Assignment #F: All-Killed 满分

Updated 1844 GMT+8 May 20, 2024

2024 spring, Complied by 赵语涵 生命科学学院

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

操作系统：windows 11

Python编程环境：Spyder IDE 5.2.2

1. 题目

22485: 升空的焰火，从侧面看

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

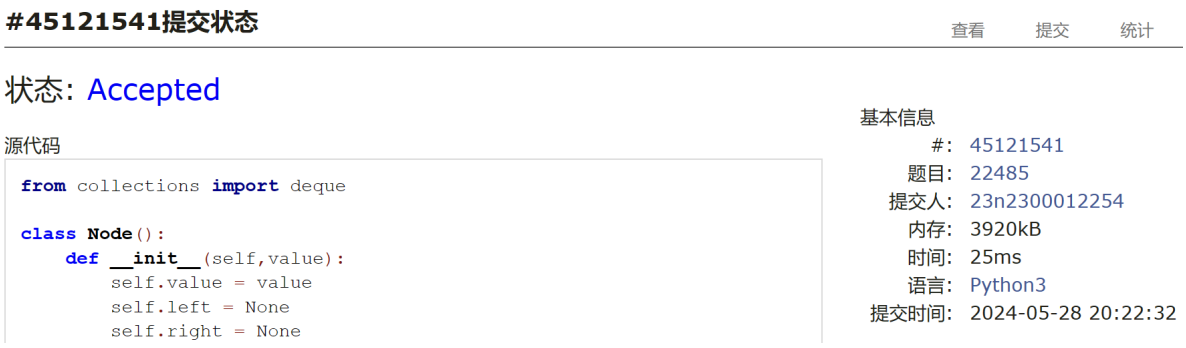
思路：原思路是通过建立Node类，并记录层信息以输出某层最后一个值。参考chatGPT以后发现可以利用deque弹出上一层，剩下最开始元素个数即为当层个数，每建立一层把最后一个值放入输出中即可

代码

```
from collections import deque

class Node():
    def __init__(self,value):
        self.value = value
        self.left = None
        self.right = None

n = int(input())
tree = [Node(x+1) for x in range(n)]
for i in range(n):
    a,b = map(int,input().split())
    if a != -1:
        tree[i].left = tree[a-1]
```

```
    if b != -1:
        tree[i].right = tree[b-1]

bfs = deque([tree[0]])
answer = ['1',]
while bfs:
    level_size = len(bfs)
    for _ in range(level_size):
        node = bfs.popleft()
        if node.left:
            bfs.append(node.left)
        if node.right:
            bfs.append(node.right)
    if bfs:
        answer.append(str(bfs[-1].value))

print(' '.join(answer))
```

代码运行截图



#45121541提交状态

查看 提交 统计

状态: Accepted

源代码

```
from collections import deque

class Node():
    def __init__(self,value):
        self.value = value
        self.left = None
        self.right = None
```

基本信息

#: 45121541
题目: 22485
提交人: 23n2300012254
内存: 3920kB
时间: 25ms
语言: Python3
提交时间: 2024-05-28 20:22:32

28203:【模板】单调栈

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

思路：原本想到用heapq按照从大到小输出带有下标的数字，然后用另一从小到大弹出heapq存储已经存在的下标，即将最小下标的存在的大于当前数字的下标计入答案，但是涉及到存下标heapq弹出后需要依次放回或者使用copy，造成超时。

学习了题解中的单调栈模版，从前向后遍历，对于某遍历的数字，依次弹出stack中的已遍历数字，满足已遍历数字小于当前数字的话就存储该已遍历数字的答案下标为当前数字下标；不能在已遍历数字中找到符合要求下标的数字存在栈stack中。最后对于stack中剩下的不能找到更大数字的数答案存为0即可

代码

```
#赵语涵2300012254
n = int(input())
a = list(map(int, input().split()))
```

```python
stack = []

#f = [0]*n
for i in range(n):
    while stack and a[stack[-1]] < a[i]:
        #f[stack.pop()] = i + 1
        a[stack.pop()] = i + 1


    stack.append(i)

while stack:
    a[stack[-1]] = 0
    stack.pop()

print(*a)
```

代码运行截图

#45122387提交状态

查看 提交 统计

状态: Accepted

源代码

```python
#赵语涵2300012254
n = int(input())
a = list(map(int, input().split()))
stack = []

#f = [0]*n
for i in range(n):
```

基本信息

#: 45122387
题目: 28203
提交人: 23n2300012254
内存: 368460kB
时间: 2882ms
语言: Python3
提交时间: 2024-05-28 21:14:37

09202: 舰队、海域出击!

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

思路：按照群里的提示学习了拓扑排序的相关知识，这里用heap排序，为了使排序依照入点数目进行，建立Node类，并改写lt函数然而超时了。用题解中给的拓扑排序标准模版做的

```python
from collections import defaultdict
from queue import Queue   #FIFO
def build_graph(n,m):
    graph=defaultdict(list)
    for _ in range(m):
        x,y=map(int,input().split())
        graph[x].append(y)
    return graph

def topological_sort(graph):
    indegree = defaultdict(int)
    result = []
    queue = Queue()
```

```
    # 计算每个顶点的入度
    for u in graph:
        for v in graph[u]:
            indegree[v] += 1
    # 将入度为 0 的顶点加入队列
    for u in graph:
        if indegree[u] == 0:
            queue.put(u)
    # 执行拓扑排序
    while not queue.empty():
        u = queue.get()
        result.append(u)
        for v in graph[u]:
            indegree[v] -= 1
            if indegree[v] == 0:
                queue.put(v)
    # 检查是否存在环
    if len(result) == len(graph):
        return 'No'
    else:
        return 'Yes'
t=int(input())
for _ in range(t):
    n,m=map(int,input().split())
    graph=build_graph(n,m)
    print(topological_sort(graph))
```

代码运行截图

#45123910提交状态

查看　提交　统计

状态: Accepted

源代码

```
from collections import defaultdict
from queue import Queue   #FIFO
def build_graph(n,m):
    graph=defaultdict(list)
    for _ in range(m):
        x,y=map(int,input().split())
```

基本信息

#: 45123910
题目: 09202
提交人: 23n2300012254
内存: 84692kB
时间: 5213ms
语言: Python3
提交时间: 2024-05-28 23:25:40

04135: 月度开销

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

思路：最开始想到以前某个题的做法把间隔找出来以后按需要的数目将最小间隔的2个连起来。但是当m比较小只连接2个不能满足的时候就没法用这个方法于是WA了。看了题解自己完全没想到用二分查找做..感觉之前模版题练少了且没有互相联系。

代码

```python
#赵语涵2300012254
n,m = map(int, input().split())
expenditure = []
for _ in range(n):
    expenditure.append(int(input()))

def check(x):
    num, s = 1, 0
    for i in range(n):
        if s + expenditure[i] > x:
            s = expenditure[i]
            num += 1
        else:
            s += expenditure[i]

    return [False, True][num > m]

lo = max(expenditure)
hi = sum(expenditure) + 1
ans = 1
while lo < hi:
    mid = (lo + hi) // 2
    if check(mid):
        lo = mid + 1
    else:
        ans = mid
        hi = mid

print(ans)
```

代码运行截图

#45189975提交状态

查看 提交 统计

状态: Accepted

源代码

```python
#赵语涵2300012254
n,m = map(int, input().split())
expenditure = []
for _ in range(n):
    expenditure.append(int(input()))

def check(x):
```

基本信息

#: 45189975
题目: 04135
提交人: 23n2300012254
内存: 7492kB
时间: 522ms
语言: Python3
提交时间: 2024-06-03 17:26:05

07735: 道路

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

思路：因为要去道路最小金额也不能超，不能直接更新更新某节点的路径值，而dfs用heap操作的方法，只是把较大路径值的节点放在heap堆后面没有删去，可以得出答案。

代码

```python
import heapq
from collections import defaultdict

MAX_COINS = int(input())  # 最大金币数
CITY_COUNT = int(input())  # 城市数目
ROAD_COUNT = int(input())

# 存储道路信息的字典，使用 defaultdict 初始化
roads = defaultdict(list)

for _ in range(ROAD_COUNT):
    start, end, length, money = map(int, input().split())
    start, end = start - 1, end - 1
    roads[start].append((end, length, money))


def bfs(start, end, max_coins):
    queue = [(0, max_coins, start)]  # (距离，剩余金币，当前城市)
    visited = set()

    while queue:
        distance, coins, city = heapq.heappop(queue)

        if city == end:
            return distance

        visited.add((city, coins))

        for next_city, road_length, road_money in roads[city]:
            if coins >= road_money:
                new_distance = distance + road_length
                if (next_city, coins - road_money) not in visited:
                    heapq.heappush(queue, (new_distance, coins - road_money,
next_city))

    return -1


print(bfs(0, CITY_COUNT - 1, MAX_COINS))
```

代码运行截图

#45190482提交状态 查看 提交 统计

状态: Accepted

源代码

```
import heapq
from collections import defaultdict

MAX_COINS = int(input())  # 最大金币数
CITY_COUNT = int(input())  # 城市数目
ROAD_COUNT = int(input())
```

基本信息

#: 45190482
题目: 07735
提交人: 23n2300012254
内存: 5760kB
时间: 44ms
语言: Python3
提交时间: 2024-06-03 18:18:30

01182: 食物链

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

思路：自己是用了parent指示类别，然后用各种分类讨论的方法最后TLE了，而题解用了并查集解决耗时过长问题，另外原本思路中的re()整理过程用并查集的合并完成。

代码

```
class DisjointSet:
    def __init__(self, n):
        #设[1,n] 区间表示同类，[n+1,2*n]表示x吃的动物，[2*n+1,3*n]表示吃x的动物。
        self.parent = [i for i in range(3 * n + 1)] # 每个动物有三种可能的类型，用 3 * n 来表示每种类型的并查集
        self.rank = [0] * (3 * n + 1)

    def find(self, u):
        if self.parent[u] != u:
            self.parent[u] = self.find(self.parent[u])
        return self.parent[u]

    def union(self, u, v):
        pu, pv = self.find(u), self.find(v)
        if pu == pv:
            return False
        if self.rank[pu] > self.rank[pv]:
            self.parent[pv] = pu
        elif self.rank[pu] < self.rank[pv]:
            self.parent[pu] = pv
        else:
            self.parent[pv] = pu
            self.rank[pu] += 1
        return True


def is_valid(n, k, statements):
    dsu = DisjointSet(n)

    def find_disjoint_set(x):
```

```
        if x > n:
            return False
        return True

    false_count = 0
    for d, x, y in statements:
        if not find_disjoint_set(x) or not find_disjoint_set(y):
            false_count += 1
            continue
        if d == 1:  # X and Y are of the same type
            if dsu.find(x) == dsu.find(y + n) or dsu.find(x) == dsu.find(y + 2 * n):
                false_count += 1
            else:
                dsu.union(x, y)
                dsu.union(x + n, y + n)
                dsu.union(x + 2 * n, y + 2 * n)
        else:  # X eats Y
            if dsu.find(x) == dsu.find(y) or dsu.find(x + 2*n) == dsu.find(y):
                false_count += 1
            else: #[1,n] 区间表示同类，[n+1,2*n]表示x吃的动物，[2*n+1,3*n]表示吃x的动物
                dsu.union(x + n, y)
                dsu.union(x, y + 2 * n)
                dsu.union(x + 2 * n, y + n)

    return false_count


if __name__ == "__main__":
    N, K = map(int, input().split())
    statements = []
    for _ in range(K):
        D, X, Y = map(int, input().split())
        statements.append((D, X, Y))
    result = is_valid(N, K, statements)
    print(result)
```

代码运行截图

#45191013提交状态

查看 提交 统计

状态: Accepted

源代码

```
class DisjointSet:
    def __init__(self, n):
        #设[1,n] 区间表示同类，[n+1,2*n]表示x吃的动物，[2*n+1,3*n]表示吃x的动物
        self.parent = [i for i in range(3 * n + 1)] # 每个动物有三种可能的类
        self.rank = [0] * (3 * n + 1)

    def find(self, u):
```

基本信息

#:	45191013
题目:	01182
提交人:	23n2300012254
内存:	20324kB
时间:	679ms
语言:	Python3
提交时间:	2024-06-03 19:29:42

2. 学习总结和收获

感觉这一套题目很难，大部分是自己写了思路以后再去参考题解完成的。准备后面的上机考试，现在要整理下cheating paper记一些模版，但是遇到思路上没有头绪的题目可能还是会比较麻烦。