Assignment #F: All-Killed 满分

Updated 1844 GMT+8 May 20, 2024

2024 spring, Complied by 郑铭毅 数学科学学院

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

Windows 11

PyCharm

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:

代码

```
from collections import deque
class TreeNode:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
    def right_view(self):
        queue=deque([(self,0)])
        result=[]
        current_level=-1
        while queue:
            node,level=queue.popleft()
            if level!=current_level:
                result.append(node.value)
                current_level=level
            if node.right:
                queue.append((node.right,level+1))
            if node.left:
                queue.append((node.left,level+1))
        return result
N=int(input())
L=[TreeNode(i) for i in range(N)]
for _ in range(N):
    a,b=map(int,input().split())
    if a!=-1 and b!=-1:
        L[_].left=L[a-1]
        L[_].right=L[b-1]
    elif a==-1 and b!=-1:
        L[_].right = L[b - 1]
    elif a!=-1 and b==-1:
        L[_].left = L[a - 1]
R=L[0].right_view()
R1=[i+1 for i in R]
print(*R1)
```

代码运行截图

#45113795提交状态

状态: Accepted

源代码

```
from collections import deque
class TreeNode:
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
    def right_view(self):
        queue=deque([(self,0)])
        result=[]
        current_level=-1
        while queue:
            node,level=queue.popleft()
            if level!=current_level:
                result.append(node.value)
                current_level=level
            if node.right:
                queue.append((node.right,level+1))
            if node.left:
                queue.append((node.left,level+1))
        return result
N=int(input())
L=[TreeNode(i) for i in range(N)]
for _ in range(N):
    a,b=map(int,input().split())
    if a!=-1 and b!=-1:
        L[_].left=L[a-1]
        L[_].right=L[b-1]
    elif a==-1 and b!=-1:
        L[_].right = L[b - 1]
    elif a!=-1 and b==-1:
        L[_].left = L[a - 1]
R=L[0].right_view()
R1=[i+1 for i in R]
print(*R1)
```

28203: 【模板】单调栈

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

思路:

代码

```
n=int(input())
nums=[int(i) for i in input().split()]
def f(i):
    global nums
    k=0
    if i<n:
        for j in range(i,n):
            if nums[i-1]<nums[j]:
                k=j+1
                break
    return k
result=[]
for _ in range(n):
    result.append(f(_+1))
print(*result)
```

代码运行截图

09202: 舰队、海域出击!

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

```
class Graph:
    def __init__(self):
        self.adj_list={}
    def add_vertex(self,v):
        if v not in self.adj_list:
            self.adj_list[v]=[]
    def add_edge(self,u,v):
        if u not in self.adj_list:
            self.add_vertex(u)
        if v not in self.adj_list:
            self.add_vertex(v)
        self.adj_list[u].append(v)
    def has_cycle(self):
        def dfs(v):
            if  visited[v]==1:
                return True
            if visited[v]==2:
                return False
            visited[v]=1
            for neighbor in self.adj_list[v]:
                if dfs(neighbor):
                    return True
            visited[v]=2
            return False
        visited={v:0 for v in self.adj_list}
        for v in self.adj_list:
            if visited[v]==0:
                if dfs(v):
                    return True
        return False
T=int(input())
for _ in range(T):
    n,m=map(int,input().split())
    graph=Graph()
    for i in range(m):
        u,v=map(int,input().split())
        graph.add_edge(u,v)
    if graph.has_cycle():
        print('Yes')
    else:
        print('No')
```

#45114113提交状态

状态: Accepted

基

源代码

```
class Graph:
    def __init__(self):
        self.adj_list={}
    def add_vertex(self,v):
        if v not in self.adj_list:
            self.adj_list[v]=[]
    def add_edge(self,u,v):
        if u not in self.adj_list:
            self.add_vertex(u)
        if v not in self.adj_list:
            self.add_vertex(v)
        self.adj_list[u].append(v)
    def has_cycle(self):
        def dfs(v):
            if  visited[v]==1:
                return True
            if visited[v]==2:
                return False
            visited[v]=1
            for neighbor in self.adj_list[v]:
                if dfs(neighbor):
                    return True
            visited[v]=2
            return False
        visited={v:0 for v in self.adj_list}
        for v in self.adj_list:
            if visited[v]==0:
                if dfs(v):
                    return True
        return False
T=int(input())
for _ in range(T):
    n,m=map(int,input().split())
    graph=Graph()
    for i in range(m):
        u,v=map(int,input().split())
        graph.add_edge(u,v)
    if graph.has_cycle():
        print('Yes')
    else:
        print('No')
```

代码运行截图

04135: 月度开销

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

思路:

代码

```
def f(nums,N,M,x):
    current_sum=0
    count=1
    for num in nums:
        if num+current_sum>x:
            count+=1
            current_sum=num
            if count>M:
                return False
        else:
            current_sum+=num
    return True
def m(N,M,nums):
    left=max(nums)
    right=sum(nums)
    while left<right:
        mid=(left+right)//2
        if f(nums,N,M,mid):
            right=mid
        else:
            left=mid+1
    return left
N,M=map(int,input().split())
nums=[int(input()) for _ in range(N)]
print(m(N,M,nums))
```

代码运行截图 

07735: 道路

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

思路:

代码

```
#

```

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:

代码

```
#

```

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

题目难度不大，适合用来复习和查缺补漏