```

题号 题目 知识点 难度 通过率

BL1 扭蛋机 递归 中等 32.67%

BL2 小A最多会新认识的多少人 图 中等 15.20%

BL3 简单表达式计算 字符串模拟栈 中等 29.30%

BL4 山寨金闪闪 数组 中等 17.53%

BL5 顺时针打印数字矩阵 递归数组模拟 中等 16.87%

BL6 精灵鼠从入口到出口的最少减少速度 动态规划字符串 中等 35.73%

BL7 比较两个版本字符串version1和version2 字符串 中等 21.86%

BL8 写一段程序判断IP字符串是否属于内网IP 字符串模拟 中等 39.96%

BL9 给定一个整数数组,判断其中是否有3个数和为N 排序数组哈希 中等 22.39%

BL10 脸滚键盘 字符串模拟 中等 15.64%

BL11 实现一个HTML语法检查器 字符串模拟栈 中等 10.45%

BL12 翻转链表 模拟链表 中等 21.86%

BL13 孙悟空的徒弟 数组穷举 中等 7.67%

BL14 ん...红茶？ 排序数组穷举 中等 14.96%

```

### BL1 扭蛋机

#### 题目描述

```

22娘和33娘接到了小电视君的扭蛋任务：

一共有两台扭蛋机，编号分别为扭蛋机2号和扭蛋机3号，22娘使用扭蛋机2号，33娘使用扭蛋机3号。

扭蛋机都不需要投币，但有一项特殊能力：

扭蛋机2号：如果塞x（x范围为>=0正整数）个扭蛋进去，然后就可以扭到2x+1个

扭蛋机3号：如果塞x（x范围为>=0正整数）个扭蛋进去，然后就可以扭到2x+2个

22娘和33娘手中没有扭蛋，需要你帮她们设计一个方案，两人“轮流扭”（谁先开始不限，扭到的蛋可以交给对方使用），用“最少”的次数，使她们能够最后恰好扭到N个交给小电视君。

```

#### 输入描述:

```

输入一个正整数，表示小电视君需要的N个扭蛋。

```

#### 输出描述:

```

输出一个字符串，每个字符表示扭蛋机，字符只能包含"2"和"3"。

```

#### 示例1

#### 输入

```

10

```

#### 输出

```

233

备注:

1<=N<=1e9

```

```java

import java.util.\*;

import java.io.\*;

//逆向思考，奇偶敏感

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

StringBuffer sb = new StringBuffer();

play(n, sb);

sb.reverse();

System.out.println(sb);

}

public static void play(int N, StringBuffer sb) {

if (N <= 0) return;

if (N % 2 == 0) {

sb.append(3);

play((N - 2) / 2, sb);

} else {

sb.append(2);

play((N - 1) / 2, sb);

}

}

}

```

### BL2 小A最多会新认识的多少人

#### 题目描述

```

小A参加了一个n人的活动，每个人都有一个唯一编号i(i>=0 & i<n)，其中m对相互认识，在活动中两个人可以通过互相都认识的一个人介绍认识。现在问活动结束后，小A最多会认识多少人？

```

#### 输入描述:

```

第一行聚会的人数：n（n>=3 & n<10000）；

第二行小A的编号: ai（ai >= 0 & ai < n)；

第三互相认识的数目: m（m>=1 & m

< n(n-1)/2）；

第4到m+3行为互相认识的对，以','分割的编号。

```

#### 输出描述:

```

输出小A最多会新认识的多少人？

```

#### 示例1

#### 输入

```

7

5

6

1,0

3,1

4,1

5,3

6,1

6,5

```

#### 输出

```

3

```

```java

import java.util.Scanner;

import java.io.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int N = Integer.parseInt(br.readLine());

int num = Integer.parseInt(br.readLine());

int m = Integer.parseInt(br.readLine());

int[] a = new int[N];

for (int i = 0; i < N; i++) {

a[i] = i;

}

int oldFriends = 1;

for (int i = 0; i < m; i++) {

String[] string = br.readLine().split(",");

int first = Integer.parseInt(string[0]);

int second = Integer.parseInt(string[1]);

if (findRoot(a, first) != findRoot(a, second)) a[findRoot(a, second)] = findRoot(a, first);

if (first == num || second == num) oldFriends++;

}

int friends = 0;

int tag = findRoot(a, num);

for (int i = 0; i < N; i++) {

if (findRoot(a, a[i]) == tag) friends++;

}

System.out.println(friends - oldFriends);

}

static int findRoot(int[] a, int ele) {

if (a[ele] != ele) return a[ele] = findRoot(a, a[ele]);

else return a[ele];

}

}

```

### BL3 简单表达式计算

#### 题目描述

```

给定一个合法的表达式字符串，其中只包含非负整数、加法、减法以及乘法符号（不会有括号），例如7+3\*4\*5+2+4-3-1，请写程序计算该表达式的结果并输出；

```

#### 输入描述:

```

输入有多行，每行是一个表达式，输入以END作为结束

```

#### 输出描述:

```

每行表达式的计算结果

```

#### 示例1

#### 输入

```

7+3\*4\*5+2+4-3-1

2-3\*1

END

```

#### 输出

```

69

-1

备注:

每个表达式的长度不超过10000，保证所有中间结果和最后结果在[-2e9,2e9]范围内

```

```java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

public static char[] arr;

public static int index = 0;

public static int length;

public static void main(String[] args) throws IOException {

BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));

String str = bf.readLine();

while (!str.equals("END")) {

arr = str.toCharArray();

length = str.length();

index = 0;

int res = getNum();

while (index < arr.length) {

if (arr[index] == '-') {

index++;

res -= getNum();

} else {

index++;

res += getNum();

}

}

System.out.println(res);

str = bf.readLine();

}

}

public static int getNum() {

int num = arr[index] - '0';

int res = num;

index++;

while (index < length) {

num = arr[index] - '0';

if (0 <= num && num <= 9) {

res = res \* 10 + num;

index++;

} else if (arr[index] == '\*') {

index++;

return res \* getNum();

} else {

break;

}

}

return res;

}

}

```

### BL4 山寨金闪闪

#### 题目描述

```

金闪闪死后，红A拿到了王之财宝，里面有n个武器，长度各不相同。红A发现，拿其中三件武器首尾相接，组成一个三角形，进行召唤仪式，就可以召唤出一个山寨金闪闪。（例如，三件武器长度为10、15、20，可以召唤成功。若长度为10、11、30，首尾相接无法组成三角形，召唤失败。）红A于是开了一个金闪闪专卖店。他把王之财宝排成一排，每个客人会随机抽取到一个区间[l,r],客人可以选取区间里的三件武器进行召唤（客人都很聪慧，如果能找出来合适的武器，一定不会放过）。召唤结束后，客人要把武器原样放回去。m个客人光顾以后，红A害怕过多的金闪闪愉悦太多男人，于是找到了你，希望你帮他统计出有多少山寨金闪闪被召唤出来。

```

#### 输入描述:

```

第一行武器数量:n <= 1\*10^7

第二行空格分隔的n个int，表示每件武器的长度。

第三行顾客数量：m <= 1\*10^6

后面m行，每行两个int l，r，表示每个客人被分配到的区间。（l<r）

```

#### 输出描述:

```

山寨金闪闪数量。

```

#### 示例1

#### 输入

```

5

1 10 100 95 101

4

1 3

2 4

2 5

3 5

```

#### 输出

```

3

```

```java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Arrays;

public class Main {

//1.把读取的武器长度存入到一个长度为武器数量的数组中去

//2.判断客人分配的区间是否可以组成一个三角形：把这个区间里的武器长度排序，每三个逐个往后移动，如果能组成则这个区间内一定能组成三角形

public static int[] arr;

public static void main(String[] args) throws IOException {

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(reader.readLine()); //把读取到的字符串的n转换为int类型的

//把读取的武器长度存入到一个长度为武器数量的数组中去

String[] str = reader.readLine().split(" ");

arr = new int[n];

for (int i = 0; i < n; i++) {

arr[i] = Integer.parseInt(str[i]);

}

//判断每个区间内的武器长度能否组成三角形,并计算数量

int count = 0;

int m = Integer.parseInt(reader.readLine());

for (int i = 0; i < m; i++) {

String[] str1 = reader.readLine().split(" ");

int l = Integer.parseInt(str1[0]);

int r = Integer.parseInt(str1[1]);

//当区间长度大于一定值的时候一定会有三角形

if (r - l + 1 > 46) count++;

else if (isTriango(l, r)) count++;

}

System.out.println(count); //输出能组成三角形的数量

}

//把这个区间里的武器长度排序，每三个逐个往后移动，如果能组成则这个区间内一定能组成三角形

private static boolean isTriango(int l, int r) {

int[] newArr = Arrays.copyOfRange(arr, l - 1, r);

//排序

Arrays.sort(newArr);

//每三个遍历一下

for (int i = 0; i < newArr.length - 2; i++) {

if (newArr[i] + newArr[i + 1] > newArr[i + 2]) return true;

}

return false;

}

}

```

### BL5 顺时针打印数字矩阵

#### 题目描述

```

给定一个数字矩阵，请设计一个算法从左上角开始顺时针打印矩阵元素

```

#### 输入描述:

```

输入第一行是两个数字，分别代表行数M和列数N；接下来是M行，每行N个数字，表示这个矩阵的所有元素；当读到M=-1，N=-1时，输入终止。

```

#### 输出描述:

```

请按逗号分割顺时针打印矩阵元素（注意最后一个元素末尾不要有逗号！例如输出“1，2，3”，而不是“1，2，

3，”），每个矩阵输出完成后记得换行

```

#### 示例1

#### 输入

```

3 3

1 2 3

4 5 6

7 8 9

-1 -1

```

#### 输出

```

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

备注:

M,N为正整数且 M\*N<=300000

```

```java

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

while (br.ready()) {

String[] s = br.readLine().split(" ");

int m = Integer.parseInt(s[0]);

int n = Integer.parseInt(s[1]);

if (m == -1 && n == -1) break;

int[][] matrix = new int[m][n];

for (int i = 0; i < m; i++) {

String[] temp = br.readLine().split(" ");

for (int j = 0; j < n; j++) matrix[i][j] = Integer.parseInt(temp[j]);

}

printMatrix(matrix, m, n);

}

}

public static void printMatrix(int[][] matrix, int row, int col) {

//创建StringBuffer存储，因为有,号

StringBuilder sb = new StringBuilder();

if (matrix == null || row == 0 || col == 0) return;

//记录圈数

int start;

//其他人的方法

start = (Math.min(col, row) - 1) / 2 + 1;//确定需要的层数

for (int i = 0; i < start; i++) {

//从左到右打印

for (int k = i; k < col - i; k++) sb.append(matrix[i][k]).append(",");

//从右上到右下

for (int j = i + 1; j < row - i; j++) sb.append(matrix[j][col - i - 1]).append(",");

//从右到左

for (int k = col - i - 2; (k >= i) && (row - i - 1 != i); k--)

sb.append(matrix[row - i - 1][k]).append(",");

//从左下到左上

for (int j = row - i - 2; (j > i) && (col - i - 1 != i); j--) sb.append(matrix[j][i]).append(",");

}

//输出，记得去掉最后一个“，”

System.out.println(sb.substring(0, sb.length() - 1));

}

}

```

### BL6 精灵鼠从入口到出口的最少减少速度

#### 题目描述

```

猛兽侠中精灵鼠在利剑飞船的追逐下逃到一个n\*n的建筑群中，精灵鼠从（0,0）的位置进入建筑群，建筑群的出口位置为（n-1,n-1），建筑群的每个位置都有阻碍，每个位置上都会相当于给了精灵鼠一个固定值减速，因为精灵鼠正在逃命所以不能回头只能向前或者向下逃跑，现在问精灵鼠最少在减速多少的情况下逃出迷宫？

```

#### 输入描述:

```

第一行迷宫的大小: n >=2 & n <= 10000；

第2到n+1行，每行输入为以','分割的该位置的减速,减速f >=1 & f < 10。

```

#### 输出描述:

```

精灵鼠从入口到出口的最少减少速度？

```

#### 示例1

#### 输入

```

3

5,5,7

6,7,8

2,2,4

```

#### 输出

```

19

```

```java

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.\*;

public class Main {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

int[][] w = new int[n][n];

int[] dp = new int[n + 1];

dp[0] = 0;

for (int i = 1; i <= n; i++) dp[i] = 100000;

for (int i = 1; i <= n; i++) {

String s = br.readLine();

for (int j = 0; j < s.length(); j += 2) {

int x = j / 2 + 1;

dp[x] = Math.min(dp[x - 1], dp[x]) + (s.charAt(j) - '0');

}

dp[0] = 100000;

}

System.out.println(dp[n]);

}

}

```

### BL7 比较两个版本字符串version1和version2

#### 题目描述

```

如果version1 > version2 返回1，如果 version1 < version2 返回-1，不然返回0.

输入的version字符串非空，只包含数字和字符.。.字符不代表通常意义上的小数点，只是用来区分数字序列。例如字符串2.5并不代表二点五，只是代表版本是第一级版本号是2，第二级版本号是5.

```

#### 输入描述:

```

两个字符串，用空格分割。

每个字符串为一个version字符串，非空，只包含数字和字符.

```

#### 输出描述:

```

只能输出1, -1，或0

```

#### 示例1

#### 输入

```

0.1 1.1

```

#### 输出

```

-1

备注:

version1和version2的长度不超过1000，由小数点'.'分隔的每个数字不超过256。

```

```java

import java.io.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String[] inarr = br.readLine().split(" ");

String[] str1 = inarr[0].split("\\.");

String[] str2 = inarr[1].split("\\.");

//先直接进行比较，看看谁的版本高

for (int i = 0; i < str1.length && i < str2.length; i++) {

if (Integer.parseInt(str1[i]) < Integer.parseInt(str2[i])) {

System.out.println("-1");

return;

} else if (Integer.parseInt(str1[i]) > Integer.parseInt(str2[i])) {

System.out.println("1");

return;

}

}

//当前面的比较没有成功，有如下情况

//长度相同时版本一致，

//一长一短，并且短的部分与长的前面相同

if (str1.length == str2.length) System.out.println("0");

else if (str1.length < str2.length) System.out.println("-1");

else System.out.println("1");

}

}

```

### BL8 写一段程序判断IP字符串是否属于内网IP

#### 题目描述

```

从业 666 年的 BILIBILI 网络安全工程师 KindMo 最近很困惑，公司有一个业务总是受到 SSRF 攻击。请帮他写一个程序，判断输入的字符串是否属于内网IP，用于防御该漏洞。

我们知道常见的内网IP有，127.0.0.1，192.168.0.1 等。

```

#### 输入描述:

```

每次输入仅包含一个IP字符串，即一个测试样例

```

#### 输出描述:

```

对于每个测试实例输出整数1或0，1代表True，即输入属于内网IP，0代表False，即输入不属于内网IP或不是IP字符串。

```

#### 示例1

#### 输入

```

42.96.146.169

```

#### 输出

```

0

```

```java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

public static void main(String[] args) throws IOException {

/\*

思路：ip转换为整数，判断数字是否属于内网地址对应的数字：

10.0.0.0 - 10.255.255.255: 需要判断：10

127.0.0.0 - 127.255.255.255 需要判断： 127

172.16.0.0 - 172.31.255.255 需要判断： 127 16 和 127 31

192.168.0.0 - 192.168.255.255 需要判断： 192 168

\*/

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String line = br.readLine();

String[] ips = line.split("\\.");

int v1 = Integer.parseInt(ips[0]);

int v2 = Integer.parseInt(ips[1]);

if (v1 == 10 || (v1 == 172 && (v2 >= 16 && v2 <= 31)) || (v1 == 192 && v2 == 168)) System.out.println(1);

else System.out.println(0);

}

}

```

### BL9 给定一个整数数组,判断其中是否有3个数和为N

#### 题目描述

```

给定一个整数数组,判断其中是否有3个数和为N

```

#### 输入描述:

```

输入为一行

逗号前为一个整数数组，每个元素间用空格隔开；逗号后为N

```

#### 输出描述:

```

输出bool值

True表示存在3个和为N的数

False表示不存在3个和为N的数

```

#### 示例1

#### 输入

```

1 2 3 4 5,10

```

#### 输出

```

True

备注:

数组长度不超过2000，所以数均为int范围的正整数

```

```java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Arrays;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String[] s = br.readLine().split(",");

String[] arr = s[0].split(" ");

int[] a = new int[arr.length];

int n = Integer.parseInt(s[1]);

for (int i = 0; i < arr.length; i++) {

a[i] = Integer.parseInt(arr[i]);

}

Arrays.sort(a);

for (int i = 0; i < a.length - 2; i++) {

int k = a.length - 1;

for (int j = i + 1; j < k; ) {

if (a[i] == n - a[j] - a[k]) {

System.out.println("True");

return;

} else if (a[i] < n - a[j] - a[k]) j++;

else k--;

}

}

System.out.println("False");

}

}

```

### BL10 脸滚键盘

#### 题目描述

```

av394281 中，充满威严的蕾米莉亚大小姐因为触犯某条禁忌，被隙间妖怪八云紫（紫m……èi）按住头在键盘上滚动。

同样在弹幕里乱刷梗被紫姐姐做成罪袋的你被指派找到大小姐脸滚键盘打出的一行字中的第 `k` 个仅出现一次的字。

(为简化问题，大小姐没有滚出 ascii 字符集以外的字)

```

#### 输入描述:

```

每个输入都有若干行，每行的第一个数字为`k`，表示求第`k`个仅出现一次的字。然后间隔一个半角空格，之后直到行尾的所有字符表示大小姐滚出的字符串`S`。

```

#### 输出描述:

```

输出的每一行对应输入的每一行的答案，如果无解，输出字符串`Myon~`

(请不要输出多余的空行）

为了方便评测，如果答案存在且为c，请输出[c]

```

#### 示例1

#### 输入

```

2 misakamikotodaisuki

3 !bakabaka~ bakabaka~ 1~2~9!

3 3.1415926535897932384626433832795028841971693993751o582097494459211451488946419191919l91919hmmhmmahhhhhhhhhh

7 www.bilibili.com/av170001

1 111

```

#### 输出

```

[d]

[9]

[l]

[7]

Myon~

备注:

字符串S仅包含可见ascii码，长度不超过100000

```

```java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

while (reader.ready()) {

String line = reader.readLine();

int index = 0;

while (line.charAt(index) != ' ') ++index;

int k = Integer.parseInt(line.substring(0, index));

System.out.println(appearFirst(k, line.substring(index + 1)));

}

reader.close();

}

private static String appearFirst(int k, String s) {

char[] ar = s.toCharArray();

int[] hash = new int[128];

for (char c : ar) hash[c]++;

for (char c : ar) {

if (hash[c] == 1 && k == 1) return "[" + c + "]";

else if (hash[c] == 1) --k;

}

return "Myon~";

}

}

```

### BL11 实现一个HTML语法检查器

#### 题目描述

```

实现一个HTML语法检查器。HTML语法规则简化如下：标签必须闭合，可以由开始和结束两个标签闭合，如<div></div>，也可以自闭合，

如<div />

标签可以嵌套如<div><a></a></div>或者 <div><a/></div>，但是标签不能交叉：<div><a></div></a>是不允许的标签里可以有属性

如<div id="a<1"></div>

属性的规则是name="任意非引号字符"，多属性声明之间必须有空格，属性声明不符合规则时，整段HTML都算语法错误

输入文本只会出现字母a-z和<>"=

请用任意语言实现一个HTML语法检查器函数，有语法错误返回1，没有语法错误返回0

```

#### 输入描述:

```

一行，一个HTML字符串

```

#### 输出描述:

```

有语法错误返回1，没有语法错误返回0

```

#### 示例1

#### 输入

```

<div><a></a></div>

```

#### 输出

```

0

示例2

输入

<div><a></div></a>

```

#### 输出

```

1

备注:

字符串长度不超过100

```

```java

import java.util.\*;

import java.util.Stack;

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

String str = input.nextLine();

int result = 1;

if (solve(str)) result = 0;

System.out.println(result);

}

public static boolean solve(String str) {

str = str.trim(); //消除前后的空格

Stack<String> stack = new Stack<>(); //记录标签的入栈顺序

int i = 0;

while (i < str.length()) {

if (i == 0 && str.charAt(i) != '<') return false; //第一个字符不是<，则表明不是以标签开始

if (i == str.length() - 1 && str.charAt(i) != '>') return false; //最后的字符不是>，则表明不是以标签结束

if (str.charAt(i) == '<') {

int j = i + 1;

//'<'之后如果是标签，紧跟在'<'之后的字符必定是a-z或者'/'

if (j < str.length() && (str.charAt(j) >= 'a' && str.charAt(j) <= 'z' || str.charAt(j) == '/')) {

//获取标签字符串<xx xx="">

boolean inStr = false; //是否处于字符串"里面

while (j < str.length()) {

//标签结束

if (!inStr && str.charAt(j) == '>') {

//传入的标签已经去除'<'和'>'字符

if (!checkAtt(str.substring(i + 1, j), stack)) return false; //属性有语法错误

i = j;

break;

} else if (str.charAt(j) == '"') inStr = !inStr;

j++;

}

}

}

i++;

}

return stack.isEmpty();

}

public static boolean checkAtt(String str, Stack<String> stack) { //处理标签里面的属性

if (str == null && str.length() == 0) {

return true;

}

int type = 0; //记录标签的类型，0为开始标签<x>，1为结束标签</x>,2为自闭合标签<x/>

if (str.charAt(0) == '/') { //判断是否是结束标签

if (!(str.length() >= 2 && str.charAt(1) >= 'a' && str.charAt(1) <= 'z')) {

return false; //结束标签的'/'之后必须紧跟字母

}

type = 1; //为结束标签

str = str.substring(1);

}

str = str.trim(); //去除字符串后面的空格

if (str.charAt(str.length() - 1) == '/') { //判断是否是自闭合标签

if (type == 1) return false; //不能同时为结束标签，又是自闭合标签

type = 2;

str = str.substring(0, str.length() - 1);// 去除最后的'/'

}

int i = 0;

//获取标签的名字，并判断是否符合闭合规则

while (i < str.length() && str.charAt(i) != ' ') i++;

String name = str.substring(0, i);

if (type == 0) stack.push(name);

else if (type == 1) {

if (!stack.isEmpty() && stack.peek().equals(name)) stack.pop();

else return false;

}

//判断其属性是否有语法错误

boolean hasAtt = false; //判断该标签是否有属性，针对结束标签不能有属性

boolean hasBlank = false; //判断属性name之前是否有空格

while (i < str.length()) {

while (i < str.length() && str.charAt(i) == ' ') {

hasBlank = true;

i++;

}

while (i < str.length() && str.charAt(i) != '=' && str.charAt(i) != ' ') i++; //获取属性名字

if (i < str.length() && str.charAt(i) == '=') { //如果没有'='说明不是属性，不进行处理

i++;

if (i >= str.length() || str.charAt(i) != '"') return false; //'='之后必须是双引号

i++;

while (i < str.length() && str.charAt(i) != '"') i++; //寻找下一个双引号

if (i >= str.length()) return false;

if (!hasBlank) return false; //属性之前没有空格，语法错误

}

hasBlank = false;

hasAtt = true;

i++;

}

return type != 1 || !hasAtt; //结束标签不能有属性

}

}

```

### BL12 翻转链表

#### 题目描述

```

对于一个链表 L: L0→L1→…→Ln-1→Ln,

将其翻转成 L0→Ln→L1→Ln-1→L2→Ln-2→…

输入是一串数字，请将其转换成单链表格式之后，再进行操作

```

#### 输入描述:

```

一串数字，用逗号分隔

```

#### 输出描述:

```

一串数字，用逗号分隔

```

#### 示例1

#### 输入

```

1,2,3,4,5

```

#### 输出

```

1,5,2,4,3

备注:

数组长度不超过100000

```

```java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String[] strings = reader.readLine().split(",");

StringBuilder buffer = new StringBuilder();

int i = 0;

int left = 0;

int right = strings.length - 1;

while (left < right) {

if (i % 2 == 0) buffer.append(strings[left++]);

else buffer.append(strings[right--]);

buffer.append(",");

i++;

}

buffer.append(strings[left]);

System.out.println(buffer);

}

}

```

### BL13 孙悟空的徒弟

#### 题目描述

```

打败魔人布欧以后，孙悟空收了n个徒弟，每个徒弟战斗力各不相同。他教导所有的徒弟和体术，合体后战斗力为原战斗力相乘。任何两个徒弟都可以合体，所以一共有n\*(n-1)/2种合体徒弟。有一天，他想考验一下孙悟天战斗力如何，希望在所有n\*(n-1)/2种合体徒弟中选择战斗力第k高的，与孙悟天对战。可是孙悟空徒弟太多了，他已然懵逼，于是找到了你，请你帮他找到对的人。

```

#### 输入描述:

```

第一行两个int。徒弟数量：n <= 1\*10^6；战斗力排名:k <= n\*(n-1)/2

第二行空格分隔n个int，表示每个徒弟的战斗力。

```

#### 输出描述:

```

战斗力排名k的合体徒弟战斗力。

```

#### 示例1

#### 输入

```

5 2

1 3 4 5 9

```

#### 输出

```

36

```

```java

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String[] s = br.readLine().split(" ");

int n = Integer.parseInt(s[0]);

long k = Long.parseLong(s[1]);

long[] num = new long[n];

String[] s1 = br.readLine().split(" ");

for (int i = 0; i < n; i++) {

num[i] = Long.parseLong(s1[i]);

}

Arrays.sort(num);

long min = 0, max = num[n - 1] \* num[n - 2];

while (min < max) {

long mid = (min + max + 1) / 2;

long cur = 0;

int left = 0, right = n - 1;

while (left < right && cur < k) {

while (left < right && num[left] \* num[right] < mid) left++;

cur += Math.max(right - left, 0);

int i = right--;

}

if (cur >= k) min = mid;

else max = mid - 1;

}

System.out.println(min);

}

}

```

### BL14 ん...红茶？

#### 题目描述

```

高贵的蕾米莉亚大小姐每天需要饮用定量 B 型血的红茶以保持威严，并且要分两杯在不同时段饮用。

女仆长十六夜咲夜每天可以制作很多杯不同剂量 B 型血的红茶供蕾米莉亚大小姐饮用。

某日，你和天才妖精琪露诺偷偷潜入红魔馆被咲夜抓住，要求在今日份的红茶中挑出所有满足大小姐要求的茶杯，否则……

```

#### 输入描述:

```

每个样例有三行输入，第一行输入表示茶杯个数，第二行输入表示每份茶杯里的 B 型血剂量，第三行表示大小姐今天的定量

```

#### 输出描述:

```

对每一个样例，输出所有可能的搭配方案，如果有多种方案，请按每个方案的第一杯 B 型血剂量的大小升序排列。

如果无法找到任何一种满足大小姐的方案，输出"NO"(不包括引号)并换行。

```

#### 示例1

#### 输入

```

7

2 4 6 1 3 5 7

7

```

#### 输出

```

1 6

2 5

3 4

备注:

茶杯个数不超过100000，保证茶杯里的B型血剂量两两不同。

```

```java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Arrays;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

String[] s = br.readLine().split(" ");

int[] nums = new int[s.length];

for (int i = 0; i < s.length; i++) {

nums[i] = Integer.parseInt(s[i]);

}

int r = Integer.parseInt(br.readLine());

Arrays.sort(nums);

int left = 0;

int right = nums.length - 1;

boolean flag = false;

while (left < right) {

if (nums[left] + nums[right] > r) right--;

else if (nums[left] + nums[right] < r) left++;

else {

flag = true;

System.out.println(nums[left] + " " + nums[right]);

left++;

}

}

if (!flag) System.out.println("NO");

}

}

```