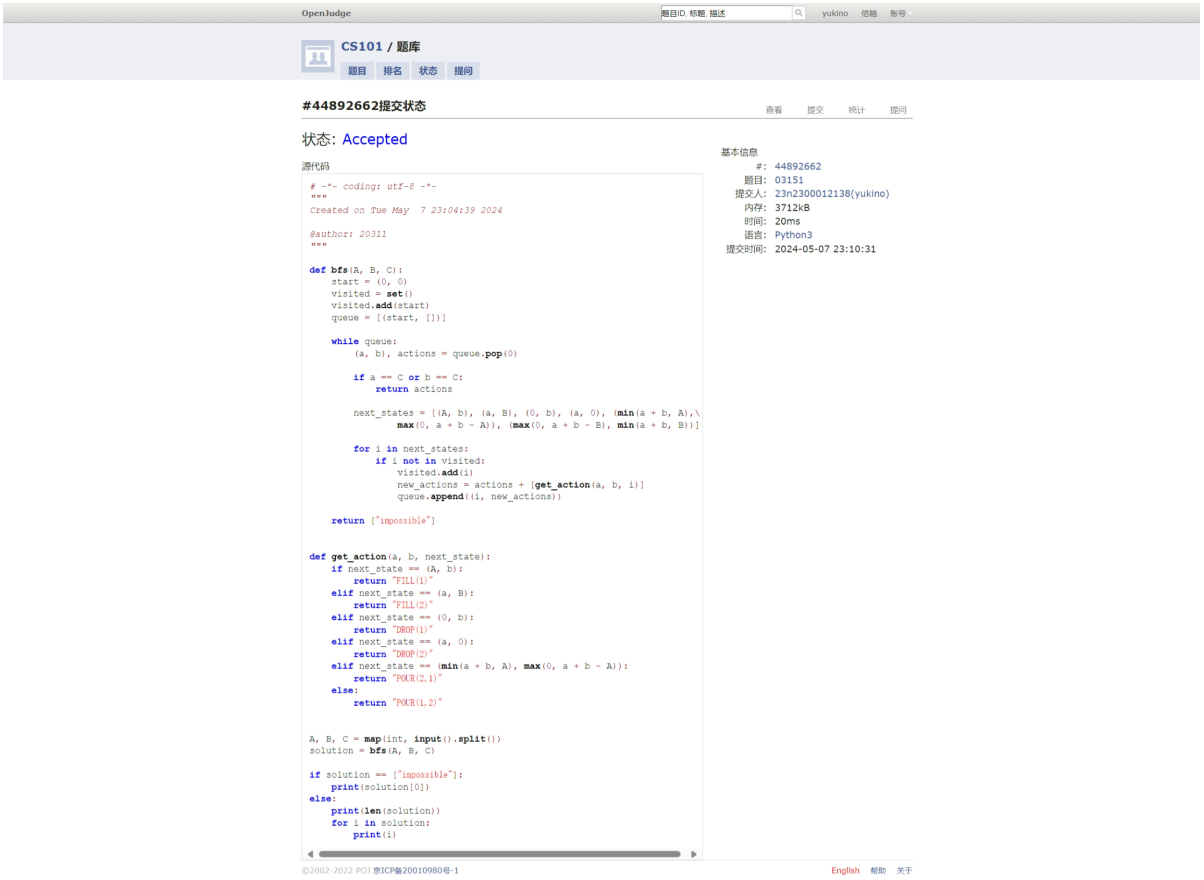
思路：利用bfs进行穷举来找到最短的步骤数，visited中储存的是两个锅中储存的水的数组

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Tue May  7 23:04:39 2024

@author: 20311
"""

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"）==



05907: 二叉树的操作

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

思路：直接用列表嵌套字典来建树，交换就直接交换key和value

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Tue May  7 23:21:25 2024

@author: 20311
"""

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"）==

状态: Accepted

源代码

```
# -*- coding: utf-8 -*-
"""
Created on Tue May  7 23:21:25 2024

@author: 20311
"""

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)
```

基本信息

#:	44892755
题目:	05907
提交人:	23n2300012138(yukino)
内存:	3836kB
时间:	72ms
语言:	Python3
提交时间:	2024-05-07 23:21:36

18250: 冰阔落 I

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

思路：一道比较典型的并查集题目

代码

```
# # -*- coding: utf-8 -*-
"""
```

```
Created on Tue May  7 23:29:48 2024

@author: 20311
"""

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"）==

状态: Accepted

基本信息

#: 44892854
题目: 18250
提交人: 23n2300012138(yukino)
内存: 5500kB
时间: 377ms
语言: Python3
提交时间: 2024-05-07 23:30:21

源代码

```
# -*- coding: utf-8 -*-
"""
Created on Tue May  7 23:29:48 2024

@author: 20311
"""

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/

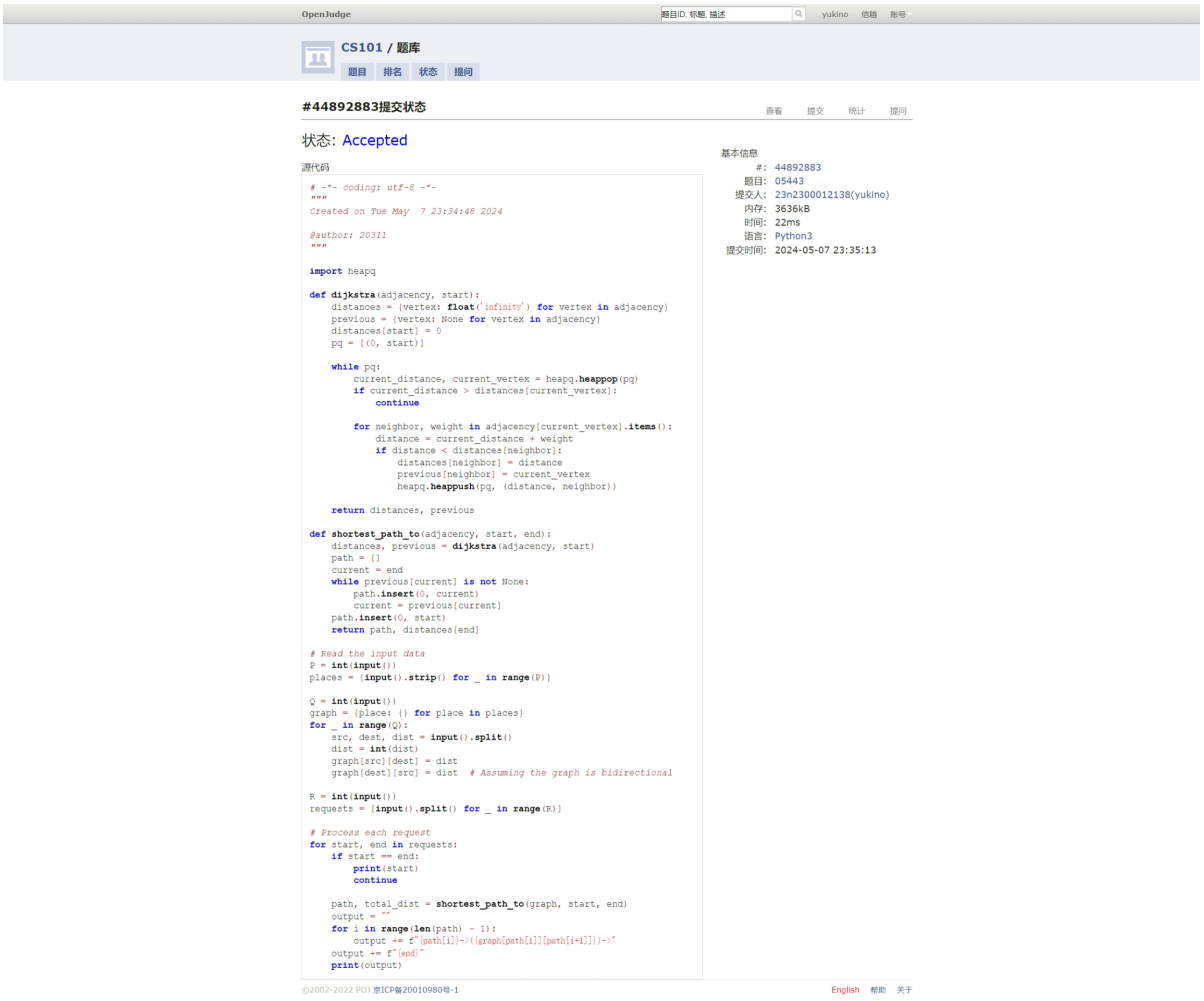
思路：使用dijkstra算法解决

代码

```
# # -*- coding: utf-8 -*-
"""
Created on Tue May  7 23:34:48 2024

@author: 20311
"""

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
    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"）==




```python
# -*- coding: utf-8 -*-
"""
Created on Tue May  7 23:34:48 2024

@author: 20311
"""

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
    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)
```

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。==

五一回校之后一直在发烧，康复之后时间比较赶，有些题目就直接抄了题解，现在基本能一下看出来不同题目应该用哪些算法来做，但是手搓的速度和熟练度准确度还需要提升