梯度求解

时间限制： 1.0 秒

空间限制： 512 MiB

刷新

下载题目目录（样例文件） (/staticdata/down/CSP202309-3.zip)

题目背景

西西艾弗岛运营公司近期在大力推广智能化市政管理系统。这套系统是由西西艾弗岛信息中心研发的。它的主要目的是，通过详细评估岛上各处的市政设施的状况，来指导市政设施的维护和更新。这套系统的核心是一套智能化的传感器网络，它能够自动地对岛上的市政设施进行评估。对市政设施的维护是需要一定成本的，而年久失修的市政设施也可能给岛上的居民造成损失。为了能够平衡成本和收益，信息中心研发了一款数学模型，描述这些变量和损益之间的复杂数学关系。要想得到最优化的成本，就要依靠梯度下降算法来求解。

梯度下降算法中，求解函数在一点处对某一自变量的偏导数是十分重要的。小 C 负责实现这个功能，但是具体的技术实现，他还是一头雾水，希望你来帮助他完成这个任务。

题目描述

设被求算的函数 $u = f(x_1, x_2, \ldots, x_n)$，本题目要求你求出 u 对 x_i 在 $(a_1, a_2, \ldots, a_n)$ 处的偏导数 $\frac{\partial u}{\partial x_i}(a_1, a_2, \ldots, a_n)$。

求算多元函数在一点处对某一自变量的偏导数的方法是：将函数的该自变量视为单一自变量，其余自变量认为是常数，运用一元函数求导的方法求出该偏导数表达式，再代入被求算的点的坐标即可。

例如，要求算 $u = x_1 \cdot x_1 \cdot x_2$ 对 x_1 在 $(1, 2)$ 处的偏导数，可以将 x_2 视为常数，依次应用求导公式。先应用乘法的求导公式：$(x_1 \cdot (x_1 \cdot x_2))' = x_1'(x_1 \cdot x_2) + x_1(x_1 \cdot x_2)'$；再应用常数与变量相乘的求导公式，得到 $x_1' \cdot x_1 \cdot x_2 + x_1 \cdot x_2 \cdot x_1'$；最后应用公式 $x' = 1$ 得到 $1 \cdot x_1 \cdot x_2 + x_1 \cdot x_2 \cdot 1$。整理得 $\frac{\partial u}{\partial x_1} = 2x_2 \cdot x_1$。再代入 $(1, 2)$ 得到 $\frac{\partial u}{\partial x_1}(1, 2) = 4$。

常见的求导公式有：

- $c' = 0$ （c是常数）
- $x' = 1$
- $(u + v)' = u' + v'$
- $(cu)' = cu'$ （c是常数）
- $(u - v)' = u' - v'$
- $(uv)' = u'v + uv'$

本题目中，你需要求解的函数 f 仅由常数、自变量和它们的加法、减法、乘法组成。且为程序识读方便，函数表达式已经被整理为逆波兰式（后缀表达式）的形式。例如，$x_1 \cdot x_1 \cdot x_2$ 的逆波兰式为 `x1 x1 * x2 *`。逆波兰式即为表达式树的后序遍历的结果。若要从逆波兰式还原原始计算算式，可以按照这一方法进行：假设存在一个空栈 S，依次读取逆波兰式的每一个元素，若读取到的是变量或常量，则将其压入 S

中；若读取到的是计算符号，则从 S 中取出两个元素，进行相应运算，再将结果压入 S 中。最后，若 S 中存在唯一的元素，则该表达式合法，其值即为该元素的值。例如对于逆波兰式 `x1 x1 * x2 *`，按上述方法读取，栈 S 的变化情况依次为（左侧是栈底，右侧是栈顶）：

1. x_1；
2. x_1，x_1；
3. $(x_1 \cdot x_1)$；
4. $(x_1 \cdot x_1)$，x_2；
5. $((x_1 \cdot x_1) \cdot x_2)$。

输入格式

从标准输入读入数据。

输入的第一行是由空格分隔的两个正整数 n、m，分别表示要求解函数中所含自变量的个数和要求解的偏导数的个数。

输入的第二行是一个逆波兰式，表示要求解的函数 f。其中，每个元素用一个空格分隔，每个元素可能是：

- 一个自变量 x_i，用字符 `x` 后接一个正整数表示，表示第 i 个自变量，其中 $i = 1, 2, \ldots, n$。例如，`x1` 表示第一个自变量 x_1。
- 一个整常数，用十进制整数表示，其值在 -10^5 到 10^5 之间。
- 一个运算符，用 `+` 表示加法，`-` 表示减法，`*` 表示乘法。

输入的第三行到第 $m + 2$ 行，每行有 $n + 1$ 个用空格分隔的整数。其中第一个整数是要求偏导数的自变量的编号 $i = 1, 2, \ldots, n$，随后的整数是要求算的点的坐标 $a_1, a_2, \ldots, a_n$。 输入数据保证，对于所有的 $i = 1, 2, \ldots, n$，a_i 都在 -10^5 到 10^5 之间。

输出格式

输出到标准输出。

输出 m 行，每行一个整数，表示对应的偏导数对 $10^9 + 7$ 取模的结果。即若结果为 y，输出为 k，则保证存在整数 t，满足 $y = k + t \cdot (10^9 + 7)$ 且 $0 \le k < 10^9 + 7$。

样例1输入

```
2 2
x1 x1 x1 * x2 + *
1 2 3
2 3 4
```

样例1输出

```
15
3
```

样例1解释

读取逆波兰式，可得被求导的式子是：$u = x_1 \cdot (x_1 \cdot x_1 + x_2)$，即 $u = x_1^3 + x_1 x_2$。

对 x_1 求偏导得 $\frac{\partial u}{\partial x_1} = 3x_1^2 + x_2$。代入 $(2,3)$ 得到 $\frac{\partial u}{\partial x_1}(2,3) = 15$。

对 x_2 求偏导得 $\frac{\partial u}{\partial x_2} = x_1$。代入 $(3,4)$ 得到 $\frac{\partial u}{\partial x_2}(3,4) = 3$。

样例2输入

```
3 5
x2 x2 * x2 * 0 + -100000 -100000 * x2 * -
3 100000 100000 100000
2 0 0 0
2 0 -1 0
2 0 1 0
2 0 100000 0
```

样例2输出

```
0
70
73
73
999999867
```

样例2解释

读取逆波兰式，可得被求导的式子是：$u = x_2 \cdot x_2 \cdot x_2 + 0 - (-10^5) \cdot (-10^5) \cdot x_2$，即 $u = x_2^3 - 10^{10} x_2$。

因为 u 中实际上不含 x_1 和 x_3，对这两者求偏导结果均为 0。

对 x_2 求偏导得 $\frac{\partial u}{\partial x_2} = 3x_2^2 - 10^{10}$。

子任务

测试点	n	m	**表达式的性质**
1, 2	$= 1$	≤ 100	仅含有 1 个元素
3, 4			仅含有一个运算符
5, 6	≤ 10		含有不超过 120 个元素，且不含乘法
7, 8			含有不超过 120 个元素
9, 10	≤ 100		

提示

C++ 中可以使用 `std::getline(std::cin, str)` 读入字符串直到行尾。

当计算整数 n 对 M 的模时，若 n 为负数，需要注意将结果调整至区间 $[0, M)$ 内。

语言和编译选项

#	名称	编译器	额外参数	代码长度限制
0	g++	`g++`	`-O2 -DONLINE_JUDGE`	65536 B
1	gcc	`gcc`	`-O2 -DONLINE_JUDGE`	65536 B
2	java	`javac`		65536 B
3	python3	`python3`		65536 B

递交历史

#	状态	时间

当前没有提交权限，请返回认证首页 (/contest/31/home)检查是否已开启模拟认证 或 可以进行自由练习。