

2017 年全国青少年信息学奥林匹克

浙江省队选拔赛第一试题解

竞赛时间：3 月 23 日 8:00 – 13:00

题目名称	仙人掌	树状数组	多项式
目录	cactus	bit	poly
可执行文件名	cactus	bit	poly
输入文件名	cactus.in	bit.in	poly.in
输出文件名	cactus.out	bit.out	poly.out
每个测试点时限	1s	4s	3s
内存限制	512MB	512MB	512MB
测试点数目	10	10	10
每个测试点分值	10	10	10
是否有部分分	否	否	否
题目类型	传统型	传统型	传统型
是否有附加文件	是	是	是

提交源程序必须加后缀

对于 C++ 语言	cactus.cpp	bit.cpp	poly.cpp
对于 C 语言	cactus.c	bit.c	poly.c
对于 Pascal 语言	cactus.pas	bit.pas	poly.pas

编译开关

对于 C++ 语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 C 语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 Pascal 语言	-O2	-O2	-O2

1 仙人掌

如果输入的图不是仙人掌，那么方案数一定为 0，否则可以发现可以把仙人掌上所有的环扣掉，变成若干棵树的问题。

令 f_i 为以 i 为根的子树有一条从 i 子树内部（不包括第 i 个节点）出发连向外部的边的方案数， g_i 则为没有这样的边的方案数。直接这么 DP 的话不好转移，可能需要 $O(n^2)$ 的时间复杂度，因为要知道 i 的孩子中有多少个有边延伸出来，才能计算把这些边配对的方案数。可以使用分治 FFT 优化这一部分的时间复杂度，即把每一个节点看成 $f_i x + g_i$ ，然后把所有孩子相乘，能做到 $O(n \log^2 n)$ 。

比较好的转化是，可以发现题目中要求没有重边，我们对于每一个方案，把每一条没有被非树边覆盖的边都变成重边，可以发现答案等于每一条边都被覆盖的仙人掌个数。相当于每一个子树必须要有一条边从树内延伸到树外（这儿要包括子树的根节点），这样就可以做到线性了。

2 树状数组

不难发现这个写错的树状数组求的其实就是单点加，询问后缀异或和（注意特判 $l = 1$ ）。

证明：考虑在 x 这个位置进行了 Add，然后在 y 这个位置进行 Find。分三种情况：

- 1) $x < y$ ，那么显然不会被统计进答案。
- 2) $x = y$ ，那么一定会被统计进答案。
- 3) $x > y$ ，取最高的 x, y 不同的二进制位，那么这一位 x 一定是 1， y 一定是 0。考虑 Add 和 Find 的过程。一个相当于每次抹掉最低的二进制为，另一个相当于给最低的二进制为加一。这时再分两种情况：
 - a) y 在这一位后的所有二进制位都是 0，那么 x 在抹掉这一位（以及后面的所有 1）后就出现在了 Find 的路径里。
 - b) y 在这一位后存在一个二进制 1，那么不难发现 x 在抹掉这一位后面的所有 1 后得到的数，一定在 y 的 Find 路径上（经过若干次增加一定会产生在这一位的进位）。

所以在这种情况 x 一定会被统计进答案。

考虑什么情况下答案会错误，令 t_i 为 i 处的修改次数。那么当 $l \neq 1$ 的时候，答案不合法当且仅当 $(t_{l-1} + r) \bmod 2 = 1$ ； $l = 1$ 时，答案不合法当且仅当 $(\sum_{i \neq r} x_i) \bmod 2 = 0$ ，因此我们只需要维护 $(x_{l-1} + x_r) \bmod 2$ 即可。

接下来就是基础的动态二维数点（或者说三维数点）了，把修改和询问看成二维平面上的点 (l, r) ，对于每一个询问，若干个矩形内的修改点会产生贡献，其中贡献可以表示成 (a, b) 二元组，表示有 b 的概率产生影响， a 的概率不影响（当然维护一个也可以）。然后进行数点就可以了，树套树或者分治都可以。

一个细节是二元组 $(\frac{1}{2}, \frac{1}{2})$ 不可逆，因为不能简单的用直接使用树状数组。可以用线段树替代或者记录一下矩形内的零元个数。

时间复杂度 $O(n \log^2 n)$ ，标程常数写的非常丑，时限开了标程两倍，应该不会有卡常数的情况吧。

3 多项式

首先需要发现的性质是，对于 $f(x) = \sum x^{t_i}$, $f(x)^2 = \sum x^{2t_i}$ ，因为交叉项都被消掉了。这样求 $f(x)^m$ 我们就可以用快速幂在 $O(\log m)$ 次乘法下解决，时间复杂度为 $O(nm \log m)$ 。

对于 45 两个点，应该是一道 SRM 的题，考虑计算 $[x^t]f(x)^m$ ，当 m 是偶数的时候，等价于计算 $[x^{\frac{t}{2}}]f(x)^{\frac{m}{2}}$ 。当 m 是奇数的时候，等价于对于 $i \in [0, n]$ ，计算 $[x^{t-i}]f(x)^{m-1}$ 。记忆化后递归处理。令 $w = R - L$ ，可以发现需要计算的状态数为 $O(\log^2 m + w \log m)$ 。随便使用 map 什么的记忆化一下就可以了。

当询问全局的时候，考虑维护在当前串中，所有长度为 18 的不同 01 串各出现了多少次，这样的 01 串只有 2^{18} 个。考虑倍增，在把次数乘 2 的时候，每一个长度为 18 的 01 串变成了长度为 37 位的 01 串（算上头尾的 0），这个时候截取高位的 18 位即可。把次数乘 2 加 1 时，可以发现高 18 位是由低 19 位计算得到的，因此直接拿每一个长度为 37 的串与 $f(x)$ 相乘截取高位的两个 18 位转移即可。同时考虑一些边界问题，需要维护开头的一段和末尾的一段的 01 串，因为这部分复杂度不是瓶颈，随便维护个几十位几百位都可以。

当询问区间时，首先把问题转化为询问前缀。同样还是维护在一个前缀中，所有长度为 18 的不同 01 串各出现多少次，在倍增的时候我们把长度过长的部分给截取掉就可以了。时间复杂度为 $O(2^{\max(n, K)} \log m)$ 。