题解

前言

本次比赛题目名字怀古，难度适中，正解全部为提高组知识点，码量较小，希望快退役的 wzy 能给在 CSP 之路上的你提供一个有力的援助。

小凯的疑惑

P.S. 题目并没有名字那么吓人。

Subtask1

不难发现答案不会超过 10^{18}，由于 n 很小，直接 2^n 枚举选择并判断是否为独立集即可。

Subtask2

由于答案不会超过 10^{18}，直接 dp 即可。
$f[i][0]$ 表示点 i 必须不选的最大值，$f[i][1]$ 表示点 i 无限制的最大值。则：

$$f[u][0]=\prod f[v][1], f[u][1]=\max\left\{f[u][0], v[u]\cdot\prod f[v][0]\right\}$$

因此直接 $O(n)$ 即可解决问题。

Subtask3

链的情况，就是把树 dp 变成了一维 dp 而已。

Subtask4

解法1

套路，看到乘积容易想到取对数，如取 $\ln$。把树上的点权全部取对数，就转化成了最大权独立集。因此在 dp 的时候顺便记一下乘积 $\bmod 998244353$ 的值即可。

由于点权随机，因此不会出现最大值和次大值乘积相差很小让 double 对数无法区分。复杂度 $O(nlogn)$，其中 $O(logn)$ 是求对数的复杂度。

解法2

（来自 lqs 的神仙做法）

不妨考虑 $subtask2$ 的 dp 方程，不难发现难以处理的地方就在于 $f[u][1]$ 需要取 $\max$。但是比较 $\max$ 的两个参数会发现，我们实际上就是比较 $\prod \frac{f[v][1]}{f[v][0]}$ 与 $v[u]$ 的大小关系。不妨令 $g[u] = \frac{f[u][1]}{f[u][0]}$，则若 $\prod g[v] > v[u]$，表示 $f[u][0]$ 更优，且此时 $g[u] = 1$；否则 $v[u] \cdot \prod f[v][0]$ 更优，且此时 $g[u] = \frac{v[u]}{\prod g[v]}$。

此时比较涉及到的数字大小在 $double$ 范围内，可以直接比较。总复杂度 $O(n)$。但是如果最大和次大的答案相隔太近，仍会超出 $double$ 精度导致无法比较。因此还是需要保证随机。

时间复杂度

P.S. 这题是由 2019 多校第二场最后一题改编来的。由于 wzy 在比赛的时候想出了一个比较妙的做法，然而发现正解很暴力，于是就出了这一题。

Subtask1

直接枚举左右端点，然后暴力统计中间每个数字的出现次数。总复杂度 $O(n^3)$。

Subtask2

枚举左端点，右端点从左往右扫，用数据结构（如 map）维护当前出现过的数字出现次数的最小值，总复杂度 $O(n^2logn)$。

Subtask3

（来自 lqs 的神仙做法）
观察到 b 是具有单调性的，如果一个数字 p 导致一个区间不可行，那么这个区间所有包含 p 的子区间必然也不可行。

考虑分治，$f(l,r)$ 表示当前分治到 l,r，那么我们把 $[l,r]$ 中所有出现次数 $< b_{r-l+1}$ 的数字全部删掉，区间就变成了若干段，分别递归处理即可。如果当前区间没有删掉任何数字，那么说明当前区间就是合法的，直接更新答案。

考虑这个做法的复杂度，即考虑递归层数。不难发现当删除数字的时候递归层数才会增加，而最多有 $O(\sqrt{n})$ 种不同的出现次数，因此这个做法的复杂度为 $O(n\sqrt{n})$。

Subtask4

这部分是多校原题。不妨考虑从左到右枚举右端点的同时维护合法左端点集合，即对于每个位置，维护当前右端点到它的区间中，有多少个合法数字。

考虑右端点移动一位，则当前右端点的合法数字个数为 n（即 0 到 n，除去它自己出现了 1 次），并且序列的某个区间的合法数字个数会同时 $+1$。由于合法数字任何时刻不会超过 $n+1$，而我们求的是对于每个右端点，合法数字个数为 $n+1$ 的最小左端点。因此直接线段树维护上述操作即可，复杂度 $O(nlogn)$。

其实知道原题之后，由于 b 的单调性，可以先二分答案区间长度把 b 统一，然后用 $subtask4$ 的做法做 $subtask3$ 也是可以在 $O(nlog^2n)$ 的时间内通过的。

Subtask5

线段树做法只能做到 $O(nlog^2n)$（至少我们暂时没想到更好的办法，如果神仙吊打了 wzy，可以来给出解法，感激不尽），考虑优化 $subtask3$ 的分治做法。

首先一次找出所有出现次数较小的数字比较耗时，我们可以找到任意一个出现次数不满足要求的数字，并且从这个数字处把区间分成两半进行递归，并不会影响正确性，还大大降低了编程复杂度。

但是由于分治两边可能不均等，会导致时间复杂度退化成 $O(n^2)$，但如果我们在 $\min(\text{左右两区间长度})$ 的时间内完成当前层的操作，复杂度就是启发式合并的复杂度，即 $O(nlogn)$。

于是利用类似于 meet in the middle 的思想，我们从区间两端同时找出现次数不满足要求的数字，显然寻找的时间就是 $\min(\text{左右两区间长度})$ 了。但是我们还需要维护每个数字的出现次数，考虑记 cnt 数组为每个数字的出现次数，并且假定每次递归时 cnt 数组恰好是当前区间中所有数字的出现次数，并且返回时清空 cnt 数组。

把拆开的两区间中较大的称为大区间，较小的称为小区间，每次找到分割点后，先把小区间中所有数字从 cnt 当中去掉，就可以直接递归大区间了。之后 cnt 数组被清空，我们只需要再遍历一遍小区间，把所有数字加到 cnt 中，递归完毕后 cnt 也恰好被清空，符合返回要求。

特殊的，如果一个区间找不到分割点，则需要遍历整个区间清空 cnt，复杂度显然没有问题。初始时，cnt 必须设置为每个数字的出现次数再进行递归。

由于每一步花费的复杂度都是 $O(\text{小区间长度})$，因此总复杂度为 $O(nlogn)$，可以通过。

逛公园

这题是常州集训完全没改动过的原题。

Subtask1

直接各种爆搜即可通过。

Subtask2

$m = 1$ 就相当于判断是否可行，可行输出 n，不可行输出 GG。

不难发现如果所有右端点都不相同的话，显然每个区间选择右端点就可以满足要求。考虑如何把右端点相同转化为右端点不同。

如果两个区间右端点都相同，那么让左端点较小的区间拥有较小的右端点显然更优，并且原问题的答案一定能够转化成这样之后的答案。因此先把所有区间按照右端点从大到小排序，对于第 i 个右端点 r_i，让它变成 $\min(r_{i-1} - 1, r_i)$ 并不断递推下去，就可以获得最后的右端点集合。

把右端点集合加到 set 里，把所有区间按照左端点从大到小排序，依次从 set 中选择小于等于它且离他原来右端点最近的新右端点，作为它现在的右端点。

经过这次操作之后，所有区间的右端点变得不同了。并且特殊的，如果在操作过程中某个区间选择的右端点小于其左端点（即操作后变为空区间），显然原问题无解，否则必然有解。总复杂度 $O(nlogn)$。

Subtask3

会做 $subtask2$ 应该就会 $subtask3$ 了。对于每种类型，把所有这种类型的区间变成右端点互不相同的。然后把所有类型的区间混在一起按照右端点排序，每次取所有还没有被满足的区间中右端点最小的那个，在这一天上班。之后对于每个类型中包含这个点的所有区间，可以把其中右端点最小的给删掉。

具体的话，对于每个类型维护一个优先队列，存左端点在当前点左边的所有区间，并且按照右端点从小到大排序。但是我们不能每次都查找每种类型，因此我们需要额外维护一个 $multiset$，对于每种类型都把右端点最小的扔进去，$multiset$ 也按照右端点排序，因此每次遍历 $multiset$ 即可维护所有类型。注意右端点移动时要把新的左端点扔到优先队列里去并维护 $multiset$。

不难发现每个区间最多被加入一次，删除一次，总复杂度 $O(nlogn)$。