Assignment #D: May月考

Updated 1654 GMT+8 May 8, 2024

2024 spring, Complied by 尹柚鑫 2100015878 光华管理学院

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：Win11

Python编程环境：jupter notebook

1. 题目

02808: 校门外的树

http://cs101.openjudge.cn/practice/02808/

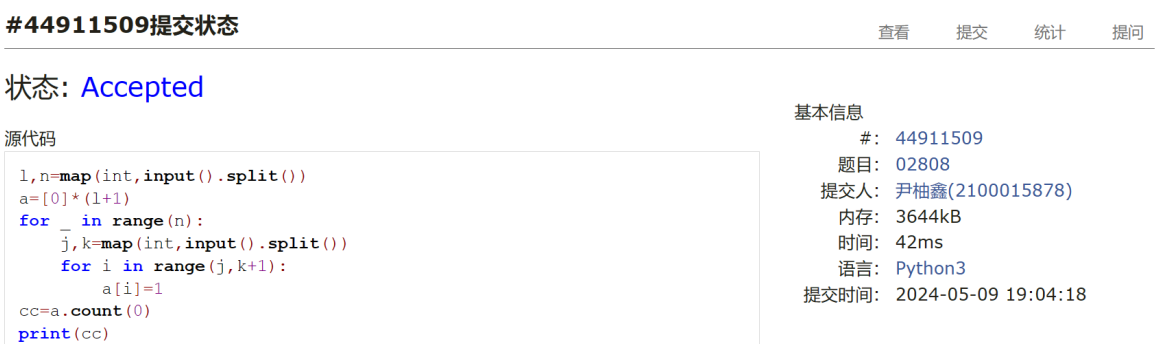
耗时：8mins

思路：最直接的方法，数数。之后可以想想怎么优化时间

代码

```
#
l,n=map(int,input().split())
a=[0]*(l+1)
for _ in range(n):
    j,k=map(int,input().split())
    for i in range(j,k+1):
        a[i]=1
cc=a.count(0)
print(cc)
```

代码运行截图 ==（至少包含有"Accepted"）==



#44911509提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
l,n=map(int,input().split())
a=[0]*(l+1)
for _ in range(n):
    j,k=map(int,input().split())
    for i in range(j,k+1):
        a[i]=1
cc=a.count(0)
print(cc)
```

基本信息

#: 44911509
题目: 02808
提交人: 尹柚鑫(2100015878)
内存: 3644kB
时间: 42ms
语言: Python3
提交时间: 2024-05-09 19:04:18

20449: 是否被5整除

http://cs101.openjudge.cn/practice/20449/

耗时：15mins

思路：一开始，转换二进制时，次数搞反了，一直WA，但是巧合的是，这个反例不多，白费很多时间

代码

```
#
numlist=[*map(int,input())]
cal=[2**i for i in range(len(numlist))]
ans=[]

for i in range(len(numlist)):
    result=0
    for j in range(i+1):
        result+=numlist[j]*cal[len(numlist)-1-j]
    index=0
    if result % 5==0:
        index=1
    ans.append(str(index))
print(''.join(ans))
```

代码运行截图 ==（至少包含有"Accepted"）==

#44918112提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
numlist=[*map(int,input())]
cal=[2**i for i in range(len(numlist))]
ans=[]

for i in range(len(numlist)):
    result=0
    for j in range(i+1):
```

基本信息

#: 44918112
题目: 20449
提交人: 尹柚鑫(2100015878)
内存: 3616kB
时间: 23ms
语言: Python3
提交时间: 2024-05-10 11:54:18

01258: Agri-Net

http://cs101.openjudge.cn/practice/01258/

耗时：25mins

思路：经典的最小生成树

代码

```
#
import heapq
def prim(graph,start):
    pq=[]
    start.distance=0
    heapq.heappush(pq,(0,start))
    visited=set()

    while pq:
        currentDist,currentVert=heapq.heappop(pq)
        if currentVert in visited:
            continue
        visited.add(currentVert)

        for nextVert in currentVert.getConnections():
            weight=currentVert.getWeight(nextVert)
            if nextVert not in visited and weight<nextVert.distance:
                nextVert.distance=weight
                nextVert.pred=currentVert
                heapq.heappush(pq,(weight,nextVert))

class Vertex:
    def __init__(self,key):
        self.id=key
        self.connectedTo={}
        self.distance=100001
        self.pred=None

    def addNeighbor(self,nbr,weight=0):
        self.connectedTo[nbr]=weight

    def getConnections(self):
        return self.connectedTo.keys()

    def getWeight(self,nbr):
        return self.connectedTo[nbr]

    def __lt__(self,other):
        return self.distance<other.distance

class Graph:
    def __init__(self):
        self.vertList={}
        self.numVertices=0
```

```
    def addVertex(self,key):
        newVertex=Vertex(key)
        self.vertList[key]=newVertex
        self.numVertices+=1
        return newVertex

    def getVertex(self,n):
        return self.vertList.get(n)

    def addEdge(self,f,t,cost=0):
        if f not in self.vertList:
            self.addVertex(f)
        if t not in self.vertList:
            self.addVertex(t)

        self.vertList[f].addNeighbor(self.vertList[t],cost)
        self.vertList[t].addNeighbor(self.vertList[f],cost)



while True:
    try:
        n=int(input())
        matrix=[]
        for _ in range(n):
            aaa=list(map(int,input().split()))
            matrix.append(aaa)

        # 建图
        graph=Graph()
        for i in range(n):
            for j in range(i+1,n):
                graph.addEdge(i,j,matrix[i][j])
                graph.addEdge(j,i,matrix[i][j])



        prim(graph,graph.getVertex(0))

        res=0
        for vertex in graph.vertList.values():
            if vertex.pred:
                res+=vertex.distance
        print(res)
    except EOFError:
        break
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
import heapq
def prim(graph,start):
    pq=[]
    start.distance=0
    heapq.heappush(pq,(0,start))
    visited=set()

    while pq:
        currentDist,currentVert=heapq.heappop(pq)
        if currentVert in visited:
```

基本信息

#:	44950961
题目:	01258
提交人:	尹柚鑫(2100015878)
内存:	5508kB
时间:	55ms
语言:	Python3
提交时间:	2024-05-13 15:34:41

27635: 判断无向图是否连通有无回路(同23163)

http://cs101.openjudge.cn/practice/27635/

耗时：20mins

思路：

（1）判断无向图是否联通，随便找一个点，做BFS遍历，之后看visited列表，如果每个点都是visited，就联通，否则不联通

（2）判断无向图是否有回路，对还没有visited的点做DFS，看遍历过程中点的邻居有几个已经被visited了，如果超过一个邻居被visited，表明成环

代码

```
#
from collections import deque

def liantong(start):
    visited[start]=1

    queue=deque()
    queue.append(start)

    while queue:
        now=queue.popleft()
        for i in graph[now]:
            if visited[i]==0:
                visited[i]=1
                queue.append(i)


def huan(start):
    if visited[start]==1:
        return

    tem=0
    for nbr in graph[start]:
        tem+=visited[nbr]
    if tem>1:
        return True
```

```
    visited[start]=1
    for nbr in graph[start]:
        if huan(nbr):
            return True

    return False


n,m=map(int,input().split())
graph=[[] for i in range(n)]
for _ in range(m):
    a,b=map(int,input().split())
    graph[a].append(b)
    graph[b].append(a)

visited=[0]*n
liantong(0)
if sum(visited)==n:
    print("connected:yes")
else:
    print("connected:no")

visited=[0]*n
for i in range(n):
    if huan(i):
        print("loop:yes")
        break
else:
    print("loop:no")
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
from collections import deque

def liantong(start):
    visited[start]=1

    queue=deque()
    queue.append(start)

    while queue:
        now=queue.popleft()
```

基本信息

#:	44951500
题目:	27635
提交人:	尹柚鑫(2100015878)
内存:	3700kB
时间:	27ms
语言:	Python3
提交时间:	2024-05-13 16:07:19

27947: 动态中位数

http://cs101.openjudge.cn/practice/27947/

耗时：30mins

思路：用一个最大堆和一个最小堆得到动态中位数的方法非常有趣。

一开始想的是，能不能像维护堆一样，维护一个中位数堆，但是比较困难

用两个堆分别维护这个二叉树的左子树和右子树非常妙

代码

```
#
import heapq
def insert_heap(num):
    if not heap_max or num<=-heap_max[0]:
        heapq.heappush(heap_max,-num)
        if len(heap_max)>len(heap_min)+1:
            heapq.heappush(heap_min,-heap_max[0])
            heapq.heappop(heap_max)

    else:
        heapq.heappush(heap_min,num)
        if len(heap_min)>len(heap_max):
            heapq.heappush(heap_max,-heap_min[0])
            heapq.heappop(heap_min)

n=int(input())
for _ in range(n):
    numlist=list(map(int,input().split()))
    if len(numlist)%2==0:
        print(len(numlist)//2)
    else:
        print((len(numlist)+1)//2)
    heap_max=[]
    heap_min=[]
    res=[]
    for i in range(len(numlist)):
        insert_heap(numlist[i])
        if i%2==0:
            res.append(str(-heap_max[0]))
    aaa=' '.join(res)
    print(aaa)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#44952522提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq
def insert_heap(num):
    if not heap_max or num<=-heap_max[0]:
        heapq.heappush(heap_max,-num)
        if len(heap_max)>len(heap_min)+1:
            heapq.heappush(heap_min,-heap_max[0])
            heapq.heappop(heap_max)
```

基本信息

#: 44952522
题目: 27947
提交人: 尹柚鑫(2100015878)
内存: 12616kB
时间: 332ms
语言: Python3
提交时间: 2024-05-13 17:48:11

28190: 奶牛排队

http://cs101.openjudge.cn/practice/28190/

耗时：30mins

思路：我没有用单调栈做，是用一个简单的递归做的，思路如下

对于奶牛列表，找最小的数最后一次出现的位置（定义为min），再找这个最小的数的右边的数中最大的数（定义为max_right），这两个数之间，就是一个符合题意的一串奶牛，更新一下result。之后，min左边的数，形成一个奶牛列表，max_right右边的数，也形成一个奶牛列表，递归套用函数就行。

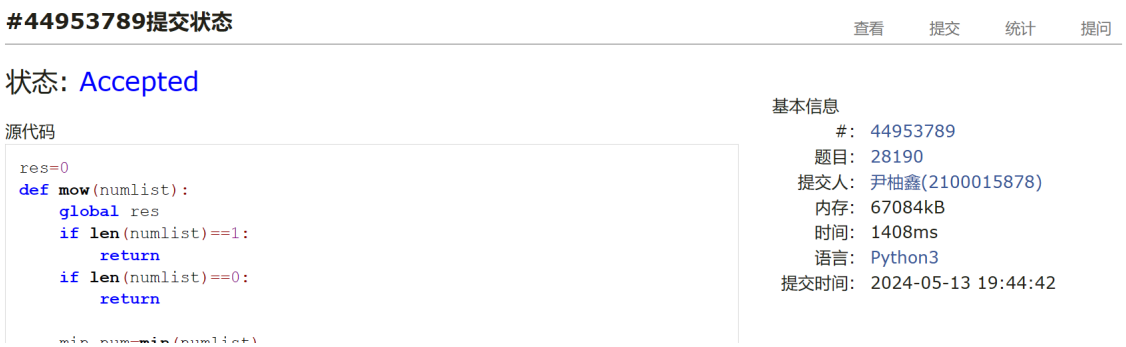
这个做法OJ是能过的，而且挺快的，但是在洛谷上，最后两个数据，会MLE，我不太知道怎么能减少空间的使用。

代码

```
#
res=0
def mow(numlist):
    global res
    if len(numlist)==1:
        return
    if len(numlist)==0:
        return

    #找最小的牛最后一次出现的位置
    min_num=min(numlist)
    min_ind=len(numlist)-numlist[::-1].index(min_num)-1
    if min_ind==len(numlist)-1:
        mow(numlist[:len(numlist)-1])
        return
    #找最小的牛右边最大的牛
    max_right_num=max(numlist[min_ind+1:])
    max_right_ind=numlist.index(max_right_num)

    if min_num==max_right_num:
        return
    #更新结果
    res=max(res,max_right_ind-min_ind+1)
    leftlist=numlist[:min_ind]
    rightlist=numlist[max_right_ind+1:]
    #递归
    mow(leftlist)
    mow(rightlist)
    return

n=int(input())
numlist=[]
for _ in range(n):
    a=int(input())
    numlist.append(a)
```

```
mow(numlist)
print(res)
```

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。==

考试还是挺难的，前四题和第六题还好，动态中位数这个，因为我做的题太少，确实没有想到怎么样实现这种结构，还是需要多做题。俩小时能勉强AC5，动态中位数不会写

奶牛这个题，让我想到了之前的一个题目，照着之前的写法想了想，其实想法有些相似，都是要找找最高或者最低的柱子，这个题也比较有意思，我摘下来

[42. 接雨水 - 力扣（LeetCode）]

42. 接雨水

已解答

困难 相关标签 相关企业

给定 `n` 个非负整数表示每个宽度为 `1` 的柱子的高度图，计算按此排列的柱子，下雨之后能接多少雨水。

示例 1：

> **输入：** `height = [0,1,0,2,1,0,1,3,2,1,2,1]`
> **输出：** 6
> **解释：** 上面是由数组 `[0,1,0,2,1,0,1,3,2,1,2,1]` 表示的高度图，在这种情况下，可以接 6 个单位的雨水（蓝色部分表示雨水）。

示例 2：

> **输入：** `height = [4,2,0,3,2,5]`
> **输出：** 9

这个柱子感觉和奶牛挺像的，这个接雨水，可以动态规划写，但是我当时想到了一个投机取巧的办法，很快就解决了。

想法就是，找最高的柱子max，对于max这个柱子左边的数，从左向右的顺序，第一根除外，当前数如果比左边的数小，就补齐为左边的数值大小；对于max这个柱子右边的数，从右向左的顺序，第一根除外，当前数如果比右边的数小，就补齐为右边的数。最后数一下一共补齐了多少就可以

```
class Solution:
 def trap(self, height: List[int]) -> int:
  maxval=max(height)
  ind_maxval=height.index(maxval)
  new_height=1*height
  for i in range(1,ind_maxval):
   if new_height[i]<new_height[i-1]:
    new_height[i]=1*new_height[i-1]
  for j in range(len(height)-2,ind_maxval,-1):
   if new_height[j]<new_height[j+1]:
    new_height[j]=1*new_height[j+1]
  old=sum(height)
  new=sum(new_height)
```

```
now=new-old

return now
```