Assignment #A: 图论：遍历，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by ==经济学院 王一粟==

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：用stack，遇到右括号弹出左括号之前的所有，再反向弹入

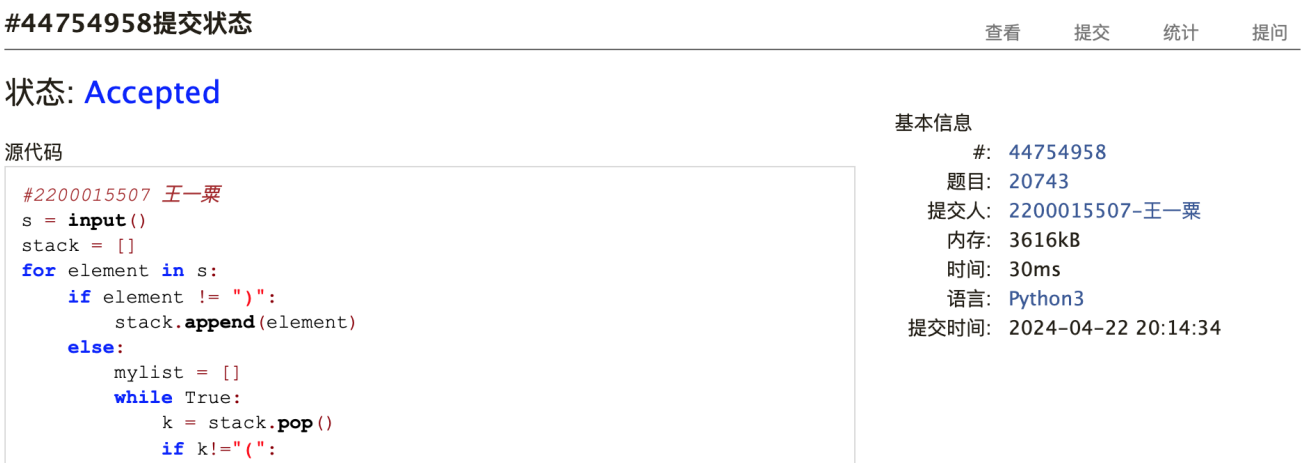
耗时：10min

代码

```
#2200015507 王一粟
s = input()
stack = []
for element in s:
```

```
    if element != ")":
        stack.append(element)
    else:
        mylist = []
        while True:
            k = stack.pop()
            if k!="(":
                mylist.append(k)
            else:
                break
        stack.extend(mylist)
print("".join(stack))
```

代码运行截图 ==（至少包含有"Accepted"）==



#44754958提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
#2200015507 王一粟
s = input()
stack = []
for element in s:
    if element != ")":
        stack.append(element)
    else:
        mylist = []
        while True:
            k = stack.pop()
            if k!="(":
```

基本信息

#: 44754958
题目: 20743
提交人: 2200015507-王一粟
内存: 3616kB
时间: 30ms
语言: Python3
提交时间: 2024-04-22 20:14:34

02255: 重建二叉树

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

思路：经典算法。根据pre的第一个元素为根节点对两部分做拆解，递归处理即可

耗时：20min

代码

```
#2200015507 王一粟
def post(pre,middle):
    if pre == "":
        return ""
    root = pre[0]
    middle_id = middle.find(root)
    left_mid = middle[:middle_id]
    right_mid = middle[middle_id+1:]
    left_pre = pre[1:len(left_mid)+1]
    right_pre = pre[len(left_mid)+1:]
    return post(left_pre,left_mid)+post(right_pre,right_mid)+root
while True:
    try:
```

```
        pre,middle = input().split()
        print(post(pre,middle))
    except:
        break
```

代码运行截图 ==（至少包含有"Accepted"）==

#44755177提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
#2200015507 王一粟
def post(pre,middle):
    if pre == "":
        return ""
    root = pre[0]
    middle_id = middle.find(root)
    left_mid = middle[:middle_id]
    right_mid = middle[middle_id+1:]
    left_pre = pre[1:len(left_mid)+1]
    right_pre = pre[len(left_mid)+1:]
    return post(left_pre,left_mid)+post(right_pre,right_mid)+root
while True:
    try:
        pre,middle = input().split()
```

基本信息

#: 44755177
题目: 02255
提交人: 2200015507-王一粟
内存: 3576kB
时间: 29ms
语言: Python3
提交时间: 2024-04-22 20:26:05

01426: Find The Multiple

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

要求用bfs实现

思路：比较偏bfs的想法。从位数小起步，如果找不到的话，在后面加0或者1。一个简化算法的方式是，如果有mod相同的可以剔掉

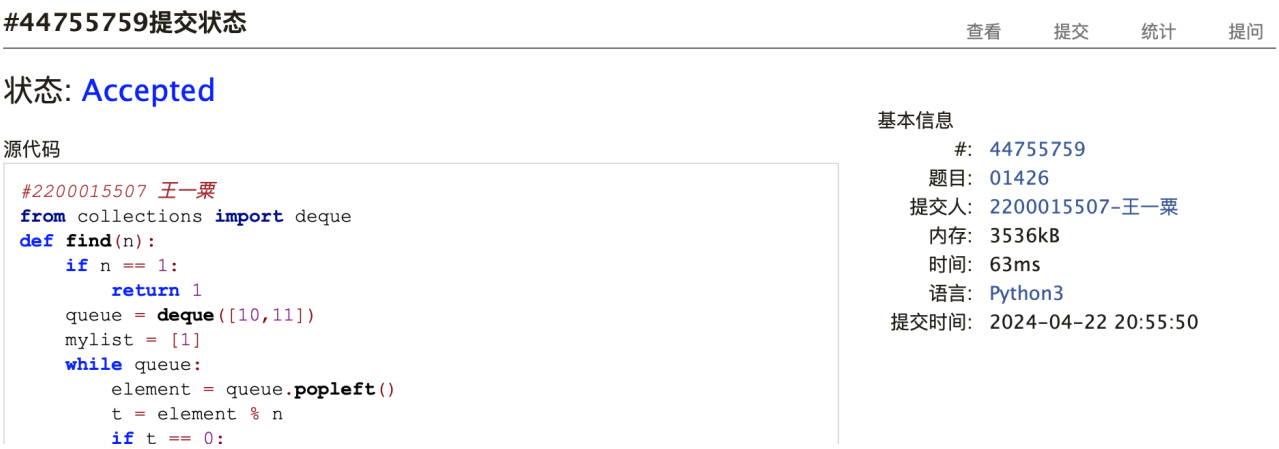
耗时：25min

代码

```
#2200015507 王一粟
from collections import deque
def find(n):
    if n == 1:
        return 1
    queue = deque([10,11])
    mylist = [1]
    while queue:
        element = queue.popleft()
        t = element % n
        if t == 0:
            return str(element)
        else:
            if t not in mylist:
                mylist.append(t)
```

```
                queue.append(element*10+1)
                queue.append(element*10)
while True:
    n = int(input())
    if n == 0:
        break
    else:
        print(find(n))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



#44755759提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
#2200015507 王一粟
from collections import deque
def find(n):
    if n == 1:
        return 1
    queue = deque([10,11])
    mylist = [1]
    while queue:
        element = queue.popleft()
        t = element % n
        if t == 0:
```

基本信息

#: 44755759
题目: 01426
提交人: 2200015507-王一粟
内存: 3536kB
时间: 63ms
语言: Python3
提交时间: 2024-04-22 20:55:50

04115: 鸣人和佐助

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

思路：本题难度个人认为不小。主要问题在于如何去构建bfs，其实本质上来看，该bfs去记录是否访问的变量有三个维度：两个空间维度和一个所谓的查克拉数量

耗时：1h

代码

```
#2200015507 王一粟
from collections import deque
m,n,p = [int(i) for i in input().split()]
mylist = []
cnt1 = 0
cnt2 = 0
for _ in range(m):
    t = list(input())
    if cnt1 == 0 and "@" in t:
        start = (_,t.index("@"))
        cnt1 = 1
    if cnt2 == 0 and "+" in t:
        end = (_,t.index("+"))
        cnt2 = 1
    mylist.append(t)
visited = [[[0]*(p+1) for i in range(n)] for j in range(m)]
```

```
    queue = deque([[start,p,0]])
    while queue:
        position,num,cnt = queue.popleft()
        x,y = position
        for dx,dy in [(1,0),(-1,0),(0,1),(0,-1)]:
            if 0<=x+dx<m and 0<=y+dy<n:
                if mylist[x+dx][y+dy] == "+":
                    print(cnt+1)
                    exit(0)
                elif mylist[x+dx][y+dy] == "#":
                    if num != 0:
                        if visited[x+dx][y+dy][num-1] == 0:
                            queue.append([(x+dx,y+dy),num-1,cnt+1])
                            visited[x+dx][y+dy][num-1] = 1
                else:
                    if visited[x+dx][y+dy][num] == 0:
                        queue.append([(x+dx,y+dy),num,cnt+1])
                        visited[x + dx][y + dy][num] = 1
    print(-1)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44756673提交状态

查看　提交　统计　提问

状态: Accepted

源代码

```
from collections import deque
m,n,p = [int(i) for i in input().split()]
mylist = []
cnt1 = 0
cnt2 = 0
for _ in range(m):
    t = list(input())
    if cnt1 == 0 and "@" in t:
        start = (_,t.index("@"))
        cnt1 = 1
    if cnt2 == 0 and "+" in t:
```

基本信息

#: 44756673
题目: 04115
提交人: 2200015507-王一粟
内存: 7200kB
时间: 111ms
语言: Python3
提交时间: 2024-04-22 21:59:01

20106: 走山路

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

思路：好难好烦...因为之前没有接触过这种算法，对图也不熟悉，最初做甚至把图和顶点的类都写全了，后来发现根本不需要，把边和点的数据用列表/字典保存好就可以。

首先，对这个图做解析，生成mylist矩阵和有关于到达相邻节点权重的列表套字典（方便索引）；

然后对任何给定的坐标顶点都去采用Dijkstra算法。具体的思路就是，每次找到distance最小的，确认distance，并去看和这个顶点相邻的顶点是否distance可以更小（用egdes），如果有更小，更新distance；

然而实现起来时间复杂度很高。一个关键问题在于，每次更新的数据都要去更新二叉堆，然而在heapq包中，直接对二叉堆其中一项更新数据就需要对二叉堆做排序，时间复杂度很高。

对此，我的处理是用一个distance_dict实时更新每个位置的distance，然后如果遇到distance更新，直接在二叉堆中做插入，对原先的distance的元素不做任何处理。每次删除最小值时，都去判断该点的distance是否小于距离字典中的值，如果不是，则为未删除的冗余元素，不做处理。

其实也可以考虑直接手搓二叉堆，用内置的percUp，但实用性太低了，而且手搓很容易出错。

另一个节约的办法是，没有必要全部进行遍历。如果我们在二叉堆中删除的元素就是要找到的终点元素，则可以直接跳出循环，直接输出该距离，后面的遍历都是没有必要的。

个人感觉Dijkstra算法要比Prim难，我是先学了第六题怎么做再来想第五题的

耗时：一天晚上2h无果，第二天思考了45min搞定

代码

```
#2200015507 王一粟
import sys
import heapq
m,n,p = [int(i) for i in input().split()]
mylist = []
for i in range(m):
    t = [int(i) if i != "#" else i for i in input().split()]
    mylist.append(t)
edges = [[{} for j in range(n)] for i in range(m)]
for i in range(m):
    for j in range(n):
        if mylist[i][j] != "#":
            for dx,dy in [(-1,0),(1,0),(0,-1),(0,1)]:
                if 0<=i+dx<m and 0<=j+dy<n and mylist[i+dx][j+dy] != "#":
                    edges[i][j][(i+dx,j+dy)] = abs(mylist[i+dx][j+dy]-mylist[i][j])
for _ in range(p):

    start_i,start_j,end_i,end_j = [int(i) for i in input().split()]
    if mylist[start_i][start_j] == "#" or mylist[end_i][end_j] == "#":
        print("NO")
        continue
    distance_dict = {(i, j): sys.maxsize if (i,j) not in edges[start_i][start_j] else edges[st
    distance_dict[(start_i, start_j)] = 0
    priority = [[distance,x[0],x[1]] for x,distance in edges[start_i][start_j].items()]
    heapq.heapify(priority)
    while priority:
        distance,position_i,position_j = heapq.heappop(priority)
        if distance_dict[(position_i,position_j)] < distance:
            continue
        if (position_i,position_j) == (end_i,end_j):
            distance_dict[(position_i,position_j)] = distance
            break
        distance_dict[(position_i,position_j)] = distance
        for neighbor_position,distance_from_current in edges[position_i][position_j].items():
            if distance_dict[neighbor_position] > distance_from_current + distance:
                distance_dict[neighbor_position] = distance_from_current + distance
                heapq.heappush(priority,[distance_from_current + distance,neighbor_position[0]
    result = distance_dict[(end_i,end_j)]
```

```
    if result < sys.maxsize:
        print(result)
    else:
        print("NO")
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44762540提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
#2200015507 王一粟
import sys
import heapq
m,n,p = [int(i) for i in input().split()]
mylist = []
for i in range(m):
    t = [int(i) if i != "#" else i for i in input().split()]
    mylist.append(t)
edges = [[{} for j in range(n)] for i in range(m)]
for i in range(m):
    for j in range(n):
        if mylist[i][j] != "#":
            for dx,dy in [(-1,0),(1,0),(0,-1),(0,1)]:
                if 0<=i+dx<m and 0<=j+dy<n and mylist[i+dx][j+dy] != "#
                    edges[i][j][(i+dx,j+dy)] = abs(mylist[i+dx][j+dy]-my
for _ in range(p):
```

基本信息

#:	44762540
题目:	20106
提交人:	2200015507-王一粟
内存:	4776kB
时间:	188ms
语言:	Python3
提交时间:	2024-04-23 14:21:15

05442: 兔子与星空

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

思路：参考了题解的做法。直接采用一个邻接字典去存储图的所有信息（键为顶点的名字，值为一个字典：每个键值对表示一条边及其权重）。解析后，从一个顶点开始，先将所有的边的信息输入并构建二叉堆，然后每次弹出最小的权重：先判断到达顶点是否已经被访问，没有被访问则添加，并将该顶点的所有可到达、但未被访问的信息依次插入二叉堆中。对每次访问过的结点，加总权重即可。

耗时：40min

代码

```
#2200015507 王一粟
import heapq
n = int(input())
graph = {chr(65+i):{} for i in range(n)}
for _ in range(n-1):
    s = input().split()
    key = s[0]
    num = int(s[1])
    for i in range(num):
        graph[key][s[i*2+2]] = int(s[i*2+3])
        graph[s[i*2+2]][key] = int(s[i*2+3])
visited = ["A"]
mylist = [(value,"A",key_to) for key_to,value in graph["A"].items()]
```

```
weight = 0
heapq.heapify(mylist)
while mylist:
    distance,fro,to = heapq.heappop(mylist)
    if to not in visited:
        visited.append(to)
        weight += distance
        for next_to,next_distance in graph[to].items():
            if next_to not in visited:
                heapq.heappush(mylist,(next_distance,to,next_to))
print(weight)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）== Q6.py

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。==

昨天这份作业就差最后两题了，今天下午沉思了一下终于搞定了。大概期中季结束后的第一周只够先把图学完，毕竟以前完全没接触过，量还是挺大的；

没做每日选做导致现在对图的理解还是很生疏，尤其是可能很多题的方法上还是很繁琐，五一打算有时间的话把每日选做补掉。

不知道是不是自己的原因，感觉图的代码长度都很长，而且比较烦。每次写完这堆代码开始跑程序就疯狂报错，就得一处一处改...