第 32 届全国信息学奥林匹克竞赛

CCF NOI 2015

第一试

竞赛时间：2015 年 7 月 17 日 8:00-13:00

题目名称	程序自动分析	软件包管理器	寿司晚宴
目录	prog	manager	dinner
可执行文件名	prog	manager	dinner
输入文件名	prog.in	manager.in	dinner.in
输出文件名	prog.out	manager.out	dinner.out
每个测试点时限	2 秒	1 秒	1 秒
内存限制	512MB	512MB	512MB
测试点数目	10	20	10
每个测试点分值	10	5	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	否

提交源程序须加后缀

对于 Pascal 语言	prog.pas	manager.pas	dinner.pas
对于 C　　语言	prog.c	manager.c	dinner.c
对于 C++　语言	prog.cpp	manager.cpp	dinner.cpp

注意：最终测试时，所有编译命令均不打开任何优化开关。

程序自动分析

【问题描述】

在实现程序自动分析的过程中，常常需要判定一些约束条件是否能被同时满足。

考虑一个约束满足问题的简化版本：假设 $x_1, x_2, x_3, \cdots$ 代表程序中出现的变量，给定 n 个形如 $x_i = x_j$ 或 $x_i \neq x_j$ 的变量相等/不等的约束条件，请判定是否可以分别为每一个变量赋予恰当的值，使得上述所有约束条件同时被满足。例如，一个问题中的约束条件为：$x_1 = x_2, x_2 = x_3, x_3 = x_4, x_1 \neq x_4$，这些约束条件显然是不可能同时被满足的，因此这个问题应判定为不可被满足。

现在给出一些约束满足问题，请分别对它们进行判定。

【输入格式】

从文件 ***prog.in*** 中读入数据。

输入文件的第 1 行包含 1 个正整数 t，表示需要判定的问题个数。**注意这些问题之间是相互独立的。**

对于每个问题，包含若干行：

第 1 行包含 1 个正整数 n，表示该问题中需要被满足的约束条件个数。

接下来 n 行，每行包括 3 个整数 i, j, e，描述 1 个相等/不等的约束条件，相邻整数之间用单个空格隔开。若 $e = 1$，则该约束条件为 $x_i = x_j$；若 $e = 0$，则该约束条件为 $x_i \neq x_j$；

【输出格式】

输出到文件 ***prog.out*** 中。

输出文件包括 t 行。

输出文件的第 k 行输出一个字符串“YES”或者“NO”**（不包含引号，字母全部大写）**，“YES”表示输入中的第 k 个问题判定为可以被满足，“NO”表示不可被满足。

【样例输入 1】

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

【样例输出 1】

```
NO
YES
```

【样例说明 1】

在第一个问题中，约束条件为：$x_1 = x_2, x_1 \neq x_2$。这两个约束条件互相矛盾，因此不可被同时满足。

在第二个问题中，约束条件为：$x_1 = x_2, x_2 = x_1$。这两个约束条件是等价的，可以被同时满足。

【样例输入 2】

```
2
3
1 2 1
2 3 1
3 1 1
4
1 2 1
2 3 1
3 4 1
1 4 0
```

【样例输出 2】

```
YES
NO
```

【样例说明 2】

在第一个问题中，约束条件有三个：$x_1 = x_2, x_2 = x_3, x_3 = x_1$。只需赋值使得 $x_1 = x_2 = x_3$，即可同时满足所有的约束条件。

在第二个问题中，约束条件有四个：$x_1 = x_2, x_2 = x_3, x_3 = x_4, x_1 \neq x_4$。由前三个约束条件可以推出 $x_1 = x_2 = x_3 = x_4$，然而最后一个约束条件却要求 $x_1 \neq x_4$，因此不可被满足。

【样例输入输出 3】

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

【数据规模与约定】

所有测试数据的范围和特点如下表所示

测试点编号	n 的规模	i,j 的规模	约定
1	$1 \le n \le 10$	$1 \le i,j \le 10{,}000$	$1 \le t \le 10$ $e \in \{0,1\}$
2			
3	$1 \le n \le 100$		
4			
5	$1 \le n \le 100{,}000$		
6			
7			
8	$1 \le n \le 100{,}000$	$1 \le i,j \le 1{,}000{,}000{,}000$	
9			
10			

软件包管理器

【问题描述】

Linux 用户和 OS X 用户一定对软件包管理器不会陌生。通过软件包管理器，你可以通过一行命令安装某一个软件包，然后软件包管理器会帮助你从软件源下载软件包，同时自动解决所有的依赖（即下载安装这个软件包的安装所依赖的其它软件包），完成所有的配置。Debian/Ubuntu 使用的 apt-get，Fedora/CentOS 使用的 yum，以及 OS X 下可用的 homebrew 都是优秀的软件包管理器。

你决定设计你自己的软件包管理器。不可避免地，你要解决软件包之间的依赖问题。如果软件包 A 依赖软件包 B，那么安装软件包 A 以前，必须先安装软件包 B。同时，如果想要卸载软件包 B，则必须卸载软件包 A。现在你已经获得了所有的软件包之间的依赖关系。而且，由于你之前的工作，除 0 号软件包以外，在你的管理器当中的软件包都会依赖一个且仅一个软件包，而 0 号软件包不依赖任何一个软件包。依赖关系不存在环（若有 $m\ (m \geq 2)$ 个软件包 $A_1, A_2, A_3, \cdots, A_m$，其中 A_1 依赖 A_2，A_2 依赖 A_3，A_3 依赖 A_4，……，A_{m-1} 依赖 A_m，而 A_m 依赖 A_1，则称这 m 个软件包的依赖关系构成环），当然也不会有一个软件包依赖自己。

现在你要为你的软件包管理器写一个依赖解决程序。根据反馈，用户希望在安装和卸载某个软件包时，快速地知道这个操作实际上会改变多少个软件包的安装状态（即安装操作会安装多少个未安装的软件包，或卸载操作会卸载多少个已安装的软件包），你的任务就是实现这个部分。**注意，安装一个已安装的软件包，或卸载一个未安装的软件包，都不会改变任何软件包的安装状态，即在此情况下，改变安装状态的软件包数为 0。**

【输入格式】

从文件 ***manager.in*** 中读入数据。

输入文件的第 1 行包含 1 个整数 n，表示软件包的总数。软件包从 0 开始编号。

随后一行包含 $n-1$ 个整数，相邻整数之间用单个空格隔开，分别表示 $1, 2, 3, \cdots, n-2, n-1$ 号软件包依赖的软件包的编号。

接下来一行包含 1 个整数 q，表示询问的总数。

之后 q 行，每行 1 个询问。询问分为两种：

- `install x`：表示安装软件包 x
- `uninstall x`：表示卸载软件包 x

你需要维护每个软件包的安装状态，一开始所有的软件包都处于未安装状态。对于每个操作，你需要输出这步操作会改变多少个软件包的安装状态，随后应用这个操作（即改变你维护的安装状态）。

【输出格式】

输出到文件 ***manager.out*** 中。

输出文件包括 q 行。

输出文件的第 i 行输出 1 个整数，为第 i 步操作中改变安装状态的软件包数。

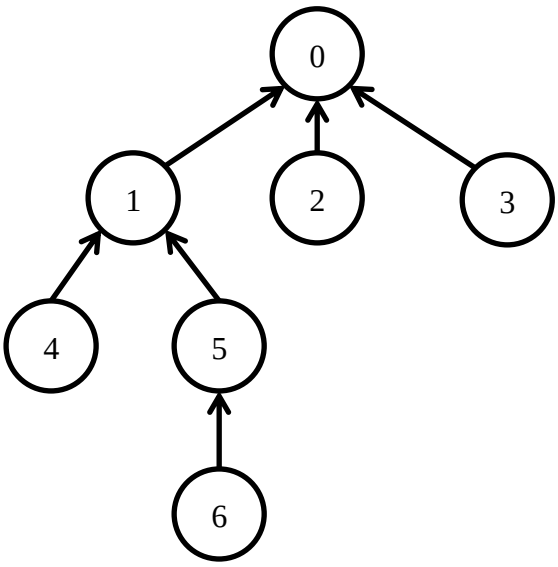
【样例输入 1】

```
7
0 0 0 1 1 5
5
install 5
install 6
uninstall 1
install 4
uninstall 0
```

【样例输出 1】

```
3
1
3
2
3
```

【样例说明 1】



一开始所有的软件包都处于未安装状态。

安装 5 号软件包，需要安装 $0, 1, 5$ 三个软件包。

之后安装 6 号软件包，只需要安装 6 号软件包。此时安装了 $0, 1, 5, 6$ 四个软件包。

卸载 1 号软件包需要卸载 $1, 5, 6$ 三个软件包。此时只有 0 号软件包还处于安装状态。

之后安装 4 号软件包，需要安装 1, 4 两个软件包。此时 0, 1, 4 处在安装状态。最后，卸载 0 号软件包会卸载所有的软件包。

【样例输入 2】

```
10
0 1 2 1 3 0 0 3 2
10
install 0
install 3
uninstall 2
install 7
install 5
install 9
uninstall 9
install 4
install 1
install 9
```

【样例输出 2】

```
1
3
2
1
3
1
1
1
0
1
```

【样例输入输出 3】

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

【数据规模与约定】

测试点编号	n 的规模	q 的规模	备注
1	$n = 5{,}000$	$q = 5{,}000$	
2			
3	$n = 100{,}000$	$q = 100{,}000$	数据不包含卸载操作
4			
5	$n = 100{,}000$	$q = 100{,}000$	编号为 i 的软件包所依赖的软件包编号在 $[0, i-1]$ 内均匀随机 每次执行操作的软件包编号在 $[0, n-1]$ 内均匀随机
6			
7			
8			
9	$n = 100{,}000$	$q = 100{,}000$	
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			

寿司晚宴

【问题描述】

为了庆祝 NOI 的成功开幕，主办方为大家准备了一场寿司晚宴。小 G 和小 W 作为参加 NOI 的选手，也被邀请参加了寿司晚宴。

在晚宴上，主办方为大家提供了 $n-1$ 种不同的寿司，编号 $1,2,3,\cdots,n-1$，其中第 i 种寿司的美味度为 $i+1$ （即寿司的美味度为从 2 到 n）。

现在小 G 和小 W 希望每人选一些寿司种类来品尝，他们规定一种品尝方案为不和谐的当且仅当：**小 G 品尝的寿司种类中存在一种美味度为 x 的寿司，小 W 品尝的寿司中存在一种美味度为 y 的寿司，而 x 与 y 不互质**。

现在小 G 和小 W 希望统计一共有多少种和谐的品尝寿司的方案（对给定的正整数 p 取模）。注意一个人可以不吃任何寿司。

【输入格式】

从文件 ***dinner.in*** 中读入数据。

输入文件的第 1 行包含 2 个正整数 n,p，中间用单个空格隔开，表示共有 n 种寿司，最终和谐的方案数要对 p 取模。

【输出格式】

输出到文件 ***dinner.out*** 中。

输出一行包含 1 个整数，表示所求的方案模 p 的结果。

【样例输入 1】

```
3 10000
```

【样例输出 1】

```
9
```

【样例输入 2】

```
4 10000
```

【样例输出 2】

```
21
```

【样例输入 3】

```
100 100000000
```

【样例输出 3】

```
3107203
```

【数据规模与约定】

测试点编号	n 的规模	约定
1	$2 \leq n \leq 30$	$0 < p \leq 10,000,000,000$
2		
3		
4	$2 \leq n \leq 100$	
5		
6	$2 \leq n \leq 200$	
7		
8	$2 \leq n \leq 500$	
9		
10		