题目大意

称一个序列 $(a_1, a_2, \cdots, a_n)$ 是避免 120 模式的，当且仅当不存在 $1 \le i < j < k \le n$ 使得 $a_k < a_i < a_j$。

给定质数 P。q 次询问，每次给定 n, m，求有多少个长度为 n 的、值域在 $[0, m]$ 内的整数序列 a 是避免 120 模式的，结果对 P 取模。

数据范围

对于全部数据，满足 $10^8 \le P \le 10^9$，$1 \le q \le 8 \times 10^4$，$1 \le n \le 100$，$0 \le m \le 10^9$。

测试点编号	$n \le$	$m \le$
$1, 2$	16	16
$3, 4$	18	18
$5, 6$	18	10^9
$7, 8$	20	20
$9, 10$	20	10^9
$11 \sim 16$	100	100
$17 \sim 20$	100	10^9

解题过程

算法一

考虑从前往后逐位确定序列 a。当考虑到第 $k+1$ 位时，一方面，为了避免存在 $1 \le i < j \le k$ 使得 $(i, j, k+1)$ 形成 120 模式，会要求 $a_{k+1} \ge L$，其中 $L = \max_{1 \le i < j \le k \wedge a_i < a_j} a_i$；另一方面，在确定了 a_{k+1} 后，为了同时考虑到 L 的变化，需要知道 $a_1, \cdots, a_k$ 中小于 a_{k+1} 的最大数是多少，并将 L 对它取 $\max$。

于是可以设计 DP 状态 $f_{k,L,S}$，其中 k, L 的含义同上一段中所述，而 S 记录了 $\{a_1, \cdots, a_k\}$。转移时枚举 a_{k+1} 的值即可。

于是该算法的复杂度为 $O(nm^2 2^m + q)$，可以通过 $10\% \sim 20\%$ 的数据。

算法二

进一步优化算法一。考虑优化转移时的复杂度，先枚举 k, S, a_{k+1}，求出 S 中小于 a_{k+1} 的最大的数 x，那么转移到 $f_{k+1,\max(L,x),S|2^{a_{k+1}}}$。对于 $L \le x$ 的情况，可以对 $f_{k,L,S}$ 在 L 这一维做前缀和，然后统一转移到 $f_{k+1,x,S|2^{a_{k+1}}}$；对于 $L > x$ 的情况，由于 $L \in S$（这里规定初始 S 不是空集而是 $\{0\}$），而 $a_{k+1} \ge L$，所以一定有 $a_{k+1} = L$。这样就能把转移的复杂度降至 $O(1)$。

算法的时间复杂度降至 $O(nm2^m + q)$，可以通过 20% 的数据。

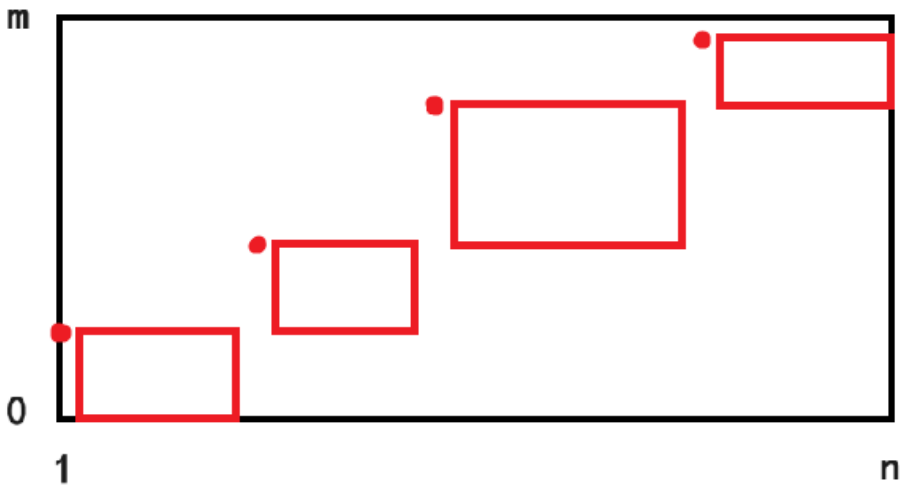
注意到 S 中我们并不需要记录 $< L$ 的数：因为 S 的用处只有每次“找到 S 中小于 a_k 的最大数 x 是多少，并将 L 对它取 $\max$”。同时，舍弃掉 S 中 $< L$ 的数后，可以发现 L 就是 S 的最小值：因为 L 变成某个数 x 的前提条件是 S 中存在这个数。于是 L 也无需记录了。

算法的时间复杂度降至 $O(n2^m + q)$，可以通过 30% 的数据。

算法三

采用从前往后逐位确定 a 的思路，问题本身的复杂性致使 DP 时无法避免地要记录指数级的状态。这启示我们从整体的角度入手。

考察该序列的所有严格前缀最大值，如图：



红点分别代表的是 a 所有的严格前缀最大值 $a_{i_1}, \cdots, a_{i_k}$，记 $a_0 = 0, i_0 = 0, i_{k+1} = n + 1$。

那么可以看出，序列 a 是避免 120 的，当且仅当，对于任意 $1 \le j \le k$，相邻两个红点 $a_{i_j}, a_{i_{j+1}}$ 之间的红框部分，即连续子序列 $(a_{i_j+1}, a_{i_j+2}, \cdots, a_{i_{j+1}-1})$，值域在 $[a_{i_{j-1}}, a_{i_j}]$ 内，且是避免 120 模式的。

于是序列 a 事实上有一个类似递归的结构，据此我们可以设计 $g_{n,m}$ 表示长度为 n 的、值域在 $[0, m]$ 内的、避免 120 模式的整数序列个数。枚举最后一个矩形的长和宽，可以得到转移式：

$$g_{n,m} = \sum_{a=1}^{n-1} \sum_{b=1}^{m} g_{n-a,m-b} g_{a-1,b} + \sum_{b=0}^{m} g_{n-1,b}$$

加号前代表的是若最后一个矩形不是第一个矩形，那么此时矩形的高 $b \ge 1$（因为是严格前缀最大值）；加号后代表的是若最后一个矩形是唯一的矩形，那么此时矩形的高 $b = a_1 \ge 0$ 即可。

初始状态为：$g_{0,m} = 1$。

该算法的时间复杂度为 $O(n^2 m^2 + q)$，可以通过 60% 的数据。

算法四

由于 a 是否是避免 120 模式的只与 a 各个元素之间的相对大小关系有关，所以当 n 固定时，答案 $g_{n,m}$ 应该是关于 m 的 n 次多项式。

具体来说，设 $h_{n,m}$ 表示长度为 n 的、避免 120 模式的本质不同的整数序列 a 的个数，使得 a 中不同元素个数为 m。这里两个序列 a, b 本质不同指的是，存在 $1 \le i, j \le n$ 使得 $[a_i < a_j] \ne [b_i < b_j]$，即 a, b 离散化后的序列不同。那么有：

$$g_{n,m} = \sum_{i=0}^{n} \binom{m+1}{i} h_{n,i}$$

即为那 i 个本质不同的元素选出它们分别是 $[0,m]$ 中的哪个整数。

于是可以看到，$g_{n,m}$ 是关于 m 的 n 次多项式。那么当 m 很大时，只需算出 $g_{n,0},\cdots,g_{n,n}$，再使用拉格朗日插值 $O(n)$ 求出 $g_{n,m}$ 即可。

结合算法二，可以通过 $30\%\sim 50\%$ 的数据。结合算法三，时间复杂度为 $O(n^4+qn)$，可以通过 100% 的数据。

算法五

观察算法三中分析的 a 的递归结构，或者 g 的转移式，容易发现可以使用二元生成函数来解决这个问题。

设 $G(x,y)$ 为 $g_{n,m}$ 的二元生成函数，那么有：

$$G=\left(G-\frac{1}{1-y}\right)\left(G-\frac{1}{1-x}\right)x+\frac{x}{1-y}G+\frac{1}{1-y}$$

解得：

$$G=\frac{2}{1-y+\sqrt{(1-y)(1-y-4(1-x)x)}}$$

在合适的固定模数下，可以使用二元生成函数的运算加速，和算法四结合，算法的时间复杂度为 $O(n^2\operatorname{polylog}(n)+qn)$。

参考资料

[1] 金策，《生成函数的运算与组合计数问题》，2015 年信息学奥林匹克中国国家队候选队员论文集。