Assignment #A: 图论：遍历，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by ==药学院 胡景博==

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

20743: 整人的提词本

http://cs101.openjudge.cn/practice/20743/

思路： 按题意操作反转，栈。

代码

```
# def reverse_parentheses(s):
    stack = []
    for char in s:
        if char == ')':
            temp = []
            while stack and stack[-1] != '(':
                temp.append(stack.pop())
            if stack:
                stack.pop()
            stack.extend(temp)

        else:
            stack.append(char)
    return ''.join(stack)
```

s = input().strip()print(reverse_parentheses(s))

代码运行截图 ==（至少包含有"Accepted"）==

#44785545提交状态

状态: Accepted

原代码

```
def reverse_parentheses(s):
    stack = []
    for char in s:
        if char == ')':
            temp = []
            while stack and stack[-1] != '(':
                temp.append(stack.pop())
            if stack:
                stack.pop()
            stack.extend(temp)

        else:
            stack.append(char)
    return ''.join(stack)


s = input().strip()
print(reverse_parentheses(s))
```

002-2022 PQ1 京ICP备20010980号-1

02255: 重建二叉树

http://cs101.openjudge.cn/practice/02255/

思路： 递归

代码

```
# def build_tree(preorder,inorder):
    if not preorder:
        return ''
    root = preorder[0]
    root_index = inorder.index(root)

    left_preorder = preorder[1:1+root_index]
    right_preorder = preorder[1+root_index:]

    left_inorder = inorder[:root_index]
    right_inorder = inorder[root_index+1:]

    left_tree = build_tree(left_preorder,left_inorder)
    right_tree = build_tree(right_preorder,right_inorder)
    return left_tree + right_tree + root



while True:
    try:
        preorder,inorder = input().split()
        post_tree = build_tree(preorder, inorder)
        print(post_tree)
    except EOFError:
        break
```

代码运行截图 ==（至少包含有"Accepted"）==

#44785854提交状态

状态: Accepted

源代码

```
def build_tree(preorder,inorder):
    if not preorder:
        return ''
    root = preorder[0]
    root_index = inorder.index(root)

    left_preorder = preorder[1:1+root_index]
    right_preorder = preorder[1+root_index:]

    left_inorder = inorder[:root_index]
    right_inorder = inorder[root_index+1:]

    left_tree = build_tree(left_preorder,left_inorder)
    right_tree = build_tree(right_preorder,right_inorder)
    return left_tree + right_tree + root



while True:
```

01426: Find The Multiple

http://cs101.openjudge.cn/practice/01426/

要求用 bfs 实现

思路： bfs

代码

import sysfrom collections import deque

sys.setrecursionlimit(1<<10)
flag = Falsedef main():
    global flag
    n = int(input())
    m = '1'
    if n == 0:

```
        flag = True
        return
    else:
        print(bfs_find(n,m))
        returndef check(n,m):
    if int(m)%n == 0:
        return True
    else:
        return Falsedef bfs_find(n,m):
    m = str(m)
    n = int(n)
    res = ''
    if int(m)%n == 0:

        return m
    queue = deque()
    queue.append(m)
    while queue:
        m = queue.popleft()
        if check(n,m):
            res = m
            break
        else:
            queue.append(m+'0')

            queue.append(m+'1')
    return mwhile not flag:
    main()
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
import sys
from collections import deque

sys.setrecursionlimit(1<<10)
flag = False
def main():
    global flag
    n = int(input())
    m = '1'
    if n == 0:
        flag = True
        return
    else:
        print(bfs_find(n,m))
        return
def check(n,m):
    if int(m)%n == 0:
        return True
    else:
```

04115: 鸣人和佐助

bfs, http://cs101.openjudge.cn/practice/04115/

思路： bfs+限定，开始用矩阵记录位置 TE，但超时，后来改用'@','+'记录位置。

代码

```
# from collections import dequedef is_vaild_move(x,y,M,N):
    return 0<=x<M and 0<=y<Ndef min_time_to_reach_sasuke(grid,M,N,T):
    directions = [(0,1),(0,-1),(1,0),(-1,0)]
    start_x,start_y = 0,0
    sasuke_x,sasuke_y = 0,0
    for i in range(M):
        for j in range(N):
            if grid[i][j] == '@':
                start_x,start_y = i,j
            elif grid[i][j] == '+':
```

```
                    sasuke_x,sasuke_y = i,j
    visited = set()
    queue = deque([(start_x,start_y,T,0)])
    while queue:
        x,y,chakra,time = queue.popleft()
        if (x,y) == (sasuke_x,sasuke_y):
            return time
        if (x,y,chakra) in visited:
            continue
        visited.add((x,y,chakra))
        for dx,dy in directions:
            new_x,new_y = x+dx,y+dy
            if is_vaild_move(new_x,new_y,M,N):
                if grid[new_x][new_y] == '#':
                    if chakra>0:
                        queue.append((new_x,new_y,chakra-1,time+1))
                else:
                    queue.append((new_x,new_y,chakra,time+1))
    return -1

M,N,T = map(int,input().split())
grid = [False]*Mfor i in range(M):
    grid[i] = list(input())print(min_time_to_reach_sasuke(grid, M, N, T))
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==

态: Accepted

代码

```
from collections import deque
def is_vaild_move(x,y,M,N):
    return 0<=x<M and 0<=y<N
def min_time_to_reach_sasuke(grid,M,N,T):
    directions = [(0,1),(0,-1),(1,0),(-1,0)]
    start_x,start_y = 0,0
    sasuke_x,sasuke_y = 0,0
    for i in range(M):
        for j in range(N):
            if grid[i][j] == '@':
                start_x,start_y = i,j
            elif grid[i][j] == '+':
                sasuke_x,sasuke_y = i,j
    visited = set()
    queue = deque([(start_x,start_y,T,0)])
    while queue:
        x,y,chakra,time = queue.popleft()
        if (x,y) == (sasuke_x,sasuke_y):
            return time
        if (x,y,chakra) in visited:
            continue
        visited.add((x,y,chakra))
```

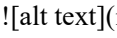
20106: 走山路

Dijkstra, http://cs101.openjudge.cn/practice/20106/

思路：

Dijkstra,bfs+堆；如非必要，勿加代码，否则可能在调试时遇到极大干扰 qwq. debug 时考虑 0,None 等 case！代码

#



代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==

05442: 兔子与星空

Prim, http://cs101.openjudge.cn/practice/05442/

思路： 最小生成树算法

代码

```
# import heapqdef prim(graph,start):
    mst = []
    used = set([start])
    edges = [(cost,start,to) for to,cost in graph[start].items()]
    heapq.heapify(edges)
    while edges:
        cost,frm,to = heapq.heappop(edges)
        if to not in used:
            used.add(to)
            mst.append((frm,to,cost))
            for to_next,cost2 in graph[to].items():
                if to_next not in used:
                    heapq.heappush(edges,(cost2,to,to_next))
    return mst

def main():
    n = int(input())
    graph = {chr(i+65):{} for i in range(n)}
    for i in range(n-1):
        data = input().split()
        star = data[0]
        m = int(data[1])
        for j in range(m):
            to_star = data[2+2*j]
            cost = int(data[3+2*j])
            graph[star][to_star] = cost
            graph[to_star][star] = cost
    mst = prim(graph,'A')
    print(sum(x[2] for x in mst))
main()#nums = [(1,1),(2,8),(9,9),(3,7)]#heapq.heapify(nums)#for i in range(4):#
print(heapq.heappop(nums))
```

代码运行截图 ==（AC 代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
import heapq
def prim(graph,start):
    mst = []
    used = set([start])
    edges = [(cost,start,to) for to,cost in graph[start].items()]
    heapq.heapify(edges)
    while edges:
        cost,frm,to = heapq.heappop(edges)
        if to not in used:
            used.add(to)
            mst.append((frm,to,cost))
            for to_next,cost2 in graph[to].items():
                if to_next not in used:
                    heapq.heappush(edges,(cost2,to,to_next))
    return mst


def main():
    n = int(input())
    graph = {chr(i+65):{} for i in range(n)}
    for i in range(n-1):
        data = input().split()
        star = data[0]
```

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。==

复习了树和最短路径等算法， 代码规范和时间复杂度优化很重要。