2 paper

这是一道小水题。

由于出题人在写 std 时没有想到算法 5，所以写在 std 里的是一个容斥做法。

2.1 算法 1

暴力。枚举每条可能的直线选还是不选，然后对每一段求出可以填的数字个数，乘起来加进答案。

时间复杂度 $\mathcal{O}(n2^n)$。

2.2 算法 2

考虑 $m = 0$ 的情况。

动态规划。设 f_i 表示 $i = n$ 或存在一条与矩形左边界距离为 i 的直线时，只考虑与左边界距离为 i 的直线左边的部分，划分为若干段并在每段上写一个数的方案数。

我们定义 $f_0 = 1$。对于 $i > 1$，如果存在 k 使得 $p_k + 1 = i$，那么 $f_i = 0$，否则，设 $d(n)$ 为正整数 n 不超过 c 的约数个数。有 $f_i = \sum_{k \le i} d(k) f_{i-k}$。

时间复杂度 $\mathcal{O}(n^2)$。

2.3 算法 3

算法 2 中的转移方程可以改写为：

$$f_i = \sum_{k=1}^{c} \sum_{q \ge 1, qk \le i} f_{i-qk}$$

如果 $c = 1$，这是一个简单的可以前缀和优化的 dp。不难发现 $c > 1$ 时也可以类似地前缀和优化。

设 $g_{i,j} = \sum_{q \ge 0, qj \le i} f_{i-qj}$ $(1 \le j \le c)$。有 $g_{i,j} = f_i + g_{i-j,j}$，$f_i = \sum_{k=1}^{c} g_{i-k,k}$。一边递推 f 的值一边更新 g 即可。

在不能放直线的位置上，把 f_i 设为 0 即可。

时间复杂度 $\mathcal{O}(nc)$。

2.4 算法 4

考虑 $m = 0$ 的情况。

算法三中的 dp 算法可以用矩阵乘法优化。

对于一个 i，只需要知道，对于所有的 $1 \le x \le y \le c$，$g_{i-x+1,y}$ 的值，就能求出 f_{i+1} 和所有的 $1 \le x \le y \le c$，$g_{i-x,y}$ 的值。所以只需要记这样的 $\frac{c(c+1)}{2}+1$ 个值（为了方便最后统计答案，还要记一个 f_i）就能进行转移。用矩阵优化转移即可。

时间复杂度 $\mathcal{O}(c^6 \log n)$。因为有很多 $\frac{1}{2}$ 的常数实际上运算次数不会太多。

2.5 算法 5

只需要在 p_k+1 处改变一下转移方程即可。这样需要做 m 次矩阵快速幂，复杂度太高。有一个优化，我们最终只需要求转移矩阵的幂乘以一个向量的结果，倍增地预处理出转移矩阵的 2^k 次幂。在求矩阵的幂乘以向量时，先把指数二进制拆分，然后依次用预处理的矩阵去乘以向量。这样就避免了矩阵与矩阵的乘法，降低了复杂度。

时间复杂度 $\mathcal{O}(c^4 m \log n + c^6 \log n)$。可以获得 100 分。