Assignment #F: All-Killed 满分

Updated 1844 GMT+8 May 20, 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）==

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

22485: 升空的焰火，从侧面看

http://cs101.openjudge.cn/practice/22485/

思路：记录树的高度变化的时刻，存储对应的前一节点的信息即可。也就是题目中所说的输出广度优先搜索中每一层最后一个节点。

代码

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

def bfs(trees):
    q = -1
    queue = [(trees[0], 0)]
    result = []
```

```
    last_t = None
    while queue:
        cur_t, cur_q = queue.pop(0)
        if not q == cur_q and last_t:
            q = cur_q
            result.append(last_t.root)
        last_t = cur_t
        if cur_t.left:
            queue.append((trees[cur_t.left-1], cur_q + 1))
        if cur_t.right:
            queue.append((trees[cur_t.right-1], cur_q + 1))
    result.append(cur_t.root)
    return result

N = int(input())
trees = [0]*N
for idx in range(N):
    trees[idx] = TreeNode(str(idx+1))
    left, right = map(int,input().split())
    trees[idx].left = left if not left == -1 else None
    trees[idx].right = right if not right == -1 else None
print(' '.join(bfs(trees)))
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

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

def bfs(trees):
    q = -1
```

基本信息

#:	45027634
题目:	22485
提交人:	2200013720
内存:	3780kB
时间:	22ms
语言:	Python3
提交时间:	2024-05-20 19:44:01

28203:【模板】单调栈

http://cs101.openjudge.cn/practice/28203/

思路：自己写了一个，但感觉和之前看到的不太一样，然后翻了翻发现其实就是stack储存的东西不一样，我重点储存了数值，而模板储存了位置，对于单调栈确实位置更重要一点（毕竟是目标）

代码

```
#
def humdrum(num):
    stack = []
    result = [0]*len(num)
    for idx in range(len(num)):
        while stack and stack[-1][1] < num[idx]:
            i, _ = stack.pop()
            result[i] = idx + 1
```

```
        stack.append((idx, num[idx]))
    return result

def humd(num): # 模板
    stack = []
    result = [0]*len(num)
    for idx in range(len(num)):
        while stack and num[stack[-1]] < num[idx]:
            result[stack.pop()] = idx + 1
        stack.append(idx)
    return result

n = int(input())
num = list(map(int, input().split()))
print(*humd(num))
```

代码运行截图 ==（至少包含有"Accepted"）==

#45070899提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def humdrum(num):
    stack = []
    result = [0]*len(num)
    for idx in range(len(num)):
        while stack and stack[-1][1] < num[idx]:
            i, _ = stack.pop()
            result[i] = idx + 1
        stack.append((idx, num[idx]))
    return result

n = int(input())
num = list(map(int, input().split()))
print(*humdrum(num))
```

基本信息

#: 45070899
题目: 28203
提交人: 2200013720
内存: 383576kB
时间: 2816ms
语言: Python3
提交时间: 2024-05-24 20:27:22

09202: 舰队、海域出击!

http://cs101.openjudge.cn/practice/09202/

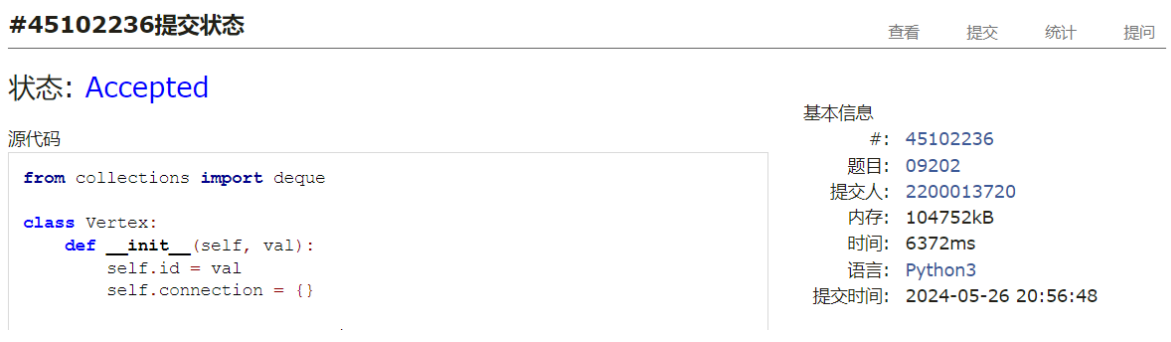
思路：本来尝试用dfs的，但是好像还得打标记，所以最后还是用了拓扑。感觉每次写Vertex和Graph有点麻烦，看了下题解发现其实直接建字典也可以，以后多试试

代码

```
#
from collections import deque

class Vertex:
    def __init__(self, val):
        self.id = val
        self.connection = {}

    def addneigh(self, val, weight):
        self.connection[val] = weight
```

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

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

    def addEdge(self, fr, to):
        if fr not in self.ids:
            self.addVertex(fr)
        if to not in self.ids:
            self.addVertex(to)
        self.ids[fr].addneigh(to, 1)

def check(graph):
    in_degree = {v: 0 for v in graph.ids}
    for up in graph.ids:
        for v in graph.ids[up].connection:
            in_degree[v] += 1
    q = deque([v for v in in_degree if in_degree[v] == 0])
    visited = []
    while q:
        cur = q.popleft()
        visited.append(cur)
        for v in graph.ids[cur].connection:
            in_degree[v] -= 1
            if in_degree[v] == 0:
                q.append(v)
    if not len(visited) == len(graph.ids):
        return 'Yes'
    return 'No'

T = int(input())
for _ in range(T):
    N, M = map(int,input().split())
    g = Graph()
    g.ids = {i:Vertex(i) for i in range(1,N+1)}
    for _ in range(M):
        i, j = map(int,input().split())
        g.addEdge(i, j)
    print(check(g))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



#45102236提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

class Vertex:
    def __init__(self, val):
        self.id = val
        self.connection = {}
```

基本信息

#: 45102236
题目: 09202
提交人: 2200013720
内存: 104752kB
时间: 6372ms
语言: Python3
提交时间: 2024-05-26 20:56:48

04135: 月度开销

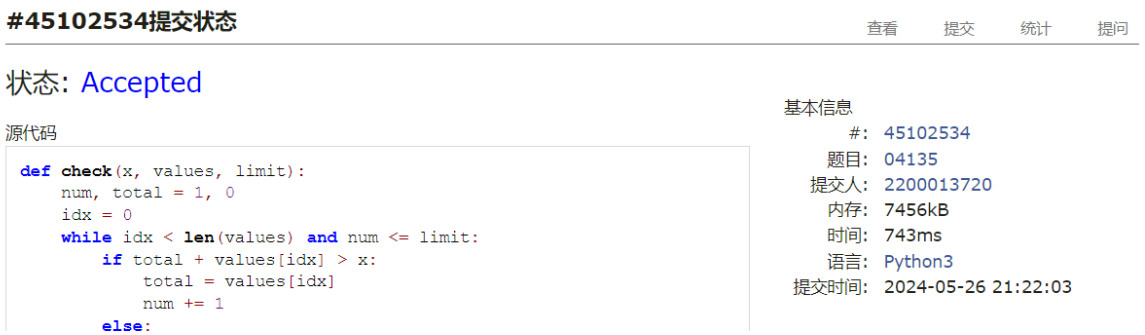
http://cs101.openjudge.cn/practice/04135/

思路：之前计概接触过，没想到用二分，以为是很难很难的一道题。然后现如今看了题解发现是用二分，写一下就过了。二分提的速度太快了

代码

```
#
def check(x, values, limit):
    num, total = 1, 0
    idx = 0
    while idx < len(values) and num <= limit:
        if total + values[idx] > x:
            total = values[idx]
            num += 1
        else:
            total += values[idx]
        idx += 1
    return num <= limit

def binary(values, limit):
    result = 1
    left, right = max(values), sum(values)
    while left < right:
        mid = (left + right)//2
        if check(mid, values, limit): # 说明mid值偏大，可以涵盖
            result = mid
            right = mid
        else:
            left = mid + 1 # 说明mid值偏小，可以涵盖
    return result

N, M = map(int, input().split())
values = []
for _ in range(N):
    values.append(int(input()))
print(binary(values, M))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



07735: 道路

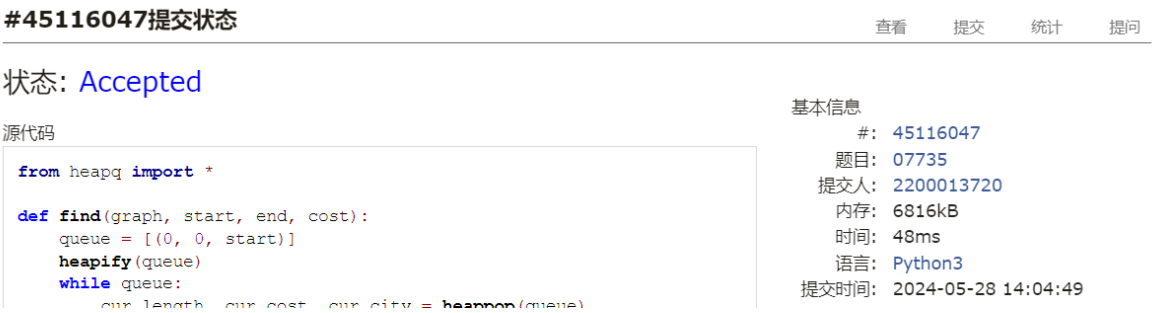
http://cs101.openjudge.cn/practice/07735/

思路：Dijkstra算法，一开始TLE了很奇怪（因为已经删了visited，也规定找到N就返回），结果发现是“不同的道路可能有相同的起点和终点”导致输入的时候覆盖了道路信息，改了一下就好了。

代码

```
#
from heapq import *

def find(graph, start, end, cost):
    queue = [(0, 0, start)]
    heapify(queue)
    while queue:
        cur_length, cur_cost, cur_city = heappop(queue)
        if cur_city == end:
            return cur_length
        for nei in graph[cur_city]:
            for value in graph[cur_city][nei]:
                add_length, add_cost = value
                if cur_cost + add_cost > cost:
                    continue
                else:
                    heappush(queue, (cur_length + add_length, cur_cost +
add_cost, nei))
    return -1

max_K = int(input())
N = int(input())
Roads = int(input())
city = {i:{} for i in range(1,N+1)}
for _ in range(Roads):
    start, end, length, value = map(int,input().split())
    if end in city[start]:
        city[start][end].append((length, value))
    else:
        city[start][end] = [(length, value)]
print(find(city, 1, N, max_K))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==



01182: 食物链

http://cs101.openjudge.cn/practice/01182/

思路：只知道用并查集，但确实不知道怎么用。于是看了题解，看完后只觉得这种思路太伟大了（）。这想必也就是编程的乐趣吧，通过几条简单的规则实现一个复杂的过程，而不失严谨与迅捷。

代码

```
#
def find(x):    # 并查集查询
    if p[x] == x:
        return x
    else:
        p[x] = find(p[x])   # 父节点设为根节点。目的是路径压缩。
        return p[x]

n,k = map(int, input().split())

# [0,n)表示同类，[n,2*n)表示x吃的动物，[2*n,3*n)表示吃x的动物
p = [0]*(3*n + 1)
for i in range(3*n+1):  #并查集初始化
    p[i] = i

ans = 0
for _ in range(k):
    a,x,y = map(int, input().split())
    if x>n or y>n: # 编号大于N
        ans += 1; continue

    if a==1: # x和y是同类
        if find(x+n)==find(y) or find(y+n)==find(x): # x、y存在捕食关系
            ans += 1; continue

        # 合并
        p[find(x)] = find(y)
        p[find(x+n)] = find(y+n)
        p[find(x+2*n)] = find(y+2*n)
    else: # a==2，即x吃y
        if find(x)==find(y) or find(y+n)==find(x): # x、y是同类或者y吃x
            ans += 1; continue
        p[find(x+n)] = find(y)
        p[find(y+2*n)] = find(x)
```

```
            p[find(x+2*n)] = find(y+n)

print(ans)
```

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。==

感觉这次作业考验模板理解的同时，也考验了模板的灵活运用与变体，比如食物链，哪怕了解并查集也不一定想得出很好的实现思路。总体难度偏大。

最近期末季又开始忙碌起来了，但不至于让人绝望，不过面对期末考还是很焦虑，很担心没能达到自己的预期（底线AC4，争取AK）。周末要花时间去整理一些模板了，希望考试多考点模板题（），目前除了并查集以及树的一些特殊实现之外，其他我还是挺有信心的。

（作法企图吸吸大佬的神力，再度声明我们是平行班！）