A

A:number of good intervals

被卡的思路：线段树/st 表加二分（复杂度 n * logn3，而且使用大量空间,对于 400 万读入可能不太好过)

正解：注意到对任何一个数它的不同的 gcd 的数量都是 logn 级别的那么我们就可以用 map 或者数组进行一边读入一边处理数据（读入新数的时候，每次读入的复杂度是 logn2 级，共 n 次

新解：用一个二维数组顺序记录对于某个整数 x 有哪些 a[i]具有因子 x，然后对每个 x 遍历一次，对每个 x 用 cnt 记录下连续数字有因子是 i 的个数，再用 ans[i]+ = 0.5 * cnt * (cnt − 1) 得到区间 gcd 可以被 i 整除的数，最后再将 ans[i]剪掉已经处理好的 ans[k * i]即可（复杂度为 n*因数个数）

B

广义 sam 正确姿势：
https://www.cnblogs.com/Xing-Ling/p/12038349.html

由于要用到多个字符串的所有子串，所以我们很容易想到广义 SAM.对于一个长度 m,那么它的贡献为长度 1~m 所有子串的贡献和,考虑它的分母就是 26,26^2,26^3...26^m 的和，我们可以预处理出来。

考虑它的分子就是后缀自动机上面所有出现的长度小于等于 m 的子串的贡献和。在后缀自动机中的 parent 树中,如果 p 所代表的子串出现的话，那么 fa[p]所代表的的子串一定出现，len[fa[p]]+1~len[p]的长度都会出现,且贡献相等，贡献为 len[p]的贡献.

在造好自动机后，我们再让每个 dictionary 的字符串在 sam 上爬一遍，一边爬的同时去累乘 parent 树上的贡献（即遍历到一个节点 x，先对其该节点累乘贡献，然后再迭代统计 parent[x] 的贡献直到 root，注意要打个时间标记，标记上一次累乘某个节点的贡献用的是哪个字符串，避免重复乘。）

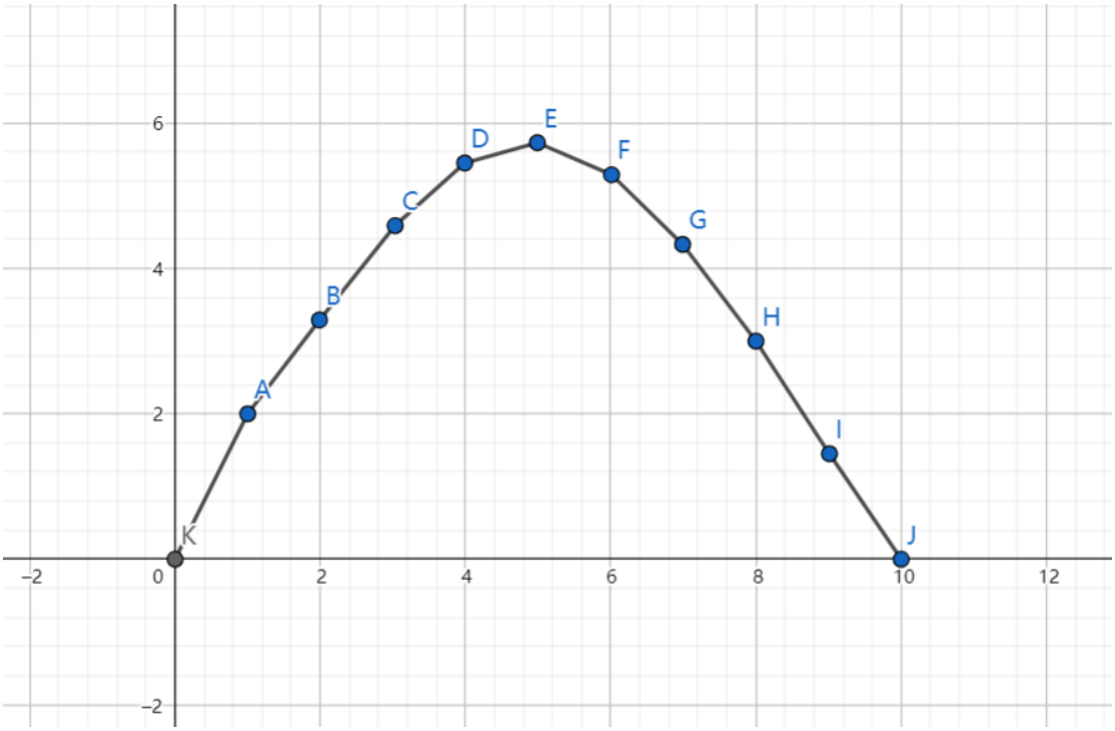
我们用一个前缀和数组 sum，记录对应长度的贡献。对应的，区间 [len[fa[x]]+1,len[x]] 上的贡献都是 y，表现在 sum 上面就是两个端点一加一减。统计完毕后，对 sum 求一遍前缀和，之后 sum[i]表示所有长度为 i 的串的贡献。然后再次对 sum 求一次前缀和，这样的话 sum[i] 就表示所有长度为 1~i 的串的贡献。（我本来是打算放用线段树做区间加的过去的，但是好像有几个人被卡了）。

另外注意一点做前缀和时候的循环上界不能只设到 sam 的节点个数，要开到 m 的上界才行，很多人一开始 wa 在这里。

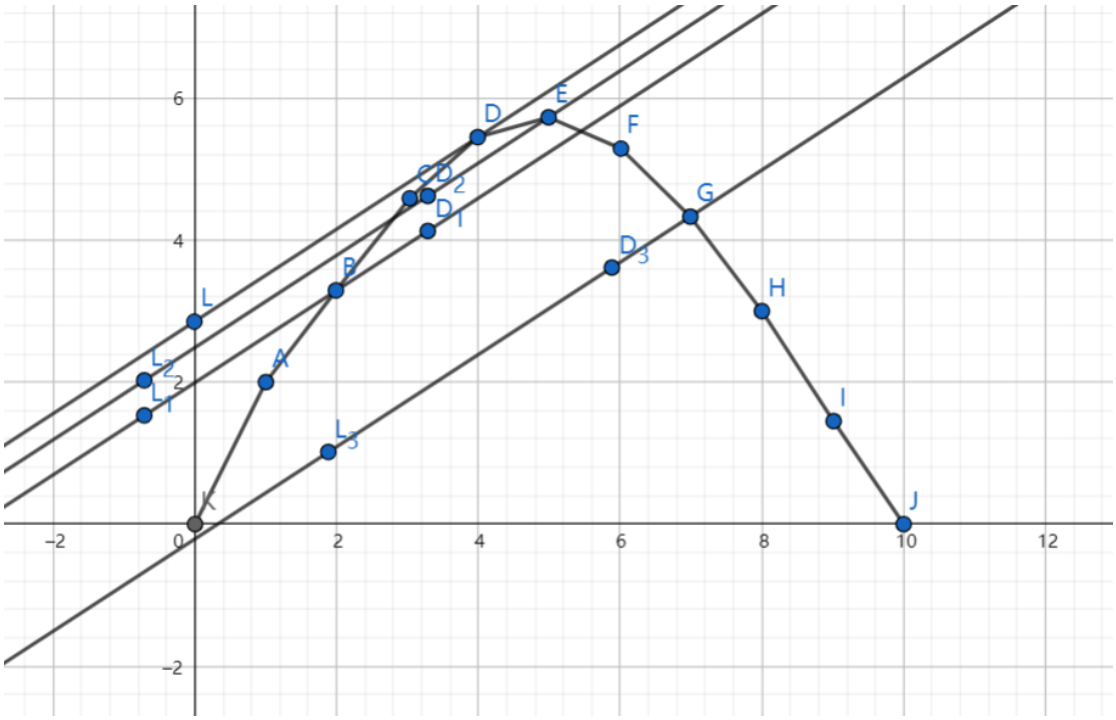
C

1.一开始有好多选手输出的错误答案都是一样的，一度以为自己题面描述出了什么偏差

2.简单 wqs 二分+DP
考虑对于 K 从 1 到 n，答案在一个凸壳上



由这一点，可以发现所求的点应该是横坐标为 K 的点，当然，也只是知道了形状而已，并不知道上凸的点坐标分别是什么，于是考虑二分切线斜率，让答案点是切点，有这样的一个性质：切线斜率单调不升，当切线斜率减小时，切得的点横坐标增大，反之减小，所以每次二分判断切点横坐标即可。



3.如何输出方案
注意可能有多个点在一条直线上，那么如果直接二分可能不能刚好切到 K 这个点，那么在输出方案上会造成困难!

1.输入时对权值进行微小扰动，消除多个点在一条直线的情况(可能不太稳，对微小扰动的要求很高，很可能 WA)。

2.
考虑二分的时候在某一斜率下，切到的点的 K 坐标是一个区间
Dp 时可以记录区间的左右端点
这样回溯方案的时候可以找到要求的 K 值所需要的区间的状态

4.我构造的数据
构造了一组全是 0 的数据好像卡了几位小哥哥

D

题意：$n \times n$棋盘上的$(n+1)^2$个点，从$(0,0)$出发的直线，恰好通过m个点的方案数。定义两个方案是不同的，当且仅当存在一个点属于方案 A 不属于方案 B。

即：求有多少符合条件的数对(i,j)，满足(i,j)互质且$floor(\frac{n}{max(i,j)})+1=m$。。

- $O(Tn)$

  枚举 $t = max(i,j)$，当$floor(\frac{n}{t}) + 1 = m$时，对答案的贡献为 $2\varphi(t)$ 。

- $O(n+T)$

  对于每一个询问，符合条件的 t 落在一个区间 $[l,r]$ 内。

  对$2\varphi(t)$求前缀和 sum，答案为 $sum[r] - sum[l-1]$ 。

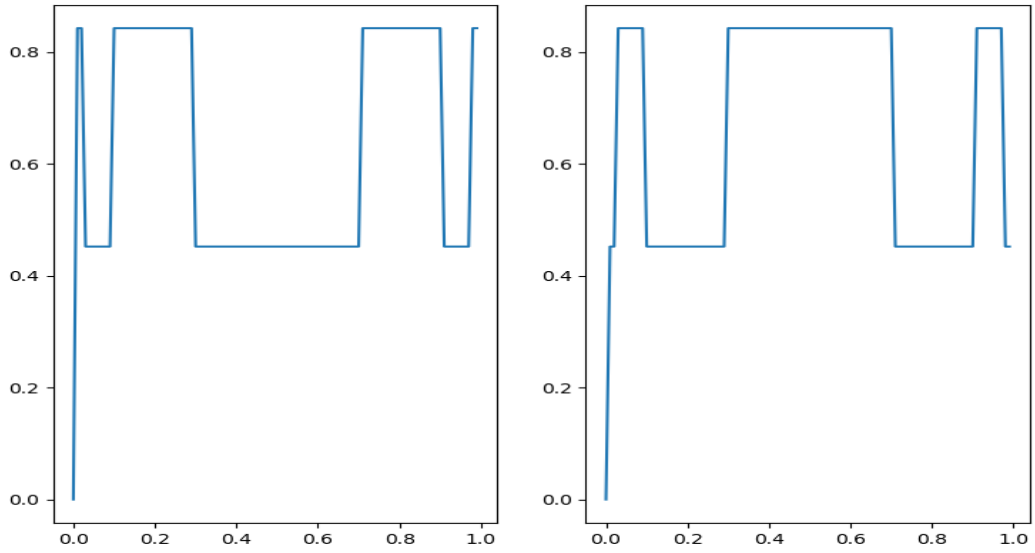
$$l = floor(\frac{n}{m}) + 1, r = floor(\frac{n}{m-1})$$

细节：$m = 1$时，$ans = 1$；$m = n+1$时$ans = 3$。

E

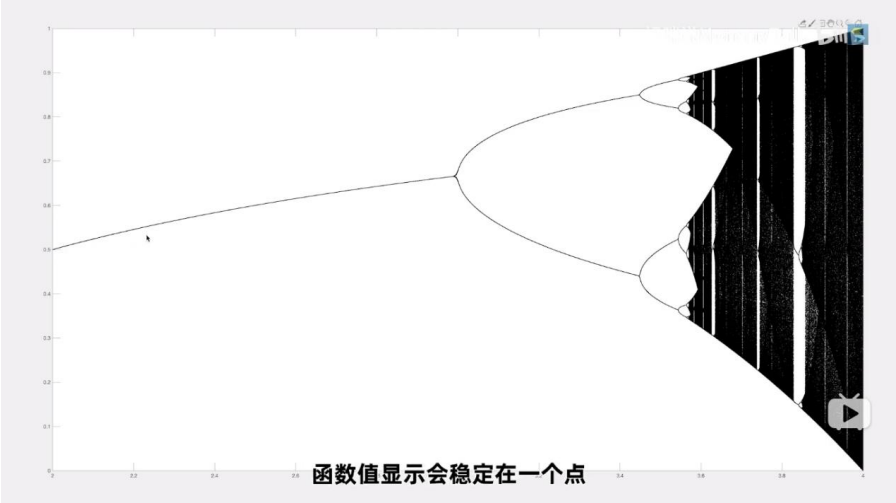
定位：签到题（但是开局榜歪了导致实际做出人数<<应做出人数）

O(Tnk)做法：打表找规律，发现无论初值如何，经过多次迭代后都会收敛到一些吸引子上（下图横轴为 x_0，分别为 mu=3.4 时迭代 10000 次与 10001 次的结果）。



容易发现每个吸引子对应的区间长度较长，所以可以选择小步长采样或随机采样模拟。

O(Tk)做法：发现所给的 x_n 是吸引子，找出其变化的周期（下图横轴为 mu，纵轴为吸引子的值）。



错误做法：以为周期为 2，按 n 的奇偶性处理 x_n。
错误做法：直接解 n 个二次方程。每个方程对应两个取值，如果随便选的话可能会收敛到 0 或者 nan。

细节：需要考虑 corner case 即 x_n=0 的情形。

其他：不要盲目调参，要思考思路是不是正确的。
其他：这种数据较简单的题目可以本地生成一些数据跑跑看（好多同学 wa 1 了

F

用德扑和车万给大家带来温暖
writer:l1ll5

注意到可能的两张公共牌只有 990 种可能
7 选 5 的 C(7,2)也很小，直接搜索所有可能的对局进行判定即可。

模拟。

实现相关：

不可取的行为 —— 手写 C(7,2)

```
return std::max({
    hand({A, B, C, D, E}),
    hand({A, B, C, D, F}),
    hand({A, B, C, D, G}),
    hand({A, B, C, E, F}),
    hand({A, B, C, E, G}),
    hand({A, B, C, F, G}),
    hand({A, B, D, E, F}),
    hand({A, B, D, E, G}),
    hand({A, B, D, F, G}),
    hand({A, B, E, F, G}),
    hand({A, C, D, E, F}),
    hand({A, C, D, E, G}),
    hand({A, C, D, F, G}),
    hand({A, C, E, F, G}),
    hand({A, D, E, F, G}),
    hand({B, C, D, E, F}),
    hand({B, C, D, E, G}),
    hand({B, C, D, F, G}),
    hand({B, C, E, F, G}),
    hand({B, D, E, F, G}),
    hand({C, D, E, F, G}),
});
```

可取的行为

编码牌型

这是两份代码

```
LL cm[maxn];
int SCORE(int i, int a = 0, int b = 0, int c = 0, int d = 0, int e = 0){
    return (i << 20) | (a << 16) | (b << 12) | (c << 8) | (d << 4) | e;
}
int sm[maxn], m;
void fuck(){
    FOR(E, 8, 60) FOR(D, 8, E) FOR(C, 8, D) FOR(B, 8, C) FOR(A, 8, B){
        int PA = A >> 2, PB = B >> 2, PC = C >> 2, PD = D >> 2, PE = E >> 2;
        LL s = SCORE(0, PE, PD, PC, PB, PA);

        if(PA == PB) s = SCORE(1, PA, PE, PD, PC);
        if(PB == PC) s = SCORE(1, PB, PE, PD, PA);
        if(PC == PD) s = SCORE(1, PC, PE, PB, PA);
        if(PD == PE) s = SCORE(1, PD, PC, PB, PA);

        if(PA == PB and PC == PD) s = SCORE(2, PC, PA, PE);
        if(PA == PB and PD == PE) s = SCORE(2, PD, PA, PC);
        if(PB == PC and PD == PE) s = SCORE(2, PD, PB, PA);

        if(PA == PC) s = SCORE(3, PA, PE, PD);
        if(PB == PD) s = SCORE(3, PB, PE, PA);
        if(PC == PE) s = SCORE(3, PC, PB, PA);

        int STRAIGHT = 0;
        if(PA + 1 == PB and PB + 1 == PC and PC + 1 == PD){
            if(PD + 1 == PE) STRAIGHT = PE;
            if(PE == 14 and PD == 5) STRAIGHT = PD;
        }
        if(STRAIGHT) s = SCORE(4, STRAIGHT);

        int FLUSH = (A & 3) == (B & 3) and (B & 3) == (C & 3) and (C & 3) == (D & 3) and (D & 3) == (E & 3);
        if(FLUSH) s = SCORE(5, PE, PD, PC, PB, PA);

        if(PA == PC and PD == PE) s = SCORE(6, PA, PD);
        if(PC == PE and PA == PB) s = SCORE(6, PC, PA);

        if(PA == PD) s = SCORE(7, PA, PE);
        if(PB == PE) s = SCORE(7, PB, PA);

        if(STRAIGHT and FLUSH) s = SCORE(8, STRAIGHT);

        cm[m] = BIT(A) | BIT(B) | BIT(C) | BIT(D) | BIT(E);
        sm[m] = s;
        m += 1;
    }
}
```

```
#define CB1(d)                  vs[d]
#define CB2(d, a)               vs[d] | vs[a] << 4
#define CB3(d, a, k)            vs[d] | vs[a] << 4 | vs[k] << 8
#define CB4(d, a, k, l)         vs[d] | vs[a] << 4 | vs[k] << 8 | vs[l] << 12
#define CB5(d, a, k, l, q)      vs[d] | vs[a] << 4 | vs[k] << 8 | vs[l] << 12 | vs[q] << 16
#define CB6(d, a, k, l, q, w)   vs[d] | vs[a] << 4 | vs[k] << 8 | vs[l] << 12 | vs[q] << 16 | vs[w] << 20
```