Assignment #A: 图论：遍历，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by 李佳霖，心理与认知科学学院

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：macOS Sonoma

Python编程环境：VSCode

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/

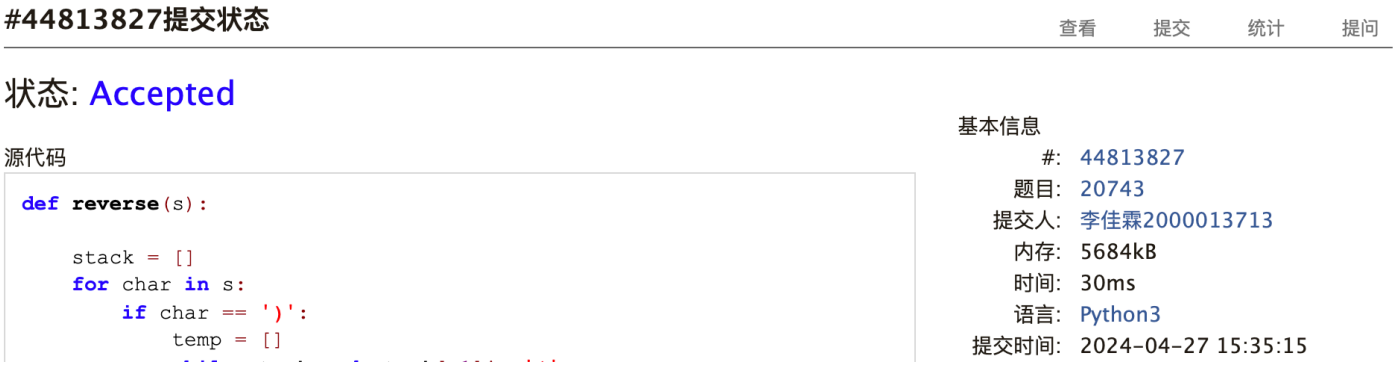
思路：类似于嵌套括号表达式，用栈处理这种括号问题即可

代码

```
#
def reverse(s):

    stack = []
```

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

    return ''.join(_ for _ in stack)

s = input().strip()
print(reverse(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：之前作业5中出现过这道题，这次又重新练习了这道题

代码

```
#
class TreeNode():
    def __init__(self, x):
        self.val = x
        self.left = None
```

```
        self.right = None

def build_tree(preorder, inorder):

    if not preorder or not inorder:
        return None

    root = TreeNode(preorder.pop())
    index = inorder.index(root.val)
    root.left = build_tree(preorder, inorder[:index])
    root.right = build_tree(preorder, inorder[index+1:])

    return root

def postTraversal(root):
    if root is None:
        return []

    res = []
    res.extend(postTraversal(root.left))
    res.extend(postTraversal(root.right))
    res.append(root.val)

    return res

while True:
    try:
        preorder, inorder = map(list, input().split(' '))
        root = build_tree(preorder[::-1], inorder)
        res = postTraversal(root)
        print(''.join(res))
    except:
        break
```

代码运行截图 （至少包含有"Accepted"）

#44814681提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class TreeNode():
    def __init__(self, x):
        self.val = x
        self.left = None
        self.right = None

def build_tree(preorder, inorder):
```

基本信息

#: 44814681
题目: 02255
提交人: 李佳霖2000013713
内存: 3592kB
时间: 27ms
语言: Python3
提交时间: 2024-04-27 16:17:32

01426: Find The Multiple

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

要求用bfs实现

思路：队列比较适合处理广度优先搜索的问题而栈适合深度优先搜索

代码

```
#
from collections import deque

def bfs(n):
    q = deque()
    inq = [False]*n #一个数的余数一定比数本身小
    q.append('1')

    while q:
        num = q.popleft()
        if int(num)!=0 and int(num)%n == 0:
            print(num)
            break
        a = num + '0'
        if not inq[int(a)%n]:
            inq[int(a)%n] = True
            q.append(a)
        b = num + '1'
        if not inq[int(b)%n]:
            inq[int(b)%n] = True
            q.append(b)

while True:
    try:
        n = int(input())
        if n == 0:
            break
        bfs(n)
    except EOFError:
        break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44816601提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

def bfs(n):
    q = deque()
    inq = [False]*n #一个数的余数一定比数本身小
    q.append('1')
```

基本信息

#: 44816601
题目: 01426
提交人: 李佳霖2000013713
内存: 3656kB
时间: 64ms
语言: Python3
提交时间: 2024-04-27 18:01:03

04115: 鸣人和佐助

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

思路：用双向队列实现了一个bfs

代码

```
#
from collections import deque

def bfs(grid, M, N, T):
    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
    start_pos = None
    sasuke_pos = None

    # Finding start and sasuke positions
    for i in range(M):
        for j in range(N):
            if grid[i][j] == '@':
                start_pos = (i, j)
            elif grid[i][j] == '+':
                sasuke_pos = (i, j)

    queue = deque([(start_pos[0], start_pos[1], T, 0)])  # (x, y, chakra, time)
    visited = [[[False] * (T + 1) for _ in range(N)] for __ in range(M)]
    visited[start_pos[0]][start_pos[1]][T] = True

    while queue:
```

```
        x, y, chakra, time = queue.popleft()

        for dx, dy in directions:
            nx, ny = x + dx, y + dy
            if 0 <= nx < M and 0 <= ny < N:
                if grid[nx][ny] == '+':
                    return time + 1
                elif grid[nx][ny] == '*' and not visited[nx][ny][chakra]:
                    visited[nx][ny][chakra] = True
                    queue.append((nx, ny, chakra, time + 1))
                elif grid[nx][ny] == '#' and chakra > 0 and not visited[nx][ny][chakra - 1]:
                    visited[nx][ny][chakra - 1] = True
                    queue.append((nx, ny, chakra - 1, time + 1))

    return -1

M, N, T = map(int, input().split())
grid = []
for i in range(M):
    grid.append(input())

result = bfs(grid, M, N, T)
print(result)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44817291提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

def bfs(grid, M, N, T):
    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
    start_pos = None
    sasuke_pos = None
```

基本信息

#: 44817291
题目: 04115
提交人: 李佳霖2000013713
内存: 7032kB
时间: 80ms
语言: Python3
提交时间: 2024-04-27 18:53:30

20106: 走山路

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

思路：

- 使用优先队列（最小堆）来保持当前待处理的节点，优先处理体力消耗较小的路径。

- 从起点开始，迭代地更新相邻节点的最小体力消耗。
- 对于每个节点，考虑所有可能的移动方向（上、下、左、右），并计算体力消耗。
- 如果新计算的体力消耗小于已知的体力消耗，则更新该节点的消耗，并将其添加到优先队列中。

代码

```
#
import heapq

def dijkstra(grid, start, end):
    m, n = len(grid), len(grid[0])
    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
    min_energy = [[float('inf')] * n for _ in range(m)]
    start_x, start_y = start
    end_x, end_y = end
    if grid[start_x][start_y] == '#' or grid[end_x][end_y] == '#':
        return "NO"

    priority_queue = []
    heapq.heappush(priority_queue, (0, start_x, start_y))
    min_energy[start_x][start_y] = 0

    while priority_queue:
        curr_energy, x, y = heapq.heappop(priority_queue)

        if (x, y) == (end_x, end_y):
            return curr_energy

        for dx, dy in directions:
            nx, ny = x + dx, y + dy
            if 0 <= nx < m and 0 <= ny < n and grid[nx][ny] != '#':
                new_energy = curr_energy + abs(int(grid[nx][ny]) - int(grid[x][y]))
                if new_energy < min_energy[nx][ny]:
                    min_energy[nx][ny] = new_energy
                    heapq.heappush(priority_queue, (new_energy, nx, ny))

    return "NO"

m, n, p = map(int, input().split())
grid = []
queries = []
for _ in range(m):
    grid.append(input().split())
for _ in range(p):
    sx, sy, ex, ey = map(int, input().split())
```

```
res = dijkstra(grid, (sx, sy), (ex, ey))
print(res)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44818524提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq

def dijkstra(grid, start, end):
    m, n = len(grid), len(grid[0])
    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
    min_energy = [[float('inf')] * n for _ in range(m)]
    start_x, start_y = start
    end_x, end_y = end
    if grid[start_x][start_y] == '#' or grid[end_x][end_y] == '#':
        return "NO"
```

基本信息

#: 44818524
题目: 20106
提交人: 李佳霖2000013713
内存: 3748kB
时间: 229ms
语言: Python3
提交时间: 2024-04-27 20:39:30

05442: 兔子与星空

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

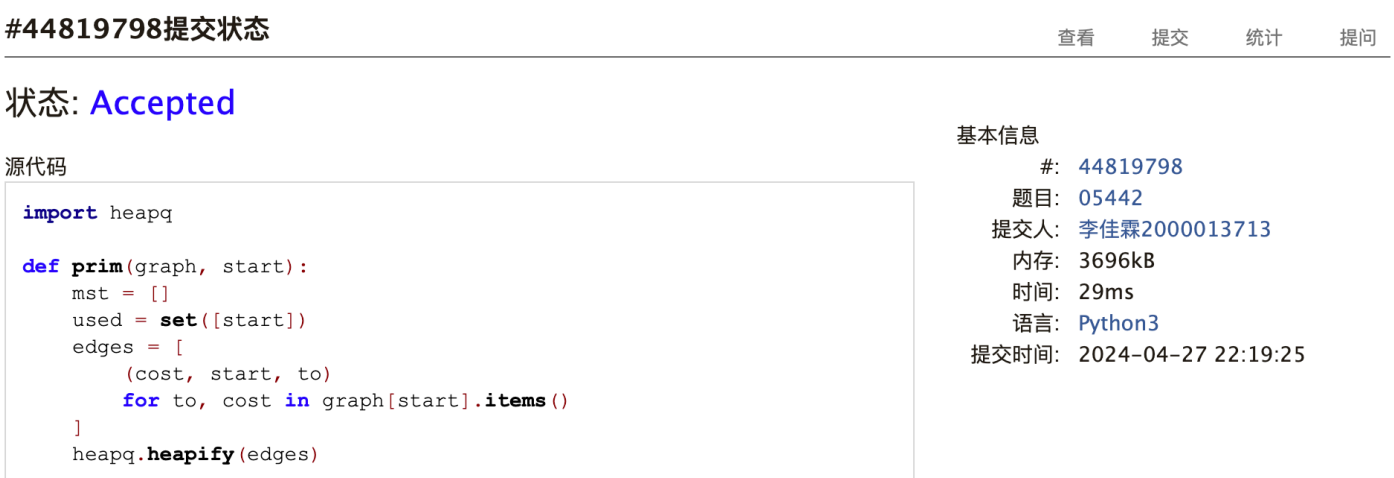
思路：看了题解，首先创建一个列表 edges ，包含从起始节点 start 出发的所有边，每个边表示为一个三元组 (cost, start, to) ，其中 cost 是边的权重， to 是边的目标节点。然后使用 heapq.heapify 函数将列表转换成一个最小堆，确保可以高效地获取最小边。在每次循环中，使用 heapq.heappop 从堆中弹出最小的边 (cost, frm, to) 。如果目标节点 to 还未被访问（即不在 used 集合中），则将其添加到 used 中，并将当前边 (frm, to, cost) 添加到 mst 列表中。接着遍历从节点 to 出发的所有边，如果这条边的另一端节点 to_next 尚未加入到 used 中，将这条边 (cost2, to, to_next) 加入到堆 edges 中。当所有节点都被访问后，即edges为空时循环结束，函数返回 mst ，即包含了最小生成树的所有边。

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)} # ASCII encoding
for i in range(n-1):
    data = input().split()
    star = data[0]
    m = int(data[1]) # how many stars is connected to current star
    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"）



#44819798提交状态

查看 提交 统计 提问

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)
```

基本信息

#: 44819798
题目: 05442
提交人: 李佳霖2000013713
内存: 3696kB
时间: 29ms
语言: Python3
提交时间: 2024-04-27 22:19:25

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。

本周作业的难度还是挺大的，虽然题目有一些之前做过的影子，但是换了一种表述方式做起来就不太能做得出来（泛化能力不太行），需要多加练习。做后面几道bfs题目的时候感觉思路都很接近，比如维护一个队列结构，一个最小堆等等，找到一点感觉，但还必须要自己实践总结争取下次能把它从头到尾写出来。