DAY06

#A、 闰年产生的原因

令 $curh, curm, curs$ 分别表示 小时,分钟,秒钟 产生的误差

考虑第 m 年

每年误差

每一年会导致误差增加 $5h, 48min, 46s$

共有 m 次这样的误差

```
curh += m * 5;
curm += m * 48;
curs += m * 46;
```

每四年误差

每四年润一年,使误差减少 $24h$

但是注意润的年也有 $5h, 48min, 46s$ 误差

所以不需要画蛇添足减掉

共有 $\lfloor \frac{m}{4} \rfloor$ 次这样的误差

每一百年误差

每100年不润,由于上述操作每100年也润了

所以每100年误差增加 $24h$ 即可,相当于撤销闰年

共有 $\lfloor \frac{m}{100} \rfloor$ 次这样的误差

每四百年误差

每400年再润,由于上述操作,我们将400年的闰年撤销了

所以每400年误差减小 $24h$,相当于再加上闰年

共有 $\lfloor \frac{m}{400} \rfloor$ 次这样的误差

注意处理后输出

注意 $0 \leq curs, curm < 60$

所以 $curs$ 先向 $curm$ 进位, $curm$ 再向 $curh$ 进位

不难发现 $curs, curm$ 必为非负数,题目就是诈骗

#B、 大分数比较

作差法:对于两式 A, B ,有

- $A - B > 0 \implies A > B$
- $A - B = 0 \implies A = B$
- $A - B < 0 \implies A < B$

显然有 $\dfrac{x}{a} - \dfrac{y}{b} = \dfrac{bx - ay}{ab}$

由于 ab 必为正数, $bx - ay$ 的正负性即为 $\dfrac{x}{a} - \dfrac{y}{b}$ 的正负性

查看 $bx - ay$ 正负性即可

但是 $bx - ay$ 最坏情况下可到 10^{27} ,long long无法储存

方法1:__int128秒掉

__int128 是 C++ 的一种数据类型,可储存大约到 10^{35} 的整数

现在 __int128 允许在 CCF 系列赛事中使用

以下为 CCF 在 2024联合省选 中给出的 __int128 标准 IO(输入输出) 板子

```
void read(__int128 &x){
    // read a __int128 variable
    char c; bool f = 0;
    while(((c = getchar()) < '0' || c > '9') && c != '-');
    if(c == '-'){f = 1; c = getchar();}
    x = c - '0';
    while((c = getchar()) >= '0' && c <= '9')x = x * 10 + c - '0';
    if(f) x = -x;
}

void write(__int128 x){
    // print a __int128 variable
    if(x < 0){putchar('-'); x = -x;}
    if(x > 9)write(x / 10);
    putchar(x % 10 + '0');
}
```

有了数据类型的加持,其他都不是难事

方法2:转换成带分数运算

注意到 $a, b \le 10^9$,这是本题的突破口

不难将 $\dfrac{n}{m}$ 写为 $(\lfloor \dfrac{n}{m} \rfloor)\dfrac{n\%m}{m}$ 的带分数形式

同理, $\dfrac{x}{a} = (\lfloor \dfrac{x}{a} \rfloor)\dfrac{x\%a}{a}, \dfrac{y}{b} = (\lfloor \dfrac{y}{b} \rfloor)\dfrac{y\%b}{b}$

优先比较 $\lfloor \dfrac{x}{a} \rfloor$ 与 $\lfloor \dfrac{y}{b} \rfloor$ 的大小关系,能比较出则直接得出答案

否则则比较 $\dfrac{x\%a}{a}$ 和 $\dfrac{y\%b}{b}$ 的关系

不难发现 $(x\%a), (y\%b), a, b \le 10^9$,作差法利用long long存储比较即可

#C、 击杀怪兽

考虑一只目前高度为 h 的怪兽,写成 $h=pk+q(p,q\in\mathbb{N},1\leq q\leq k)$ 的形式

请务必注意上式和带余除法不完全相同 ,如 $h=5,k=5$,则 $h=0k+5$

若 $p=0$,则不难发现受到一次攻击后即死亡

否则,受到一次攻击后,血量变为 $h'=(p-1)k+q$,不难发现 $h\equiv h'\equiv q(mod\ k)$

即怪兽受到一次攻击后,其血量 $mod\ k$ 的结果不变,皆为 q

若原来有 n 只怪兽,血量分别为
$h_1=p_1k+q_1,h_2=p_2k+q_2,...,h_n=p_nk+q_n(1\leq q_1,q_2,...,q_n\leq k)$

不难发现对于 i,j 而言,若 $p_i\geq q_i$,则当前情况下 i 必然比 j 先受到攻击

所以必有一时刻,满足 **所有怪物都存活** ,且 $p_1=p_2=...=p_n=0$

即

$h'_1=q_1,h'_2=q_2,...,h'_n=q_n$

由于 $1\leq q_1,q_2,...,q_n\leq k$,所以当前所有怪物必将被 **一击必杀**

Mas的攻击顺序即为怪物死亡顺序,Mas将先按生命值(h'_i)攻击,相同时按编号攻击

使用结构体的多关键字排序即可,时间复杂度 $O(nlogn)$

千万注意 ,由于 $1\leq q_1,q_2,...,q_n\leq k$,所以若 $h_i\ mod\ k=0$,实际上 $q_i=k$

#D、 又是田忌赛马

前 $20pts$ 纯靠判断解决问题

如果你会搜索,那么 $40pts$ 是很好拿的

其次考虑可被优化的通解

为了利用单调性解决问题,不妨让田忌和齐威王的马都由速度值由小到大排序

计田忌的马的速度值为 $a_1,a_2,...,a_n$,齐威王马的速度值为 $b_1,b_2,...,b_n$

现在考虑田忌的每只马能产生的贡献,显然,该贡献不是 1 (打赢了齐威王的某只马)就是 0 (没有马让他打败了)

我们从前至后地考虑田忌的 n 只马,若当前考虑到第 i 只,齐威王未应战的马的集合为 T

首先尝试寻找 T 中 **最大的** $<a_i$ 的元素 b_j ,让 a_i 去击败 b_j 即可

不难发现,对于任何 k 满足 $b_k\leq b_j$,交换 b_k,b_j ,局面一定不可能变得更优

当然,该操作最多只可能让将来田忌的一只马有可能找不到能击败的对手,产生的贡献由 1 变为 0 ,但是由于这只马 a_i 已经产生了 1 的贡献,所以可证明让 a_i 击败 b_j 一定最优

若无法在 T 中找到合适的元素,说明这只马已经不可能产生 1 的贡献,即对结果无影响,直接跳过即可

直至将 a 数组遍历完毕,检查田忌所有马一共产生了多少贡献,即为可能的最大分数

若考虑最小分数,我们可以将 **齐威王与田忌交换** ,此时即求齐威王的马最多能产生多少贡献

注意 ,由于我们反着思考,所以双方平局的情况,对于齐威王的马是会产生为 1 的贡献的(即我们在 T 中寻找的是最大的 $\le b_i$ 的元素,此处取等)

最终用 n 减去齐威王的马产生的最大贡献,即为可能的最小分数

不难发现这种算法的时间复杂度主要依赖于在 T 中寻找元素的时间复杂度,若这一步时间复杂度为 $O(g(n))$,则总时间复杂度为 $O(ng(n))$

不难发现若对于每一次都把 T 暴力扫一遍, $g(n)=n$,总时间复杂度 $O(n^2)$,可拿到 $70pts$

优化思路1:数据结构优化

不难发现我们是在 T 中寻找第一个 $<$ 或 $\le$ 定值的元素

考虑维护单调性后二分,但是由于 T 会有删除操作,所以考虑使用 set 维护

若 set 名为 s

使用 `s.lower_bound(s.begin(),s.end(),val)` 可寻找第一个 $\ge val$ 的元素 **位置** ,返回迭代器,向前寻找一个即为最大的 $< val$ 的元素位置

使用 `s.upper_bound(s.begin(),s.end(),val)` 可寻找第一个 $> val$ 的元素 **位置** ,返回迭代器,向前寻找一个即为最大的 $\le val$ 的元素位置

$g(n)=logn$,总时间复杂度 $O(nlogn)$,可以通过

优化思路2:双指针优化

由大至小考虑 a 数组,即从 a_n 至 a_1 遍历 a

同理,由大到小考虑 b

假如某一时刻考虑 a_l 和 b_r ,初始时 $l=r=n$

以考虑田忌的最优解为例,若考虑齐威王,将下列各个大于,小于号与大于等于,小于等于号互换即可

若 $a_l \le b_r$,由于 a 数组有序,必有 $a_1 \le a_2 \le ... \le a_l \le b_r \le b_{r+1} \le .. \le b_n$

不难发现对于 $i \in [1,l)$,考虑 b_r 及 b_r 以后的马无意义

此时 $r--$,对于田忌而言没有贡献产生

否则, 此时的 b_r 必为最大的 $< a_l$ 的元素(用反证法不难证明)

$l--, r--$,完成一次赛马,田忌的马增加了 1 的贡献

若 $l<1 \lor r<1$,则遍历结束,统计贡献即可

由于平均下来每个数组元素只访问一次,均摊时间复杂度 $g(n)=1$,总时间复杂度 $O(n)$,可以通过