Assignment #A: 图论：遍历，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by ==张宇帆-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）==

Python编程环境：Spyder IDE 5.2.2

1. 题目

20743: 整人的提词本

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

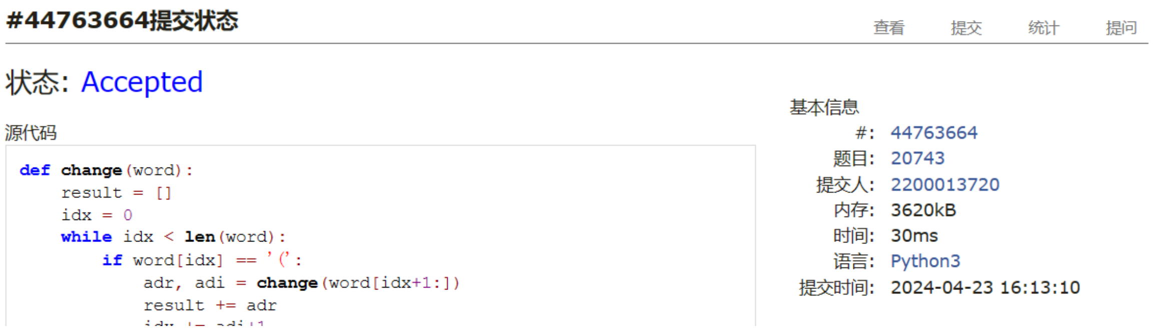
思路：简单的递归，遇见左括号重新建个result，遇见右括号把result反转然后返回即可。（不过我第一次做是在每日选做，那个时候的AC代码更短，看来有所退步了）

代码

```python
# 
def change(word):
    result = []
    idx = 0
    while idx < len(word):
        if word[idx] == '(':
            adr, adi = change(word[idx+1:])
            result += adr
            idx += adi+1
        elif word[idx] == ')':
            result.reverse()
            return result, idx+1
        else:
            result.append(word[idx])
            idx += 1
    return result, idx
```

```python
result, _ = change(input())
print(''.join(result))
```

代码运行截图 == （至少包含有"Accepted"） ==



#44763664提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
def change(word):
    result = []
    idx = 0
    while idx < len(word):
        if word[idx] == '(':
            adr, adi = change(word[idx+1:])
            result += adr
```

基本信息

#:	44763664
题目:	20743
提交人:	2200013720
内存:	3620kB
时间:	30ms
语言:	Python3
提交时间:	2024-04-23 16:13:10

02255: 重建二叉树

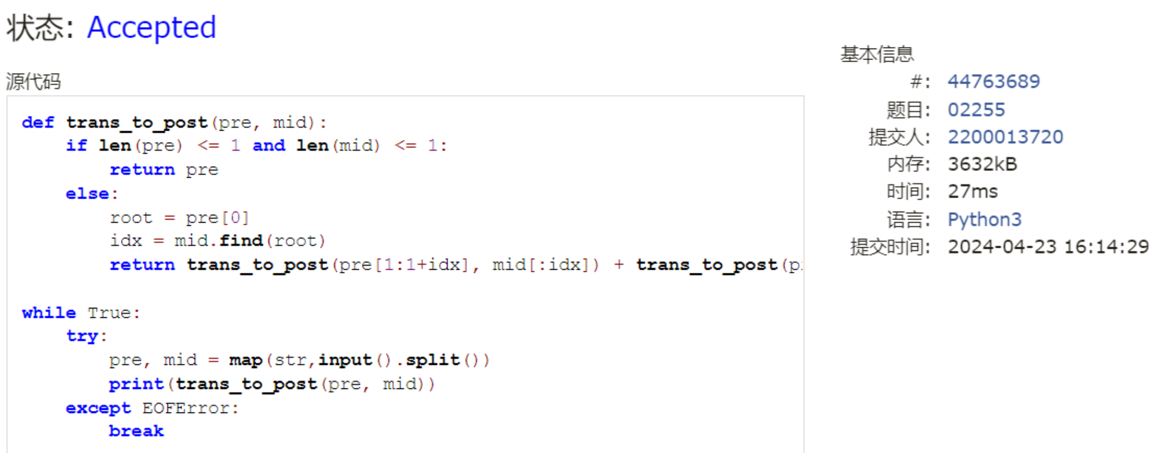
http://cs101.openjudge.cn/practice/02255/

思路：依据前序遍历和中序遍历的特点即可输出后序遍历

代码

```python
#
def trans_to_post(pre, mid):
    if len(pre) <= 1 and len(mid) <= 1:
        return pre
    else:
        root = pre[0]
        idx = mid.find(root)
        return trans_to_post(pre[1:1+idx], mid[:idx]) +
trans_to_post(pre[1+idx:], mid[idx+1:]) + root

while True:
    try:
        pre, mid = map(str,input().split())
        print(trans_to_post(pre, mid))
    except EOFError:
        break
```

代码运行截图 == （至少包含有"Accepted"） ==


状态: Accepted

源代码

```
def trans_to_post(pre, mid):
    if len(pre) <= 1 and len(mid) <= 1:
        return pre
    else:
        root = pre[0]
        idx = mid.find(root)
        return trans_to_post(pre[1:1+idx], mid[:idx]) + trans_to_post(p

while True:
    try:
        pre, mid = map(str,input().split())
        print(trans_to_post(pre, mid))
    except EOFError:
        break
```

基本信息

#:	44763689
题目:	02255
提交人:	2200013720
内存:	3632kB
时间:	27ms
语言:	Python3
提交时间:	2024-04-23 16:14:29

01426: Find The Multiple

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

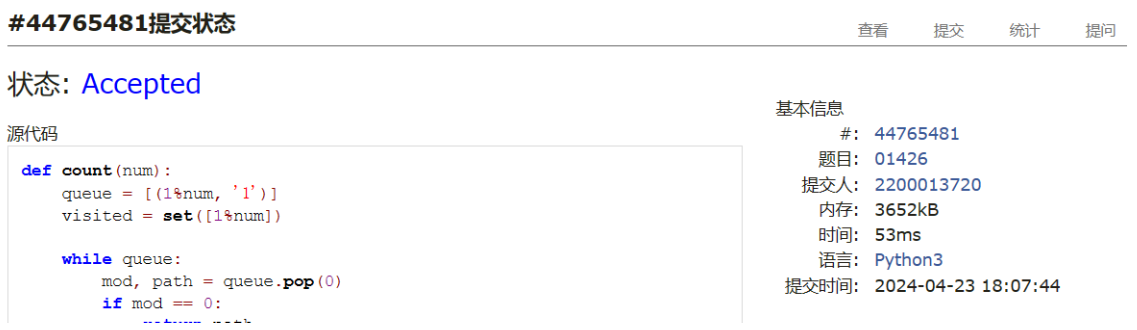
要求用bfs实现

思路：常规的队列实现bfs即可，本来一开始尝试用的递归，结果因为递归次数太多导致RE了，遂换成bfs

代码

```python
#
def count(num):
    queue = [(1%num, '1')]
    visited = set([1%num])

    while queue:
        mod, path = queue.pop(0)
        if mod == 0:
            return path
        for i in ['0','1']:
            new_path = path + i
            new_mod = (mod*10 + int(i))%num
            if new_mod not in visited:
                queue.append((new_mod, new_path))
                visited.add(new_mod)

n = int(input())
while not n == 0:
    print(count(n))
    n = int(input())
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==


#44765481提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def count(num):
    queue = [(1%num, '1')]
    visited = set([1%num])

    while queue:
        mod, path = queue.pop(0)
        if mod == 0:
```

基本信息

#:	44765481
题目:	01426
提交人:	2200013720
内存:	3652kB
时间:	53ms
语言:	Python3
提交时间:	2024-04-23 18:07:44

04115: 鸣人和佐助

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

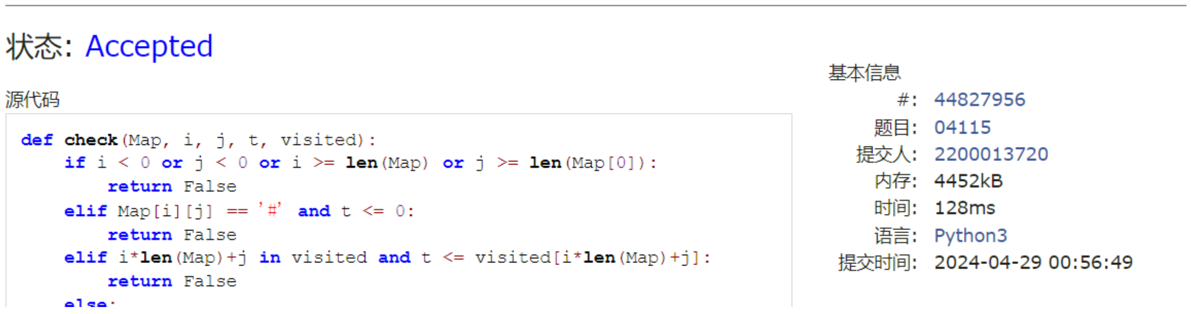
思路：一开始用bfs建立了传统的visited然后WA了，翻了群记录发现可以走重复后删掉visited结果TLE了，遂一番思考后添加visited，剪枝方式就是到达当前位置的查克拉小于先前抵达过的查克拉则剪枝，否则更新并进行移动。

代码

```python
# 
def check(Map, i, j, t, visited):
    if i < 0 or j < 0 or i >= len(Map) or j >= len(Map[0]):
        return False
    elif Map[i][j] == '#' and t <= 0:
        return False
    elif i*len(Map)+j in visited and t <= visited[i*len(Map)+j]:
        return False
    else:
        return True

def bfs(Map, start, target, T):
    queue = [(0, start[0], start[1], T)]
    move = [[0,1],[0,-1],[1,0],[-1,0]]
    visited = {start[0]*len(Map)+start[1]: 0}
    result = -1
    while not queue == []:
        length, i, j, t = queue.pop(0)
        if Map[i][j] == target:
            result = length
            break
        for m in move:
            new_i, new_j = i + m[0], j + m[1]
            if check(Map, new_i, new_j, t, visited):
                if Map[new_i][new_j] == '#':
                    new_t = t - 1
                else:
                    new_t = t
                queue.append((length + 1, new_i, new_j, new_t))
                visited[new_i*len(Map)+new_j] = new_t ## 更新或记录
    return result
```

```python
M, N, T = map(int,input().split())
Map = []
for i in range(M):
    path = input()
    if '@' in path:
        start = [i, path.index('@')]
    Map.append(list(path))
print(bfs(Map, start, '+', T))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



状态: Accepted

源代码

```python
def check(Map, i, j, t, visited):
    if i < 0 or j < 0 or i >= len(Map) or j >= len(Map[0]):
        return False
    elif Map[i][j] == '#' and t <= 0:
        return False
    elif i*len(Map)+j in visited and t <= visited[i*len(Map)+j]:
        return False
    else:
```

基本信息

#:	44827956
题目:	04115
提交人:	2200013720
内存:	4452kB
时间:	128ms
语言:	Python3
提交时间:	2024-04-29 00:56:49

20106: 走山路

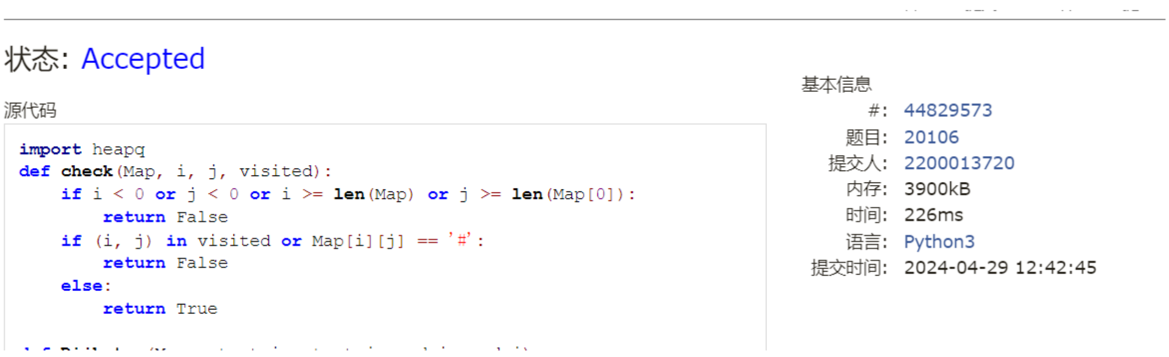
Dijkstra, http://cs101.openjudge.cn/practice/20106/

思路：一开始想复杂了，把visited跟鸣人和佐助那道一样设了个字典结果一直WA，最后看了题解发现他的visited是正常的集合，所以改了，中途RE和MLE一次，最后终于AC了，这山路走得饭都忘了吃了)

代码

```python
#
import heapq
def check(Map, i, j, visited):
    if i < 0 or j < 0 or i >= len(Map) or j >= len(Map[0]):
        return False
    if (i, j) in visited or Map[i][j] == '#':
        return False
    else:
        return True

def Dijkstra(Map, start_i, start_j, end_i, end_j):
    move = [(0,1),(0,-1),(1,0),(-1,0)]
    queue = [(0, start_i, start_j)]
    heapq.heapify(queue)
    visited = set()
    while not queue == []:
        path, cur_i, cur_j = heapq.heappop(queue)
        if cur_i == end_i and cur_j == end_j:
            return path
        if (cur_i, cur_j) in visited:
            continue
        visited.add((cur_i, cur_j))
        for m in move:
            i, j = cur_i + m[0], cur_j + m[1]
            if check(Map, i, j, visited):
                new_path = path + abs(int(Map[i][j]) - int(Map[cur_i][cur_j]))
                heapq.heappush(queue, (new_path, i, j))
    return 'NO'

m, n, p = map(int,input().split())
Map= []
f_result = []
for _ in range(m):
    Map.append(input().split())
for _ in range(p):
    start_i, start_j, end_i, end_j = map(int,input().split())
    if Map[start_i][start_j] == '#' or Map[end_i][end_j] == '#':
        result = 'NO'
    else:
        result = Dijkstra(Map, start_i, start_j, end_i, end_j)
    print(result)
```

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 check(Map, i, j, visited):
    if i < 0 or j < 0 or i >= len(Map) or j >= len(Map[0]):
        return False
    if (i, j) in visited or Map[i][j] == '#':
        return False
    else:
        return True
```

基本信息

#:	44829573
题目:	20106
提交人:	2200013720
内存:	3900kB
时间:	226ms
语言:	Python3
提交时间:	2024-04-29 12:42:45

05442: 兔子与星空

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

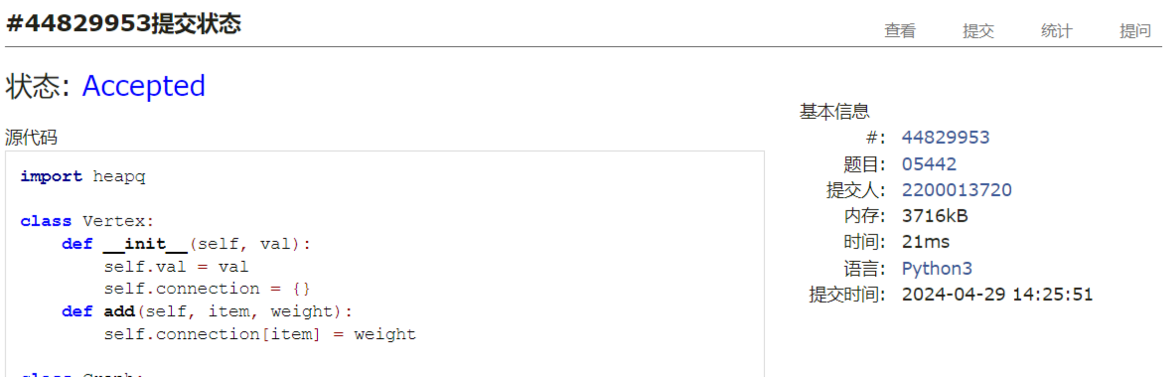
思路：了解了Prim算法之后利用图进行实现就可以了，感觉Prim和Dijkstra好像，二者的思路都是取局部最小，有点类似于贪心了，所以堆的重要性就凸显出来了（这个一遍就AC了，吃饱饭感觉就是不一样）

代码

```
#
import heapq

class Vertex:
    def __init__(self, val):
        self.val = val
        self.connection = {}
    def add(self, item, weight):
        self.connection[item] = weight
```

```python
class Graph:
    def __init__(self):
        self.ids = {}

    def addVertex(self, val):
        self.ids[val] = Vertex(val)

    def addEdge(self, fr, to, weight):
        if fr not in self.ids:
            self.addVertex(fr)
        if to not in self.ids:
            self.addVertex(to)
        self.ids[fr].add(to, weight)

def Prim(graph, start):
    queue = [(0, start)]
    heapq.heapify(queue)
    visited = set()
    result = 0
    while queue:
        weight, cur = heapq.heappop(queue)
        if cur in visited:
            continue
        result += weight
        visited.add(cur)
        for c in graph.ids[cur].connection:
            if not c in visited:
                w = graph.ids[cur].connection[c]
                heapq.heappush(queue, (w, c))
    return result

n = int(input())
graph = Graph()
record = float('inf')
for _ in range(n-1):
    keylist = input().split()
    val = keylist[0]
    num = int(keylist[1])
    for idx in range(num):
        weight = int(keylist.pop())
        if weight < record:
            start = val
        target = keylist.pop()
        graph.addEdge(val, target, weight)
        graph.addEdge(target, val, weight)
print(Prim(graph, start))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



状态: Accepted

源代码

```python
import heapq

class Vertex:
    def __init__(self, val):
        self.val = val
        self.connection = {}
    def add(self, item, weight):
        self.connection[item] = weight

class Graph:
```

基本信息

#:	44829953
题目:	05442
提交人:	2200013720
内存:	3716kB
时间:	21ms
语言:	Python3
提交时间:	2024-04-29 14:25:51

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。==

作业总是因为其他一些事情就中断或忘了做了，当然最主要的是因为我在学计概的时候对dfs和bfs有阴影，导致我现在看见这两就脑袋疼，不过这次作业做起来感觉了解算法后bfs倒也不是很难，最主要的是注意visited的使用（+必要的剪枝）。Prim和Dijkstra本质上也只是在选取节点的时候不是按一定的移动规则，而是选取局部最优。

最后，祝老师和同学们五一快乐！（尽管我五一假期一天一个ddl）