Solution

Shopping

这个题解法很多。有一个动态规划的做法。这里说贪心+调整 的做法。

首先按照 Cantor 的喜好降序排序，并满足 Cantor 喜好相同的物品 Lagrange 的喜好小的排在前面。

以这个顺序先强制给他们两个分配物品，但这样这样很可能不是最优的。

然后倒序扫描：

对于当前属于 Cantor 的物品，在这个状态下他所选择的物品一定是这个物品，可以不考虑。

对于当前属于 Lagrange 的物品，在这个状态下他不一定会取这个物品，而是会取这个物品的后面所有 Cantor 的物品中对于他价值最大的那一个。因为通过这种策略，他的总价值一定增大，并且使得下一次 Cantor 取物品是一定会取当前属于 Lagrange 的这个物品，并没有给 Lagrange 造成损失（他已经舍弃这个物品了）。

这样做复杂度$O(nlog_n + n^2)$，后面扫描的过程可以用数据结构做到$O(nlog_n)$。

Game

这个题目真是个很有趣的题目。

先来讨论放球的过程。由于最多有 n 个球放进去，并且取球又是一次一个，所以放进来的球的总个数是$O(q)$级别的。并且当树的形态一定时，放球的顺序是一定的。所以可以通过预处理放球的先后顺序。以这个先后顺序为优先级用堆维护一下当前将放入哪个结点即可，这样放一个球的复杂度是$O(log_n)$的。

然后再来讨论取球的情况。以根为最上端考虑。那么每从树中取出一个球，他的上方的所有球都会向下掉一格。这其实是相当于将这个结点最上方的那个点拿掉而已。所以可以开一个倍增数组 num[son][k]，表示结点 son 向上走2^k所到的祖先，一个布尔数组 d[son]表示结点 son 是否有球。就可以从当前要删的结点开始向上跳，在$O(log_n)$的时间复杂度上跳到该节点最上方有球的结点。把这个结点的球删掉，把这个结点加入堆，输出答案即可。

Function

这个题目就是我所说的唯一一道要用平衡树的题目。

题目描述有点诡异，其实就是：给定 n 个数对(x_i, y_i)，求最长的序列满足

$(x_i < x_j, y_i < y_j, i < j)$。

其实也就是 LIS 的二维版本。。

在一维的LIS中有一个$O(nlog_n)$的二分做法，维护D[i]表示长度为 i 的LIS，尾结点最小是多少，显然 D 数组单调。

将这种思路扩展到二维，由于点和点的大小关系要满足两个条件，所以这个二分数组里每个位置不能只维护一个点，而要维护一棵平衡树。每次插入点(x, y)的时候如果当前树中存在点(x', y')，使得$(x' < x \&\& y' < y)$，那么这个点可以插到这棵树后面的树中去。这是满足二分性的。所以可以二分。

数据是随机数据，所以这样很可能就可以过了。还有一种树套树的方法 可能这个数据也可以过，不过跑原数据还要在以上二分的基础上加优化才能过。

我们发现，每颗平衡树上会有大量无用结点。如果没有无用节点，那么将平衡树按 x 升序排序，y 应该满足降序。按照这个原则每加入一个元素时删掉加入所产生的无用结点即可。