Assignment #7: 贪心和DP

Updated 0919 GMT+8 Oct 24, 2023

2023 fall, Complied by Xinjie Song, Phy

说明：

1）请把每个题目解题思路（可选），源码Python, 或者C++/C（已经在Codeforces/Openjudge上AC），截图（包含Accepted, 学号），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。

3）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、作业评论有md或者doc。

4）如果不能在截止前提交作业，请写明原因。

另外，CF的题目，在洛谷有中文翻译，例如 https://www.luogu.com.cn/problem/CF1764C

编程环境

操作系统：Windows 11 22H2

Python编程环境：PyCharm 2023.2 (Community Edition)

C/C++编程环境：g++ (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0

1. 必做题目

158B. Taxi

*special problem, greedy, implementation, 1100

https://codeforces.com/problemset/problem/158/B

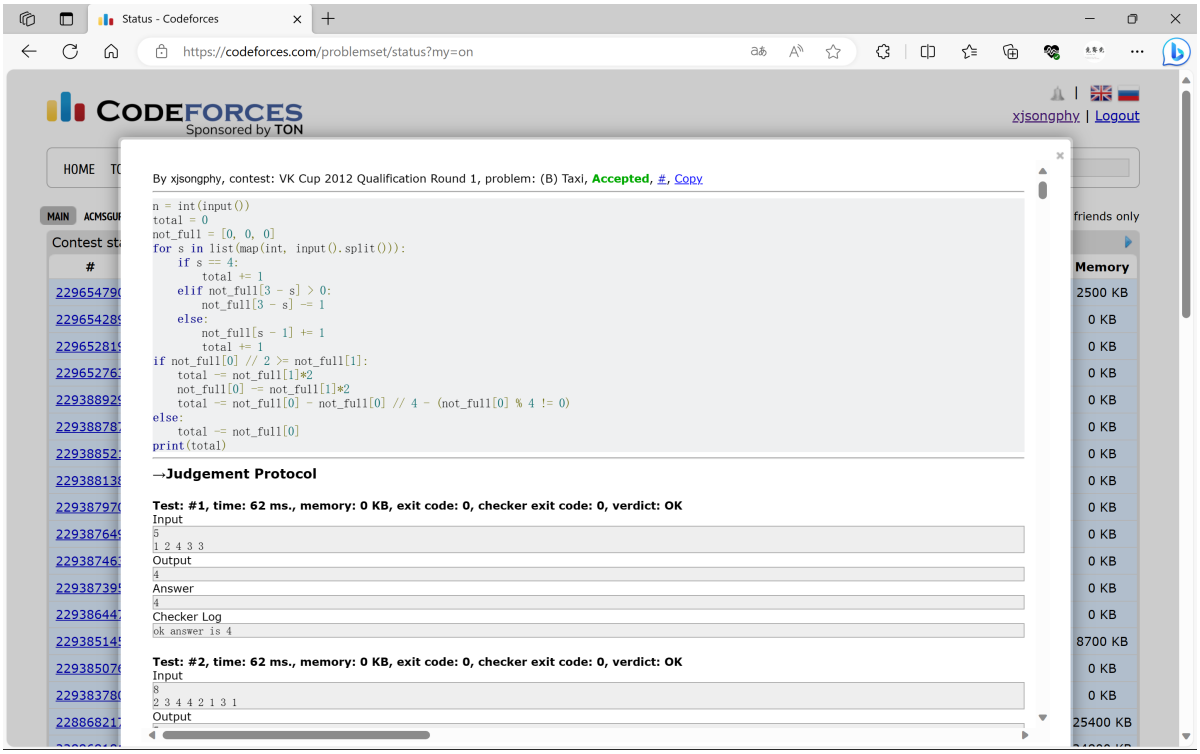
思路：4个人的小组坐一辆车；$1-3$个人的小组优先坐只坐了$4-i$个人的车，否则单独坐一辆车；处理完所有数据后，如果有只坐了1个人或2个人的车，则继续拼车，直至不能再减少车的数量为止。

代码

```
n = int(input())
total = 0
not_full = [0, 0, 0]
for s in list(map(int, input().split())):
    if s == 4:
        total += 1
    elif not_full[3 - s] > 0:
        not_full[3 - s] -= 1
    else:
```

```
        not_full[s - 1] += 1
        total += 1
if not_full[0] // 2 >= not_full[1]:
    total -= not_full[1]*2
    not_full[0] -= not_full[1]*2
    total -= not_full[0] - not_full[0] // 4 - (not_full[0] % 4 != 0)
else:
    total -= not_full[0]
print(total)
```

代码运行截图



545D. Queue

greedy, implementation, sortings, 1300

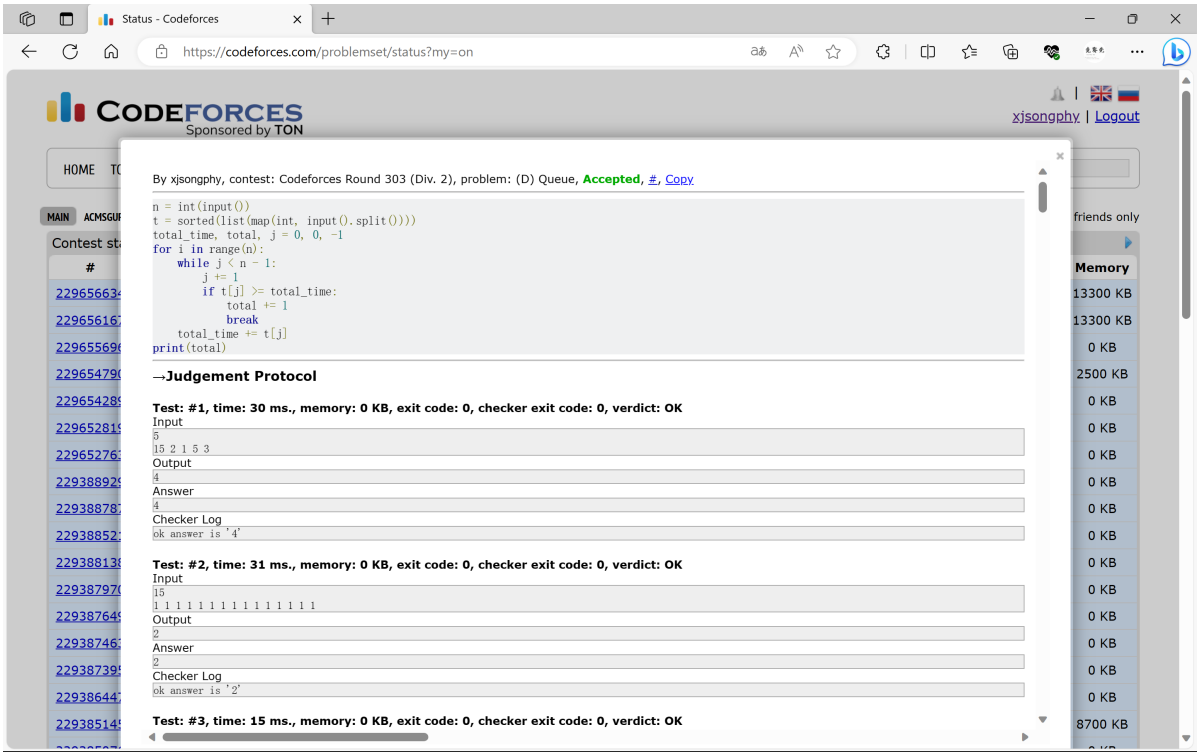
https://codeforces.com/problemset/problem/545/D

思路：将t_i由小到大排序，由时间最小的人开始，如果某个人A不满意，向后找到第一个在A的位置上能够满意的人B，此时，A原位置和B原位置之间的人不论是否交换A和B，都不会满意，否则，如果A和B之间由某个人C在原位置上满意，将C和A交换位置后，C在A的位置上一定满意，与B是A后面第一个在A的位置上能够满意的人矛盾；另外，不论B是否满意，交换B和A的位置后（其实最终会把A放到队尾），B以及B原位置之后的人满意的人数只会增多，不会变少；由于t_i由小到大排序，因此这个不满意的人和满意的人原来位置之间的人在原位置都不会满意，下一个位置可能满意的人一定在这个位置满意的人原来的位置之后，转化为双指针问题。

代码

```
n = int(input())
t = sorted(list(map(int, input().split())))
total_time, total, j = 0, 0, -1
for i in range(n):
    while j < n - 1:
        j += 1
        if t[j] >= total_time:
            total += 1
            break
    total_time += t[j]
print(total)
```

代码运行截图



803A. Maximal Binary Matrixcon

structive algorithms, 1400

https://codeforces.com/problemset/problem/803/A

思路：如果$k > n^2$，输出-1并退出；如果$k \leq n^2$，优先横向填充，否则填充在下一行对角线处。

代码

```
n, k = map(int, input().split())
if k > n**2:
    print(-1)
    exit()
matrix = [['0']*n for i in range(n)]
i, j = 0, 0
while k > 0:
    if i == j:
        matrix[i][j] = '1'
        k -= 1
    elif k > 1:
        matrix[i][j] = matrix[j][i] = '1'
        k -= 2
    else:
        matrix[i + 1][i + 1] = '1'
        k -= 1
    j += 1
    if j == n:
        i += 1
        j = i
for i in range(n):
    print(' '.join(matrix[i]))
```

代码运行截图

1793C. Dora and Search

constructive algorithms, data structures, two pointers, 1200,

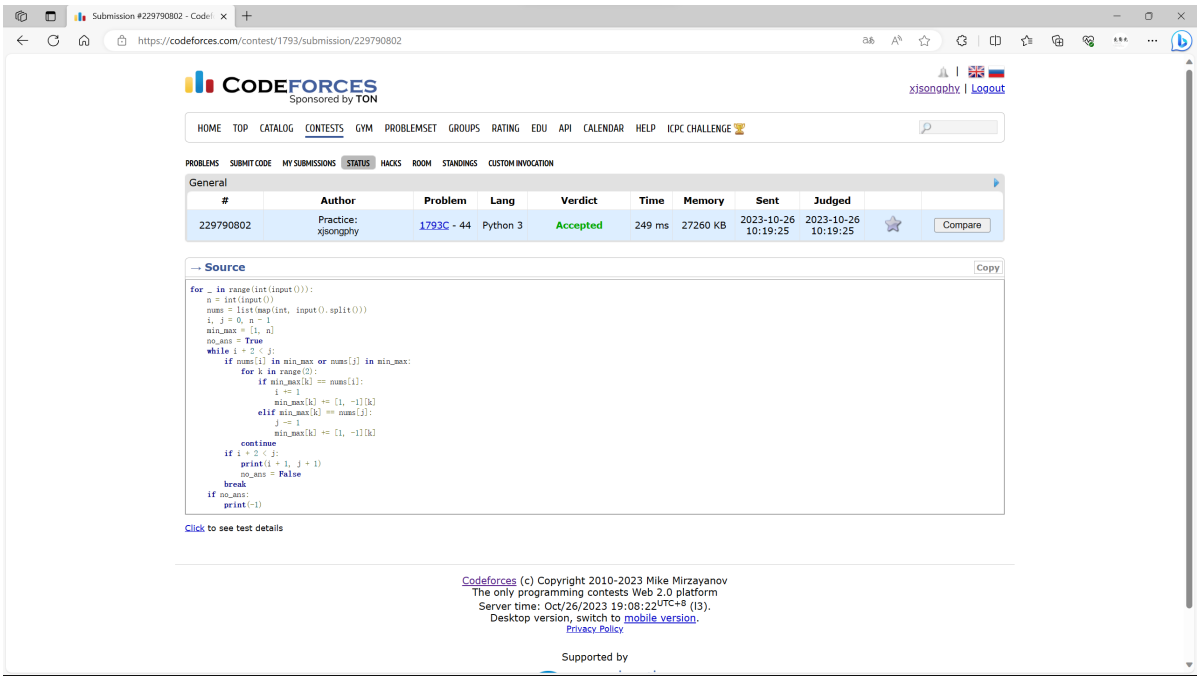
https://codeforces.com/problemset/problem/1793/C

思路：由于排列必为$1-n$，考虑1和n均不位于两端的情况，此时两端肯定不会是最大值或最小值，直接输出即可；若某一侧是1，检查其相邻位置是不是$1+1=2$或n，若不是，输出该位置和另一侧；若某一侧是n，检查其相邻位置是不是1或$n-1$，若不是，输出该位置和另一侧；若是，以此类推即可，注意最终输出时还需要判断片段中数字数量是否大于等于4。

代码

```
for _ in range(int(input())):
    n = int(input())
    nums = list(map(int, input().split()))
    i, j = 0, n - 1
    min_max = [1, n]
    no_ans = True
    while i + 2 < j:
        if nums[i] in min_max or nums[j] in min_max:
            for k in range(2):
                if min_max[k] == nums[i]:
                    i += 1
                    min_max[k] += [1, -1][k]
                elif min_max[k] == nums[j]:
                    j -= 1
                    min_max[k] += [1, -1][k]
            continue
        if i + 2 < j:
            print(i + 1, j + 1)
            no_ans = False
        break
    if no_ans:
        print(-1)
```

代码运行截图



2. 选做题目

368B. Sereja and Suffixes

data structures/dp, 1100

https://codeforces.com/problemset/problem/368/B

思路：读取输入后，找到最小的索引，从末尾开始，利用前一个索引的结果和字典快速查找的特性，反向依次计算出最小的索引及以后所有索引的答案存储在列表中，再根据输入顺序输出答案。

代码

```
n, m = map(int, input().split())
nums = list(map(int, input().split()))
indexes = [int(input()) - 1 for i in range(m)]
min_index = min(indexes)
distinct_nums, outputs = {}, [0]
i = n - 1
while i >= min_index:
    if distinct_nums.get(nums[i]):
        outputs.append(outputs[-1])
    else:
        distinct_nums[nums[i]] = 1
        outputs.append(outputs[-1] + 1)
    i -= 1
outputs.reverse()
for i in indexes:
    print(outputs[i - min_index])
```

代码运行截图

1764C. Doremy's City Construction

graphs, greedy, 1400

https://codeforces.com/problemset/problem/1764/C

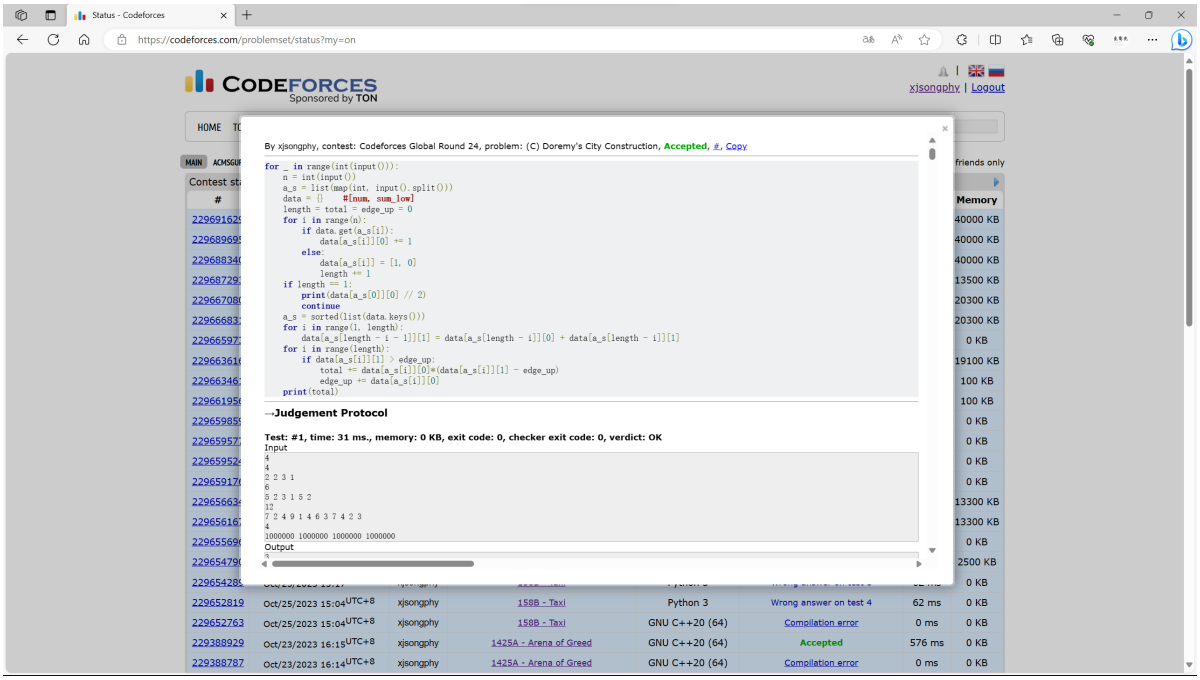
思路：首先对输入的数据进行处理，合并相同的高度，并记录每个高度出现几次，然后由低至高计算小于某个高度的位置总和（注意这里计算时可以用到上一个高度的数据，以减少用时），然后，由高至低判断某个高度应该向上联通还是向下联通，若向下则在edge_up中记录，以待下一个高度计算时使用（这里如果只有一个高度，那么输出这个高度数量整除2的值；若不同的高度大于一种，可以证明相互联通获得的边数必定大于某一高度内部两两一组联通得到的边数），最后输出总数即可。

代码

```
for _ in range(int(input())):
    n = int(input())
    a_s = list(map(int, input().split()))
    data = {}    #[num, sum_low]
    length = total = edge_up = 0
    for i in range(n):
        if data.get(a_s[i]):
            data[a_s[i]][0] += 1
        else:
            data[a_s[i]] = [1, 0]
            length += 1
    if length == 1:
        print(data[a_s[0]][0] // 2)
        continue
    a_s = sorted(list(data.keys()))
```

```
    for i in range(1, length):
        data[a_s[length - i - 1]][1] = data[a_s[length - i]][0] + data[a_s[length
- i]][1]
    for i in range(length):
        if data[a_s[i]][1] > edge_up:
            total += data[a_s[i]][0]*(data[a_s[i]][1] - edge_up)
            edge_up += data[a_s[i]][0]
    print(total)
```

代码运行截图



3. 学习总结和收获

这周自学了正则表达式，并利用正则表达式重做了邮箱验证题目；初步了解了lru_cache的使用方法，虽然还没有实际使用过。这次感觉作业题目难度有所上升，但解法应该都是合理的，没有出现类似背包问题贪心算法那样可能得到错误答案的解法（因为虽然我知道猜出一种贪心策略可以提交测试，但内心深处仍然抵触这种做法，感觉就是在赌，按下提交按钮的时候没有安全感）。

截至2023年10月26日，OJ完成题目84道，CF完成题目39道。