A.茵蒂克丝

这是一道原创题。

暴力可以考虑直接区间 dp，时间复杂度 $O(n^2+q)$，期望得分 50 分。

引理：对于正数 a,b,c,d 满足 $\frac{a}{b}\le\frac{c}{d}$，有 $\frac{a}{b}\le\frac{a+c}{b+d}\le\frac{c}{d}$。

证明可以直接代数推导，也可以考虑组合意义为溶液浓度。

所以，最优的合法区间的长度一定 $<2k$，否则可以分裂成两个合法区间，其中必然有一个不劣。

这时有用的区间只有 $O(nk)$ 个，这是可以接受的。

设 $t=2k-1$，对于一组询问 $[l,r]$，考虑所有的 $l'\le r-t+1$，以这些位置为左端点的有用区间一定被询问区间包含。可以对每个左端点预处理其的最优区间，然后使用 st 表询问。这一部分复杂度为 $O(nk+n\log n)$。

对于 $l'>r-t+1$，我们可以优化暴力的区间 dp。注意到所有在此时访问到的区间的长度均 $<t$，我们可以 $O(nk)$ 地 dp。

总时间复杂度 $O(nk+n\log n+q)$，期望得分 100 分。

B.御坂美琴

首先，所有 0 出度点的答案显然是 -1。我们删去这些点和与其相关的边，这会导致新的 0 出度点出现，重复下去。使用类似拓扑排序的方法便可以找出所有这样的点，而剩下的点都在至少一个环上，所以他们一定有解。

考虑一个点 u 的答案 ans_u 可能如何贡献而成：

1. 其的一条出边的 r_i。
2. 经过一条抵达 v 的出边，ans_v-p_i 的值。

对于第二种情况，肯定是 ans 较大的点贡献到 ans 较小的点，这启发我们从大往小做。

倒序遍历值域，考虑遍历到 x，如何 check 每个的点的答案是否可以 $\le x$。

对于一个点 u，考虑其所有出边 (u,v)。

1. 若 $r_i > x$，肯定不能走这条边。
2. 若 v 的答案已经确定，且 $ans_v - p_i > x$ 则不能走这条边。

如果点 u 没有任何可走的出边，那么他的答案就被确定为 $x+1$。如果确定了某个点的答案，再次重复上述过程。

这是一个有正确性的算法，但我们显然不能真的遍历一遍值域。

考虑加速模拟这一过程。类似地，我们每次取出并删去限制最严的边，如果这是出发点的最后一条剩余出边，那么他的答案就可以被确定。当一个点的答案被确定时，所有连向他的边的限制可能会变严，总变化次数显然不超过 m。

使用堆维护，时间复杂度 $O(n+m\log m)$，期望得分 100 分。

C.欧提努斯

记 $H=\max_{i=1}^{n} h_i$

考虑一个给定容器，每个位置的水位有多高。这个值等于其前缀最大值与后缀最大值的较小值。我们不妨只计算水位高度之和，输出时再减去积木高度之和。

如果确定了最大值位置，那么其左边的答案就是所有前缀最大值之和，右边同理。

注意到我们事实上可以将左边的前缀最大值与右边的后缀最大值归并起来，于是可以得到结论：**所有答案一定存在一种构造方式，使得最大值在最后一个位置。**

考虑从小往大插数，此时过程可以如此描述：一个数 x 如果不作为前缀最大值，便将其放入一个集合 S 中；如果作为前缀最大值，选择一个 $T\subseteq S$，对总和贡献 $x\times(|T|+1)$，然后将 S 设为 $S-T$。

这时就可以 dp 了，记 $f(i,j,k)$ 表示做到第 i 个数，$|S|=j$，总和为 k 是否可行。转移可以贡献到 $f(i+1,j+1,k)$ 与 $\forall t\le j, f(i+1,j-t,k+h_i\times(t+1))$。时间复杂度 $O(n^3H)$，期望得分 40 分。

由于这是 01 的 dp，自然想到 bitset 优化。可以用类似完全背包的思想，从 $f(i,j)$ 贡献到 $f(i,j-1)$。时间复杂度 $O(\frac{n^3H}{\omega})$，期望得分 60 分。

注意到值域较小，可能的前缀最大值只有 H 个，我们可以只在每个值的最后一个位置进行第二种转移。对于第一种转移，一种比较好写且常数较小的做法是改记 $i-|S|=j$，这样我们可以不对 dp 数组做任何事。

时间复杂度 $O(\frac{n^2H^2}{\omega})$，期望得分 100 分。

D.食蜂操祈

我们可以发现 1 前面的数必须递减，否则栈内元素会乱序。类似地，2 的前面去掉 1 后（如果存在），也必须降序。

于是可以得到结论：**一个排列 P 可以被排序的充要条件是，对于每个 i，其前面所有 $>P_i$ 的数必须递减**。这个定理的另一种表述形式是，不存在 $i<j<k$，使得 $P_k<P_i<P_j$。

考虑保证 A 升序怎么做，此时字典序的限制便会消失。

考虑一个生成合法排列的过程：从 n 到 1 加数。在 x 加入之前，序列的前缀极长下降段的长度为 len，则 x 在新序列的位置可以是 1 到 $len+1$ 中的任何位置，而新序列的前缀极长下降段的长度则是 x 的位置。

这是一个可以进行 dp 的过程，但容易发现这类似栈维护括号匹配，答案即是卡特兰数。可以得到特殊性质的 10 分。

对于字典序的限制，经典套路是枚举 LCP 的长度，于是我们考虑确定了排列的一个前缀时的答案。

首先我们需要考察是否存在一个合法的排列。对于 $\forall i<j$ 且 $P_i<P_j$，若在 j 之前，所有 $<P_i$ 的数并未全部出现，则必然无解。反之则一定有解。

考虑计算有解时的答案。在倒序加数时，与没有限制的情况的不同之处在于：没在这个前缀出现过的数无法将前缀内的数弹出，而前缀内的数的决策一定，且必须将所有不在前缀内的数弹出。于是我们可以直接不将前缀内的数加入栈中，遇到一个前缀内的数时就清空栈。可以验证这一定生成的是合法排列。

此时，不在前缀内出现的数会在值域上形成若干连续段，他们之间的决策相互独立。于是，方案数就是卡特兰数的乘积。

如果我们暴力枚举 LCP 的后一位 i 如何选取，精细实现可以做到 $O(n^2)$，加上特殊性质期望得分 50 分。

注意到我们从前往后确定排列里数的过程，每次选择的数一定处于现在可选的数的第一个连续段中，否则一定会出现 231 状物。也就是说是，如果有解 A_i 一定需要在第一个连续段内，否则 $> A_i$ 的数定然不会出现在第一个连续段内。

计算完这部分答案后，A_i 将会把第一个连续段分裂成两部分。而计算答案时，我们只需要枚举该连续段中 $> A_i$ 的部分即可。这引导我们思考启发式复杂度算法：假设我们知道卡特兰数自卷积的值，我们便可以枚举连续段中 $\leq A_i$ 的部分，再扣掉这些值。于是这部分复杂度同启发式分裂，为 $O(n \log n)$。

至于卡特兰数自卷积的计算，并不需要 NTT，而是可以通过组合意义简单推导的。

总时间复杂度 $O(n \log n)$，期望得分 100 分。