4 QOJ1832 Crab's Cannon 解题报告

4.1 题面描述

给定 n, l 以及 n 个互不相同的正整数 a_i，称一个集合 S 是好的当且仅当存在一个长为 l 的字符集不限的字符串满足 S 恰好是其所有回文前缀的长度构成的集合。

求 a_i 构成的集合 A 的超集里好的集合的最小大小。

$2 \le n \le 3 \times 10^5$，$1 \le a_i \le l \le 10^{18}$。时间限制 3s，空间限制 512MB。

4.2 题目解法

4.2.1 约定

- s' 表示字符串 s 的反串。
- $|s|$ 表示字符串 s 的长度。
- $a + b$ 表示字符串 a 和 b 按顺序拼接的结果。
- s_i 表示串 s 的第 i 个字符，如无特殊说明，下标从 1 开始。
- s 有周期 x 表示 $\forall 1 \le i \le |s| - x, s_i = s_{i+x}$。

4.2.2 判定性问题

我们先考虑判定性问题，即判断集合 A 本身是否是好的，显然 A 集合必须包含 1。

平凡地，我们有如下的暴力做法：

直接考虑 A 中每个回文前缀造成的字符等价关系，然后把每个等价类赋值成不同的字符，最后检查整个串是否只有这些前缀是回文的。

考虑如何优化这个判定过程，我们有引理如下：

引理 4.1. *若串 s 有长为 x, y 的两个回文前缀，且 $x < y, 2x > y$，那么 s 也有长为 $2x - y$ 的回文前缀。*

证明. 当 $i \le 2x - y$ 时，有 $s_i = s_{x+1-i} = s_{y-x+i} = s_{2x-y+1-i}$。 □

利用这个结论，考虑把 A 集合的元素从小到大排序，对于相邻的 $x < y$，我们判断是否有 $2x - y \le 0$ 或 $2x - y \in A$。

全都满足时考虑增量构造 s。

如果 $2x - y \le 0$，直接令新的 s 为 $s + t + s'$，其中 t 是一个长为 $y - 2x$ 的回文串，且前一半的字符不重复且未出现过。

如果 $2x-y>0$，设 t 是 s 长为 $y-x$ 的前缀，我们令新的 s 为 $s+t'$。此时 s 长为 $y-x$ 的前缀为 t，而中间剩的部分是长为 $2x-y$ 的后缀，因为 s 回文它也是长为 $2x-y$ 的前缀，因此中间也是回文的。

因此在我们构造的过程中，新的等价类中的字符是从未出现过的，且旧的等价类不会合并，因此这个暴力构造被优化到了 $O(l)$ 的复杂度。

那么这个引理是否是充要的呢？不幸的是，我们有如下反例：

$A=\{1,2,4\}$，可以发现 s 只能形如"aaaa"。

这告诉我们，有可能在复制长为 $y-x$ 的前缀的反串（也即长为 $y-x$ 的后缀）到 s 末尾时，也产生了一些介于 x,y 之间的多余的回文前缀。

在进行进一步的分析之前，我们先给出下面的引理并证明：

引理 4.2 (弱周期引理). *若 s 有周期 x,y，且 $x+y\le|s|$，那么 s 也有周期* $\gcd(x,y)$。

证明. 不妨设 $x>y$，为了方便起见，我们下面认为字符串下标从 0 开始。

考虑对每个 i 说明 $s_i=s_{i \bmod \gcd(x,y)}$，首先我们令 $j=i \bmod x$，依定义有 $s_i=s_j$。

由裴蜀定理，存在正整数 k 满足 $ky \bmod x=\gcd(x,y)$，于是存在非负整数 k 满足 $(j+ky) \bmod x=i \bmod \gcd(x,y)$。

考虑从 $z=j$ 开始令 z 加 k 次 y，每当 $z\ge x$ 时，就让 z 减去 x。可以发现这个过程中始终有 $s_z=s_j$，且最后 $z=i \bmod \gcd(x,y)$。于是得证。 □

回到刚才的问题，我们要分析在复制长为 $y-x$ 的前缀的反串（也即长为 $y-x$ 的后缀）到 s 末尾时，什么时候会产生一些介于 x,y 之间的多余的回文前缀。

首先，我们有 $y-x\le x$，即 $2x\ge y$。不妨设存在 $z\in(x,y)$ 使得 z 也是回文前缀，那么有 $y-z+y-x\le 2(y-x)\le y$，于是有 $y-\gcd(y-z,y-x)$ 也是回文前缀，且容易发现 $\gcd(y-z,y-x)$ 是 s 长为 y 的前缀的周期。

类似上面的证明，我们也能说明多出来的所有前缀 z 加上 x,y 本身构成了一个等差数列，且公差为长为 y 的前缀的一个周期以及这个前缀长为 $y-x$ 的后缀的最小整周期。

不妨设公差为 d，那么最短的多出来的回文前缀是 $x+d$，而由引理 4.1 我们知道 $x-d$ 也是回文前缀。

由于我们只关心之前的判据不对的情况，我们有 $2d\le y-x\le x$。

设长度小于 x 的最长回文前缀为 w，显然有 $w\ge x-d$。

因为 $x-(x-d)+x-w\le 2d\le x$，我们可以再次使用弱周期引理，得到 $x-\gcd(d,x-w)$ 也是回文前缀，进一步有 $x-w\mid d$、长为 x 的前缀有长为 $x-w$ 的周期、长为 y 的前缀长为 $y-x$ 的后缀也有长为 $x-w$ 的周期。

而因为 d 是最小整周期，$x-w\mid d$ 且也是周期，所以必然有 $x-w=d$。

于是此时一定有 $x-w\mid y-x$，且 $x-w\ne y-x$。

总结一下，集合 A 是否是好的可以用如下条件判定：

- 判断是否有 $1 \in A$。
- 判断对于 A 中排序后相邻的任意 $x < y$ 是否都有 $2x \le y$ 或者 $2x - y \in A$。
- 判断对于 A 中排序后相邻的任意 $x < y < z$ 是否都有 $2y < z$ 或者 $y - x = z - y$ 或者 $y - x \nmid z - y$。

如果以上条件都满足，那么 A 就是好的集合。为了方便起见，我们称第二个条件为 C_1，第三个条件为 C_2。

4.2.3 原问题

下面让我们解决原问题。

首先，我们考虑怎么优化回文前缀集合的存储，因为它最大可以达到 $|s|$ 的大小。

由 C_1，我们首先能推出回文前缀长度排序后的差分是不降的，更进一步地，我们有以下引理：

引理 4.3. *一个回文串 s 所有回文前缀的长度构成了 $O(\log|s|)$ 个等差数列。*

证明. 设 s 长度为 n，且 s 有长为 x 的回文前缀，那么有：$s_i = s_{x+1-i} = s_{n-x+i}$，于是 s 有长为 $n - x$ 的周期，反之亦然。

考虑 s 长度在 $[2^k, 2^{k+1})$ 之间的所有回文前缀，不妨设最长的长度为 y，那么对于任意两个长度在 $[2^k, y)$ 的回文前缀 u, v，s 长为 y 的前缀有周期 $y-u, y-v$，于是有周期 $\gcd(y-u, y-v)$，即有回文前缀 $y - \gcd(y-u, y-v) \ge u$。

若长度在 $[2^k, 2^{k+1})$ 之间的回文前缀数量 ≤ 2，这些回文前缀构成了至多一个等差数列。

否则，考虑取 u 为次长回文前缀，v 为任意 $[2^k, u)$ 的回文前缀，那么由 u 是最长有 $y - \gcd(y-u, y-v) = u$，即 $y - u \mid y - v$。

而一个串有周期 x 时显然也有周期 px，于是长度在 $[2^k, 2^{k+1})$ 之间的回文前缀 v 满足 $y - u \mid y - v$，且任意 $y - v = k(y-u)$ 的 v 都是回文前缀，于是长度在 $[2^k, 2^{k+1})$ 之间的回文前缀构成了一个公差为 $y - u$ 的等差数列。

于是每个 k 对应的 $[2^k, 2^{k+1})$ 区间里只有一个等差数列，总共至多有 $O(\log|s|)$ 个等差数列。 $\square$

结合这个引理，回文前缀长度排序后的差分只有 $O(\log|s|)$ 个值不同的连续段，且值单调不降。

下面我们把输入的回文前缀长度排序后做差分，然后依次把这些差分加入到回文前缀长度差分的集合里，分类讨论发生了什么。

我们称一个差分是“大的”当且仅当它大于之前的所有差分之和，设当前插入的差分为 x，有如下几种情况：

1. x 是大的。

2. 设上一个差分是 y，如果有 $y \mid x$，把 x 劈成 $\frac{x}{y}$ 个 y 直接加入即可。

3. 若 $y < x$，我们对 x 使用 C_1。如果存在一段差分数组的后缀和为 x，直接加入 x，否则找到对应的位置把那个差分按照 x 的位置劈成 u, v，并把在它之后的连续段先暂时取出来。我们依次加入 u, v 以及后面的连续段和 x，因为后面的连续段已经满足了 C_1，所以只需依次检查是否会被第二种情况劈开即可。

4. 若 $y > x$，我们先用 C_1 把 y 劈成 z 与 k 个 x（$z < x$），然后加入 z 和 $k+1$ 个 x。

我们容易说明上面的做法是对的，但是其复杂度并不正确，因为把一个差分劈开的情况会造成多次递归。

下面，我们再次使用弱周期引理来优化：

如果把一个差分 y 劈成若干个差分 $b_1, b_2, \ldots, b_k$，且 y 不大于之前的所有差分之和，那么直接把 y 劈成若干个 $\gcd(b_1, b_2, \ldots, b_k)$ 也是正确的。证明和之前的情况类似。

于是，上面第三条把 y 劈成 u, v 的时候，如果 y 不是大的，那么我们直接把它们劈成若干 $\gcd(u, v)$ 并只做一次递归，否则依次加入 u, v。第四条把 y 劈成 z 和 k 个 x 的情况同理，如果 y 不是大的，直接加入 $\frac{x+y}{\gcd(x,y)}$ 个 $\gcd(x, y)$。

此时我们的算法只有在劈开一个“大的”差分的时候会产生多于一个递归，不产生额外递归的时候至多把所有段弹出再插入，一次只会贡献 $O(\log |s|)$ 的总复杂度，而上述的两种情况都至多会贡献两个新的“大的”差分。

如果两个差分中有大的，只会产生 $O(1)$ 的额外复杂度。否则这次操作完全销毁了一个“大的”差分。

这里我们认为两个差分中只有一个“大的”的情况相当于继承了原来的“大的”差分，而两个差分都是“大的”的情况是先继承了原来的差分，然后创建了一个新的差分。

因为整个算法流程仅仅在劈开原有的差分，所以所有创建的“大的”差分之间是不交的，于是它们的总数就只有 $O(\log |s|)$。

而我们产生额外递归的情况会销毁一个大的差分，并且可以视作每额外递归一次会产生 $O(\log |s|)$ 的复杂度，于是总复杂度就为 $O(n \log |s| + \log^2 |s|)$，足以通过这道题。

参考资料

[1] Mateusz Radecki (Radewoosh), Marek Sokolowski (mnbvmar), “XX Open Cup Grand Prix of Warsaw Editorial”, QOJ,

https://qoj.ac/download.php?type=attachments&id=1510&r=1

[2] AThousandMoons, “Gym102341【杂题】”, 博客园,

https://www.cnblogs.com/AThousandMoons/p/14629050.html

[3] Wikipedia contributors, “Maximum flow problem”, Wikipedia,

https://en.wikipedia.org/wiki/Maximum_flow_problem

[4] kefaa2, antontrygubO_o, and gepardo, “Problem Tutorial: Crab’s Cannon”, QOJ, https://qoj.ac/download.php?type=solution&id=1832

[5] Yikun Hou, “Border Theory 学习笔记”, 洛谷, https://www.luogu.com.cn/article/nbz2k022