冲刺 CCF NOI2025 全国赛模拟题

梦熊集训–Day5

时间：2025 年 07 月 10 日 08:00 ～ 13:00

题目名称	最大公约数	杏酥桐	矩形
题目类型	传统型	传统型	交互型
目录	`gcd`	`irris`	`rect`
可执行文件名	`gcd`	`irris`	`rect`
输入文件名	`gcd.in`	`irris.in`	`N/A`
输出文件名	`gcd.out`	`irris.out`	`N/A`
每个测试点时限	1.0 秒	4.0 秒	3.0 秒
内存限制	1024 MiB	1024 MiB	1024 MiB
测试点数目	20	25	10
测试点是否等分	是	是	是

提交源程序文件名

对于 C++ 语言	`gcd.cpp`	`irris.cpp`	`rect.cpp`

编译选项

对于 C++ 语言	`-O2 -std=c++14 -static`

注意事项（请仔细阅读）

1. 选手提交的源程序必须存放在以自己姓名命名的文件夹中，文件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，但你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的相邻整数均使用一个空格分隔。
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. 评测时采用的机器配置为：Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz，内存 32GB。上述时限以此配置为准。

最大公约数（gcd）

【题目描述】

对于一个正整数序列 $b_1, b_2 \cdots, b_m$ ，定义函数 $F(b)$ ：

- 定义一次操作为，选择序列中的两个相邻元素 b_i, b_{i+1} ，将它们替换成一个整数 $\gcd(b_i, b_{i+1})$ 。
- 将序列 b 的元素变为全部相同，所需的最少操作次数即为函数 $F(b)$ 的值。

给定一个长度为 n 的正整数序列 $a_1, a_2, \cdots, a_n$ ，你需要回答 q 次询问，每次询问给出两个数 l, r ，你需要计算 $F(a[l, r])$ 。

【输入格式】

从文件 ***gcd.in*** 中读入数据。
第一行两个正整数 n, q ，表示序列 a 的长度和询问次数。
第二行 n 个正整数表示序列 $a_1, a_2, \cdots, a_n$ 。
接下来 q 行，每行两个正整数 l, r 表示一次询问。

【输出格式】

输出到文件 ***gcd.out*** 中。
对于每次询问，输出一行一个整数表示答案。

【样例 1 输入】

```
8 3
12 18 36 24 42 60 54 9
1 5
3 7
1 8
```

【样例 1 输出】

```
3
3
7
```

【样例 1 解释】

对于第一组询问，可能的操作为：
$(12, 18, 36, 24, 42) \to (6, 36, 24, 42) \to (6, 12, 42) \to (6, 6)$

【样例 2】

见选手目录下的 ***gcd/gcd2.in*** 与 ***gcd/gcd2.ans***。

【样例 3】

见选手目录下的 ***gcd/gcd3.in*** 与 ***gcd/gcd3.ans***。

【数据范围】

对于所有数据，保证 $1 \le n \le 10^5, 1 \le q \le 3 \times 10^5, 1 \le a_i \le 10^5, 1 \le l \le r \le n$ 。

测试点编号	$n \le$	$q \le$	特殊性质
$1, 2$	15	120	无
3	300	300	A
4			B
5			无
6	2000	2000	A
7			B
8			无
9	60000	60000	A
10			B
$11, 12$			无
$13, 14$	10^5	10^5	A
$15, 16$			B
$17, 18$			无
$19, 20$		3×10^5	

特殊性质 A：满足 a_i 都是 2 的幂次；
特殊性质 B：满足 $a_i \le 36$。

杏酥桐（irris）

【题目描述】

Yuki 有一棵仅包含根结点 1 的有根树 T 和一个变量 n，初始时 $n=1$。
给定 q 次操作。操作有以下 2 种：

- 1 u_i x_i：在 u_i 的第 x_i 个儿子后插入结点 $n+1$；特殊地，若 $x_i=0$，则表示将结点 $n+1$ 作为 u_i 的第 1 个儿子插入。u_i 的其余儿子的相对顺序不变。设 u_i 的儿子个数为 s_{u_i}，则保证 $1\le u_i\le n$ 且 $0\le x_i\le s_{u_i}$。在执行此操作后 n 的值变为 $n+1$。
- 2 v_i k_i：查询对树 T 进行 k_i 次左儿子右兄弟变换后结点 v_i 的父亲结点。其中，左儿子右兄弟变换指：对于树 T 上的结点 u，将结点 u 在原树中的第一个儿子作为结点 u 在新树上的左儿子，将结点 u 在原树中的下一个兄弟作为结点 u 在新树上的右儿子。保证 $2\le v_i\le n$ 且 $1\le k_i\le 10^9$。**注意，此操作不会真的对树 T 进行 k_i 次左儿子右兄弟变换，也就是说在执行此操作后树形态不变。**

你需要对于每个 2 操作求出答案。

【输入格式】

从文件 ***irris.in*** 中读入数据。
本题有多组测试数据。
输入的第一行包含两个正整数 c,T，分别表示测试点编号和测试数据组数。样例满足 $c=0$。
接下来依次输入每组测试数据。对于每组测试数据：

- 第一行一个正整数 q。
- 接下来 q 行，第 i 行三个整数 o_i,u_i,x_i 或 o_i,v_i,k_i，格式同题目描述。

【输出格式】

输出到文件 ***irris.out*** 中。
对于每组测试数据中的每个 2 操作，输出一行一个整数表示答案。

【样例 1 输入】

```
0 2
8
1 1 0
1 2 0
```

```
2 3 1
1 3 0
1 1 0
1 4 0
1 4 1
2 7 1
8
1 1 0
2 2 2
2 2 2
1 2 0
2 3 1
1 3 0
1 4 0
2 4 3
```

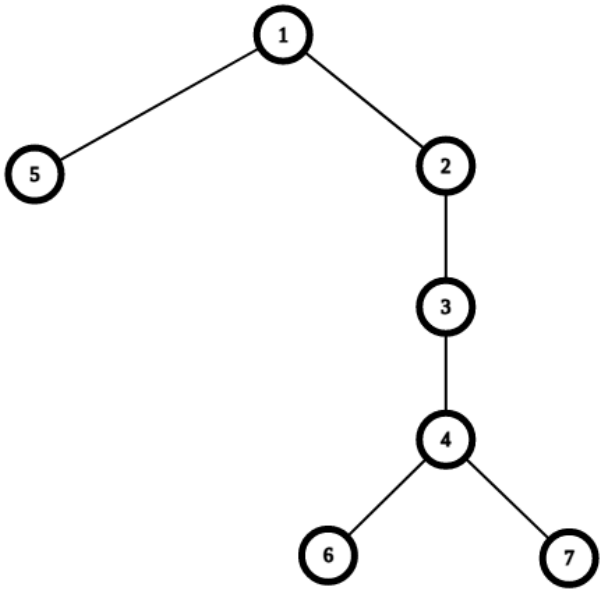
【样例 1 输出】

```
2
6
1
1
2
3
```

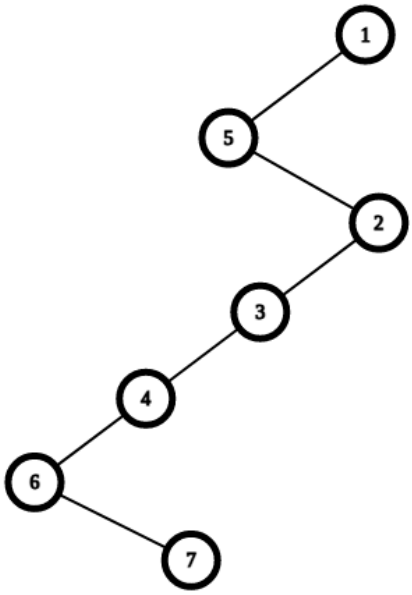
【样例 1 解释】

该样例包含两组测试数据，对于第一组测试数据：

- 第 1 次操作插入结点 2 作为结点 1 的儿子结点。
- 第 2 次操作插入结点 3 作为结点 2 的儿子结点。
- 此时树包含 2 条边 $(1,2),(2,3)$，经过 1 次左儿子右兄弟变换后，树仍为 $(1,2),(2,3)$，3 的父亲结点为 2。
- 接下来进行 4 次结点插入操作，操作结束后的树形如：



- 经过 1 次左儿子有兄弟变换后，树形如：



此时结点 7 的父亲结点为 6。

【样例 2】

见选手目录下的 ***irris/irris2.in*** 与 ***irris/irris2.ans***。

【样例 3】

见选手目录下的 ***irris/irris3.in*** 与 ***irris/irris3.ans***。

【样例 4】

见选手目录下的 ***irris/irris4.in*** 与 ***irris/irris4.ans***。

【样例 5】

见选手目录下的 ***irris/irris5.in*** 与 ***irris/irris5.ans***。

【样例 6】

见选手目录下的 ***irris/irris6.in*** 与 ***irris/irris6.ans***。

【样例 7】

见选手目录下的 ***irris/irris7.in*** 与 ***irris/irris7.ans***。

【数据范围】

对于所有测试数据，保证：$1 \le T \le 3$，$1 \le q \le 10^6$，$o_i \in \{1, 2\}$，$1 \le u_i \le n$，$0 \le x_i \le s_{u_i}$，$2 \le v_i \le n$，$1 \le k_i \le 10^9$。

测试点编号	$q \le$	k_i	特殊性质
$1, 2, 3$	10^2	$\le 10^2$	无
$4, 5$	3×10^3	$= 1$	
$6, 7$		$= 10^9$	
$8, 9, 10$		$\le 10^9$	
$11, 12$	5×10^5	$= 1$	
$13, 14$		$= 10^9$	
15		$\le 10^9$	A
$16, 17$			B
$18, 19$			C
$20, 21, 22$			无
$23, 24, 25$	10^6		

特殊性质 A：对于所有 1 操作，均有 $u_i = 1$。

特殊性质 B：对于所有满足 $1 \le i < j \le q$ 的正整数 i, j，均有 $op_i \le op_j$。

特殊性质 C：对于所有 1 操作，均有 $x_i = cnt_{u_i}$。

矩形（rect）

【题目描述】

这是一道交互题。

二维平面上初始有 n 个点 (i, p_i)，每个点有权值 d_i，$0 \le i < n$；

共 m 次操作，操作编号为 0 到 $m-1$，按编号升序执行；

编号为 w 的操作给出 x_1, x_2, y_1, y_2，满足 $0 < x1 < x2 < n$，$0 < y1 < y2 < n$。

一个点 (x, y) 的网格坐标被定义为 $((x \ge x1) + (x \ge x2), (y \ge y1) + (y \ge y2))$。

依次进行以下步骤：

1. 查询网格坐标为 (X, Y) 的点的权值之和，记录在 $ans[X][Y]$。
2. 对每个网格坐标为 (X, Y) 的点，进行修改 $o[X][Y]$。
3. 所有点的坐标同时发生改变，对于一个网格坐标为 (X, Y) 的点，它的坐标从 (x, y) 变为 $(x + dx[X], y + dy[Y])$；

其中

- $dx[0] = 0$，$dx[1] = n - x2$，$dx[2] = x1 - x2$。
- $dy[0] = 0$，$dy[1] = n - y2$，$dy[2] = y1 - y2$。
- $x1, x2, y1, y2, o, ans$ 都是数组，下标对应操作的编号。

其中 d_i 属于抽象的数据类型 D，$o[X][Y]$ 属于抽象的数据类型 O；

D 上定义了抽象的运算 $+ : D + D \to D$；

D, O 上定义了抽象的运算 $\cdot : O \cdot D \to D$；

O 上定义了抽象的运算 $\cdot : O \cdot O \to O$；

ϵ_D 是 D 中的一个特殊的元素，称为单位元；

ϵ_O 是 O 中的一个特殊的元素，称为单位元；

这些操作满足性质：

对任意 $a, b, c \in D$，有 $a + b = b + a$，$(a + b) + c = a + (b + c)$，$a + \epsilon_D = \epsilon_D + a = a$；

对任意 $u, v, w \in O$，有 $(u \cdot v) \cdot w = u \cdot (v \cdot w)$，$u \cdot \epsilon_O = \epsilon_O \cdot u = u$；

对任意 $u, v \in O$，$a, b \in D$，有 $(u \cdot v) \cdot a = u \cdot (v \cdot a)$，$u \cdot (a + b) = (u \cdot a) + (u \cdot b)$，$\epsilon_O \cdot a = a$，$u \cdot \epsilon_D = \epsilon_D$；

执行每次 $+$ 或 $\cdot$ 运算有一定的代价，具体地，在计算 $a + b$ 或 $a \cdot b$ 时如果 a, b 都不是 ϵ_D 或 ϵ_O，则代价为 1，否则代价为 0。你需要保证每个答案正确，且总代价不能超过当前子任务的**代价上限**。

【输入格式】

下发的交互库以如下格式读取输入数据：

- 第一行：n
- 接下来 n 行：p_i d_i（d_i 由两个整数表示）
- 第 $n+2$ 行：m
- 接下来 m 行：$x1_i$ $x2_i$ $y1_i$ $y2_i$ o_i（o_i 由 9 * 4 个整数表示）

D 中的元素是 2×1 的矩阵，O 中元素是 2×2 的矩阵，矩阵中的元素是对 2^{32} 取模的整数；

$+$ 对应矩阵加法，$\cdot$ 对应矩阵乘法，具体可以参考下发的交互库的实现；

实际评测环境中输入输出格式以及 D, O 等可能有不同的定义；

【输出格式】

下发的交互库以如下格式打印你的答案：

- 对每个询问，输出 13 行，其中第 1 , 2 , 3 , 5 , 6 , 7 , 9 , 10 , 11 行有两个整数，依次表示这次询问对应的 $ans[0][0], ans[0][1], ans[0][2], ans[1][0], ans[1][1], ans[1][2], ans[2][0], ans[2][1], ans[2][2]$，其余为空行；
- 向 stderr 打印总代价，以及在总代价超过代价上限时进行提示。

【实现细节】

你必须引用 `data.h` 头文件。

头文件中定义了数据类型 `Data(D)` 和 `Operation(O)`，你可以使用以下已定义的成员函数对类型为 `Data` 和 `Operation` 的数据进行操作：

```
1 void Data::add_eq(const Data &a)
```

`w.add(a)` 计算 $w+a$，并将结果保存在 w，每次调用的代价在 w, a 都不是单位元时为 1，否则为 0；

```
1 void Data::add(const Data &a,const Data &b)
```

`w.add(a,b)` 计算 $a+b$，并将结果保存在 w，每次调用的代价在 a, b 都不是单位元时为 1，否则为 0；

```
1 void Data::clr()
```

`w.clr()` 可以将 ϵ_D 保存在 w，每次调用的代价为 0；

```
1 bool Data::empty()const
```

`w.empty()` 判断 w 是否为 ϵ_D，若是则返回 true，否则返回 false，每次调用的代价为 0；

```
void Operation::apply(Data &a)const
```

`w.apply(a)` 计算 $w \cdot a$，并将结果保存在 a，每次调用的代价在 w, a 都不是单位元时为 1，否则为 0；

```
void Operation::apply(Operation &u)const
```

`w.apply(u)` 计算 $w \cdot u$，并将结果保存在 u，每次调用的代价在 w, u 都不是单位元时为 1，否则为 0；

```
void Operation::clr()
```

`w.print()` 可以将 ϵ_O 存储在 w，每次调用的代价为 0；

```
bool Operation::empty()const
```

`w.empty()` 判断 w 是否为 ϵ_O，若是则返回 true，否则返回 false，每次调用的代价为 0；

另外，你还可以使用 `Data` 或 `Operation` 类型的赋值运算符、拷贝构造函数或无参构造函数，以 `Data` 类型为例：

`w=u` 可以将 u 复制一份存储在 w，每次调用的代价为 0；

`Data w(u);` 或 `Data w=u;` 可以将 u 复制一份存储在新定义的 w，每次调用的代价为 0；

`Data w;` 可以将 ϵ_D 存储在新定义的 w，代价为 0；

`Operation w;` 可以将 ϵ_O 存储在新定义的 w，代价为 0；

除了以上描述的对 `Data` 或 `Operation` 的操作外，其余操作根据情况可能被视为攻击评测系统。

`sizeof(Data)` 和 `sizeof(Operation)` 都不超过 64，此外交互库还需要不超过 64MB 的空间。时间和空间限制包括交互库使用的时间和空间。仅使用赋值、构造函数、`apply` 和 `add` 就可以写出正确的程序，其它函数可能可以提供便利。

你需要实现以下函数：

```
void solve(
        const int n,
        const int m,
        const int p[],
        const Data d[],
        const int x1[],
        const int x2[],
        const int y1[],
        const int y2[],
```

```
10          const Operation o[][3][3],
11          Data ans[][3][3])
```

- n：点的个数；
- m：操作的个数；
- p：$(i, p[i])$ 表示每个点的坐标，$0 \le i < n$；
- d：$d[i]$ 表示每个点的初始权值，$0 \le i < n$；
- $x1, x2, y1, y2, o, ans$：$x1[i], y1[i], x2[i], y2[i], o[i]$ 表示每次操作的输入，你需要将相应的答案存储到 $ans[i]$，$0 \le i < m$。

【数据范围】

对于 100% 的数据，满足 $n \le 10^5, m \le 2 \times 10^4$。
共 10 组数据，满足 $n = 10^5$；
每组数据的 m 分别为 $10, 100, 1000, 2000, 5000, 10000, 12500, 15000, 17500, 20000$；
所有数据的代价上限为 10^8。
每组数据对应 10 分。