2016 年清华大学计算机系推研

上机测试

时间：2016 年 9 月 19 日 17:30 ~ 21:30

题目名称	众数	文件系统	表达式求值	矩阵树	距离
题目类型	传统型	传统型	传统型	传统型	传统型
输入	标准输入	标准输入	标准输入	标准输入	标准输入
输出	标准输出	标准输出	标准输出	标准输出	标准输出
每个测试点时限	1 秒	1 秒	1 秒	1 秒	1 秒
内存限制	512 MB	512 MB	512 MB	512 MB	512 MB
测试点/包数目	10	10	10	10	10
测试点是否等分	是	是	是	是	是

众数（modevalue）

【题目描述】

输入 n 个数，问哪个数出现的次数最多。如果有多个出现次数最多的，输出最大的那个数。

【输入格式】

从标准输入读入数据。
第一行有一个正整数 n。
第二行有 n 个正整数，表示每一个数，相邻两个数之间用一个空格隔开。
保证 n 和这 n 个正整数都不会超过 2048。

【输出格式】

输出到标准输出。
只输出一个数即答案。

【样例输入】

```
7
2 3 6 2 3 6 7
```

【样例输出】

```
6
```

文件系统（filesystem）

【题目描述】

B 君在设计一个文件系统。

B 君在这个文件系统里，有 n 个文件夹。B 君想知道表示这 n 个文件夹所在路径的字符串，长度之和是多少。

一个文件夹的路径是，自己和所有祖先文件夹名字，以/分割连接起来，其中最前的一个文件夹前加/，最后一个文件夹后不加/。

比如对于样例，它的文件结构为

```
Users
    wwwwodddd
        Documents
        Downloads
System
```

5 个文件夹的的路径分别为（按照输入的顺序）

```
/Users
/Users/wwwwodddd
/Users/wwwwodddd/Documents
/System
/Users/wwwwodddd/Downloads
```

他们的长度分别是 $6, 16, 26, 7, 26$，所以所有长度和为 81。

特别注意，根目录/不是一个文件夹，他的路径长度不应被计算在最终答案中。每个字符串开头的/不是根目录的意思，这里的/应该被计入答案。

【输入格式】

从标准输入读入数据。

输入第一行一个正整数 n，表示一共有 n 个文件夹。保证 $1 \le n \le 1000$。

以下 n 行每行描述一个文件夹，第 i（$1 \le i \le n$）行描述第 i 个文件夹，有一个整数 f_i 和一个字符串 s_i。

f_i 表示第 i 个文件夹的父文件夹是第 f_i 个文件夹，特别的，如果 f_i 为 0，那么说明这个文件夹在根目录。保证 $0 \le f_i < i$。

s_i 表示第 i 个文件夹的名字，这个名字一定由数字和大小写字母组成。保证在同一个文件夹下不存在多个文件夹同名，保证每个文件夹的长度至多为 12。

【输出格式】

输出到标准输出。
输出一行一个整数，表示所有路径长度的和。

【样例输入】

```
5
0 Users
1 wwwwodddd
2 Documents
0 System
2 Downloads
```

【样例输出】

```
81
```

表达式求值（eval）

【题意简述】

在本题当中，你需要设法描述一个表达式的计算过程。这个表达式可能含有四则运算（加、减、乘、除，不会出现负号）、括号、函数调用（普通函数调用和成员函数调用），和一些常量。

例如，`(a+f((b-c+e)*d/c.h(d,d)).g(e)).g(d).h(f(a,c),f(b)/f(c),f(d))` 就是一个可能出现的表达式，其中 `a`,`b`,`c`,`d`,`e` 是常量，`f` 是普通函数，`g`,`h` 是成员函数。

运算的优先级为：括号 > 函数调用 > 乘除法 > 加减法。注意运算的优先级的大小不代表计算顺序的先后。

你不需要求出这个表达式的具体值，你只需要描述它的计算过程，即将这个表达式分解为若干次四则运算与函数调用，且每次运算或调用的操作数都是常量或之前的运算或调用的结果。

你可能需要一些编译原理的相关知识来解决这道题目。

【表达式的定义】

首先，一个常量、普通函数、成员函数只可能是一个小写英文字母。在任何一个表达式中，一个字母至多只可能是常量、普通函数、成员函数的一种。

我们用递归的方式定义合法的表达式：

- 单独的一个常量是一个合法的表达式。
- 如果 `[EXPR]` 是一个合法的表达式，那么 `([EXPR])` 是一个合法的表达式，其值与 `[EXPR]` 相同。
- 如果 `[EXPR_1]`、`[EXPR_2]`、……、`[EXPR_n]` 是 n 个合法的表达式（n 至少为 1），`[FUNC]` 是一个普通函数，那么 `[FUNC]([EXPR_1],[EXPR_2],……,[EXPR_n])` 是一个合法的表达式，其值为将 `[EXPR_1]`、`[EXPR_2]`、……、`[EXPR_n]` 依次作为参数，调用普通函数 `[FUNC]` 所得的结果。注意同一个函数在不同的调用中可能接受不同个数的参数。
- 如果 `[EXPR]` 是一个上述三条规则或本条规则或本条规则定义的一个合法的表达式，`[EXPR_1]`、`[EXPR_2]`、……、`[EXPR_n]` 是 n 个合法的表达式（n 至少为 1），`[FUNC]` 是一个成员函数，那么 `[EXPR].[FUNC]([EXPR_1],[EXPR_2],……,[EXPR_n])` 是一个合法的表达式，其值为将 `[EXPR_1]`、`[EXPR_2]`、……、`[EXPR_n]` 依次作为参数，调用 `[EXPR]` 的成员函数 `[FUNC]` 所得的结果。注意同一个函数在不同的调用中可能接受不同个数的参数。
- 如果 `[EXPR_0]`、`[EXPR_1]`、……、`[EXPR_n]` 是 $n+1$ 个上述四条规则定义的合法的表达式（n 至少为 1），`[OPR_1]`、`[OPR_2]`、……、`[OPR_n]` 是 n 个乘号或除号运算符，那么 `[EXPR_0][OPR_1][EXPR_1][OPR_2]……[OPR_n][EXPR_n]`

是一个合法的表达式，其值为将 [EXPR_0]、[EXPR_1]、……、[EXPR_n] 依次进行 [OPR_1]、[OPR_2]、……、[OPR_n] 运算的结果。

- 如果 [EXPR_0]、[EXPR_1]、……、[EXPR_n] 是 $n+1$ 个上述五条规则定义的合法的表达式（n 至少为 1），[OPR_1]、[OPR_2]、……、[OPR_n] 是 n 个加号或减号运算符，那么 [EXPR_0][OPR_1][EXPR_1][OPR_2]……[OPR_n][EXPR_n] 是一个合法的表达式，其值为将 [EXPR_0]、[EXPR_1]、……、[EXPR_n] 依次进行 [OPR_1]、[OPR_2]、……、[OPR_n] 运算的结果。
- 只有符合以上几条定义的表达式才是合法的。

容易看到，这样定义的每个合法的表达式都有唯一一种解读方式，即不会引起歧义。

【计算顺序的规定】

上述规定确定了一个表达式的值，接下来我们确定一个表达式的求值顺序。我们用与定义类似的方式规定这个顺序：

- 对于单独的一个常量，不需要计算。
- 对于 ([EXPR])，只需计算 [EXPR]。
- 对于 [FUNC]([EXPR_1],[EXPR_2],……,[EXPR_n])，先依次计算 [EXPR_1]、[EXPR_2]、……、[EXPR_n]，再调用 [FUNC]。
- 对于 [EXPR].[FUNC]([EXPR_1],[EXPR_2],……,[EXPR_n]),先依次计算 [EXPR]、[EXPR_1]、[EXPR_2]、……、[EXPR_n]，再调用 [FUNC]。
- 对于 [EXPR_0][OPR_1][EXPR_1][OPR_2]……[OPR_n][EXPR_n]（其中 [OPR_1]、[OPR_2]、……、[OPR_n] 全为乘除或全为加减），先计算 [EXPR_0]，再计算 [EXPR_1]，再调用 [OPR_1] 得出中间结果，再计算 [EXPR_2]，再用上述中间结果和 [EXPR_2] 的结果调用 [OPR_2]……直到计算完毕。

可以看出，除函数外，上述运算顺序均与我们日常使用的顺序相同；函数的运算顺序在不同的标准中不同，这里我们规定为从左至右。

【输入格式】

输入文件只有一行（以换行符结束），只包含一个需要处理的表达式。

表达式的长度不超过 100，不会出现任何空格等多余字符。

【输出格式】

按计算顺序输出每一次的运算符与函数调用。每次调用的参数只能是常量或之前某一次的运算结果，其中运算结果我们用从小到大的正整数依次表示。即：我们用单个英文字母（与输入中的相同）表示一个常量，用一个正整数表示之前某一次的运算结果，设第 i 次的调用产生的结果为 i。

以下用 [VALUE] 表示一个参数，[OPR] 表示一个运算符，[FUNC] 表示一个函数。

如果是运算符调用，设这次要计算的是 [VALUE_1][OPR][VALUE_2]，则你需要输出一行 [OPR][空格][VALUE_1][空格][VALUE_2]。

如果是普通函数调用,设这次要计算的是 [FUNC]([VALUE_1],[VALUE_2],……,[VALUE_n])，则你需要输出一行 [FUNC][空格][VALUE_1][空格][VALUE_2][空格]……[空格][VALUE_n]。

如果是成员函数调用,设这次要计算的是 [VALUE_0].[FUNC]([VALUE_1],[VALUE_2],……,[VAL
则你需要输出一行 [FUNC][空格][VALUE_0][空格][VALUE_1][空格][VALUE_2][空格]……[空格][VALUE_n]。

【样例 1 输入】

```
(a+f((b-c+e)*d/c.h(d,d)).g(e)).g(d).h(f(a,c),f(b)/f(c),f(d))
```

【样例 1 输出】

```
- b c
+ 1 e
* 2 d
h c d d
/ 3 4
f 5
g 6 e
+ a 7
g 8 d
f a c
f b
f c
/ 11 12
f d
h 9 10 13 14
```

【样例 1 解释】

```
- b c          //  1  b-c
+ 1 e          //  2  b-c+e
* 2 d          //  3  (b-c+e)*d
h c d d        //  4  c.h(d,d)
/ 3 4          //  5  (b-c+e)*d/c.h(d,d)
```

```
f 5           //  6  f((b-c+e)*d/c.h(d,d))
g 6 e         //  7  f((b-c+e)*d/c.h(d,d)).g(e)
+ a 7         //  8  a+f((b-c+e)*d/c.h(d,d)).g(e)
g 8 d         //  9  (a+f((b-c+e)*d/c.h(d,d)).g(e)).g(d)
f a c         // 10  f(a,c)
f b           // 11  f(b)
f c           // 12  f(c)
/ 11 12       // 13  f(b)/f(c)
f d           // 14  f(d)
h 9 10 13 14  // 15  (a+f((b-c+e)*d/c.h(d,d)).g(e)).g(d).h(f(a,c),f(b)/f(c),
```

【样例 2 输入】

```
a+b+c+d+e*f*g*h*i*(j-k-l-m-n)
```

【样例 2 输出】

```
+ a b
+ 1 c
+ 2 d
* e f
* 4 g
* 5 h
* 6 i
- j k
- 8 l
- 9 m
- 10 n
* 7 11
+ 3 12
```

【样例 3 输入】

```
(a+a+a)+a+a+a+(a+a+a)
```

【样例 3 输出】

```
+ a a
+ 1 a
```

```
+ 2 a
+ 3 a
+ 4 a
+ a a
+ 6 a
+ 5 7
```

【测试点】

测试点 1：包含常量、加减号测试点 2：包含常量、四则运算符测试点 3：包含常量、加减号、括号测试点 4：包含常量、四则运算符、括号测试点 5：包含常量、普通函数调用测试点 6：包含常量、成员函数调用测试点 7：包含常量、普通函数调用、成员函数调用、括号测试点 8：包含常量、四则运算符、普通函数调用、括号测试点 9：包含常量、四则运算符、成员函数调用、括号测试点 10：包含常量、四则运算符、普通函数调用、成员函数调用、括号

矩阵树（matrixtree）

【题意简述】

B 君有一个 n 个点的有向图。

B 君想知道有多少种不同的以 1 号点为根的生成树形图（在离散数学这门课程中，这个数据结构被叫做“根树”）。由于答案可能很大，只需输出这个数除以 10007 所得的余数。

L 君的提示：

以 1 号点为根的树形图，就是说，从有向图中选出 $n-1$ 条边，构成一个树，这个树的根节点是 1，对于所有其他点 i，存在且只存在一条从 i 到 1 的有向路径。

G 君的提示：

对于一个有向图，我们构造如下 **A** 矩阵：（其中 $a_{i,j}$ 表示矩阵第 i 行第 j 列的元素）

对于 $i \neq j$，$a_{i,j}$ 为 i 到 j 的边数的相反数；

$a_{i,i}$ 为 i 的出度。

那么这个矩阵删去第 1 行第 1 列之后的余子式为以 1 号点为根的树形图的数量。

R 君的提示：

如果你需要计算

$$(a/b) \mod p$$

其中 p 是质数，b 是 a 的约数，b 与 p 互质。

你可以先找到 c，满足 $bc \mod p = 1$。

然后可以证明

$$(a/b) \mod p = (a \mod p) \cdot c \mod p$$

其中满足 $bc \mod p = 1$ 的 c，被称为 b 的乘法逆元，满足这个条件的 c 存在且唯一。

【输入格式】

从标准输入读入数据。

输入第一行一个正整数 n，表示一共有 n 个点。

以下 n 行，每行 n 个数，表示这个有向图的邻接矩阵。

其中如果第 i 行第 j 个数是 1，表示 i 到 j 存在一条有向边，如果是 0，表示不存在有向边。

在本题中没有重边和自环，所以这个矩阵是 01 矩阵，主对角线上的数均为 0。

【输出格式】

输出到标准输出。
一行一个整数，表示答案。

【样例输入】

```
4
0 0 0 0
1 0 1 0
1 1 0 1
1 1 1 0
```

【样例输出】

```
12
```

【样例解释】

矩阵 **A** 为

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

删掉第一行第一列之后的，做行列式求值（也就是余子式）为

$$\begin{vmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{vmatrix} = 12$$

【子任务】

对于 10% 的数据 $n \le 3$；
对于 20% 的数据 $n \le 4$；
对于 40% 的数据 $n \le 10$；
对于 60% 的数据 $n \le 16$；
对于 100% 的数据 $n \le 100$。

距离（dist）

【题目描述】

在很多领域下，我们需要将高维空间中的点作为某个模型的基本研究对象。

现在我们建立一个 n 维的直角坐标系，每个点可以用它的坐标 $(x_1, x_2, \cdots, x_n)$ 来表示。如果我们设定每一维坐标都必须是不超过 m 的正整数，那么一共就有 m^n 个点。

对于两个点 $(a_1, a_2, \cdots, a_n)$ 与 $(b_1, b_2, \cdots, b_n)$，我们定义 $\sum_{i=1}^{n} |a_i - b_i|$，即 $|a_1 - b_1| + |a_2 - b_2| + \cdots + |a_n - b_n|$，为这两个点的（曼哈顿）距离。容易看出，两个点的距离至少为 0，至多为 $n(m-1)$。

m^n 个点中的每个点都具有一个权值。你需要对于每个点，对于每个满足 $0 \le d \le n(m-1)$ 的 d，求出和这个点距离为 d 的所有点的权值之和。（如果不存在这样的点则应输出 0）

【输入格式】

第一行输入两个正整数 n, m，用空格隔开。

接下来 m^n 行，每行一个正整数，表示每个点的权值。输入是按照坐标序列的字典序排列的，即先输入点 $(1, 1, \cdots, 1, 1)$ 的权值，再输入 $(1, 1, \cdots, 1, 2)$ 的权值，……，再输入 $(1, 1, \cdots, 1, m)$ 的权值，再输入 $(1, 1, \cdots, 2, 1)$ 的权值……

【输出格式】

输出 m^n 行，每行表示一个点。每行有 $n(m-1)+1$ 个整数，依次表示和这个点距离为 $0, 1, \cdots, n(m-1)$ 的所有点的权值之和。

输出的点的顺序应与输入相同；同一行相邻的两个整数之间应当用恰好一个空格隔开。

由于数据规模较大，输出规模可达上百万个整数，请务必使用快速的方式进行输入输出。

【样例 1 输入】

```
3 2
1
2
4
8
16
32
```

```
64
128
```

【样例 1 输出】

```
1 22 104 128
2 41 148 64
4 73 146 32
8 134 97 16
16 97 134 8
32 146 73 4
64 148 41 2
128 104 22 1
```

【样例 2 输入】

```
2 3
9
8
7
6
5
4
3
2
1
```

【样例 2 输出】

```
9 14 15 6 1
8 21 12 4 0
7 12 15 8 3
6 17 14 8 0
5 20 20 0 0
4 13 16 12 0
3 8 15 12 7
2 9 18 16 0
1 6 15 14 9
```

【测试点】

所有数据满足：$n \geq 1, m \geq 2$，要输出的数不会超过 1250000 个。每个点的权值不会超过 100000。

测试点 1：$n = 1, m = 1000$。测试点 2：$n = 2, m = 85$。测试点 3：$n = 3, m = 25$。测试点 4：$n = 4, m = 12$。测试点 5：$n = 5, m = 8$。测试点 6：$n = 6, m = 5$。测试点 7：$n = 8, m = 3$。测试点 8：$n = 10, m = 3$。测试点 9：$n = 13, m = 2$。测试点 10：$n = 16, m = 2$。