简单序列题

> 给定长度为 n 的非负整数序列 $\{a_n\}$，你可以进行如下操作:
>
> - 选择 $1 \le i \le n-1$，将 a_i, a_{i+1} 同时变成 $\max\{a_i, a_{i+1}\} - 1$，但你必须保证所有元素时刻为非负整数。
>
> 目标是将所有元素全变成 0，求最少操作次数。
>
> $n \le 100000$。

我们发现每次都是操作最大值 a_i，如果 $a_{i-1} \ge a_{i-1}$，那么操作 $(i-1, i)$；否则操作 $(i, i+1)$。用个堆和并查集维护就好了。

简单打怪题

> 你正在玩一个 RPG 游戏。游戏由 n 层组成，你初始的生命值为 x、等级为 1，你的任务是从第 1 层开始往上走完 n 层。
>
> 第 i 层有 k_i 个怪物，它们的攻击力分别为 $a_{i,1}, a_{i,2}, \cdots, a_{i,k_i}$。到达这一层后，你需要选择这些怪物的一个**非空**子集 S，并与你选出的怪物进行战斗。一次战斗定义为不断重复以下两步，直到 S 中没有怪物存活。
>
> 1. 对于 $\forall j \in S$，若 j 依然存活，它会对你发动一次攻击，你的生命值会减少 $a_{i,j}$ 点。
> 2. 你选择一个 $j \in S$，使得 j 依然存活，并把它杀死。若你的等级 $< m$，则你的等级会上升一级。
>
> 给定 $h_1 \le h_2 \le \cdots \le h_m$，在战斗完成后，假设你的等级是 t，你的生命值会回复 h_t 点。
>
> 求使得**最终你的等级为** m 且整个过程中生命值恒为**正数**的初始生命值 x 的最小值。
>
> $n \le 100, m \le 5 \times 10^4$。

注意到 Johnson 序需要记两个信息，从前往后做的话，能做，但是需要一个二分。

但是**从后往前做，只需要记一个最小值就行了**，不需要记总和！所以设 $f_{i,j}$ 表示 i 之后的层，当前等级是 j，需要的最小初始等级。这个转移是可以决策单调性分治优化的。

简单判断题

> 给定一张无向图，判断它是否有奇度数点个数为 4 的倍数的生成树，一个测试点内有多张图需要判断。
>
> $n, m \le 2 \times 10^5$。

首先肯定是 2 的倍数。先随便找到一个生成树，如果它符合条件就好；不符合，我们就找一条非树边替换进去，如果存在一条对应的树边替换后效果很好，那就好了！所以我们随机一些生成树并枚举树边来判断就行了。

Doremy's Paint 2

> 给定一个长度为 n 的序列 $a_{1\cdots n}$，初始有 $a_i = i$。
>
> 现在有 m 个操作，第 i 个操作有参数 $l_i, r_i (l_i \le r_i)$，执行该操作会把 $a_{l_i+1\cdots r_i}$ 赋值为 a_l。
>
> 给定 k，对于所有 $x = 1 \sim m$，求如果依次执行第 x 个到第 $x+k-1$ 个操作那么序列中有多少种不同的元素。这里将第 $i+m$ 个操作视为第 i 个操作。$n, m \le 10^5$。

我们从后往前推，维护 f_i 表示 i 元素能生存到什么时间。那么一个 t 时刻的操作 $[L,R]$ 相当于令：

$$\begin{aligned}\forall i\in(L,R],a_i\leftarrow t\\ a_L\leftarrow\max_{j\in[L,R]}a_j\end{aligned}\tag{1}$$

发现 f 的连续段个数的总和只有 $O(n)$ 级别，所以我们可以用 set 来暴力维护这些连续段。在维护 set 的同时，维护一个树状数组，维护每个时间消失的元素个数，就能在树状数组上 $O(\log n)$ 进行一次查询了。

Fishermen

> 给你一个长度为 n 的序列 $\{a\}$，你可以将 $\{a\}$ 中的数重新排列，并根据重排后的序列 $\{a\}$ 生成序列 $\{b\}$，生成方法如下：
>
> - $b_1=a_1$。
> - $\forall i\in(1,n]$，b_i 是满足 $a_i|b_i$ 且 $b_i>b_{i-1}$ 的的小正整数。
>
> 求所以可能生成的序列 $\{b\}$ 中 $\sum\limits_{i=1}^n b_i$ 的最小值。$n\le 1000$。

找一个数组 $c_{1\sim n}$，使得 $b_i=c_ia_i$ 两两不同，那么肯定存在一个顺序使得 b 严格递增，代价也就是 $\sum c_ia_i$。

对于每个 a_i 枚举 $a_i\sim na_i$，分别建一个左部点，连向右部点 i。那么这张图的最小费用最大匹配就是答案。

考虑右部点数只有 n，所以我们可以在每次产生新增广路后（这只会发生 n 次），从汇点进行一次 bfs $\mathcal{O}(n^2)$，处理出所有能到达汇点的右部点；从小到大考虑这 $\mathcal{O}(n^2)$ 个左部点时，只要 $\mathcal{O}(1)$ 判断它连向的点是不是可以到达汇点就行了。总复杂度 $\mathcal{O}(n^3)$，可以用 bitset 优化掉一个 w。