Assignment #D: May月考

Updated 1654 GMT+8 May 8, 2024

2024 spring, Complied by 数学科学学院 王镜廷 2300010724

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：Windows11 专业版

Python编程环境：VSCode 1.86.2, with extension Python and python version 3.12.2

1. 题目

02808: 校门外的树

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

用时：约3分钟

思路：直接模拟

代码

```
L, M = map(int, input().split())
T = [1] * (L + 1)
for i in range(M) :
    u, v = map(int, input().split())
    for j in range(u, v + 1) :
        T[j] = 0
print(sum(T))
```

代码运行截图 （至少包含有"Accepted"）

20449: 是否被5整除

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

用时：约3分钟

思路：直接模拟

代码

```
A = input()
res = []
tmp = 0
for a in A :
    tmp = (2 * tmp + ord(a) - ord('0')) % 5
    res.append(1 if tmp == 0 else 0)
print("".join(str(a) for a in res))
```

代码运行截图 （至少包含有"Accepted"）

01258: Agri-Net

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

用时：约1小时（其中写代码写了10分钟，剩下50分钟在调程序，因为没看到多组数据，怎么交都不对）

思路：直接求最小生成树

代码

```python
class DisjointSet :
    def __init__(self, items) :
        self.rep = dict(zip(items, items))
        self.siz = dict(zip(items, [1] * len(items)))
    def getrep(self, item) :
        if self.rep[item] == item :
            return item
        else :
            self.rep[item] = self.getrep(self.rep[item])
            return self.rep[item]
    def merge(self, u, v) :
        fu = self.getrep(u)
        fv = self.getrep(v)
        su = self.siz[fu]
        sv = self.siz[fv]
        if fu == fv :
            return True
        else :
            if su > sv :
                self.rep[fv], self.siz[fu] = fu, su + sv
            else :
                self.rep[fu], self.siz[fv] = fv, su + sv
            return False

while True :
    try :
        N = int(input())
        E = []
        for i in range(N) :
            tmp = 0
            while tmp < N :
                u = list(map(int, input().split()))
                for j in range(len(u)) :
                    E.append((u[j], i, j + tmp))
                tmp += len(u)
        E = sorted(E)
        D = DisjointSet(list(i for i in range(N)))
        ans = 0
        for e in E :
            flag = D.merge(e[1], e[2])
            if not flag :
                ans += e[0]
        print(ans)
    except :
        break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44917903提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 44917903
题目: 01258
提交人: 23n2300010724
内存: 4800kB
时间: 67ms
语言: Python3
提交时间: 2024-05-10 11:25:30

源代码

```
class DisjointSet :
    def __init__(self, items) :
        self.rep = dict(zip(items, items))
        self.siz = dict(zip(items, [1] * len(items)))
    def getrep(self, item) :
        if self.rep[item] == item :
            return item
        else :
            self.rep[item] = self.getrep(self.rep[item])
            return self.rep[item]
    def merge(self, u, v) :
        fu = self.getrep(u)
        fv = self.getrep(v)
        su = self.siz[fu]
        sv = self.siz[fv]
        if fu == fv :
            return True
        else :
            if su > sv :
                self.rep[fv], self.siz[fu] = fu, su + sv
```

27635: 判断无向图是否连通有无回路(同23163)

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

用时：约13分钟

思路：用并查集实现（因为前一个题刚写过）

代码

```
class DisjointSet :
    def __init__(self, items) :
        self.rep = dict(zip(items, items))
        self.siz = dict(zip(items, [1] * len(items)))
    def getrep(self, item) :
        if self.rep[item] == item :
            return item
        else :
            self.rep[item] = self.getrep(self.rep[item])
            return self.rep[item]
    def merge(self, u, v) :
        fu = self.getrep(u)
        fv = self.getrep(v)
        su = self.siz[fu]
        sv = self.siz[fv]
        if fu == fv :
            return True
        else :
            if su > sv :
                self.rep[fv], self.siz[fu] = fu, su + sv
            else :
                self.rep[fu], self.siz[fv] = fv, su + sv
            return False

while True :
    try :
        N = int(input())
        E = []
        for i in range(N) :
            tmp = 0
            while tmp < N :
                u = list(map(int, input().split()))
                for j in range(len(u)) :
                    E.append((u[j], i, j + tmp))
                tmp += len(u)
        E = sorted(E)
        D = DisjointSet(list(i for i in range(N)))
        ans = 0
        for e in E :
            flag = D.merge(e[1], e[2])
            if not flag :
                ans += e[0]
        print(ans)
    except :
        break
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44917903提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class DisjointSet :
    def __init__(self, items) :
        self.rep = dict(zip(items, items))
        self.siz = dict(zip(items, [1] * len(items)))
    def getrep(self, item) :
        if self.rep[item] == item :
            return item
        else :
            self.rep[item] = self.getrep(self.rep[item])
            return self.rep[item]
    def merge(self, u, v) :
        fu = self.getrep(u)
        fv = self.getrep(v)
        su = self.siz[fu]
        sv = self.siz[fv]
        if fu == fv :
            return True
        else :
            if su > sv :
                self.rep[fv], self.siz[fu] = fu, su + sv
```

基本信息

#: 44917903
题目: 01258
提交人: 23n2300010724
内存: 4800kB
时间: 67ms
语言: Python3
提交时间: 2024-05-10 11:25:30

27947: 动态中位数

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

用时：约15分钟

思路：建立两个对顶堆。

代码

```python
import heapq as h

T = int(input())
for Case in range(T) :
    us = list(map(int, input().split()))
    q1 = [] # less than median
    q2 = [] # more than median
    h.heapify(q1)
    h.heapify(q2)
    siz1, siz2 = 0, 0
    res = []
    for u in us :
        if siz2 == 0 :
            h.heappush(q2, u)
            siz2 += 1
        else :
            v = h.heappop(q2)
            h.heappush(q2, v)
            if u >= v :
                h.heappush(q2, u)
                siz2 += 1
            else :
                h.heappush(q1, -u)
                siz1 += 1
        if siz1 > siz2 :
            u = -h.heappop(q1)
            siz1 -= 1
            h.heappush(q2, u)
            siz2 += 1
        if siz2 > siz1 + 1 :
            u = h.heappop(q2)
            siz2 -= 1
            h.heappush(q1, -u)
            siz1 += 1
        if siz2 == siz1 + 1 :
            u = h.heappop(q2)
            res.append(u)
            h.heappush(q2, u)
    print(len(res))
    print(" ".join(str(u) for u in res))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44908933提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
import heapq as h

T = int(input())
for Case in range(T) :
    us = list(map(int, input().split()))
    q1 = [] # less than median
    q2 = [] # more than median
    h.heapify(q1)
    h.heapify(q2)
    siz1, siz2 = 0, 0
    res = []
    for u in us :
        if siz2 == 0 :
            h.heappush(q2, u)
            siz2 += 1
        else :
            v = h.heappop(q2)
            h.heappush(q2, v)
            if u >= v :
                h.heappush(q2, u)
                siz2 += 1
```

基本信息

#: 44908933
题目: 27947
提交人: 23n2300010724
内存: 10504kB
时间: 502ms
语言: Python3
提交时间: 2024-05-09 15:58:06

28190: 奶牛排队

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

用时：约2小时

思路：这道题我前后想了好几个思路

思路1：时间复杂度$O(n\log n)$，空间复杂度$O(n\log n)$，注意到首先可以用$O(n\log n)$的时间将数据离散化同时保证无重复的值（在一点左边且与之值相同的视为比它大），之后考虑最大值所处位置M，最大值只能作为区间右端点或不在区间中，而最大值作为区间右端点的时候区间最长的左端点显然是在最大值位置以左的最小值，从而可以再递归处理$[1,M-1]$和$[M+1,N]$两个子区间。思路1是建立两个ST表分别用于O（1）时间求出最大值和最小值在每个区间中的位置，但是注意本题空间限制128M，显然会MLE，因此未实现该算法。

思路2：承接思路1（也做离散化从而使值互异），换一种方法求出每个区间中最大值与最小值位置，考虑维护一个类似于线段树但是不需要支持修改操作的树形结构，从而对于给定的区间可以在$O(\log n)$时间内查询区间内最大值的位置，之后同上递归处理，时间复杂度$O(n\log n)$，空间复杂度$O(n)$，在OJ上提交MLE了（不知道为啥），在洛谷上用PyPy 3作为语言提交，获得了30~70分不等（取决于评测机波动）。（我觉得如果改成C的话跑过1e5的数据范围应该问题不大，1e6就有点悬了，所以没有实现）

思路3：考虑计算每个点作为区间右端点的时候其所能找到的最远左端点，发现从左向右扫描的过程中维护两个单调栈（一个记录最大值，一个记录最小值）即可，这时所求的值即为在该点左侧第一个比该点的值大的点以右这个区间的最小值，也就是在该点左侧第一个比该点的值大的点以右的第一个在最小值那个单调栈里的点，可以在最小值的单调栈中二分查找得到结果。时间复杂度$O(n\log n)$，空间复杂度$O(n)$，在洛谷上用PyPy 3作为语言提交可以通过该题目，但是OJ上还是过不了。

思路4（这个复杂度感觉可能是错的，更像是乱搞，但是不知道为啥就是过了）：承接思路3，试图找一种方法快速查找每个点以右第一个在最小值那个单调栈中的点，注意每个点只会被加入一次，删除至多一次（一旦删掉就再也不会出现了），尝试维护一个数组nextmin记录这件事，之后每次更新的时候仿照并查集里面的思路做路径压缩，OJ上就通过了，洛谷上也能通过。

代码

```python
# 思路2
h = []
root = None
N = 0
class SEGtreenode :
    def __init__ (self, left, right, maxpos, minpos) :
        self.left = left
        self.right = right
        self.maxpos = maxpos
        self.minpos = minpos

def build(l, r) :
    global h
    if l == r :
        return SEGtreenode(None, None, l, l)
    m = (l + r) // 2
    lson = build(l, m)
    rson = build(m + 1, r)
    lmax, lmin = lson.maxpos, lson.minpos
    rmax, rmin = rson.maxpos, rson.minpos
    return SEGtreenode(lson, rson, lmax if h[lmax] > h[rmax] else rmax, lmin if h[lmin] < h[rmin] else rmin)
def getmaxpos(u, l, r, x, y) :
    if l >= x and r <= y :
        return u.maxpos
    if r < x or l > y :
        return -1
    m = (l + r) // 2
    resl = getmaxpos(u.left, l, m, x, y)
    resr = getmaxpos(u.right, m + 1, r, x, y)
    return resl + resr + 1 if (resr == -1 or resl == -1) else (resl if h[resl] > h[resr] else resr)
def getminpos(u, l, r, x, y) :
    if l >= x and r <= y :
        return u.minpos
    if r < x or l > y :
        return -1
    m = (l + r) // 2
    resl = getminpos(u.left, l, m, x, y)
    resr = getminpos(u.right, m + 1, r, x, y)
    return resl + resr + 1 if (resr == -1 or resl == -1) else (resl if h[resl] < h[resr] else resr)

def maxpos(l, r) :
    if l > r :
        return l
    global root, N
    return getmaxpos(root, 0, N - 1, l, r)
def minpos(l, r) :
    if l > r :
        return l
    global root, N
    return getminpos(root, 0, N - 1, l, r)
def calc(l, r) :
    if l >= r :
        return 0
    MAXpos = maxpos(l, r)
    MINpos = minpos(l, MAXpos - 1)
    return max(calc(l, MAXpos - 1), max(calc(MAXpos + 1, r), MAXpos - MINpos + 1 if MINpos != MAXpos else 0))

N = int(input())
for i in range(N) :
    h.append((int(input()), -i))
h = sorted(h)
for i in range(N) :
    h[i] = (-h[i][1], i + 1)
h = sorted(h)
for i in range(N) :
    h[i] = h[i][1]
root = build(0, N - 1)
```

```
print(calc(0, N - 1))
```

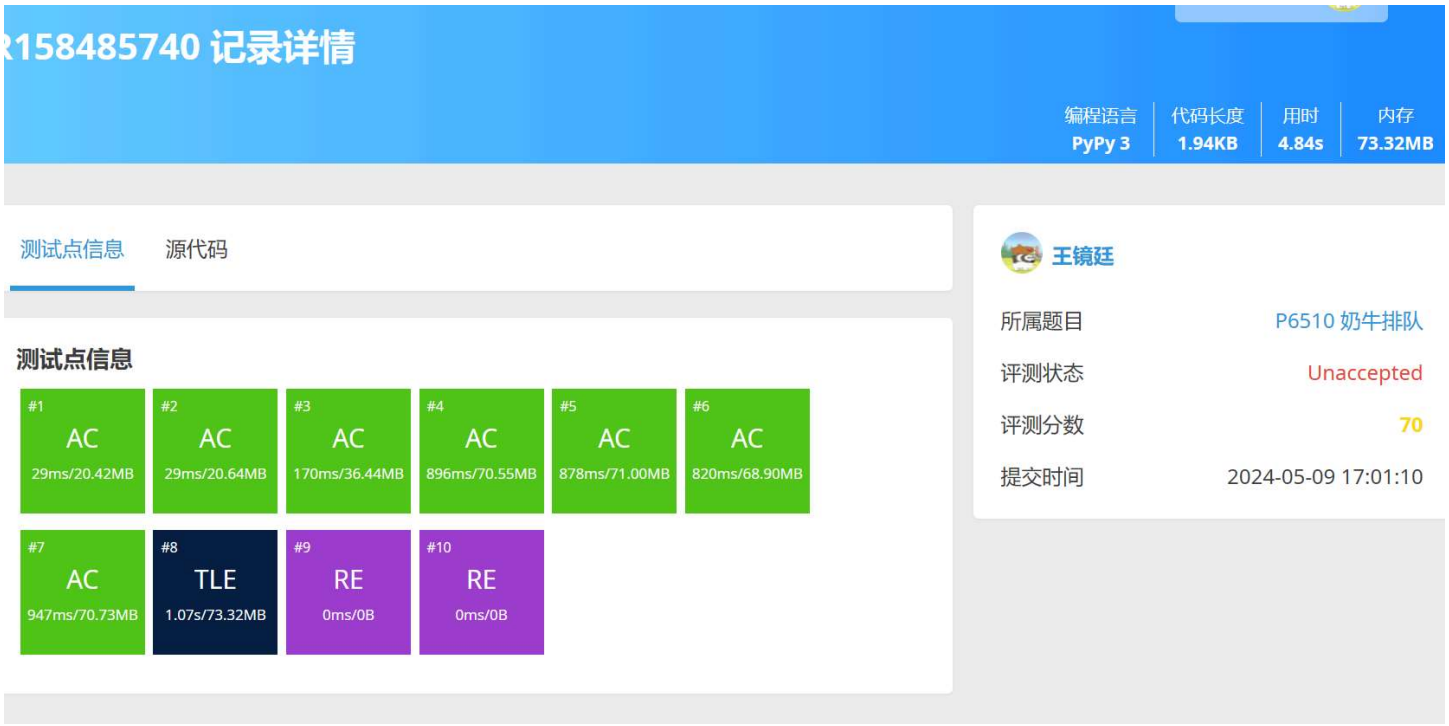
```
# 思路3
from collections import deque

N = int(input())
h = []
for i in range(N) :
    h.append(int(input()))
qmax = []
qmin = []
res = 0
for i in range(N) :
    while qmax and h[qmax[-1]] < h[i]:
        qmax.pop()
    qmax.append(i)
    while qmin and h[qmin[-1]] >= h[i] :
        qmin.pop()
    qmin.append(i)  # qmin increase
    #print(qmax, qmin)
    u = 0 if len(qmax) == 1 else qmax[-2]
    l, r = 0, len(qmin) - 1
    while r >= l + 1 :
        m = (l + r) // 2
        if qmin[m] < u :
            l = m + 1
        else :
            r = m
    res = max(res, i - qmin[l] + 1 if qmin[l] != i else 0)
print(res)
```

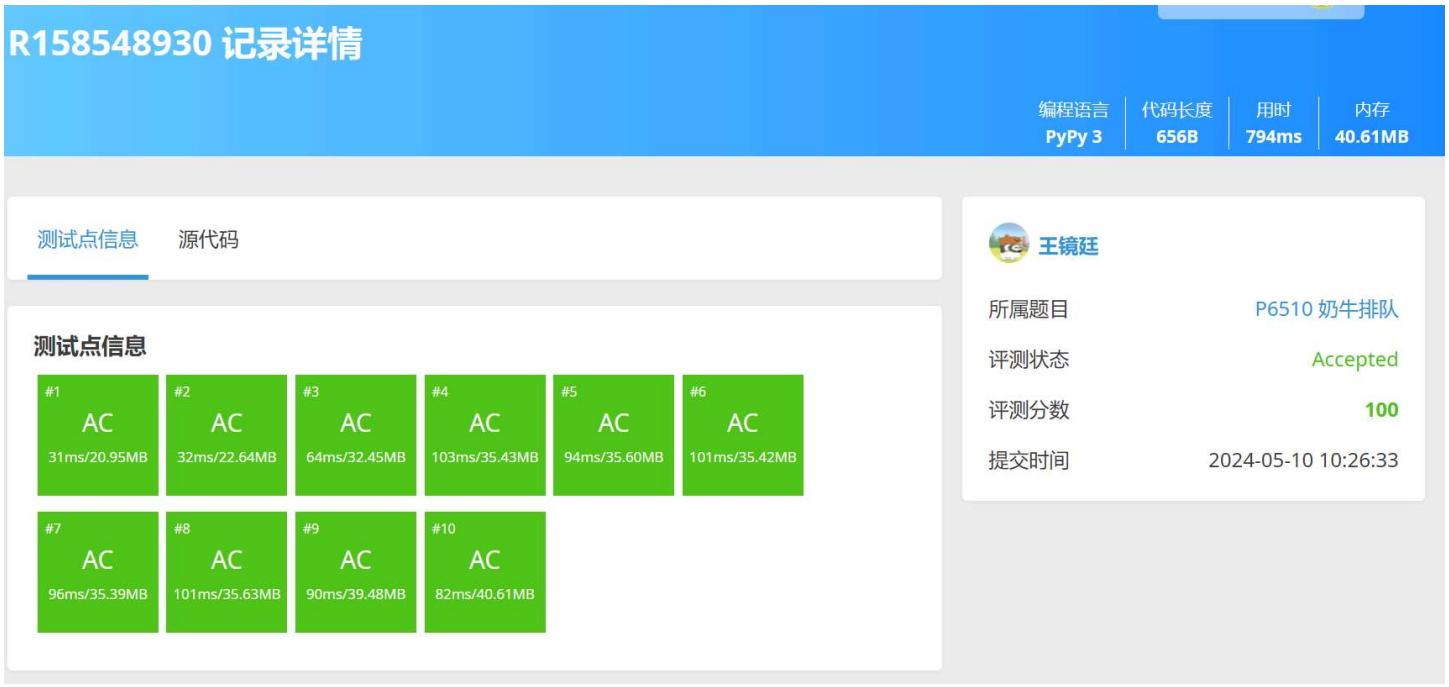
```
# 思路4
N = int(input())
h = []
for i in range(N) :
    h.append(int(input()))
qmax = []
qmin = []
nextmin = []
vismin = set()
res = 0
for i in range(N) :
    while qmax and h[qmax[-1]] < h[i]:
        qmax.pop()
    qmax.append(i)
    while qmin and h[qmin[-1]] >= h[i] :
        vismin.remove(qmin[-1])
        qmin.pop()
    qmin.append(i)  # qmin increase
    vismin.add(i)
    nextmin.append(i)
    #print(qmax, qmin)
    u = 0 if len(qmax) == 1 else qmax[-2]
    l = nextmin[u]
    tmpstack = [u]
    while l not in vismin :
        tmpstack.append(l)
        l = nextmin[l + 1]
    for tmp in tmpstack :
        nextmin[tmp] = l
    res = max(res, i - l + 1 if l != i else 0)
print(res)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

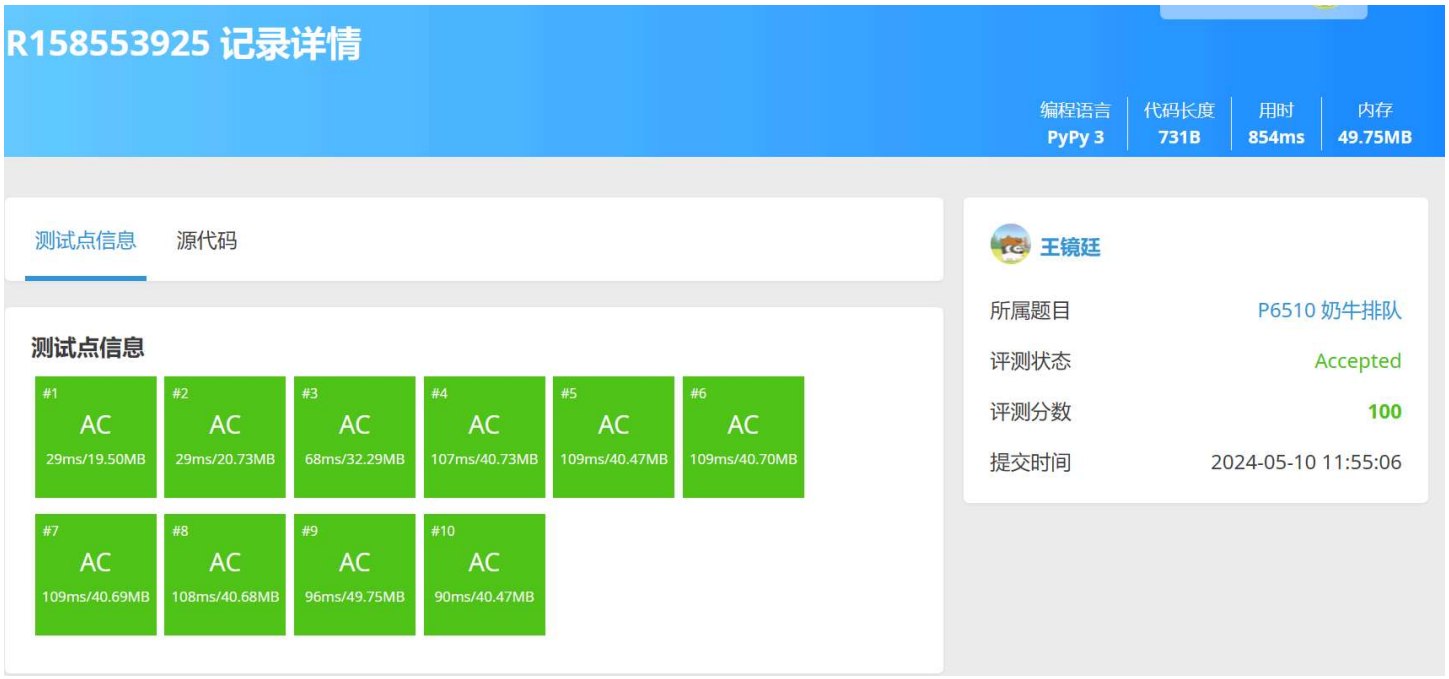
思路2在洛谷提交：



思路3在洛谷提交：



思路4在洛谷提交：



思路4在OJ提交：

2. 学习总结和收获

感觉需要加强一下基本功了。