Assignment #F: All-Killed 满分

Updated 1844 GMT+8 May 20, 2024

2024 spring, Complied by 同学的姓名、院系

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/

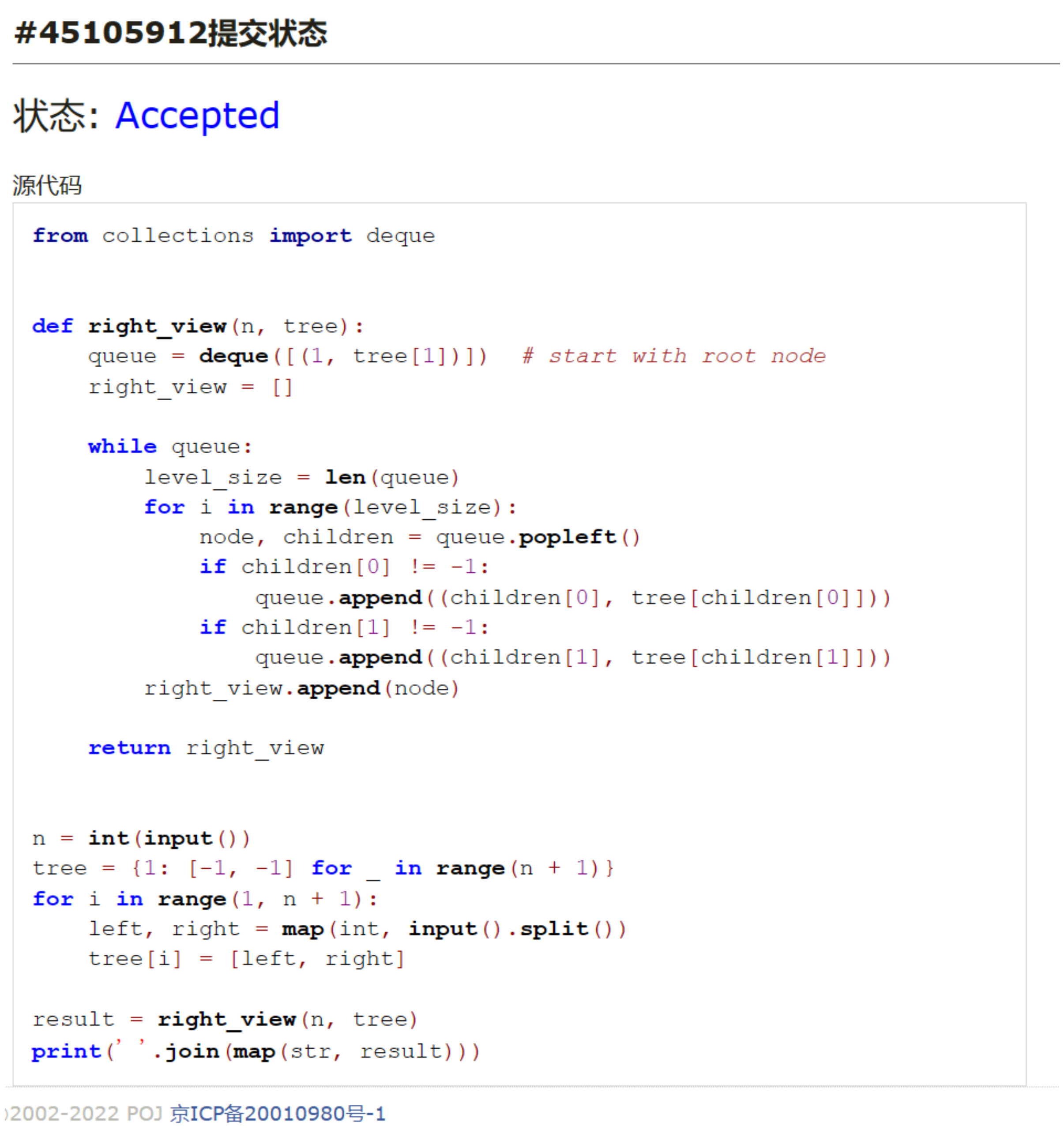
思路：

bsf遍历二叉树

代码‘

```python
# 
from collections import deque


def right_view(n, tree):
    queue = deque([(1, tree[1])])  # start with root node
    right_view = []

    while queue:
        level_size = len(queue)
        for i in range(level_size):
            node, children = queue.popleft()
            if children[0] != -1:
                queue.append((children[0], tree[children[0]]))
            if children[1] != -1:
                queue.append((children[1], tree[children[1]]))
        right_view.append(node)

    return right_view


n = int(input())
tree = {1: [-1, -1] for _ in range(n + 1)}
for i in range(1, n + 1):
    left, right = map(int, input().split())
    tree[i] = [left, right]

result = right_view(n, tree)
print(' '.join(map(str, result)))
```

代码运行截图 （至少包含有"Accepted"）



#45105912提交状态

状态: Accepted

源代码

```python
from collections import deque


def right_view(n, tree):
    queue = deque([(1, tree[1])])  # start with root node
    right_view = []

    while queue:
        level_size = len(queue)
        for i in range(level_size):
            node, children = queue.popleft()
            if children[0] != -1:
                queue.append((children[0], tree[children[0]]))
            if children[1] != -1:
                queue.append((children[1], tree[children[1]]))
        right_view.append(node)

    return right_view


n = int(input())
tree = {1: [-1, -1] for _ in range(n + 1)}
for i in range(1, n + 1):
    left, right = map(int, input().split())
    tree[i] = [left, right]

result = right_view(n, tree)
print(' '.join(map(str, result)))
```

2002-2022 POJ 京ICP备20010980号-1

28203:【模板】单调栈

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

思路：按照单调栈模版来做。www

代码

```python
# 单调栈
n = int(input())
a = list(map(int, input().split()))
stack = []

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


    stack.append(i)

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

print(*a) 这样写不会超时

# 单调栈
n = int(input())
a = list(map(int, input().split()))
b = [int(x) for x in range(1, n + 1)]
stack = []
for i in range(n):
    if a[i] == max(a):
        stack.append(0)
    for j in range(i + 1, n):
        if a[j] > a[i]:
            stack.append(j + 1)
            break
    print(stack)
print(*stack)

这样写超时
```

代码运行截图 (至少包含有"Accepted")

09202: 舰队、海域出击!

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

思路：dfs判断图中是否有环

代码

```
#
from collections import defaultdict

def dfs(p):
    vis[p] = True
    for q in graph[p]:
        in_degree[q] -= 1
        if in_degree[q] == 0:
            dfs(q)
```

```
for _ in range(int(input())):
    n, m = map(int, input().split())
    graph = defaultdict(list)
    in_degree = [0] * (n + 1)
    vis = [False] * (n + 1)
    for _ in range(m):
        x, y = map(int, input().split())
        graph[x].append(y)
        in_degree[y] += 1
    for k in range(1, n + 1):
        if in_degree[k] == 0 and not vis[k]:
            dfs(k)
    flag = any(not vis[i] for i in range(1, n + 1))
    print('Yes' if flag else 'No')
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

04135: 月度开销

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

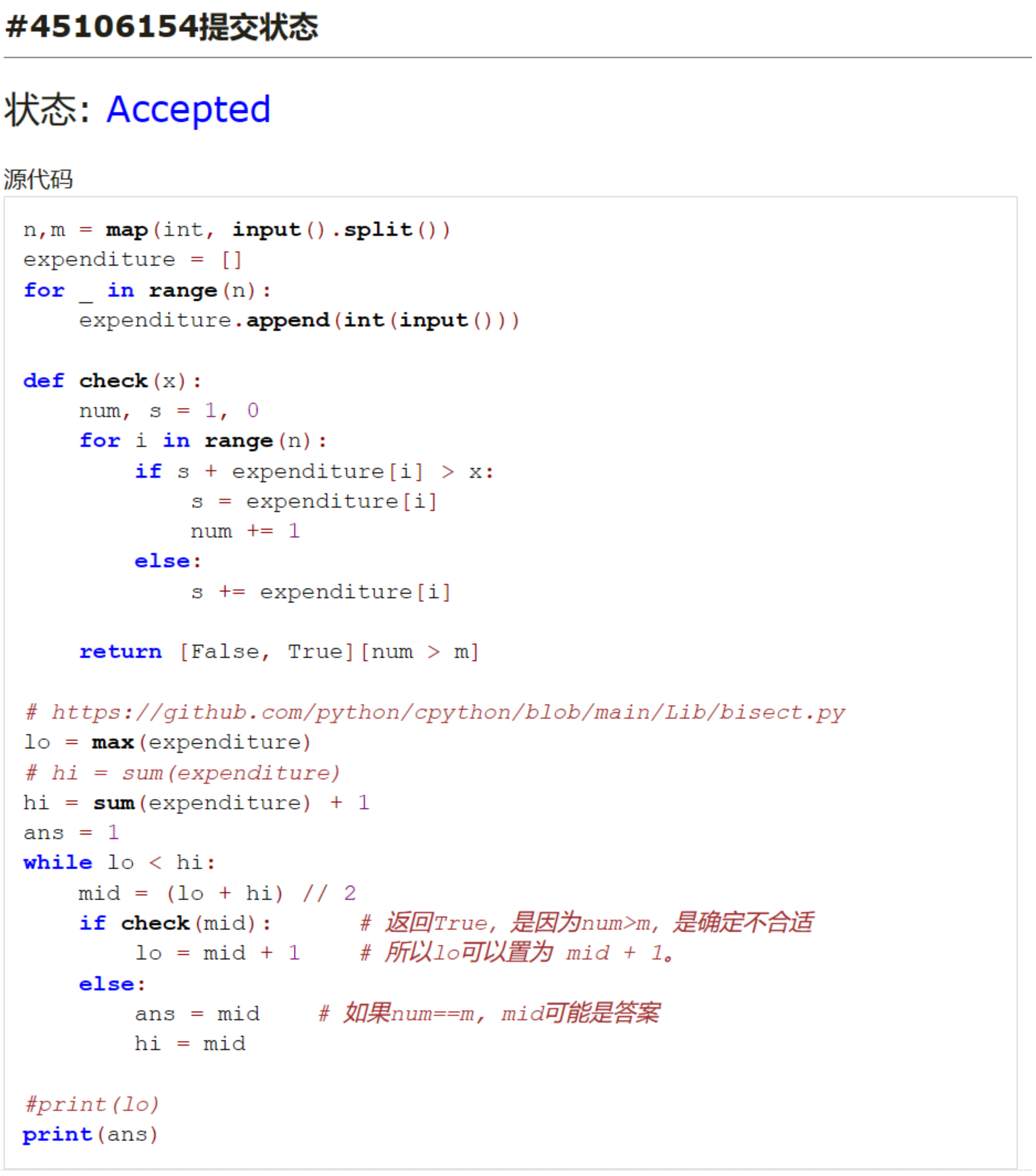
思路：二分查找，但是感觉没有接触过这类题目。

代码

```
#
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]

# https://github.com/python/cpython/blob/main/Lib/bisect.py
lo = max(expenditure)
# hi = sum(expenditure)
hi = sum(expenditure) + 1
```

```
ans = 1
while lo < hi:
    mid = (lo + hi) // 2
    if check(mid):       # 返回True，是因为num>m，是确定不合适
        lo = mid + 1     # 所以lo可以置为 mid + 1。
    else:
        ans = mid     # 如果num==m, mid可能是答案
        hi = mid

#print(lo)
print(ans)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#45106154提交状态

状态: Accepted

源代码

```
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]

# https://github.com/python/cpython/blob/main/Lib/bisect.py
lo = max(expenditure)
# hi = sum(expenditure)
hi = sum(expenditure) + 1
ans = 1
while lo < hi:
    mid = (lo + hi) // 2
    if check(mid):       # 返回True，是因为num>m，是确定不合适
        lo = mid + 1     # 所以lo可以置为 mid + 1。
    else:
        ans = mid     # 如果num==m, mid可能是答案
        hi = mid

#print(lo)
print(ans)
```

07735: 道路

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

思路：dijkstra算法

代码

```python
import heapq

def dijkstra(g):
    while pq:
        dist,node,fee = heapq.heappop(pq)
        if node == n-1 :
            return dist
        for nei,w,f in g[node]:
            n_dist = dist + w
            n_fee = fee + f
            if n_fee <= k:
                dists[nei] = n_dist
                heapq.heappush(pq,(n_dist,nei,n_fee))
    return -1

k,n,r = int(input()),int(input()),int(input())
g = [[] for _ in range(n)]
for i in range(r):
    s,d,l,t = map(int,input().split())
    g[s-1].append((d-1,l,t)) #node,dist,fee

pq = [(0,0,0)] #dist,node,fee
dists = [float('inf')] * n
dists[0] = 0
spend = 0

result = dijkstra(g)
print(result)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#45106180提交状态

状态: Accepted

源代码

```
import heapq

def dijkstra(g):
    while pq:
        dist,node,fee = heapq.heappop(pq)
        if node == n-1 :
            return dist
        for nei,w,f in g[node]:
            n_dist = dist + w
            n_fee = fee + f
            if n_fee <= k:
                dists[nei] = n_dist
                heapq.heappush(pq,(n_dist,nei,n_fee))
    return -1

k,n,r = int(input()),int(input()),int(input())
g = [[] for _ in range(n)]
for i in range(r):
    s,d,l,t = map(int,input().split())
    g[s-1].append((d-1,l,t)) #node,dist,fee

pq = [(0,0,0)] #dist,node,fee
dists = [float('inf')] * n
dists[0] = 0
spend = 0

result = dijkstra(g)
print(result)
```

01182: 食物链

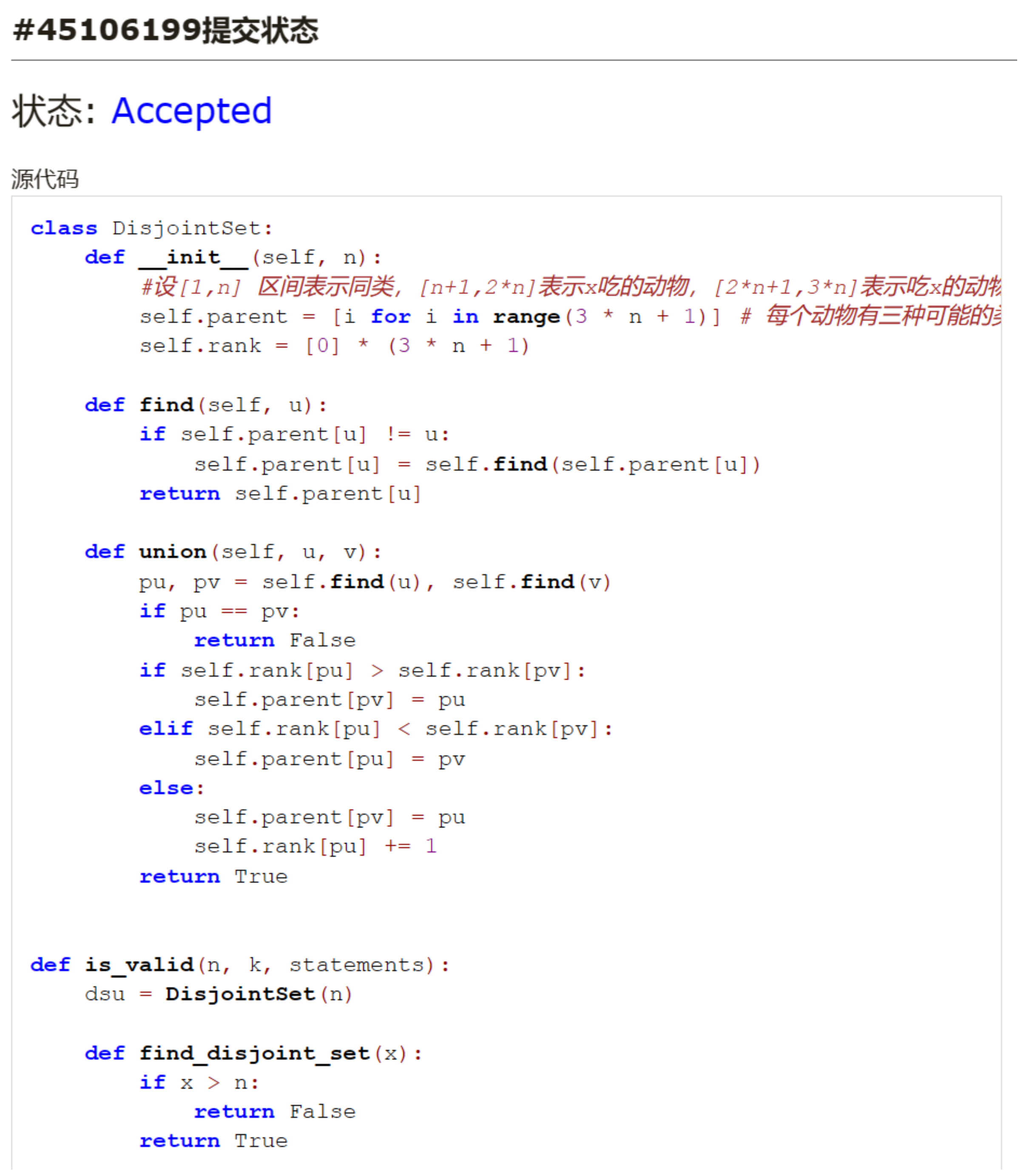
http://cs101.openjudge.cn/practice/01182/

思路：并查集

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 （AC代码截图，至少包含有"Accepted"）



#45106199提交状态

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):
        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
```

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。

本周复习了排序算法，队列栈和树的题目。但是本次作业中涉及图算法dijkstra和并查集还不太熟练。