NOIP 模拟赛
试题讨论与讲解
day1

Cydiater

清华大学

2019 年 9 月 2 日

1

Dove 的疑惑

题目大意

给定 $\{m\}$，询问有多少种可能的 $\{a\}$ 使得不存在 x 使得关于 $\{a\},\{m\}$ 的同余方程成立。

数据范围

对于全部测试点，保证 $n \leq 10^5, \prod_{i=1}^{n} m_i \leq 10^{18}, 1 \leq m_i \leq 10^{18}$。

测试点	n	$\prod_{i=1}^{n} m_i$	特殊性质
1,2,3,4	≤ 10		保证 m_i 均为质数
5,6,7	≤ 10	$\leq 10^3$	
8,9,10	$\leq 10^2$	$\leq 10^3$	
11,12,13	$\leq 10^3$	$\leq 10^3$	
14,15,16		$\leq 10^5$	
17,18,19,20			

得分情况

算法一

保证 m_i 均为质数

根据中国剩余定理，当 m_i 均为质数时，如果满足互不相同，那么当 $\{a\}$ 任意取值时我们总能构造出对应的 x 满足条件，所以我们只需要满足对于相同的 m_i 取到相同的 a_i 即可。

复杂度为 $O(n)$，期望得分 $20pts$。

算法二

$n \leq 10^3, \prod m_i \leq 10^3$

枚举所有可能的 a_i，然后利用扩展中国剩余定理验证即可。
复杂度为 $O(n \prod m_i)$，期望得分 $60pts$，结合算法一期望得分 $80pts$。

算法三

标算

我们取 $M = \mathrm{lcm}\{m\}$，容易观察到，对于任意一个 x，其 $\mathrm{mod}\{m\}$ 得到的 $\{a\}$ 总是与 $x + M \bmod \{m\}$ 得到的 $\{a\}$ 是一致的。而对于任意两个不同的数 x, y 来说，如果他们的差值不超过 M，那么则说明他们至少在模一个 m_i 时不同，也就是说存在不同的 $\{a\}$。

所以我们可以说，对于 $\{m\}$ 来说，如果保证了 m_i 互不相同，那么恰好存在 M 个 x 存在 $\{a\}$ 且互不相同。

所以，答案就是 $\prod m - M$。

复杂度为 $O(n)$，期望得分 $100pts$。

2

捉迷藏

题目大意

给定一棵 n 个节点的树，树上第 i 个点的权值为 $a_i \in \{0,1\}$。对于每个节点询问最大联通块的大小，使得该联通块包含该节点，同时含有偶数个 $a_i = 0$ 的节点。

数据范围

对于所有测试点，保证 $n \leq 10^6$。

测试点	n	$(\sum_{i=1}^{n}[a_i = 0] \bmod 2)$	特殊性质
1,2,3	≤ 20		
4,5	≤ 500		
6,7,8	≤ 1000	$= 0$	
9,10,11	≤ 1000		
12,13,14	$\leq 10^5$	$= 0$	
15,16,17,18	$\leq 10^5$		保证树的直径不超过 $2 \times \log n$。
19,20,21	$\leq 10^5$		
22,23	$\leq 5 \times 10^5$		保证树的直径不超过 $2 \times \log n$。
24,25	$\leq 10^6$		

得分情况

算法一

$(\sum_{i=1}^{n}[a_i = 0] \bmod 2) = 0$

容易发现，在保证了这个条件的前提下，对于每个节点都可以把整个树作为可以访问的联通块，直接输出 n 即可。

期望得分 $24pts$。

初步的观察

对于偶数的情况不再考虑，下面只考虑奇数的情况。

可以发现，为了实现尽量多的访问节点，那么最多会存在一个 $a_i = 0$ 的节点不被访问。那么问题转化成了对于每个节点 u，确定一个 $a_v = 0$ 的节点 v，使得在 u 为根的前提下，v 的子树大小最小。

算法二

$n \leq 1000$

利用前面提到的性质，要获得一个节点的答案，就指定当前节点为根，然后 DFS 整棵树，找到那个子树大小最小，同时 $a_v = 0$ 的节点 v。

复杂度 $O(n^2)$，期望得分 $44pts$，结合算法一期望得分 $56pts$。

进一步思考

对于每个节点来说，只有 $a_v = 0$ 的节点会对他贡献答案。对于每个 $a_v = 0$ 的节点 v 考虑，计算 v 对其他部分的贡献。

容易发现，将节点 v 删去后原树被划分为若干个联通块，那么节点 v 对每个联通块的贡献为联通块内的节点数量。

而每个节点的答案就是所有的贡献取 $\max$。

算法三

保证树的直径不超过 $2 \times \log n$

任选一个节点为根，那么每个 $a_v = 0$ 的节点 v，需要对于他所有儿子的子树取 $\max$，并且对该子树以外的部分也要取 $\max$。

儿子内的部分打标记，最后 DFS 整棵树即可。对于子树外的部分，遍历他所有的祖先，对于祖先内这个节点对应儿子以外的儿子打上标记即可。

复杂度 $O(n \log n)$，期望得分 $68pts$，结合算法一期望得分 $80pts$。

算法四

DFS 序结合 STL 容器

观察到每次的取 $\max$ 都是针对子树和子树外的部分而言的，而子树对应 DFS 序上是一段连续的区间，而子树外的部分是至多两段连续的区间。

因此本题转化成了类线段覆盖问题，差分后用 set 维护当前最大值即可。

复杂度 $O(n \log n)$，期望得分 $84pts - 92pts$

算法五

标算

可以发现，暴力遍历祖先是没有必要的，我们同样可以打一个标记来解决。
最后进行两次 DFS，一次维护上传标记，一次维护下传标记，实现的细节请参考 std。
复杂度 $O(n)$，期望得分 $100pts$。

3

Cicada 的序列

题目大意

Cicada 有一个长度为 n 的序列，序列中第 i 个数的值为 a_i。对于该序列的一个**连续子序列** $a_l, a_{l+1}, \cdots, a_r$ 来说，其带给 Cicada 的愉悦度为 $a_l \bmod a_{l+1} \bmod a_{l+2} \cdots \bmod a_r$。

现在 Cicada 想知道，这个序列的**所有连续子序列**能给他带来的愉悦度的**和**是多少。

数据范围

对于所有的测试点，保证 $n \leq 3 \times 10^5, a_i \leq 10^9$。

测试点	n	a_i	特殊性质
1,2,3	≤ 100	≤ 50	
4,5,6	≤ 5000	$\leq 10^5$	
7,8,9	$\leq 10^5$	≤ 100	保证 a_i 是随机数列
10,11		≤ 100	保证 a_i 是随机数列
12,13,14			保证 a_i 是不递减数列
15,16,17			保证 a_i 是不递增数列
18,19,20			

得分情况

算法一

$n \leq 100$

首先暴力枚举区间，然后暴力计算，复杂度 $O(n^3)$，期望得分 $15pts$。
不会真有人写这个吧?~~

算法二

$n \leq 5000$

观察到当左端点固定的时候，右端点可以 $O(1)$ 的移动和维护答案。因此可以每次枚举左端点，然后固定左端点移动右端点，移动的同时维护答案。

复杂度 $O(n^2)$，期望得分 $30pts$。

观察 $a \bmod b = c$ 的性质

$$a \bmod b = c$$

- 当 $b > a$ 时，该运算并没有任何意义，即 $a = c$。
- 当 $b \leq \frac{a}{2}$ 时，$c < b \leq \frac{a}{2}$。
- 当 $\frac{a}{2} < b \leq a$ 时，$c = a - b \leq \frac{a}{2}$。

性质的利用

因为一次有效的运算**至少**会使当前值减半，因此 $a_l \bmod a_{l+1} \bmod a_{l+2} \cdots \bmod a_r$ 中**有效**的运算不超过 $\log n$ 次。

当左端点固定的时候，可以按照右端点答案是否相同进行划分，最多只会划分 $\log n$ 个不同的连续段，每段内的答案是相同的。

问题转化为对于每个左端点，如何快速的确定所有划分的分界点。

算法四

保证 a_i 是不递减数列

这时对于每个左端点来说，只存在最右端一个分界点，直接统计即可。
复杂度 $O(n)$，期望得分 $15pts$，结合算法二期望得分 $45pts$。

算法五

保证 a_i 是随机数列 & $a_i \leq 100$

因为保证了 a_i 是随机数列同时 $a_i \leq 100$，因此在期望的意义下长度为 100 的连续子区间内**大概率**会存在一个 1。结合前面给出的性质可以发现 1 一定是最后一个分界点。因为对于每个左端点暴力查找分界点，在找到最后一个分界点时退出即可。

复杂度 $O(n\max\{a_i\})$，期望得分 $25pts$，结合算法二和算法四期望得分 $70pts$。

算法六

主席树

当固定左端点时，查询右端点本质上是在下标后缀内对权值进行区间取最小值，可以利用主席树来解决。

复杂度 $O(n\log^2 n)$，期望得分 $100pts$。

算法七

线段树/BIT

利用主席树可以不加思考的完成查找的过程，但是并没有使用主席树的必要。

可以考虑只拿线段树来维护，但是具体实现时会发现这个线段树更新的顺序和我们询问的顺序是相反的，同时因为 $\max$ 自己的性质并不好直接修改。

可以通过维护每个权值的下一个出现位置来实现更新，具体实现可以参考数据包中的 seg.cpp。

另外还有一种处理方法，观察到插入和询问恰好是逆序关系，因此我们记录下修改的过程，在询问时倒退撤销之前的修改操作即可，这种做法对空间的消耗可能比较大。

复杂度 $O(n\log^2 n)$，期望得分 $100pts$。

算法八

二分 +RMQ

可以发现本题中每次的查找具有可二分性，每次查找二分即可。注意判定时需要用 ST 表来维护 RMQ。

复杂度 $O(n\log^2 n)$，期望得分 $100pts$。

算法九

分治 [标算]

重新审视这个题目，可以发现我们查找的过程其实很符合序列分治的结构。
我们结合代码来说明。

```cpp
vector<pii> dc(int le, int ri) {
    if (le == ri) {
        if (ri == n) for (auto w : st[ord[le]]) ans += w.fi;
        std::sort(st[ord[le]].begin(), st[ord[le]].end());
        return st[ord[le]];
    }
    int mi = (le + ri) / 2, pp = le, qq = mi + 1;
    up (i, le, ri) if (ord[i] <= mi) _ord[pp++] = ord[i];
    else _ord[qq++] = ord[i];
    std::memcpy(ord + le, _ord + le, sizeof(int) * (ri - le + 1));
    vector<pii> ss = dc(le, mi), cur, tmp;
    assert(cur.empty());
    int o = mi + 1, cl = n + 1;
    for (auto w : ss) {
        while (o <= ri && a[ord[o]] <= w.fi) {
            cmin(cl, ord[o]);
            o++;
        }
        if (cl <= ri || ri == n) ans += (ll)w.fi * (cl - w.se);
        if (w.fi == 0) continue;
        if (cl <= ri) {
            st[cl].push_back(make_pair(w.fi % a[cl], cl));
        }
        else cur.push_back(w);
    }
    ss = dc(mi + 1, ri);
    pp = 0; qq = 0;
    while (pp != (int)cur.size() || qq != (int)ss.size()) {
        if (pp == (int)cur.size()) tmp.push_back(ss[qq++]);
        else if (qq == (int)ss.size()) tmp.push_back(cur[pp++]);
        else tmp.push_back(cur[pp] < ss[qq] ? cur[pp++] : ss[qq++]);
    }
    return tmp;
}
```