Assignment #F: All-Killed 满分

Updated 1844 GMT+8 May 20, 2024

2024 spring, Complied by ~~天幂~~ 化学与分子工程学院

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 23H2

Python编程环境：Visual Studio Code 1.86.2230.

1. 题目

22485: 升空的焰火，从侧面看

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

思路：只是levelOrder的话有点懒得写类，不过意思到了就行。

代码

```
def check(str):
    x = int(str)
    if x == -1:
        return False
    else:
        return x
n = int(input())
tree = [[]] + [list(map(check, input().split())) for _ in range(n)]

def s(node, ans = [], level = 0):
    global tree
    try:
        ans[level]
    except IndexError:
        ans.append(-1)
```

```
    ans[level] = str(node)
    if tree[node][0]:
        s(tree[node][0], ans, level + 1)
    if tree[node][1]:
        s(tree[node][1], ans, level + 1)
    return ans

print(' '.join(s(1)))
```

代码运行截图

状态: Accepted

源代码

```
def check(str):
    x = int(str)
    if x == -1:
        return False
    else:
        return x
n = int(input())
tree = [[]] + [list(map(check, input().split())) for _ in range(n)]

def s(node, ans = [], level = 0):
    global tree
    try:
        ans[level]
    except IndexError:
        ans.append(-1)
    ans[level] = str(node)
    if tree[node][0]:
        s(tree[node][0], ans, level + 1)
    if tree[node][1]:
        s(tree[node][1], ans, level + 1)
    return ans

print(' '.join(s(1)))
```

28203:【模板】单调栈

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

思路：确实模板题，没啥可说的。

代码

```
n = int(input())
ans = [0 for _ in range(n)]
l = list(map(int, input().split()))
stack = []
i = 0
while i < n:
    while stack and l[i] > l[stack[-1]]:
        ans[stack.pop()] = i + 1
    stack.append(i)
    i += 1
print(*ans)
```

代码运行截图

状态: Accepted

源代码

```
n = int(input())
ans = [0 for _ in range(n)]
l = list(map(int, input().split()))
stack = []
i = 0
while i < n:
    while stack and l[i] > l[stack[-1]]:
        ans[stack.pop()] = i + 1
    stack.append(i)
    i += 1
print(*ans)
```

09202: 舰队、海域出击!

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

思路：拓扑排序法。和prim/Dijkstra算法有点相似，写起来可能比搜索更加顺手，而且似乎时间复杂度一致。

代码

```
t = int(input())
for _ in range(t):
    n, m = map(int, input().split())
    indegree = [0] * n
    fromnode = {i : [] for i in range(n)}
    for _ in range(m):
        sp, tp = map(int, input().split())
        sp, tp = sp - 1, tp - 1
```

```
        fromnode[sp].append(tp)
        indegree[tp] += 1
    visited = 0
    tovisit = []
    for i in range(n):
        if indegree[i] == 0:
            tovisit.append(i)
    while tovisit:
        cnode = tovisit.pop()
        visited += 1
        for node in fromnode[cnode]:
            indegree[node] -= 1
            if indegree[node] == 0:
                tovisit.append(node)
    print(['Yes', 'No'][visited == n])
```

代码运行截图

状态：Accepted

源代码

```
t = int(input())
for _ in range(t):
    n, m = map(int, input().split())
    indegree = [0] * n
    fromnode = {i : [] for i in range(n)}
    for _ in range(m):
        sp, tp = map(int, input().split())
        sp, tp = sp - 1, tp - 1
        fromnode[sp].append(tp)
        indegree[tp] += 1
    visited = 0
    tovisit = []
    for i in range(n):
        if indegree[i] == 0:
            tovisit.append(i)
    while tovisit:
        cnode = tovisit.pop()
        visited += 1
        for node in fromnode[cnode]:
            indegree[node] -= 1
            if indegree[node] == 0:
                tovisit.append(node)
    print(['Yes', 'No'][visited == n])
```

04135: 月度开销

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

思路：计概老代码。

代码

```
def check(t):
    global l
    global x
    global y
    s = l[0]
    n = 1
    for i in range(x-1):
        temp = s + l[i+1]
        if temp > t:
            s = l[i+1]
            n += 1
            if n > y:
                return False
        else:
            s = temp
    return True
x, y = map(int, input().split())
l = [int(input()) for _ in range(x)]
sety = set()
L = max(l)
R = sum(l)
while L < R:
    t = (L + R)//2
    if check(t):
        R = t
    else:
        L = t + 1
print(L)
```

代码运行截图

状态: Accepted

源代码

```
def check(t):
    global l
    global x
    global y
    s = l[0]
    n = 1
    for i in range(x-1):
        temp = s + l[i+1]
        if temp > t:
            s = l[i+1]
            n += 1
            if n > y:
                return False
        else:
            s = temp
    return True
x, y = map(int, input().split())
l = [int(input()) for _ in range(x)]
sety = set()
L = max(l)
R = sum(l)
while L < R:
        t = (L + R)//2
        if check(t):
                R = t
        else:
                L = t + 1
print(L)
```

07735: 道路

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

思路：根据资金进行剪枝的Dijkstra算法，模板题。

代码

```
from heapq import *
class Road:
    start:int
    destination:int
    length:int
    cost:int
    def __init__(self, s, d, l, t):
        self.start = s
        self.destination = d
```

```
        self.length = l
        self.cost = t
    def __lt__(self, other):
        return self.length < other.length
    def __eq__(self, other):
        return self.length == other.length

k = int(input())
n = int(input())
r = int(input())
roads = {city : [] for city in range(1, n + 1)}
for _ in range(r):
    s, d, l, t = map(int, input().split())
    newRoad = Road(s, d, l, t)
    roads[s].append(newRoad)

def work():
    ways = [(0, 1, k, [])]
    while ways:
        info = heappop(ways)
        distance, currentcity, remain, passedcities = info
        if currentcity == n:
            print(distance)
            return
        for roadfc in roads[currentcity]:
            if roadfc.destination in passedcities or roadfc.cost > remain:
                continue
            else:
                heappush(ways, (distance + roadfc.length, roadfc.destination,
remain - roadfc.cost, passedcities + [currentcity]))
    print(-1)
    return
work()
```

代码运行截图

状态: Accepted

源代码

```
from heapq import *
class Road:
    start:int
    destination:int
    length:int
    cost:int
    def __init__(self, s, d, l, t):
        self.start = s
        self.destination = d
        self.length = l
        self.cost = t
    def __lt__(self, other):
        return self.length < other.length
    def __eq__(self, other):
        return self.length == other.length

k = int(input())
n = int(input())
r = int(input())
roads = {city : [] for city in range(1, n + 1)}
for _ in range(r):
    s, d, l, t = map(int, input().split())
    newRoad = Road(s, d, l, t)
    roads[s].append(newRoad)

def work():
    ways = [(0, 1, k, [])]
    while ways:
        info = heappop(ways)
        distance, currentcity, remain, passedcities = info
        if currentcity == n:
            print(distance)
            return
        for roadfc in roads[currentcity]:
            if roadfc.destination in passedcities or roadfc.cost > rema
                continue
            else:
                heappush(ways, (distance + roadfc.length, roadfc.destina
    print(-1)
    return
work()
```

01182: 食物链

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

思路：来自于此处，非常优美的设计，利用了ABC之间的循环关系。

代码

```
class DisjointSet(object):
    size:int
    father_dict:dict
    fake:int
    def __init__(self, n):
        self.size = n
        self.father_dict = {}
        self.fake = 0
        for i in range(3 * n):
            self.father_dict[i] = i
    def find(self, x):
        if self.father_dict[x] == x:
            return x
        else:
            self.father_dict[x] = self.find(self.father_dict[x])
            return self.father_dict[x]
    def union(self, x, y):
        px = self.find(x)
        py = self.find(y)
        if px != py:
            self.father_dict[py] = px
            return 'No'
        else:
            return 'Yes'
    def op(self, type, a, b):
        if a > self.size or b > self.size:
            self.fake += 1
        elif type == 1:
            self.checknmerge(a - 1, b - 1)
        elif type == 2:
            self.checknseteat(a - 1, b - 1)


    def checknseteat(self, a, b):
        if self.find(a) == self.find(b) or self.find(a + self.size) ==
self.find(b):
            self.fake += 1
        else:
            self.seteat(a, b)

    def seteat(self, a, b):
        self.union(a, b + self.size)
        self.union(a + self.size, b + 2 * self.size)
        self.union(a + 2 * self.size, b)

    def checknmerge(self, a, b):
        if self.find(a) == self.find(b + self.size) or self.find(a +
self.size) == self.find(b):
            self.fake += 1
        else:
            self.merge(a, b)

```

```
    def merge(self, a, b):
        self.union(a, b)
        self.union(a + self.size, b + self.size)
        self.union(a + 2 * self.size, b + 2 * self.size)

n, k = map(int, input().split())
ds = DisjointSet(n)
for _ in range(k):
    ds.op(*map(int, input().split()))
print(ds.fake)
```

代码运行截图

状态: Accepted

源代码

```
class DisjointSet(object):
    size:int
    father_dict:dict
    fake:int
    def __init__(self, n):
        self.size = n
        self.father_dict = {}
        self.fake = 0
        for i in range(3 * n):
            self.father_dict[i] = i
    def find(self, x):
        if self.father_dict[x] == x:
            return x
        else:
            self.father_dict[x] = self.find(self.father_dict[x])
            return self.father_dict[x]
    def union(self, x, y):
        px = self.find(x)
        py = self.find(y)
        if px != py:
            self.father_dict[py] = px
            return 'No'
        else:
            return 'Yes'
    def op(self, type, a, b):
        if a > self.size or b > self.size:
            self.fake += 1
        elif type == 1:
            self.checknmerge(a - 1, b - 1)
        elif type == 2:
            self.checknseteat(a - 1, b - 1)


    def checknseteat(self, a, b):
        if self.find(a) == self.find(b) or self.find(a + self.size) == s
            self.fake += 1
        else:
            self.seteat(a, b)

    def seteat(self, a, b):
        self.union(a, b + self.size)
        self.union(a + self.size, b + 2 * self.size)
        self.union(a + 2 * self.size, b)

    def checknmerge(self, a, b):
        if self.find(a) == self.find(b + self.size) or self.find(a + sel
            self.fake += 1
        else:
            self.merge(a, b)

    def merge(self, a, b):
        self.union(a, b)
        self.union(a + self.size, b + self.size)
        self.union(a + 2 * self.size, b + 2 * self.size)

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

```
n, k = map(int, input().split())
ds = DisjointSet(n)
for _ in range(k):
    ds.op(*map(int, input().split()))
print(ds.fake)
```


2. 学习总结和收获

真得找点题目做了，拖到学期末了233