T1

```
问你在一个m*n的点阵中，有多少条直线，恰好经过k个点，且斜率k > 0

（1）O(N^5)
    O(n^2)枚举斜率k = b / a，其中gcd(a,b)==1
    O(n^2)枚举直线的左下角(i,j)
    O(n)检查是否恰好经过k个点

（2）O(N^4)
    以左下角的点为原点(0,0)建系，边界为n-1和m-1。
    和n^5中的一样枚举a,b,x,y。（x为原来的i，y为原来的j）

    若直线恰好经过k个点，则应满足以下条件：
    1. 点(x-a, y-b)不合法。（保证(x,y)为直线的最左下角）
        x-a < 0 or y-b < 0
        即：x <= a-1 or y <= b-1

    2. 点(x+(k-1)*a, y+(k-1)*b)在合法范围内。（保证至少经过k个点）
        x+(k-1)*a <= n-1 and y+(k-1)*b <= m-1
        即：x <= n-(k-1)*a-1 and y <= m-(k-1)*b-1

    3. 点(x+k*a, y+k*b)不合法。（保证最多经过k个点）
       x+k*a >= n or y+k*b >= m
       即：x >= n-k*a or y >= m-k*b

    这就将O(n)检查经过点的个数变成了O(1)
```

（3）O(N^2logN) **std 做法**

在 n^4 的做法中，得出了在确定斜率 k = b / a 的情况下，合法的左下角(x,y)应该满足的条件：

1. x <= a-1 or y <= b-1
2. x <= n-(k-1)*a-1 and y <= m-(k-1)*b-1
3. x >= n-k*a or y >= m-k*b

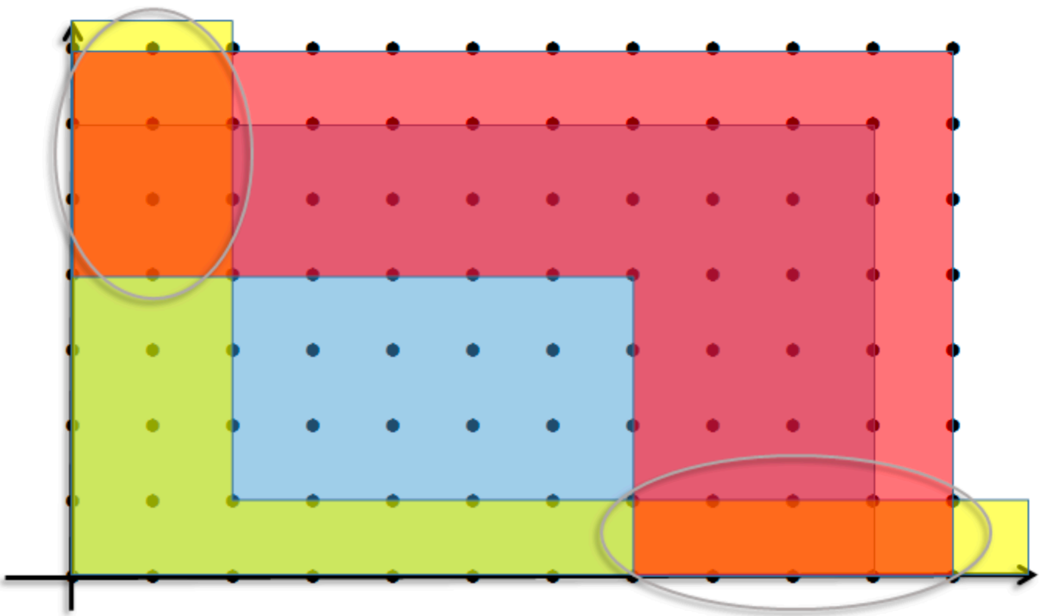
令：x1 = a-1, y1 = b-1
x2 = n-(k-1)*a-1, y2 = m-(k-1)*b-1
x3 = n-k*a, y3 = m-k*b

即：x <= x1 or y <= y1
X <= x2 and y <= y2
X >= x3 or y >= y3

在坐标系中，这三个条件限定出了一些合法的部分，我们要求的就是在这部分中的格点个数（包括边界）。

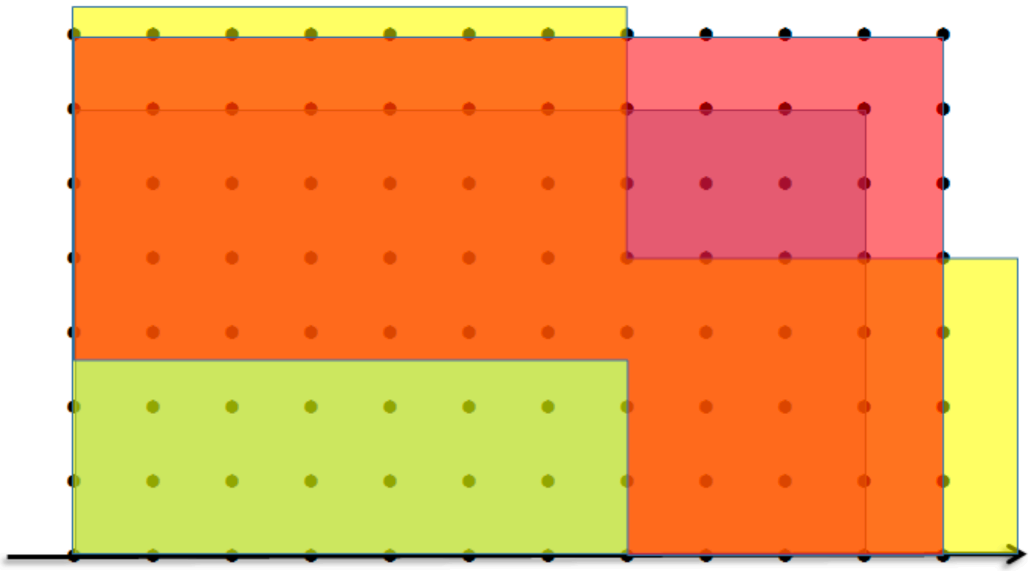
如图，要求的为画圈的部分。（黄、蓝、红分别代表限定条件 1、2、3）

（情况一）



情况（2）

整个橙色的部分为所求。



推个公式分情况求个点数就好啦。这样就不用枚举左下角(x,y)了。

枚举斜率O(n^2) * 判断gcd(a,b)==1 O(logn)
总复杂度O(n^2*logn)

T2

最短路+TSP问题

预处理两两之间的最短路，然后对特殊点进行一次状压dp

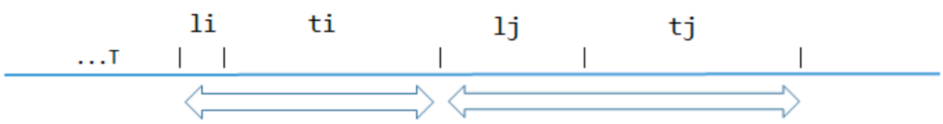
T3

给你一棵树。根节点为 1。
n 个点，每个点的点权为 w[i]。每条边花费时间 len。
对于每一条边,父节点到子节点方向需要花费时间 len,从子节点到父节点不花费时间。
对于每个点，花费 = 到达它的时刻 * 点权。
问你遍历所有点的最小总花费。

（1）O((son!)^N) son 为节点的儿子数（估值）
dfs 整棵树，对于每一个节点，枚举去他的儿子们的顺序。
复杂度 = O(son!) * ΠO(son 的 son) = ... = O((son!)^n)
（好像是这么推的吧……）

（2）O(NlogN) **std 做法**
对于每个节点，无非就是确定一个最优的，遍历儿子们的子树的顺序。
第一遍 dfs 处理这样几个值：
t[i]：表示从节点 i 开始，遍历完 i 的子树所需的时间
sum[i]：表示节点 i 的子树的点权（w[i]）之和

假设已经确定了一个遍历子树们的顺序（如图）。



那么对于顺序相邻的两个子树 i 和 j 来说，交换它们两个的顺序，并不会影响到其他子树（因为遍历 i 和 j 的时间总和是一定的）。

假设 i 在 j 之前，结果更优，则有：花费（i 在 j 之前） < 花费（j 在 i 之前）

设 ci 表示在 0 时刻，从 i 出发遍历它的子树的花费。

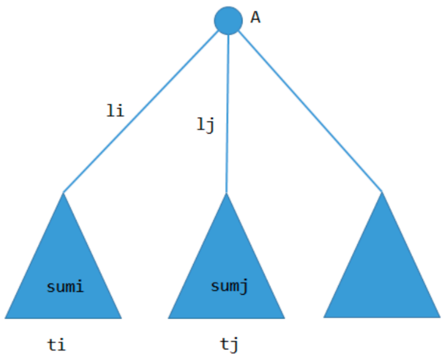
实际花费（i 在 j 之前） = ci + li*sumi + cj + (li+ti+lj)*sumj

实际花费（j 在 i 之前） = cj + lj*sumj + ci + (lj+tj+li)*sumi

代入：花费（i 在 j 之前） < 花费（j 在 i 之前）

整理得：

(ti+li)*sumj < (tj+lj)*sumi



所以对于每个节点，按照上式给儿子们排个序就好啦。（贪心）

总复杂度 = ΣO(cnt_son*log(cnt_son)) = O(NlogN)