宝物

记第 i 种宝物的数量为 cnt_i，一个非常显然的事实是：若 $cnt_i \neq 0$，那么这种宝物选一个的方案一定比不选更优。

那么我们来看一下总的可能方案数：
$max(cnt_1,1) \times max(cnt_2,1) \times ... \times max(cnt_k,1)$。其中 $\Sigma cnt_i \leq 50$。最坏情况下每一种的宝物数量相同，令他们等于 k，那么此时总的方案数不会超过 $k^{\frac{n}{k}}$，$k=3$ 时取到最大值，上限不是一个很大的数字。并且在实际实现过程中较上界还有一定差距，故搜索即可解决。

那为什么之前的搜索只能拿到 50 分呢?

值得注意的是：对于 $cnt_i = 0$ 的类，应该在搜索过程中直接跳过，否则可能导致搜索到不必要的状态从而引起超时（$50-60$ 分）。

赛车

首先把所有赛车按照加速度排序，这样就不会存在后面的车不可能超过前面车的情况。

我们维护一个栈，其中存储所有可能的领跑者。依次讨论每一辆车，如果栈顶的车加速度和初始位置都不大于当前车，那么可以直接扔掉栈顶车。

如果当前车在某个时间超过栈顶车，并且这个时间不大于栈顶车超过队中倒数第二辆车的时间，说明栈顶车也没有机会成为领跑者，故也丢掉。直到栈顶车无法排除或栈为空，将当前车加入栈顶。

最终栈中的车辆就是所有的领跑者。

法棍

问题分析 1：

观察可以发现，题目中所定义的区别，本质上就是 $l1+l2-2\times LCS$。其中 $l1$ 是其中一个字符串的长度，$l2$ 是另一个的长度，LCS 是两字符串的最长公共子序列。

所以问题就转化成了求 LCS。

那么对于询问次数极少的小数据，我们可以每次暴力用 $\mathcal{O}(nm)$ 的求 LCS 的 dp 算法来解决。

问题分析 2：

我们知道最朴素的求 LCS 的方法是 $\mathcal{O}(n^2)$，对于这道题目而言有 100000 次询问，显然不可行，所以我们需要另辟蹊径。

不妨尝试定义新的状态

$Dp[i][j][k]$：表示从 A 串的第 i 个字母开始，从 B 串的第 j 个字母开始能够得到长度为 k 的公共子序列的最小 A 下标（A_i 和 B_j 不一定相同）。

那么状态转移方程也不难得出：

若 $Ai \neq Bj$，$Dp[i][j][k] = min(Dp[i+1][j][k], Dp[i][j+1][k])$

若 $A_i = B_j$，

$Dp[i][j][k] = min(Dp[i+1][j+1][k-1], Dp[i+1][j][k], Dp[i][j+1][k])$

注意需要**特判的边界情况**：$Ai = Bj$ 且 $k = 1$ 时，$Dp[i][j][k] = i$，动规数组初始值全部为 inf（inf 是一个非常大的整数）即可，在动规过程中对边界情况的特判即可完成最简单状态的初始值设定。

最后*对于每一个询问*，从大到小枚举 k，看$Dp[l_i][1][k]$是否不超过 r_i 即可。

时间复杂度：$\mathcal{O}(n \times m^2 + q \times m)$

火锅

30 分做法：暴力枚举（搜索即可）。

60 分做法：

首先可以发现最优解可以增量构造，即往吃 k 份肉的最优解加入一份肉可以得到吃 $k+1$ 份肉的最优解，因此存在一个顺序 $ord_1, ord_2, \cdots ord_n$ 满足 $ord_1, ord_2, \cdots, ord_k$ 是吃 k 份肉的一个最优解吃的肉对应的集合。

假设确定了要吃哪些肉，那么肯定是按照捞出锅的时间从小到大吃。按照出锅时间从小到大依次考虑每份肉，在 $ord_1, ord_2, \cdots ord_{n-1}$ 中找到一个位置插入第 i 份肉。令第 i 份肉的

出锅时间为 a，吃掉它的时间为 b，$ord_1, ord_2, \cdots ord_{n-1}$ 里按照出锅时间顺序吃掉前 k 份肉 $ord_1, ord_2, \cdots ord_k$ 所需的时间为 t_k，则将 ord_k 替换成第 i 份肉后，对应的方案所需的时间为 $max(t_{k-1}, a) + b$，如果 $max(t_{k-1}, a) + b < t_k$，则第 i 份肉应该插在 ord_k 之前。再修改对应 t_k 即可。

100 分做法：

注意到满足 $max(t_{k-1}, a) + b < t_k$ 的 k 是 ord 序列的一个后缀，所以可以二分找到对应的位置，将第 i 份肉插入 ord 序列，并将后面部分的 t 都修正为 $max(t_{k-1}, a) + b$。

为了加速这个过程，可以用平衡树维护 ord 序列，每个位置记录 t_k 以及 t_{k-1} 方便二分，在修正 t 为 $max(t_{k-1}, a) + b$ 时，根据 t 的单调性将 t 的一个区间赋值为 a，再将一个后缀加上 b。最后得到的 t 序列就是答案。

时间复杂度 $\mathcal{O}(nlogn)$。