Problem A. Simple

数字较小时，可以枚举每个 c，直接用扩展欧几里得算出一个初始解，然后列个式子推一推。如果数字比较大，我们可以根据贝祖定理知道，如果 c 不是(m,n)的倍数，那么一定不是好的。否则我们考虑当 m 和 n 互质的时候，这时只要枚举 y∈[0,n-1]，那么 y*m-x*n 在[1,q]之间的数一定都不是好的。

Problem B. Walk

数据比较小的时候，直接枚举 O(n^2)条路径，dfs 一遍就好了。当 w 比较小的时候，我们可以枚举答案，然后走树上所有权值是 w 倍数的边，一棵树上的最长链相信大家都会，这里不再赘述，复杂度是 O(w*n)。你发现枚举答案，然后统计最长链的时候，有很多边其实不需要访问，一条边只会在枚举他因子作为答案的时候才有用。所以我们只要把每条边拆成因子个数条边就好了，理论时间复杂度 O(n^1.5)，但是因子的个数远远达不到这个数量级，所以只要常数不是太大，就可以通过本题。

Problem C. Travel

数据很小的时候，分别可以爆搜每一个各自走的顺序或者每一个格子向左向右走的状态解决。然后 xi=i 比较好处理，讨论即可有一种贪心的方法是每次向一个方向走到底再向反方向走然后讨论，这样的思想和标算其实很相似，但是有一些问题。比较小的数据点，

数据没有刻意卡这种贪心。

-1 的情况非常好处理显然当 l=0 但是 s≠1 的时候，是没有合法路径的。同理 l=n-1，s≠n 也是没有合法路径的。非常显然，这里不给出证明。

先考虑特殊的情况，当 s=1 的时候，显然答案的下界是 xn-x1，就是从最左边按次序一直跳到最右边，不过 L>0 的时候就不能这么做了。答案要求最小也就是说要尽量少走回头路。假如我们在 n 这个位置停下，那么中间就需要走一些回头的路类来用掉 L，我们把所有 i 到 i+1 之间的区间看成一个线段。跳的路径相当于对线段进行覆盖。显然所有的线段都必须覆盖至少 1 次，而至少有 L 个线段至少覆盖 3 次。下面给出证明。

假如某一次是从 i 向左跳，那么之前一定会从左侧跳到 i，肯定会覆盖[i-1,i]的线段，然后从 i 向左跳走回头路，又会覆盖[i-1,i]一次，最后因为一定要走到 n，所以在走完回头路后一定还会再向右跳回来，所以至少经过三次[i-1,i]。也就至少有 L 条线段被覆盖至少 3 次。这样我们就可以贪心的选取权值最小的 L 条线段计算作为答案了。显然这种情况下 L 越小，答案越优。

不过起始和结束的位置不一定在两端，那假设结束的位置是 e，根据对称性，不妨假设 s 在 e 的前面。显然 s 前面的线段和 e 后面的线段一定都至少经过两次，而且存在方案能让 s 前和 e 后一共用掉 n-e+s-1 个 L，并且每条线段只经过两次。这样不仅能减少[s,e]中间走 L 的次数，而且也让 s 和 e 两侧的花费尽量小。中间的部分

事实上和刚才讨论的从 1 开始，n 为终点的情况是完全一样的。有一些边界情况需要注意，这里不作说明。

这样我们就可以直接枚举 e 来计算答案了，继而发现 e=i 和 e=i+1 的情况只是有[i,i+1]这条线段覆盖次数限制的变化，其他线段都没有影响。所以在枚举 e 的时候，我们用一个堆或者是队列来维护权值最小的若干条线段就好了，时间复杂度 O(n*logn)。