Assignment #A: 图论：遍历，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by 潘子轩（2300012747）信息科学技术学院

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：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：

可以利用栈将左括号存储起来，遇到右括号就将左括号弹出一个与右括号组合存入列表中，最后按照列表中存储的坐标数据依次进行翻转即可。

代码

```
def f(s, start, end):
    s_list = list(s)
    s_list[start:end] = reversed(s_list[start:end])
    return ''.join(s_list)

S = input()
i = 0
stack = []
answers = []
for char in S:
    if char != '(' and char != ')':
        i += 1
    elif char == '(':
        stack.append(i)
    elif char == ')':
        a = [stack.pop(), i]
        answers.append(a)
s = ''
for char in S:
```

```
        if char != '(' and char != ')':
            s += char
    for answer in answers:
        s = f(s, answer[0], answer[1])
    print(s)
```

代码运行截图 ==（至少包含有"Accepted"）==

02255: 重建二叉树

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

思路:

根据前序和中序递归建树即可。

代码

```
    def build_postorder(preorder, inorder):
        if not preorder:
            return ''

        root = preorder[0]
        root_index = inorder.index(root)
```

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

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

        left_postorder = build_postorder(left_preorder, left_inorder)
        right_postorder = build_postorder(right_preorder, right_inorder)

        return left_postorder + right_postorder + root

    while True:
        try:
            preorder, inorder = input().strip().split()
            print(build_postorder(preorder, inorder))

        except EOFError:
            break
```

代码运行截图 ==（至少包含有"Accepted"）==

01426: Find The Multiple

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

要求用bfs实现

思路：

答案一定是0或1组成的大整数，我们可以利用bfs从短到长依次查找。

代码

```
from collections import deque
def find_multiple_bfs(n):
    if n == 1:
        return '1'
    queue = deque()
    queue.append('1')
    while queue:
        curr = queue.popleft()
        num1 = int(curr + '0')
        num2 = int(curr + '1')
        if num1 % n == 0:
            return str(num1)
        queue.append(str(num1))
        if num2 % n == 0:
            return str(num2)
        queue.append(str(num2))

while True:
    n = int(input())
    if n == 0:
        break
    print(find_multiple_bfs(n))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

04115: 鸣人和佐助

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

思路：

根据题目要求进行bfs，为了优化查找速度 vis可以设立为三维数组。

代码

```
from collections import deque

def is_valid(x, y):
    return 0 <= x < M and 0 <= y < N

M, N, T = map(int, input().split())
Map = [list(input()) for _ in range(M)]
start_x, start_y = 0, 0
for i in range(M):
    for j in range(N):
        if Map[i][j] == '@':
```

```
            start_x, start_y = i, j
            break

visited = [[[False] * (T + 1) for _ in range(N)] for _ in range(M)]

queue = deque()
queue.append([[start_x, start_y], T, 0])

while queue:
    state = queue.popleft()
    lst = state[0]
    x, y = lst[0], lst[1]
    chakra, time = state[1], state[2]
    if Map[x][y] == '+':
        print(time)
        break
    dx = [0, 1, 0, -1]
    dy = [1, 0, -1, 0]
    for i in range(4):
        nx, ny = x + dx[i], y + dy[i]
        if is_valid(nx, ny):
            if Map[nx][ny] == '*' or Map[nx][ny] == '+':
                if not visited[nx][ny][chakra] and chakra >= 0:
                    visited[nx][ny][chakra] = True
                    queue.append([[nx, ny], chakra, time + 1])
            elif Map[nx][ny] == '#' and chakra > 0:
                if not visited[nx][ny][chakra - 1]:
                    visited[nx][ny][chakra - 1] = True
                    queue.append([[nx, ny], chakra - 1, time + 1])
else:
    print("-1")
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

20106: 走山路

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

思路:

利用Dijkstra算法对每个位置维护一个距离值，表示从起点到该位置的最小体力消耗。

代码

```
from heapq import heappop, heappush
def bfs(x1, y1):
    q = [(0, x1, y1)]
    v = set()
    while q:
        t, x, y = heappop(q)
        v.add((x, y))
        if x == x2 and y == y2:
            return t
        for dx, dy in dir:
            nx, ny = x+dx, y+dy
            if 0 <= nx < m and 0 <= ny < n and ma[nx][ny] != '#' and
(nx, ny) not in v:
                nt = t+abs(int(ma[nx][ny])-int(ma[x][y]))
                heappush(q, (nt, nx, ny))
    return 'NO'

m, n, p = map(int, input().split())
```

```
    ma = [list(input().split()) for _ in range(m)]
    dir = [(1, 0), (-1, 0), (0, 1), (0, -1)]
    for _ in range(p):
        x1, y1, x2, y2 = map(int, input().split())
        if ma[x1][y1] == '#' or ma[x2][y2] == '#':
            print('NO')
            continue
        print(bfs(x1, y1))
```

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：

利用prim算法，将与生成树中的节点相连的所有边加入到一个候选边集合中。从候选边集合中选择权值最小的边，如果这条边连接了一个已经在生成树中的节点和一个不在生成树中的节点，则将这条边加入到生成树中，并将该节点也加入到生成树中。

代码

```
    import heapq
```

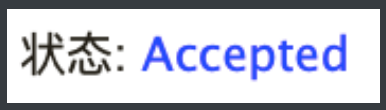
```
def prim(graph, start):
    mst = []
    used = set([start])
    edges = [
        (cost, start, to)
        for to, cost in graph[start].items()
    ]
    heapq.heapify(edges)

    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

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+j*2]
        cost = int(data[3+j*2])
        graph[star][to_star] = cost
        graph[to_star][star] = cost
mst = prim(graph, 'A')
print(sum(x[2] for x in mst))
```

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。==

由于最近外出没带电脑，只能先研究一下答案，回学校之后再自己做一遍。