

NOIP 模拟赛

时间：2024 年 7 月 24 日

题目名称	地板	异或	正方形	宝物
题目类型	传统型	传统型	传统型	传统型
目录	floor	xor	square	treasure
文件名	floor	xor	square	treasure
输入文件名	floor.in	xor.in	square.in	treasure.in
输出文件名	floor.out	xor.out	square.out	treasure.out
时间限制	1.0 秒	1.0 秒	2.0 秒	4.0 秒
内存限制	512 MiB	512 MiB	1 GiB	2 GiB
测试包数目	10	10	10	13
测试包等分	是	否	是	否

提交源程序

C++ 语言	floor.cpp	xor.cpp	square.cpp	treasure.cpp
--------	-----------	---------	------------	--------------

编译选项

C++ 语言	-O2 -lm
--------	---------

【注意事项（请仔细阅读）】

1. 选手提交的源程序必须存放在已建立好的，且带有样例文件和下发文件的文件夹中，文件夹名称与对应试题英文名一致。
2. 文件名（包括程序名和输入输出文件名）必须使用英文小写。
3. C++ 中函数 `main()` 的返回值类型必须是 `int`，值必须为 0。
4. 对于因未遵守以上规则对成绩造成的影响，相关申诉不予受理。
5. 若无特殊说明，结果比较方式为忽略行末空格、文末回车后的全文比较。
6. 程序可使用的栈空间大小与该题内存空间限制一致。
7. 在 Linux 终端中执行命令 `ulimit -s unlimited` 可将当前终端下的栈空间限制放大，但选手使用的栈空间大小不应超过题目限制。
8. 每道题目所提交的代码文件大小限制为 100KB。
9. 若无特殊说明，输入文件与输出文件中同一行内的多个整数、浮点数、字符串等均使用一个空格进行分隔。
10. 输入文件中可能存在行末空格，请选手使用更完善的读入方式（例如 `scanf` 函数）避免出错。
11. 直接复制 PDF 题面中的多行样例，数据将带有行号，建议选手直接使用对应目录下的样例文件进行测试。
12. 使用 `std::deque` 等 STL 容器时，请注意其内存空间消耗。
13. 请务必使用题面中规定的的编译参数，保证你的程序在本机能够通过编译。此外，不允许在程序中手动开启其他编译选项，一经发现，本题成绩以 0 分处理。
14. 在最新版本的 NOI Linux 下使用 LemonLime 测评。

地板 (floor)

【标准算法】

令 $m = \max_{i=1}^n s_i$ 。

显然前两个数和后两个数互相独立，因此我们可以计算满足 $\left\lfloor \frac{s_a}{s_b} \right\rfloor$ 的 (a, b) 的数量 f_i ，最后计算 $\sum_{i=0}^T f_i f_{T-i}$ 。

一种思路是枚举 s_a 进行整除分块，但是复杂度为 $O(n\sqrt{m})$ ，不能接受。

因此我们可以考虑暴力枚举 i, b ，计算满足条件的 s_a 的数量。易知若 $s_a \in [is_b, (i+1)s_b)$ ，则有 $\left\lfloor \frac{s_a}{s_b} \right\rfloor = i$ 。

由于调和级数 $\sum_{i=1}^m \frac{m}{i} \sim O(m \log m)$ ，因此我们在 $O(m \log m)$ 的时间内解决了这个问题。

异或 (xor)

【标准算法】

令 $m = \max_{i=1}^n a_i$ 。

首先相等的数可以去重，因此可以将 n 变成 $\min(n, m)$ 。

令 g_i 表示 $\sum_S [|S| \geq 2 \wedge f(S) \geq i]$ ，则最终答案即为 $\sum_{i=1}^{\infty} (g_i - g_{i+1})i = \sum_{i=1}^{\infty} g_i$ 。

考虑计算 g_i 。令 $\text{highbit}(x)$ 表示 x 的二进制表示下最高位 1 的位置。不难发现对于任意 u, v ，若 $\lfloor \frac{a_u}{2^{\text{highbit}(i)+1}} \rfloor \neq \lfloor \frac{a_v}{2^{\text{highbit}(i)+1}} \rfloor$ ，则一定有 $a_u \oplus a_v \geq i$ 。

这鼓励我们同时计算出 $i \in [2^k, 2^{k+1})$ 的 g_i ，并且由于上面的性质，我们已经将问题分成了若干份，只需要对每组 $\lfloor \frac{a_u}{2^{k+1}} \rfloor$ 相等的单独做，最后做乘法原理即可。

现在考虑每组怎么做，首先组里不可能选超过两个数，因为根据抽屉原理，一定至少会有两个数满足 $\lfloor \frac{a_u}{2^k} \rfloor = \lfloor \frac{a_v}{2^k} \rfloor$ ，从而得到 $a_u \oplus a_v < 2^k$ ，显然不合法。

同时，当我们要选择两个数时，显然它们的第 k 位是不相等的，因此我们可以将它们再分成两个集合，第一个集合里选一个数，第二个集合里选一个数。我们可以暴力枚举选哪两个数，因为每对数恰好被计算一次，因此复杂度是 $O(n^2)$ 的。

注意题目中 $|S| \geq 2$ 的限制，因此在合并答案时还需要做一个小背包。总时间复杂度 $O(n^2 \log m)$ 。

正方形 (square)

【标准算法】

定义关于矩阵的函数 $f_y(a)$ 表示右边界在 y 的正方形的最大边长。

对 x 维建线段树，考虑一个节点 $[l, r]$ 存储 $[f_1(a_{l\dots r, 1\dots m}), f_2(a_{l\dots r, 1\dots m}), \dots, f_m(a_{l\dots r, 1\dots m})]$ 。

修改和查询是简单的。关键在于如何合并两个节点的答案。

假设我们已经求出了 $[l, mid)$ 和 $[mid, r)$ 的节点，现在要求 $[l, r)$ 。

不经过 $x = mid$ 的正方形可以直接通过上下取 \max 得到，关键在于求出经过 $x = mid$ 的正方形。

令 up_i 表示从 $x = mid$ 开始向上连续的 1 的数量， $down_i$ 表示从 $x = mid$ 开始向下连续的 1 的数量。

则现在问题转为对每个 $i \in [1, m]$ 求最小的 j 满足 $\min_{j \leq k \leq i} up_k + \min_{j \leq k \leq i} down_k \geq i - j + 1$ 。这个东西可以用双指针和滑动窗口维护，直接套即可解决本题。

复杂度 $O(qm \log n)$ 。

宝物 (treasure)

【送分】

特殊性质 A 和特殊性质 B 可以直接打表找规律，因此可以拿到送的 20 分。

【标准算法】

$$f_i =$$

考虑 CDQ 分治：

1. 假设当前为 $solve(l, r)$ ，要求出 $[l, r)$ 的真实答案。
2. 若 $l = r$ 返回。
3. 令 $mid = \lfloor \frac{l+r}{2} \rfloor$ 。
4. 递归 $solve(l, mid)$ 。
5. 计算区间 $[l, mid)$ 对 $[mid, r)$ 的贡献。
6. 递归 $solve(mid, r)$ 。

CDQ 分治的关键在于步骤 4：

求出 $[l, mid)$ 的后缀最大值，和 $[mid, r)$ 的前缀最大值。

考虑枚举转移中的交易所 k ：

- 若交易所在左侧：枚举后缀最大值等于该交易所权值的所有坐标 i ，这些坐标会分别对右侧的两个不同的区间造成不同的贡献，一种是右侧花费时间更长，一种是左侧花费时间更长。贡献分别是 $A(x - k) + f_i$ 和 $B(k - i) + f_i$ ，这两种贡献都可以用线段树打 tag 直接维护。
- 若交易所在右侧：枚举前缀最大值等于该交易所权值的所有坐标 i ，这些坐标会分别从左侧的两个不同的区间受到不同的贡献，一种是左侧花费时间更长，一种是右侧花费时间更长。贡献分别是 $B(k - x) + f_x$ 和 $A(i - k) + f_x$ ，这两种贡献都可以用线段树维护区间最小值直接做。

至此，我们已经以 $O(n \log^2 n)$ 的时间复杂度解决了此问题，而要继续优化成 $O(n \log n)$ ，只需要把上面的线段树全部换成[猫树](#)即可。