A 毒奶

chenjiuri_ccpc2022_weihai_buti

20mins

- 优先贪一个冠军的，两个冠军的，3个冠军的4个冠军的，5个冠军的。
- 先选一个冠军的，和先选两个冠军的之间有什么差别？如果现在的组队情况是，组完相同情况的队之后，现在它们之间，外面的各种类型的选手的数量应该是一样的。当前的冠军总是越多越好。
- 所以优选的先选冠军少的情况即可。
- 第一个，选择顺序的先后会不会产生一些影响?
    - 后面剩余的人中能够组成单队的可能越来越小。
    - 但是这两种是否等效?
    - 假设两种不会产生影响。

J Eat, Sleep, Repeat

- 题意:
    - 给定一个数组:
    - 给定k约束，代表某一个数出现的次数不能大于某一个数数字。
    - 两个人在博弈进行一下操作:
        - 选择一个数，并且将它减少1;
    - 博弈终点:
        - 无论选择什么数字，数字都为0。
        - 无论选择哪一个操作，都会出现某一个数字的个数超出限制超出限制时。

题解

- 设置 $limt[-1]=0$，每一个数字都可以减少若干次。
- $limit[x]=$ 无穷大。
- 关注几个特殊的情形
    - 限制的个数为0，数字就不可以经过该数。
- 相关实现
    - 第一点，进行分段。
    - 第二点，将每一段的情况计算，计算出总贡献。
- 关键问题，怎么处理分段?

    - 限制的存储问题，直接用map存储即可，但是这样不好遍历。
    - 用pair类型元素的vector来存储当前点的情况。
- 有没有可能从一开始，就行不通了？直接就是先手输。
    - 看题解应该没有卡这里。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef std::pair<int, int> iip;
const int maxn = 2e5 + 10;

void solve()
{
    ll n, k;
    std::cin >> n >> k;
    std::vector<int> a(n);
    for (int i = 0; i < n; i++)
        std::cin >> a[i];
    std::vector<iip> limit(k);
    // for (auto& [x, y] : limit)
    //     std::cin >> x >> y;
    for (int i = 0; i < k; i++)
        cin >> limit[i].first >> limit[i].second;
    limit.emplace_back(-1, 0);
    limit.emplace_back(2e9, 0);
    sort(a.begin(), a.end());
    sort(limit.begin(), limit.end());
    ll tot = 0;
    for (int i = 0; i < limit.size() - 1;)
    {
        //先找到上边界
        int j = i + 1;
        ll sum = 0;
        while (j < limit.size() && limit[j].second > 0)
            j++; //定位到上边界。
        int low = lower_bound(a.begin(), a.end(), limit[i].first) -
a.begin();
        int high = lower_bound(a.begin(), a.end(), limit[j].first) -
a.begin();
        //记录其中有多少个数字
        ll cunt = high - low;
        //两种方式：
        for (int k = low; k < high; k++) //首先搞完理想状态的一步一步迁移向，
次优的状态迁移计算。
            sum += a[k] - limit[i].first - 1;
        //开始减去其它部分；
        //指导找到步限制的点或者终点。
        int x = limit[i].first + 1;
        while (cunt && i < j) // i和j管理的是限制，x和y管理的是数组。
```

```
        {
            i++;
            if (x != limit[i].first)
                break;
            cunt -= limit[i].second;
            sum -= max(cunt, 0LL);
            x++;
        }
        tot += sum;
        i = j;
    }
    if (tot & 1)
        cout << "Pico\n";
    else
        cout << "FuuFuu\n";
}
int main()
{
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr), std::cout.tie(nullptr);
    int t;
    std::cin >> t;
    while (t--)
        solve();
}
```

- 等效问题，等效处理的思想。
- 计算的一种思想
  - 对于一个整体的量；比如这里指的是：
    - 计算出一个sum。不是一层一层累加起来。
    - 而是一层一层减。
  - 关于博弈角度：
    - 不知道怎么处理但是已经总结的经验有。
      - 坚持的正确的模拟一遍，反正有可能走不动了。
      - 反正就是找规律。关注一些特殊情形，关注博弈过程中的一些关注的量。比如某些资源的奇偶性。
  - 对自己来说，写这一份代码也有一定的难度。

C Grass

ad

- 给出一堆点，选出五个点，并且确定一个中点，使得A，B，C，D，E等等只有一个交叉点。
-

G

Grade two

```
给定x
一个区间【l，r】
寻找区间中的数，满足
gcd(x*k^x,x)==1;
```