inv

sub1

$\mathcal{O}(n!)$ 暴力。

sub2

$\mathcal{O}(2^n n)$，状压还未选的位置的集合。

sub3

考虑只留下前 i 个数的位置，然后插入第 $i+1$ 个位置。
并且因为第 $i+1$ 个位置只与第 i 个位置相关，所以考虑 DP。

令 $f_{i,j}$ 表示前 i 个数中 i 的排名为 j 时最大逆序对个数。
转移考虑枚举 $i+1$ 的排名 k，找出合法的 (j,k) 转移。

复杂度 $\mathcal{O}(n^3)$。

sub4

优化上述 DP 过程至 $\mathcal{O}(n^2)$ 即可。

sub5

由上述 DP 过程可以知道，如果要求 $p_i > p_{i+1}$，那么让 $i+1$ 在前 $i+1$ 个的排名为 1 肯定是最优的。
否则若 $p_i < p_{i+1}$，那么让 $i+1$ 排名恰好比 i 大 1 是最优的。

于是可以知道实际上只需要让连续的 $p_i < p_{i+1} < \cdots < p_{i+k}$ 的限制构造成
$p_i + k = p_{i+1} + (k-1) = \cdots = p_k$，那么剩下的 $p_i > p_{i+1}$ 的限制直接构造成递减的就行了。

至于逆序对个数，可以知道只有连续的上升段内部不会有逆序对产生，其余的都会产生，统计是简单的。

时间复杂度 $\mathcal{O}(n)$，多个 $\log$ 维护应该也是放过了的。

> 一个简单的多的做法：
> 考虑答案的上界就是除掉连续上升段内部的逆序对数，然后发现能构造出来，就做完了。

beauty

首先能够贪心的知道 b 的美丽值就是将 b 排序后的 $2(\sum\limits_{i=\lceil \frac{n}{2} \rceil+1}^{n} b_i - \sum\limits_{i=1}^{\lfloor \frac{n}{2} \rfloor} b_i)$。
其实也就是最大的 $\lfloor \frac{n}{2} \rfloor$ 个 b_i 有正的贡献，最小的 $\lfloor \frac{n}{2} \rfloor$ 个 b_i 有负的贡献。

sub1

$\mathcal{O}(V^n)$ 暴力。

sub2

当 $V=1$ 时，显然答案为 0。

当 $V=2$ 时，考虑枚举 1 的个数得到 2 的个数算贡献。

当 $V=3$ 时，类似的枚举 $1,2$ 个数得到 3 的个数算贡献。

时间复杂度 $\mathcal{O}(n^2)$。

sub3

当 $V=4$ 时，已经不再能枚举 3 个值了，于是不妨试着多拆拆贡献?

这部分分并没有什么特殊的做法，其实是为了提示正解，避免选手直接去想没啥前途的背包（？）。

sub4

留给一些 $\mathcal{O}(n^2V^2)$ 的做法。

sub5

考虑对最后排序得到的序列来统计。
于是可以想到值域从小到大 DP。

即记 $f_{i,j}$ 为考虑了值域 $1\sim i$，选了 j 个数的贡献和。
转移就可以考虑 $i+1$ 选的个数 k，$\mathcal{O}(1)$ 算出对应贡献。

因为要得到最后可能的原序列的个数，所以每个数的个数是必须需要的。
于是复杂度 $\mathcal{O}(n^2V)$。

sub6

考虑拆贡献：
$a-b=(a-(a-1))+((a-1)-(a-2))+\cdots+((b+1)-b)$。

于是这说明其实只用考虑对于 $1\le v<V$ 分为 $b_i\le V$ 和 $b_i>V$ 两种来计数，最后把所有 v 合并（其实也就是相加）。

那么现在的值域就只有 $0,1$ 了，就可以考虑枚举 0 的个数 c_0，有 $c_1=n-c_0$。

知道 c_0,c_1 后算大小的贡献是简单的，若 n 为奇数取掉中间的值，贡献即为 $\min(c_0,c_1)$。

那么方案数也很简单了，这 c_0 个数值要在 $1\sim v$ 中，c_1 个数值要在 $v+1\sim n$ 中，同时要把 c_0,c_1 最后一起拼成原序列，方案数即为 $v^{c_0}\times(n-v)^{c_1}\times\binom{n}{c_0}$。

时间复杂度 $\mathcal{O}(nV)$。

max

原题：https://atcoder.jp/contests/snuke21/tasks/snuke21_j

sub1

$\mathcal{O}(2^n)$ 暴力。

sub2

因为 $a_i = b_i$，所以可以直接合并 a_i, b_i 为一个限制，直接做二维偏序即可。

时间复杂度 $\mathcal{O}(n \log n)$。

sub3

因为限制是特征值，这说明实际上 ≥ 4 个数的选取都是无效的，因为最多就只有 3 个数能分别取一个 $\max$。

直接暴力统计，复杂度 $\mathcal{O}(n^3)$。

sub4

首先对于选取 ≤ 2 个显然是简单的，主要问题在于 $= 3$ 个怎么算。

考虑枚举 $a_i = \max a$，然后从小至大枚举 $\max b = b_j$。
那么对于 $\max c = c_k$，就要满足 $a_k < a_i, b_k < b_j, c_k > \max(c_i, c_j)$。

用个树状数组之类的数据结构维护，即可做到 $\mathcal{O}(n^2 \log n)$。

sub5

留给一些奇奇怪怪的 $\mathcal{O}(n^2)$ 或者是什么 $\mathcal{O}(n\sqrt{n}\log n)$ 状物的做法。

sub6

考虑容斥，容斥掉出现过的贡献。

首先如果只选了 1 个，显然就是 n。

如果选了 2 个，一共有 $\binom{n}{2}$ 种选法。
出现过的对 (i, j) 显然就需要满足 $a_i < a_j, b_i < b_j, c_i < c_j$，可以 cdq 分治求出。

如果选了 3 个，一共有 $\binom{n}{3}$ 中选法。

那么一个出现过的对就肯定最大值一定有一个下标取到了 ≥ 2 个 $\max$。
枚举这个下标 p，那么对于另外 2 个下标 i, j 就需要满足 $a_i < a_p, a_j < a_p, b_i < b_p, b_j < b_p$（这之后的部分对于 a, c 及 b, c 都是类似的，就只以 a, b 举例来说了）。
于是可以二维偏序求出 $a_i < a_p, b_i < b_p$ 的个数 cnt，那么 i, j 都应该在这 cnt 个中选取，于是减掉 $\binom{cnt}{2}$。

需要注意的是，如果 i, j 在 a, b, c 都被 p 偏序的话会被减掉三次，于是应该补回来两次，可以复用选 2 个数的信息。

时间复杂度 $\mathcal{O}(n \log^2 n)$。

power

原题：https://loj.ac/p/3722

sub1

暴力递推 $a(i)$ 与 $b(i)$。

即 $a(2i) = a(i), a(2i+1) = a(i) + 1, b(3i) = b(i), b(3i+1) = b(i) + 1, b(3i+2) = b(i)$。

时间复杂度 $\mathcal{O}(n)$。

sub2

因为 $x=y=1$，所以求的就是 $\sum\limits_{i=1}^{n} z^{b(i)}$。

直接做三进制下的数位 DP 即可，时间复杂度 $\mathcal{O}(\log_3 n)$ 或 $\mathcal{O}(\log_3^2 n)$。

sub3

因为 $y=1$，所以求的就是 $\sum\limits_{i=1}^{n} x^i z^{b(i)}$。

因为 i 在指数，所以 x^i 其实可以拆成 $x^{\sum\limits_j 3^j v_j}$，于是能够发现还是能够将贡献统计到三进制下的每一位，于是还是可以数位 DP，时间复杂度 $\mathcal{O}(\log_3^2 n)$。

sub4, 5

给一些神秘做法或者大常数做法的部分分（但是开 3s 了应该不会出现吧？）。

sub6

一种想法是同样考虑数位 *DP*，然后中间顺便维护上对应的二进制分解。
但是能够发现 3^w 对于二进制的贡献过于混乱，不能很好的处理。
但是考虑到对于三进制下第 w 位，其产生的最大贡献 $< 3^{w+1}$，那么在这里可以对应的定义一个 S_w 为能表示为 2^x 且 $\geq 3^{w+1}$ 最小的数。
那么就能知道，不管低 w 位怎么操作，基本都只会对 $\log_2 S_w$ 位造成影响，注意的是这里可能会有对下一位的进位。

于是可以考虑 *DP* 状态设为 $f_{w,g,s}$，表示当前从高到低考虑到了三进制第 w 位，二进制意义下对 $(\log_2 S_w)+1$ 位的进位为 g，且当前二进制表示下 $\bmod S_w$ 为 s。
对于转移，就只需要考虑三进制下第 w 位选取 $0/1/2$，且下一部分是否有对这部分的进位，对于 $y^{a(i)}$ 就可以在这里先统计出 $a(i)$ 对应的 $(\log_2 S_{w-1})+1 \sim \log_2 S_w$ 位的贡献了。

看着好像还是没什么优化。
这个时候就可以考虑用朴素的数位 *DP* 和带记忆化的数位 *DP* 相结合了。
考虑设定一个阈值 B。
对于 $w \geq B$ 时，考虑直接用朴素的数位 *DP*，复杂度即为 $\mathcal{O}(3^{\log_3 n-B})$。
对于 $w \leq B$ 时，考虑用带记忆化的数位 *DP*，那么复杂度就是状态数 $\mathcal{O}(3^{B+2})$（因为 S_w-3^{w+1} 并不会多多少）。

那么平衡一下，可以取 $B=\frac{\log_3 n}{2}-1$。
时间复杂度 $\mathcal{O}(3^{\frac{\log_3 n}{2}+1})$，可以认为是 $\mathcal{O}(\sqrt{n})$ 的。