

鏖战表达式

【问题描述】

这是一道交互题。

我们需要处理这样一类表达式：它由 n 个元素和 $n - 1$ 个运算符构成，两个相邻元素之间有且仅有一个运算符，且表达式中没有括号。例如 $a + b \times c$ 就是一个这样的表达式。

这些运算符一共有 k 种，每种优先级都不同。为了方便，我们用 1 到 k 的整数来表示这些运算符，并且数字越大的优先级越高。

这些运算符都满足交换律和结合律，即对任意的元素 a, b, c 和运算符 $\sim$ ，满足

$$a \sim b = b \sim a$$

$$a \sim (b \sim c) = (a \sim b) \sim c$$

我们记一开始的表达式为第 0 个版本。

有 m 个操作，每个操作为以下几种之一：

1. 元素修改：对某一个版本，修改某个位置上的元素；
2. 运算符修改：对某一个版本，修改某两个元素之间的运算符；
3. 翻转：对某一个版本，将第 l 个元素到第 r 个元素之间的所有元素（包括第 l 个和第 r 个元素）和运算符翻转。

我们记第 i 次操作之后得到的表达式为第 i 个版本。

在每个操作之后，你需要求出当前表达式的值。

你只能通过调用一个函数 $F(a, b, op)$ 来得到 $a \ op \ b$ 的结果，且调用这个函数的次数有一定限制。

【任务描述】

你需要实现以下几个函数：

- $init(test_id, n, m, k, a, ops)$

我们首先会调用这个函数。其中：

- $test_id$ 为测试点编号， n, m, k 意义见题目描述；
- a 为一个大小为 n 的数组， $a[i]$ 表示最初表达式里的第 i 个元素的值（下标从 0 开始）；
- ops 为一个大小为 n 的数组， $ops[i]$ 表示第 i 个元素与第 $i - 1$ 个元素之间的运算符；
- 注意 $ops[0]$ 没有意义，请不要尝试去访问它。

- `modify_data(id, pos, x)`
元素修改操作：对第 id 个版本，修改第 pos ($0 \leq pos < n$) 个元素为 x ，并将修改后的表达式作为一个新的版本。你需要返回修改后表达式的值。
- `modify_op(id, pos, new_op)`
运算符修改操作：对第 id 个版本，修改第 pos ($1 \leq pos < n$) 和第 $pos - 1$ 个元素之间的运算符为 new_op ，并将修改后的表达式作为一个新的版本。你需要返回修改后表达式的值。
- `reverse(id, l, r)`
翻转操作：对第 id 个版本，将第 l 个元素到第 r 个元素之间的所有元素（包括第 l 个和第 r 个元素， $0 \leq l \leq r < n$ ）和运算符翻转，并将修改后的表达式作为一个新的版本。你需要返回修改后表达式的值。

其中表达式里的元素（`init` 函数中 a 数组里的元素，`modify_data` 中的 x ，和每次返回的表达式值）均为 `Data` 类型，这种类型在交互库里提供了定义。

你可以调用一个函数 F 来得到两个元素做某个运算的结果：

- $F(a, b, op)$
返回将 a, b 两个元素做运算 op ($1 \leq op \leq k$) 之后的值。

请注意：`Data` 类型中的变量 x 只对交互库有意义，你不需要也不应该访问这个变量。另外请保证传入函数 F 的 a, b 和三种操作函数的返回值为 `init`、`modify_data` 或 F 中提供的元素，否则，在使用下发的交互库进行测试时，会发生未知行为（如返回错误的结果，或运行错误等），在最终评测时，该测试点会被判为 0 分。

【实现细节】

你需要且只能提交一个源文件 `expr.cpp/c/pas` 实现上述函数，并遵循下面的命名和接口。

对 C/C++ 语言的选手：

你需要包含头文件 `expr.h`。

`Data` 类型的定义：

```
typedef struct {
    int x;
} Data;
```

最终评测时，`Data` 类型的定义与题面中给出的一样。

你需要实现的函数：

```
void init(  
    int test_id, int n, int m, int k,  
    const Data *a, const int *ops  
);  
Data modify_data(int id, int pos, Data x);  
Data modify_op(int id, int pos, int new_op);  
Data reverse(int id, int l, int r);
```

函数 F 的接口信息如下：

```
Data F(Data a, Data b, int op);
```

你需要在本题目录下使用如下命令编译得到可执行程序：

对于 C 语言

```
gcc grader.c expr.c -o expr -O2 -lm
```

对于 C++语言

```
g++ grader.cpp expr.cpp -o expr -O2 -lm
```

对 **Pascal** 语言的选手：

你需要使用单元 *graderhelperlib*。

Data 类型的定义：

```
type Data = record  
    x : longint;  
end;
```

最终评测时，*Data* 类型的定义与题面中给出的一样。

你需要实现的函数：

```
procedure init(  
    test_id, n, m, k : longint;  
    const a : array of Data;  
    const ops : array of longint  
);  
function modify_data(  
    id, pos : longint;  
    x : Data  
): Data;  
function modify_op(  
    id, pos, new_op : longint  
): Data;  
function reverse(id, l, r : longint) : Data;
```

函数 F 的接口信息如下：

```
function F(
    a, b : Data;
    op : longint
) : Data;
```

你需要在本题目录下使用如下命令编译得到可执行程序：

```
fpc grader.pas -o"expr" -O2
```

【如何开始答题】

本题目录下，有针对每种语言的样例源代码 *expr_sample.cpp/c/pas*，选择你所需的语言，将其复制为 *expr.cpp/c/pas*，按照本节前文中提到的方式进行编译，即能通过编译得到可执行程序。

注意：你只能选择一种语言进行作答，即你本题的试题目录下不能同时存在多个语言的 *expr.cpp/c/pas*。

接下来你需要修改这个文件的实现，以达到题目的要求。

【如何测试你的程序】

交互库将从文件 *expr.in* 读入以下格式的数据：

第 1 行包含 4 个整数 $test_id, n, m, k$ ，需保证 n, m 不超过 20000， k 不超过 100。

第 2 行仅包含 1 个整数 lim ，表示 F 函数的调用次数的限制，需保证 lim 不超过 10^7 。

第 3 行包含 $n - 1$ 个整数，第 i ($1 \leq i < n$) 个整数表示第 i 个运算符。

接下来 m 行，每行先是一个整数 id ，然后是对第 id 个版本的一个操作。

接下来 2 或 3 个整数，描述一个操作。每个操作必须为下列之一：

- 1 pos
修改第 pos ($0 \leq pos < n$) 个元素
- 2 pos new_op
修改第 pos ($1 \leq pos < n$) 和第 $pos - 1$ 个元素之间的运算符为 new_op 。
- 3 l r
将第 l 个元素到第 r 个元素之间的所有元素（包括第 l 个和第 r 个元素）和运算符翻转。

读入完成之后，交互库将调用 *init* 函数。然后再根据数据调用 m 次 *modify_data*, *modify_op* 或 *reverse* 函数。最后交互库将会用某种（选手们

不必知道的) 方式来计算你的所有函数返回值的校验和, 并输出到文件 *expr.out* 中。交互库还会输出你调用 *F* 函数的次数。

如果传入 *F* 函数的参数非法 (*op* 不在 1 到 *k* 的范围内, 或 *a, b* 不是由交互库提供的元素), 那么交互库会将校验和输出为非法值 -1 。(因此该测试点得 0 分), 然后在下面一行输出错误的详细信息。

如果要使用自己的输入文件进行测试, 请保证输入文件按照以上格式, 否则不保证程序能正确运行。

【评分方法】

最终评测时只会收取 *expr.cpp/c/pas*, 修改本题目录中的其他文件对评测无效。

题目首先会受到和传统题相同的限制。例如编译错误会导致整道题目得 0 分, 运行错误、超过时间限制、超过空间限制等会导致相应测试点得 0 分等。

你只能访问自己定义的和交互库给出的变量及其对应的内存空间, 尝试访问其他空间将可能导致编译错误或运行错误。

若程序正常结束, 则会开始检验正确性。如果答案不正确, 或 *F* 函数的调用次数超过了测试点的限制, 该测试点得 0 分。否则该测试点得满分。

题目中所给的时间、空间限制为你的代码和交互库加起来可以使用的时间和空间。我们保证, 对于任何合法的数据, 任何语言任何版本的交互库 (包括下发给选手的和最终评测时用的), 正常运行所用的时间不会超过 0.2s, 正常运行所用的空间不会超过 64MB, 也就是说, 选手实际可用的时间至少为 0.8s, 实际可用的空间至少为 960MB。

【样例输入 1】

```
0 5 5 2
1000
2 1 1 1
0 1 1
1 2 2 2
2 3 1 3
0 3 0 3
0 3 1 1
```

【样例输出 1】

```
120400404
```

【样例解释 1】

这是使用试题目录中的 *grader* 和正确的源程序得到的输出中的校验和。若你的程序得到了不同的校验和，那么你的程序在本样例下是错误的。

我们用 $+$ 和 $\times$ 表示这两种运算符，用小写字母表示元素，则每一个版本的表达式为如下形式：

- 第 0 个版本： $a \times b + c + d + e$
- 第 1 个版本： $a \times f + c + d + e$
- 第 2 个版本： $a \times f \times c + d + e$
- 第 3 个版本： $a \times d + c \times f + e$
- 第 4 个版本： $d + c + b \times a + e$
- 第 5 个版本： $a \times b + c + d + e$

【样例输入输出 2】

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

【数据规模和约定】

各测试点满足以下约定：

$test_id$	$n =$	$m =$	$k =$	$lim =$	其他约定
1	10	10	1	10^7	无
2	1000	1000	5	10^7	
3	10000	10000	1	10^7	没有 <i>reverse</i> 操作， 第 i 个操作的 id 为 $i - 1$
4	20000	20000	1	10^7	
5	10000	10000	1	10^7	第 i 个操作的 id 为 $i - 1$
6	20000	20000	1	10^7	
7	10000	10000	1	10^7	无
8	20000	20000	1	10^7	
9	15000	15000	3	10^7	第 i 个操作的 id 为 $i - 1$
10	15000	15000	5	10^7	
11	15000	15000	3	10^7	无
12	15000	15000	5	10^7	
13	15000	15000	50	10^7	第 i 个操作的 id 为 $i - 1$
14	20000	20000	50	10^7	
15	20000	20000	100	10^7	
16	20000	20000	100	10^7	无
17	15000	15000	50	10^7	
18	20000	20000	50	10^7	
19	20000	20000	100	10^7	
20	20000	20000	100	10^7	