1 修建竹林 (bamboo)

前 50pts 的暴力部分就不再详述了。

注意到对于某个特定的 h, $\lceil \frac{h}{x} \rceil$ 只有 $O(\sqrt{h})$ 种不同的取值，即一棵竹子经过多少次被裁剪只有根号种取值。同时，整片竹林也只有 $O(N\sqrt{h})$ 种可能的裁剪方式。

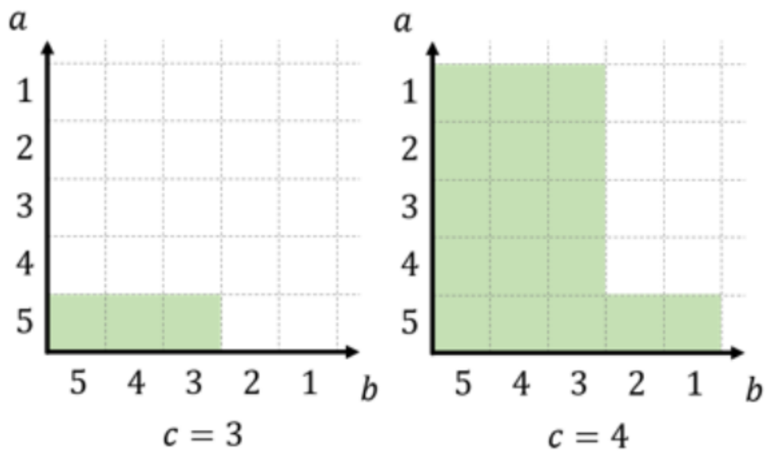
我们可以直接枚举这 $O(N\sqrt{h})$ 种裁剪方式，再计算每种裁剪方式对应的最大的 k，总复杂度为 $O(N^2\sqrt{h})$，视实现优劣可以获得 70pts-100pts.

不难发现，如果我们按 k 从小到大枚举每种裁剪方式，每次更改只会改变一根竹子的裁剪方式，可以 $O(1)$ 维护答案。时间复杂度 $O(N\sqrt{h})$。

2 选择卡牌 (card)

首先考虑 c_i 全部相同时怎么做。

举个例子，比如只有 1 张卡牌：$a = 4, b = 2, c = 3$。当小 D 选出的卡牌满足 $c = 3$ 或 4 时，对应的 a, b 可行的范围为：



其中绿色部分为可行区域，白色部分为不可行区域。当有多张卡牌时，白色区域会取并集。所以，对应一个固定的 c，我们可以通过线段树维护白色区域的和，所需的操作是区间求和和区间取最大值。由于白色区域随着 b 的增加而单调减，所以区间取最大值可以变成区间赋值从而达到 $O(N \log B)$ 的复杂度。

当 c_i 不同时，我们首先将 c_i 从大到小排序，然后依次算出每一个时刻的白色区域。当我们的 c 从大变小越过某个 c_i 时，相当于执行两个区间取最大值操作，也可以通过线段树维护。

总时间复杂度 $O(N \log B)$。

3 方形的零 (zero)

首先，通过前缀和和任意数据结构，我们可以在 $O(N \log N)$ 的时间内求出从每一列开始最小的和为 0 的矩形的位置。

我们考虑一个朴素的动态规划：记 $dp_{i,j}$ 为第一行用在 i 之前，第二行用在 j 之前的最大的矩形数量。转移是 $O(1)$ 的：枚举每种矩形，然后转移到最近的位置。朴素动态规划的时间复杂度为 $O(N^2)$，可以获得 30pts。

接下来的两个部分分思路都差不多：都保证了最优解中每个矩形的大小比较小。以 $|a_i| \leq 1$ 为例子，存在一组最优解中每个矩形的大小都不大于 2。这启发我们在动态规划时只要计算 $|i-j| < 4$ 的位置就可以考虑全所有的转移，因为不会在某一行中出现一个过大的矩形。另一个部分分只要考虑 $|i-j| < 20$ 的位置就行。所以这两个部分的动态规划都可以做到 $O(N)$。

顺着这个思路我们可以发现，实际上在朴素的动态规划中有很多的状态是没有求的意义的。若 $dp_{i,i} = k$，j_1, j_2 是最小的 j 使得 $dp_{i,j_1} = k+1, dp_{i,j_2} = k+2$。那么 j_2 是没有意义的，因为如果 i 到 j_1 这段的第一行没有其他矩形，那么 $dp_{i,j_1} = dp_{j_1,j_1}$，$dp_{i,j_2} = dp_{j_1,j_2}$。而 dp_{i,j_2} 和 dp_{j_1,j_2} 在后继的转移中的作用是完全一样的。

否则如果 i 到 j_1 这段有其他的矩形，那么会存在一个 i_1 满足 $i_1 \leq j_2$ 且 $dp_{i_1,j_1} = k+2$，从这个状态可以转移到 $dp_{i_1,j_2} = k+3$，通过这条转移链也可以跳过 dp_{i,j_2}。

所以我们需要的保留的状态仅仅是 $dp_{i,i}, dp_{i_1,j}, dp_{i,j_1}$。其中 i_1, j_1 是最小的 i, j 满足 $dp_{i_1,j} = dp_{i,j_1} = dp_{i,j} + 1$。

具体实现方式可以参考标程,动态规划复杂度为 $O(N)$,总时间复杂度为 $O(N \log N)$。