Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Complied by 赵策 数学科学学院

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

操作系统：Windows 11

Python编程环境：Visual Studio Code 1.86.2

1. 题目

28170: 算鹰

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

思路：

代码

```
#
def dfs(i,j):
    if 0<=i<10 and 0<=j<10 and not visited[i][j] and board[i][j]=='.':
        visited[i][j]=True
        dfs(i-1,j)
        dfs(i+1,j)
        dfs(i,j-1)
        dfs(i,j+1)
    return

board=[list(input()) for _ in range(10)]
visited=[[False]*10 for _ in range(10)]
count=0
for i in range(10):
    for j in range(10):
```

```
        if board[i][j]=='.' and not visited[i][j]:
            dfs(i,j)
            count+=1
print(count)
```

代码运行截图 （至少包含有"Accepted"）

02754: 八皇后

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

思路：

代码

```
#
from itertools import permutations
def is_valid(board):
    for i in range(n):
        for j in range(i+1,n):
            if board[j]==board[i] or abs(board[j]-board[i])==j-i:
                return False
    return True

def solve_n_queens(n,b):
    perms=list(permutations(range(n)))
    solutions=[i for i in perms if is_valid(i)]
    return solutions[b-1]

n=8  # Eight queens
for _ in range(int(input())):
    solution=[i+1 for i in solve_n_queens(n,int(input()))]
    print(''.join(map(str,solution)))
```

代码运行截图 （至少包含有"Accepted"）

03151: Pots

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

思路:

代码

```
#
def pour_water(A,B,C):
    visited=set()
    queue=[(0,0,[])]
    while queue:
        a,b,operations=queue.pop(0)
        if a==C or b==C:
            return operations
        if (a,b) in visited:
            continue
        visited.add((a,b))
        if a<A:
            queue.append((A,b,operations+['FILL(1)']))  # 填充A
        if b<B:
            queue.append((a,B,operations+['FILL(2)']))  # 填充B
        if a>0:
            queue.append((0,b,operations+['DROP(1)']))  # 清空A
        if b>0:
            queue.append((a,0,operations+['DROP(2)']))  # 清空B
        if a>0 and b<B:
            pour_amount=min(a,B-b)
            queue.append((a-pour_amount,b+pour_amount,operations+['POUR(1,2)']))
 # 从A向B倒水
        if b>0 and a<A:
            pour_amount=min(b,A-a)
            queue.append((a+pour_amount,b-pour_amount,operations+['POUR(2,1)']))
 # 从B向A倒水
    return 'impossible'

A,B,C=map(int,input().split())
result=pour_water(A,B,C)
print(result if result=='impossible' else
str(len(result))+'\n'+'\n'.join(result))
```

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:

代码

```
#
class Node:
    def __init__(self,val):
        self.val=val
        self.left=None
        self.right=None

def swap_nodes(nodes,x,y):
    for node in nodes:
        if node.left and node.left.val in[x,y]:
            node.left=nodes[y] if node.left.val==x else nodes[x]
        if node.right and node.right.val in[x,y]:
            node.right=nodes[y] if node.right.val==x else nodes[x]

def find_leftmost(node):
    while node and node.left:
        node=node.left
    return node.val if node else -1

for _ in range(int(input())):
    n,m=map(int,input().split())
    nodes=list(map(Node,range(n)))
    for _ in range(n):
        x,y,z=map(int,input().split())
        if y!=-1:
            nodes[x].left=nodes[y]
        if z!=-1:
            nodes[x].right=nodes[z]
    for _ in range(m):
        operation=list(map(int,input().split()))
        if operation[0]==1:
            x,y=operation[1:]
            swap_nodes(nodes,x,y)
        else:
            print(find_leftmost(nodes[operation[1]]))
```

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_parent(i):
    if parents[i]!=i:
        parents[i]=find_parent(parents[i])
    return parents[i]

def union(x,y):
    rootX=find_parent(x)
    rootY=find_parent(y)
    parents[rootY]=rootX

while True:
    try:
        n,m=map(int,input().split())
        parents=list(range(n+1))
        rank=[0]*(n+1)
        for _ in range(m):
            x,y=map(int,input().split())
            if find_parent(x)==find_parent(y):
                print('Yes')
            else:
                print('No')
                union(x,y)
        roots={find_parent(i) for i in range(1,n+1)}
        print(len(roots))
        print(*sorted(roots),sep=' ')
    except EOFError:
        break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

05443: 兔子与樱花

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

思路:

代码

```
#
import heapq
def dijkstra(graph,start):
    distances={node: float('inf') for node in graph}
    distances[start]=0
    pq=[(0,start)]
    while pq:
        curr_dist,curr_node=heapq.heappop(pq)
        if curr_dist>distances[curr_node]:
            continue
        for neighbor,distance in graph[curr_node]:
            new_dist=curr_dist+distance
            if new_dist<distances[neighbor]:
                distances[neighbor]=new_dist
                heapq.heappush(pq,(new_dist,neighbor))
    return distances

n=int(input())
locations=[input() for _ in range(n)]
roads={}
for _ in range(int(input())):
    start,end,distance=input().split()
    distance=int(distance)
    if start in roads:
        roads[start].append((end,distance))
    else:
        roads[start]=[(end,distance)]
    if end in roads:
        roads[end].append((start,distance))
    else:
        roads[end]=[(start,distance)]
routes=[input().split() for _ in range(int(input()))]
graph={location:[] for location in locations}
for start,edges in roads.items():
    graph[start]=edges
for start,end in routes:
    distances=dijkstra(graph,start)
    shortest_distance=distances[end]
    path=[]
    curr_node=end
    while curr_node!=start:
        path.append(curr_node)
        neighbors=graph[curr_node]
        prev_node=None
        for neighbor,distance in neighbors:
            if distances[neighbor] + distance==distances[curr_node]:
                prev_node=neighbor
                break
        path.append(f'({distance})')
        curr_node=prev_node
    path.append(start)
```

```
    print('->'.join(reversed(path)))
```

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

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。

作业题写起来还蛮有意思，常用算法继续练习加强