***入门训练***

序列求和

问题描述

求1+2+3+...+n的值。

输入格式

输入包括一个整数n。

输出格式

输出一行，包括一个整数，表示1+2+3+...+n的值。

样例输入

4

样例输出

10

样例输入

100

样例输出

5050

数据规模与约定

1 <= n <= 1,000,000,000。

import java.util.Scanner;

public class Main {

    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);

        long n=sc.nextLong();

        long sum=0;

         if(n>0&&n<=1000000000){

            sum=n\*(1+n)/2;

              System.out.println(sum);

          }

    }

}

圆的面积

问题描述

给定圆的半径r，求圆的面积。

输入格式

输入包含一个整数r，表示圆的半径。

输出格式

输出一行，包含一个实数，四舍五入保留小数点后7位，表示圆的面积。

样例输入

4

样例输出

50.2654825

数据规模与约定

1 <= r <= 10000。

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int r = scanner.nextInt();

        double PI = Math.PI;

        System.out.println(String.format("%.7f", (PI\*r\*r)));

    }

}

Fibonacci数列

问题描述

Fibonacci数列的递推公式为：Fn=Fn-1+Fn-2，其中F1=F2=1。

当n比较大时，Fn也非常大，现在我们想知道，Fn除以10007的余数是多少。

输入格式

输入包含一个整数n。

输出格式

输出一行，包含一个整数，表示Fn除以10007的余数。

说明：在本题中，答案是要求Fn除以10007的余数，因此我们只要能算出这个余数即可，而不需要先计算出Fn的准确值，再将计算的结果除以10007取余数，直接计算余数往往比先算出原数再取余简单。

样例输入

10

样例输出

55

样例输入

22

样例输出

7704

数据规模与约定

1 <= n <= 1,000,000。

import java.util.Scanner;

public class One {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int f3 = 1;

        int f1 = 1;

        int f2 = 1;

        if (n < 3) {

            System.out.println(f1);

        } else {

            for (int i = 2; i < n; i++) {

                f3 = f1+f2;

                f1 = f2;

                f2 = f3;

            }

            System.out.println(f3%10007);

        }

    }

}

***基础练习***

时间转换 (vip)

问题描述

　　给定一个以秒为单位的时间t，要求用“<H>:<M>:<S>”的格式来表示这个时间。<H>表示时间，<M>表示分钟，而<S>表示秒，它们都是整数且没有前导的“0”。例如，若t=0，则应输出是“0:0:0”；若t=3661，则输出“1:1:1”。

输入格式

　　输入只有一行，是一个整数t（0<=t<=86399）。

输出格式

　　输出只有一行，是以“<H>:<M>:<S>”的格式所表示的时间，不包括引号。

样例输入

0

样例输出

0:0:0

样例输入

5436

样例输出

1:30:36

import java.util.Scanner;

public class Main {

    /\*\*

     \* @param args

     \*/

    public static void main(String[] args) {

        Scanner reader = new Scanner(System.in);

        int N = reader.nextInt();

        int h=0,m=0,s=0;

        m=N/60;

        h=m/60;

        s=N%60;

        m=m%60;

        System.out.println(h+":"+m+":"+s);

    }

}

字符串对比 (vip)

问题描述

　　给定两个仅由大写字母或小写字母组成的字符串(长度介于1到10之间)，它们之间的关系是以下4中情况之一：

　　1：两个字符串长度不等。比如 Beijing 和 Hebei

　　2：两个字符串不仅长度相等，而且相应位置上的字符完全一致(区分大小写)，比如 Beijing 和 Beijing

　　3：两个字符串长度相等，相应位置上的字符仅在不区分大小写的前提下才能达到完全一致（也就是说，它并不满足情况2）。比如 beijing 和 BEIjing

　　4：两个字符串长度相等，但是即使是不区分大小写也不能使这两个字符串一致。比如 Beijing 和 Nanjing

　　编程判断输入的两个字符串之间的关系属于这四类中的哪一类，给出所属的类的编号。

输入格式

　　包括两行，每行都是一个字符串

输出格式

　　仅有一个数字，表明这两个字符串的关系编号

样例输入

BEIjing

beiJing

样例输出

3

import java.util.Scanner;

public class Main {

    /\*\*

     \* @param args

     \*/

    public static void main(String[] args) {

        Scanner reader = new Scanner(System.in);

        String str1 = reader.next();

        String str2 = reader.next();

        if (str1.length() != str2.length()) {

            System.out.println(1);

        } else if (str1.equals(str2)) {

            System.out.println(2);

        } else if (str1.regionMatches(true, 0, str2, 0, str1.length())) {

            System.out.println(3);

        }else {

            System.out.println(4);

        }

    }

}

分解质因数 (vip)

问题描述

　　求出区间[a,b]中所有整数的质因数分解。

输入格式

　　输入两个整数a，b。

输出格式

　　每行输出一个数的分解，形如k=a1\*a2\*a3...(a1<=a2<=a3...，k也是从小到大的)(具体可看样例)

样例输入

3 10

样例输出

3=3

4=2\*2

5=5

6=2\*3

7=7

8=2\*2\*2

9=3\*3

10=2\*5

提示

　　先筛出所有素数，然后再分解。

数据规模和约定

　　2<=a<=b<=10000

import java.util.Scanner;

public class Main {

    /\*\*

     \* @param args

     \*/

    public static void main(String[] args) {

        Scanner reader = new Scanner(System.in);

        int start=reader.nextInt();

        int end=reader.nextInt();

        for(int i=start;i<=end;i++){

            System.out.println(i+"="+fun(i).substring(1));

        }

    }

    static String fun(int n){

        String str="";

        for(int i=2;i<n;i++){

            if(n%i==0){

                return "\*"+str+i+fun(n/i);

            }

        }

        return "\*"+n;

    }

}

矩阵乘法 (vip)

问题描述

　　给定一个N阶矩阵A，输出A的M次幂（M是非负整数）

　　例如：

　　A =

　　1 2

　　3 4

　　A的2次幂

　　7 10

　　15 22

输入格式

　　第一行是一个正整数N、M（1<=N<=30, 0<=M<=5），表示矩阵A的阶数和要求的幂数

　　接下来N行，每行N个绝对值不超过10的非负整数，描述矩阵A的值

输出格式

　　输出共N行，每行N个整数，表示A的M次幂所对应的矩阵。相邻的数之间用一个空格隔开

样例输入

2 2

1 2

3 4

样例输出

7 10

15 22

import java.util.Scanner;

//测试通过

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        //几行

        int row = sc.nextInt();

        //相当于m

        int m = sc.nextInt();

        int[][] first = new int[row][row];

        int[][] next = new int[row][row];

        int[][] result = new int[row][row];

        for (int i = 0; i < row; i++)

            for (int j = 0; j < row; j++){

                next[i][j]=first[i][j] = sc.nextInt();

            }

        //判断是否为0

        if(m==0){

            for(int i=0;i<row;i++)

                for(int j=0;j<row;j++)

                    if(i==j)

                        result[i][j]=1;

                    else

                        result[i][j]=0;

        }

        else

        for (int h = 0; h < m - 1; h++) {

            result = new int[row][row];

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

                for (int j = 0; j < row; j++) {

                    for (int k = 0; k < row; k++) {

                        result[i][j] += next[i][k] \* first[k][j];

                    }

                }

            }

            next = result;

        }

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

            for (int j = 0; j < row; j++) {

                System.out.print(next[i][j] + "\t");

            }

            System.out.println();

        }

    }

}

矩形面积交 (vip)

问题描述

　　平面上有两个矩形，它们的边平行于直角坐标系的X轴或Y轴。对于每个矩形，我们给出它的一对相对顶点的坐标，请你编程算出两个矩形的交的面积。

输入格式

　　输入仅包含两行，每行描述一个矩形。

　　在每行中，给出矩形的一对相对顶点的坐标，每个点的坐标都用两个绝对值不超过10^7的实数表示。

输出格式

　　输出仅包含一个实数，为交的面积，保留到小数后两位。

样例输入

1 1 3 3

2 2 4 4

样例输出

1.00

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        String aline[]=sc.nextLine().split("\\s");

        String bline[]=sc.nextLine().split("\\s");

        double[] a=new double[4];

        double[] b=new double[4];

        double[] c=new double[4];

        for(int i=0;i<4;i++){

            a[i]=Double.parseDouble(aline[i]);

        }

        for(int i=0;i<4;i++){

            b[i]=Double.parseDouble(bline[i]);

        }

        c[0]=Math.max(a[0], b[0]);

        c[1]=Math.max(a[1], b[1]);

        c[2]=Math.min(a[2], b[2]);

        c[3]=Math.min(a[3], b[3]);

        if(c[0]<=c[2]&&c[1]<=c[3]){

            System.out.println((c[2]-c[0])\*(c[3]-c[1]));

        }else{

            System.out.println("error");

        }

    }

}

完美的代价 (vip)

问题描述

　　回文串，是一种特殊的字符串，它从左往右读和从右往左读是一样的。小龙龙认为回文串才是完美的。现在给你一个串，它不一定是回文的，请你计算最少的交换次数使得该串变成一个完美的回文串。

　　交换的定义是：交换两个相邻的字符

　　例如mamad

　　第一次交换 ad : mamda

　　第二次交换 md : madma

　　第三次交换 ma : madam (回文！完美！)

输入格式

　　第一行是一个整数N，表示接下来的字符串的长度(N <= 8000)

　　第二行是一个字符串，长度为N.只包含小写字母

输出格式

　　如果可能，输出最少的交换次数。

　　否则输出Impossible

样例输入

5

mamad

样例输出

3

import java.io.\*;

public class Main {

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

        BufferedReader buf = new BufferedReader(

                new InputStreamReader(System.in));

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

        String s = buf.readLine();

        char c[] = s.toCharArray();

        int count = 0, num = 0;

        char ch = '0';

        int i, k = 0, j;

        int b[] = new int[26];

        for (i = 0; i < n; i++)

            for (i = 0; i < n; i++) {

                j = c[i] - 'a';

                b[j]++;

            }

        for (j = 0; j < 26; j++) {

            if (b[j] % 2 != 0) {

                k++;

                ch = (char) (j + 'a');

            }

        }

        if (k >= 2)

            System.out.println("Impossible");

        else

            System.out.println(changes(c, ch, n));

    }

    public static int changes(char s[], char x, int n) {

        int i, change = 0, j, k;

        for (i = 0; i < n / 2; i++) {

            if (s[i] == x) {

                for (j = i; j < n - i - 1; j++)

                    if (s[n - i - 1] == s[j])

                        break;

                change += j - i;

                for (k = j; k > i; k--)

                    s[k] = s[k - 1];

                s[i] = s[n - i - 1];

            } else {

                for (j = n - i - 1; j >= i; j--)

                    if (s[i] == s[j])

                        break;

                change += n - i - 1 - j;

                for (k = j; k < n - i - 1; k++)

                    s[k] = s[k + 1];

                s[n - i - 1] = s[i];

            }

        }

        return change;

    }

}

数的读法 (vip)

问题描述

　　Tom教授正在给研究生讲授一门关于基因的课程，有一件事情让他颇为头疼：一条染色体上有成千上万个碱基对，它们从0开始编号，到几百万，几千万，甚至上亿。

　　比如说，在对学生讲解第1234567009号位置上的碱基时，光看着数字是很难准确的念出来的。

　　所以，他迫切地需要一个系统，然后当他输入12 3456 7009时，会给出相应的念法：

　　十二亿三千四百五十六万七千零九

　　用汉语拼音表示为

　　shi er yi san qian si bai wu shi liu wan qi qian ling jiu

　　这样他只需要照着念就可以了。

　　你的任务是帮他设计这样一个系统：给定一个阿拉伯数字串，你帮他按照中文读写的规范转为汉语拼音字串，相邻的两个音节用一个空格符格开。

　　注意必须严格按照规范，比如说“10010”读作“yi wan ling yi shi”而不是“yi wan ling shi”，“100000”读作“shi wan”而不是“yi shi wan”，“2000”读作“er qian”而不是“liang qian”。

输入格式

　　有一个数字串，数值大小不超过2,000,000,000。

输出格式

　　是一个由小写英文字母，逗号和空格组成的字符串，表示该数的英文读法。

样例输入

1234567009

样例输出

shi er yi san qian si bai wu shi liu wan qi qian ling jiu

import java.util.Scanner;

public class Main {

    /\*\*读数

     \*

     \*/

        public static void main(String[] args) {

            Scanner in=new Scanner(System.in);

            String s=in.next();

            int n=s.length();

            int m=n;

            int[] a=new int[n];

            StringBuffer buf=new StringBuffer();

            //将字符串放在数组里面

            for(int i=0;i<n;i++){

                if(i<n-1){

                    a[i]=Integer.parseInt(s.substring(i, i+1));

                }

                else

                    a[i]=Integer.parseInt(s.substring(n-1));

            }

            //读数

            for(int i=0;i<n;i++){

                if(m==10){

                    if(a[i]==1){

                        buf.append("shi ");

                    }

                    else{

                        buf.append("er shi ");

                    }

                }

                if(m==9){

                        if(a[i]>0){

                            buf.append(check(a[i])+" yi ");

                        }

                        else{

                            if(a[i]==0&&a[i+1]!=0)

                             buf.append("yi ling ");

                            else

                                buf.append("yi ");

                        }

                }

                if(m==8){

                    if(a[i]>0){

                        buf.append(check(a[i])+" qian ");

                    }

                    else{

                        if(a[i]==0&&a[i+1]!=0){

                            buf.append("ling ");

                        }

                }

                }

                if(m==7){

                    if(a[i]>0){

                        buf.append(check(a[i])+" bai ");

                    }

                    else{

                        if(a[i]==0&&a[i+1]!=0){

                            buf.append("ling ");

                        }

                    }

                }

                if(m==6){

                    if(n==6){

                        if(a[i]>1){

                            buf.append(check(a[i])+" shi ");

                        }

                        if(a[i]==1){

                            buf.append("shi ");

                        }

                    }

                    else{

                        if(a[i]!=0){

                            buf.append(check(a[i])+" shi ");

                        }

                        else if(a[i]==0&&a[i+1]!=0){

                            buf.append("ling ");

                        }

                    }

                }

                if(m==5){//wan的

                    if(n==5){

                        if(a[i]>0){

                            buf.append(check(a[i])+" wan ");

                        }

                    }

                    else{

                        if(a[i]!=0){

                            buf.append(check(a[i])+" wan ");

                        }

                        else{

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

                                if(a[i-j]!=0){

                                    buf.append("wan ");

                                    if(a[i+1]!=0){

                                        buf.append("ling ");

                                    }

                                    break;//注意别忘记

                                }

                            }

                        }

                    }

                }

                if(m==4){

                    if(a[i]>0){

                        buf.append(check(a[i])+" qian ");

                    }

                    else{

                        if(a[i]==0&&a[i+1]!=0){

                        buf.append("ling ");

                        }

                    }

                }

                if(m==3){

                    if(a[i]>0){

                        buf.append(check(a[i])+" bai ");

                    }

                    else{

                        if(a[i]==0&&a[i+1]!=0){

                        buf.append("ling ");

                        }

                    }

                }

                if(m==2){

                    //if(n==2)

                    if(a[i]>0){

                        buf.append(check(a[i])+" shi ");

                    }

                    else{

                        if(a[i]==0&&a[i+1]!=0){

                        buf.append("ling ");

                        }

                    }

                }

                if(m==1){

                    if(a[i]>0) buf.append(check(a[i]));

                }

            m--;

            }

            System.out.println(buf);

        }

        static String s="";

        static String check(int i){

            switch(i){

            case 1:s="yi";break;

            case 2:s="er";break;

            case 3:s="san";break;

            case 4:s="si";break;

            case 5:s="wu";break;

            case 6:s="liu";break;

            case 7:s="qi";break;

            case 8:s="ba";break;

            case 9:s="jiu";break;

            }

            return s;

        }

    }

Sine之舞 (vip)

问题描述

　　最近FJ为他的奶牛们开设了数学分析课，FJ知道若要学好这门课，必须有一个好的三角函数基本功。所以他准备和奶牛们做一个“Sine之舞”的游戏，寓教于乐，提高奶牛们的计算能力。

　　不妨设

　　An=sin(1–sin(2+sin(3–sin(4+...sin(n))...)

　　Sn=(...(A1+n)A2+n-1)A3+...+2)An+1

　　FJ想让奶牛们计算Sn的值，请你帮助FJ打印出Sn的完整表达式，以方便奶牛们做题。

输入格式

　　仅有一个数：N<201。

输出格式

　　请输出相应的表达式Sn，以一个换行符结束。输出中不得含有多余的空格或换行、回车符。

样例输入

3

样例输出

((sin(1)+3)sin(1–sin(2))+2)sin(1–sin(2+sin(3)))+1

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int n = scanner.nextInt();

        String string = f(n, 1, 1);

        System.out.println(string);

    }

    private static String f1(int n, int i, int j) {

        if (i == n) return "sin("+j+")";

        if (i % 2 ==0) return "sin("+i+"+"+f1(n, i+1,j+1)+")";

        else return "sin("+i+"-"+f1(n, i+1,j+1)+")";

    }

    private static String f(int n, int i, int j) {

        if (i == n) return f1(n, i, j)+"+"+i;

        return "("+f(n, i+1, j)+")"+f1(n, i, j)+"+"+i;

    }

}

FJ的字符串 (vip)

问题描述

　　FJ在沙盘上写了这样一些字符串：

　　A1 = “A”

　　A2 = “ABA”

　　A3 = “ABACABA”

　　A4 = “ABACABADABACABA”

　　… …

　　你能找出其中的规律并写所有的数列AN吗？

输入格式

　　仅有一个数：N ≤ 26。

输出格式

　　请输出相应的字符串AN，以一个换行符结束。输出中不得含有多余的空格或换行、回车符。

样例输入

3

样例输出

ABACABA

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int t=sc.nextInt();

        print(t);

    }

    private static void print(int t) {

        if(t==1)

            System.out.print("A");

        else if(t<=26){

            print(t-1);

            System.out.print((char)(64+t));

            print(t-1);

        }

    }

}

芯片测试 (vip)

问题描述

　　有n（2≤n≤20）块芯片，有好有坏，已知好芯片比坏芯片多。

　　每个芯片都能用来测试其他芯片。用好芯片测试其他芯片时，能正确给出被测试芯片是好还是坏。而用坏芯片测试其他芯片时，会随机给出好或是坏的测试结果（即此结果与被测试芯片实际的好坏无关）。

　　给出所有芯片的测试结果，问哪些芯片是好芯片。

输入格式

　　输入数据第一行为一个整数n，表示芯片个数。

　　第二行到第n+1行为n\*n的一张表，每行n个数据。表中的每个数据为0或1，在这n行中的第i行第j列（1≤i, j≤n）的数据表示用第i块芯片测试第j块芯片时得到的测试结果，1表示好，0表示坏，i=j时一律为1（并不表示该芯片对本身的测试结果。芯片不能对本身进行测试）。

输出格式

　　按从小到大的顺序输出所有好芯片的编号

样例输入

3

1 0 1

0 1 0

1 0 1

样例输出

1 3

import java.io.\*;

public class Main {

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

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

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

        int a[][]=new int[n][n];

        int b[]=new int[n\*n];

        int i=0,j=0;

        for(i=0;i<n;i++){

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

        for(j=0;j<n;j++){

            a[i][j]=Integer.parseInt(s[j]);

            if(a[i][j]==0)

                b[j]=b[j]+1;

    }

    }

        for(i=0;i<n;i++)

            if(b[i]<=n/2)

                System.out.print(i+1+" ");

}

}

龟兔赛跑预测 (vip)

问题描述

　　话说这个世界上有各种各样的兔子和乌龟，但是研究发现，所有的兔子和乌龟都有一个共同的特点——喜欢赛跑。于是世界上各个角落都不断在发生着乌龟和兔子的比赛，小华对此很感兴趣，于是决定研究不同兔子和乌龟的赛跑。他发现，兔子虽然跑比乌龟快，但它们有众所周知的毛病——骄傲且懒惰，于是在与乌龟的比赛中，一旦任一秒结束后兔子发现自己领先t米或以上，它们就会停下来休息s秒。对于不同的兔子，t，s的数值是不同的，但是所有的乌龟却是一致——它们不到终点决不停止。

　　然而有些比赛相当漫长，全程观看会耗费大量时间，而小华发现只要在每场比赛开始后记录下兔子和乌龟的数据——兔子的速度v1（表示每秒兔子能跑v1米），乌龟的速度v2，以及兔子对应的t，s值，以及赛道的长度l——就能预测出比赛的结果。但是小华很懒，不想通过手工计算推测出比赛的结果，于是他找到了你——清华大学计算机系的高才生——请求帮助，请你写一个程序，对于输入的一场比赛的数据v1，v2，t，s，l，预测该场比赛的结果。

输入格式

　　输入只有一行，包含用空格隔开的五个正整数v1，v2，t，s，l，其中(v1,v2<=100;t<=300;s<=10;l<=10000且为v1,v2的公倍数)

输出格式

　　输出包含两行，第一行输出比赛结果——一个大写字母“T”或“R”或“D”，分别表示乌龟获胜，兔子获胜，或者两者同时到达终点。

　　第二行输出一个正整数，表示获胜者（或者双方同时）到达终点所耗费的时间（秒数）。

样例输入

10 5 5 2 20

样例输出

D

4

样例输入

10 5 5 1 20

样例输出

R

3

样例输入

10 5 5 3 20

样例输出

T

4

import java.util.Scanner;

public class Main {

    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);

        int num[]=new int[5];

        for(int i=0;i<5;i++)

            num[i]=sc.nextInt();

        int sum1=0,sum2=0,time=1;

        for(int t=1;t<=num[4]/num[0];time++){

            sum1+=num[0]\*t;

            sum2+=num[1]\*t;

            if(sum1==num[4]||sum2==num[4])

                break;

            if(sum1-sum2>=num[2]){

                sum2+=num[1]\*num[3];

                time+=num[3];

            }

        }

        if(sum2>sum1){

            System.out.println("T");

            System.out.println(num[4]/num[1]);

        }

        else if(sum2<sum1){

            System.out.println("R");

            System.out.println(time);

        }

        else{

            System.out.println("D");

            System.out.println(num[4]/num[1]);

        }

    }

}

回形取数 (vip)

问题描述

　　回形取数就是沿矩阵的边取数，若当前方向上无数可取或已经取过，则左转90度。一开始位于矩阵左上角，方向向下。

输入格式

　　输入第一行是两个不超过200的正整数m, n，表示矩阵的行和列。接下来m行每行n个整数，表示这个矩阵。

输出格式

　　输出只有一行，共mn个数，为输入矩阵回形取数得到的结果。数之间用一个空格分隔，行末不要有多余的空格。

样例输入

3 3

1 2 3

4 5 6

7 8 9

样例输出

1 4 7 8 9 6 3 2 5

样例输入

3 2

1 2

3 4

5 6

样例输出

1 3 5 6 4 2

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Scanner;

public class Main

{

    public static boolean[][] visited;

    public static void main(String[] args)

    {

        Scanner scanner = new Scanner(System.in);

        while(scanner.hasNext())

        {

            int x = scanner.nextInt();

            int y = scanner.nextInt();

            int[][] num = new int[x][y];

            visited = new boolean[x][y];

            for(int i = 0 ; i < x; i++)

                for(int j = 0; j < y; j++)

                {

                    num[i][j] = scanner.nextInt();

                    visited[i][j] = false;

                }

            int time = 0;

            int a = -1, b = 0;

            while(time < x \* y)

            {

                while(a + 1 < x && visited[a + 1][b] == false)

                {

                    System.out.print(num[a + 1][b] + " ");

                    visited[a + 1][b] = true;

                    a++;

                    time++;

                }

                while(b + 1 < y && visited[a][b + 1] == false)

                {

                    System.out.print(num[a][b + 1] + " ");

                    visited[a][b + 1] = true;

                    b++;

                    time++;

                }

                while(a - 1 >= 0 && visited[a - 1][b] == false)

                {

                    System.out.print(num[a - 1][b] + " ");

                    visited[a - 1][b] = true;

                    a--;

                    time++;

                }

                while(b - 1 >= 0 && visited[a][b - 1] == false)

                {

                    System.out.print(num[a][b - 1] + " ");

                    visited[a][b - 1] = true;

                    b--;

                    time++;

                }

            }

        }

    }

}

报时助手 (vip)

问题描述

　　给定当前的时间，请用英文的读法将它读出来。

　　时间用时h和分m表示，在英文的读法中，读一个时间的方法是：

　　如果m为0，则将时读出来，然后加上“o'clock”，如3:00读作“three o'clock”。

　　如果m不为0，则将时读出来，然后将分读出来，如5:30读作“five thirty”。

　　时和分的读法使用的是英文数字的读法，其中0~20读作：

　　0:zero, 1: one, 2:two, 3:three, 4:four, 5:five, 6:six, 7:seven, 8:eight, 9:nine, 10:ten, 11:eleven, 12:twelve, 13:thirteen, 14:fourteen, 15:fifteen, 16:sixteen, 17:seventeen, 18:eighteen, 19:nineteen, 20:twenty。

　　30读作thirty，40读作forty，50读作fifty。

　　对于大于20小于60的数字，首先读整十的数，然后再加上个位数。如31首先读30再加1的读法，读作“thirty one”。

　　按上面的规则21:54读作“twenty one fifty four”，9:07读作“nine seven”，0:15读作“zero fifteen”。

输入格式

　　输入包含两个非负整数h和m，表示时间的时和分。非零的数字前没有前导0。h小于24，m小于60。

输出格式

　　输出时间时刻的英文。

样例输入

0 15

样例输出

zero fifteen

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Scanner;

public class Main

{

    public static String[] time = {"","zero","one","two","three","four","five","six","seven","eight","nine","ten","eleven","twelve","thirteen","fourteen","fifteen","sixteen","seventeen","eighteen","nineteen"};

    public static String[] time2 = {"twenty","thirty","forty","fifty"};

    public static void main(String[] args)

    {

        Scanner scanner = new Scanner(System.in);

        while(scanner.hasNext())

        {

            int h = scanner.nextInt();

            int m = scanner.nextInt();

            if(h < 20)

            {

                System.out.print(time[h + 1]);

            }

            else

            {

                System.out.print("twenty" + " " + time[h - 20]);

            }

            if(m == 0)

            {

                System.out.print(" " + "o'clock");

            }

            else

            {

                if(m < 20)

                {

                    System.out.print(" " + time[m + 1]);

                }

                else

                {

                    int a = m / 10;

                    int b = m % 10;

                    System.out.print(" " + time2[a - 2] + " " + time[b + 1]);

                }

            }

            System.out.println();

        }

    }

}

2n皇后问题 (vip)

问题描述

　　给定一个n\*n的棋盘，棋盘中有一些位置不能放皇后。现在要向棋盘中放入n个黑皇后和n个白皇后，使任意的两个黑皇后都不在同一行、同一列或同一条对角线上，任意的两个白皇后都不在同一行、同一列或同一条对角线上。问总共有多少种放法？n小于等于8。

输入格式

　　输入的第一行为一个整数n，表示棋盘的大小。

　　接下来n行，每行n个0或1的整数，如果一个整数为1，表示对应的位置可以放皇后，如果一个整数为0，表示对应的位置不可以放皇后。

输出格式

　　输出一个整数，表示总共有多少种放法。

样例输入

4

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

样例输出

2

样例输入

4

1 0 1 1

1 1 1 1

1 1 1 1

1 1 1 1

样例输出

0

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class L\_27 {

    static int[] table1=new int[9];

    static int sum=0,sum1=0;

    static boolean[] bool1=new  boolean[9];

    static boolean[] bool2=new  boolean[9];

    static List<String> list=new ArrayList<String>();

    public static void dfs(int k,int m,int[][] a,int[] table,boolean[] bool,boolean flag)

    {

        if(k==m+1)

        {

            if(flag){

                for (int i = 0; i <list.size(); i++) {

                    String table3=list.get(i);

                    int j = 1;

                    for (; j < m+1; j++) {

                        if(table1[j]==table3.charAt(j)-'0') break;

                    }

                    if(j==m+1)sum++;

                }

                return;//如果递归到9层返回

            }else{

                String s="";

                for (int j = 0; j <  m+1; j++) {

                    s=s+table[j];

                }

                list.add(s);

                return;

            }

        }

        for(int i=1;i<=m;i++)//试探每个位置

        {

            if(a[i-1][k-1]==0) continue; //判断可否放皇后

            int n=0;

            //判断对角线是否存在冲突

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

                if(table[k-j]+j==i || table[k-j]-j==i) break;

                n++;

            }

            if(n==k-1)

            {

                if(!bool[i])//判断每一行是否存在冲突

                {

                    table[k]=i;

                    bool[i]=true; //这行有皇后

                    dfs(k+1,m,a,table,bool,flag);

                    bool[i]=false;

                }

            }

        }

    }

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int n=Integer.parseInt(sc.nextLine());

        int[][] a=new int[n][n];

        for (int i = 0; i < n; i++) {

            String str=sc.nextLine();

            String[] strs=str.split(" ");

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

                a[i][j]=Integer.parseInt(strs[j]);

            }

        }

        long start=System.currentTimeMillis();

        int[] table2=new int[n+1];

        table1[1]=-2;

        table2[1]=-2;

        dfs(1,n,a,table2,bool2,false);

        dfs(1,n,a,table1,bool1,true);

        System.out.println(sum);

        long end=System.currentTimeMillis();

        System.out.println(end-start);

    }

}

Huffuman树 (vip)

问题描述

　　Huffman树在编码中有着广泛的应用。在这里，我们只关心Huffman树的构造过程。

　　给出一列数{pi}={p0, p1, …, pn-1}，用这列数构造Huffman树的过程如下：

　　1. 找到{pi}中最小的两个数，设为pa和pb，将pa和pb从{pi}中删除掉，然后将它们的和加入到{pi}中。这个过程的费用记为pa + pb。

　　2. 重复步骤1，直到{pi}中只剩下一个数。

　　在上面的操作过程中，把所有的费用相加，就得到了构造Huffman树的总费用。

　　本题任务：对于给定的一个数列，现在请你求出用该数列构造Huffman树的总费用。

　　例如，对于数列{pi}={5, 3, 8, 2, 9}，Huffman树的构造过程如下：

　　1. 找到{5, 3, 8, 2, 9}中最小的两个数，分别是2和3，从{pi}中删除它们并将和5加入，得到{5, 8, 9, 5}，费用为5。

　　2. 找到{5, 8, 9, 5}中最小的两个数，分别是5和5，从{pi}中删除它们并将和10加入，得到{8, 9, 10}，费用为10。

　　3. 找到{8, 9, 10}中最小的两个数，分别是8和9，从{pi}中删除它们并将和17加入，得到{10, 17}，费用为17。

　　4. 找到{10, 17}中最小的两个数，分别是10和17，从{pi}中删除它们并将和27加入，得到{27}，费用为27。

　　5. 现在，数列中只剩下一个数27，构造过程结束，总费用为5+10+17+27=59。

输入格式

　　输入的第一行包含一个正整数n（n<=100）。

　　接下来是n个正整数，表示p0, p1, …, pn-1，每个数不超过1000。

输出格式

　　输出用这些数构造Huffman树的总费用。

样例输入

5

5 3 8 2 9

样例输出

59

import java.util.Arrays;

import java.util.Scanner;

public class L\_28 {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int n=Integer.parseInt(sc.nextLine());

        String str=sc.nextLine();

        String[] strs=str.split(" ");

        int[] a=new int[n];

        int sum=0;

        for (int i = 0; i < n; i++) {

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

        }

        for (int i = 0; i < n-1; i++) {

            Arrays.sort(a);

            int f=a[0]+a[1];

            for (int j = 2; j < n-i; j++) {

                if(j==n-i-1){

                    a[j-2]=a[j];

                    a[j-1]=f;

                    a[j]=100001;

                }

                else a[j-2]=a[j];

            }

            sum+=f;

        }

        System.out.println(sum);

    }

}

高精度加法 (vip)

问题描述

　　输入两个整数a和b，输出这两个整数的和。a和b都不超过100位。

算法描述

　　由于a和b都比较大，所以不能直接使用语言中的标准数据类型来存储。对于这种问题，一般使用数组来处理。

　　定义一个数组A，A[0]用于存储a的个位，A[1]用于存储a的十位，依此类推。同样可以用一个数组B来存储b。

　　计算c = a + b的时候，首先将A[0]与B[0]相加，如果有进位产生，则把进位（即和的十位数）存入r，把和的个位数存入C[0]，即C[0]等于(A[0]+B[0])%10。然后计算A[1]与B[1]相加，这时还应将低位进上来的值r也加起来，即C[1]应该是A[1]、B[1]和r三个数的和．如果又有进位产生，则仍可将新的进位存入到r中，和的个位存到C[1]中。依此类推，即可求出C的所有位。

　　最后将C输出即可。

输入格式

　　输入包括两行，第一行为一个非负整数a，第二行为一个非负整数b。两个整数都不超过100位，两数的最高位都不是0。

输出格式

　　输出一行，表示a + b的值。

样例输入

20100122201001221234567890

2010012220100122

样例输出

20100122203011233454668012

import java.math.BigInteger;

import java.util.Scanner;

public class Main

{

    public static void main(String[] args)

    {

        Scanner scanner = new Scanner(System.in);

        while(scanner.hasNext())

        {

            String s1 = scanner.next();

            String s2 = scanner.next();

            BigInteger num1 = new BigInteger(s1);

            BigInteger num2 = new BigInteger(s2);

            System.out.println(num1.add(num2));

        }

    }

}

阶乘计算 (vip)

问题描述

　　输入一个正整数n，输出n!的值。

　　其中n!=1\*2\*3\*…\*n。

算法描述

　　n!可能很大，而计算机能表示的整数范围有限，需要使用高精度计算的方法。使用一个数组A来表示一个大整数a，A[0]表示a的个位，A[1]表示a的十位，依次类推。

　　将a乘以一个整数k变为将数组A的每一个元素都乘以k，请注意处理相应的进位。

　　首先将a设为1，然后乘2，乘3，当乘到n时，即得到了n!的值。

输入格式

　　输入包含一个正整数n，n<=1000。

输出格式

　　输出n!的准确值。

样例输入

10

样例输出

3628800

import java.util.\*;

import java.math.BigInteger;

public class Main {

public static void main(String[] args) {

    int n=new Scanner(System.in).nextInt();

    String s="1";

   BigInteger result=new BigInteger(s);

    for(int i=2;i<=n;i++){

        result=result.multiply(new BigInteger(String.valueOf(i)));

    }

System.out.println( result.toString());

    }

}

闰年判断

问题描述

给定一个年份，判断这一年是不是闰年。

当以下情况之一满足时，这一年是闰年：

1. 年份是4的倍数而不是100的倍数；

2. 年份是400的倍数。

其他的年份都不是闰年。

输入格式

输入包含一个整数y，表示当前的年份。

输出格式

输出一行，如果给定的年份是闰年，则输出yes，否则输出no。

样例输入

2013

样例输出

no

样例输入

2016

样例输出

yes

数据规模与约定

1990 <= y <= 2050。

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int year = scanner.nextInt();

        if (year % 400 == 0 || year % 4 == 0 && year % 100 != 0) {

            System.out.println("yes");

        } else {

            System.out.println("no");

        }

    }

}

01字串

问题描述

对于长度为5位的一个01串，每一位都可能是0或1，一共有32种可能。它们的前几个是：

00000

00001

00010

00011

00100

请按从小到大的顺序输出这32种01串。

输入格式

本试题没有输入。

输出格式

输出32行，按从小到大的顺序每行一个长度为5的01串。

样例输出

00000

00001

00010

00011

<以下部分省略>

public class Main {

    public static void main(String[] args) {

        for(int i=0;i<32;i++){

            System.out.println(i/16%2+""+i/8%2+""+i/4%2+""+i/2%2+""+i%2);

        }

    }

}

字母图形

问题描述

利用字母可以组成一些美丽的图形，下面给出了一个例子：

ABCDEFG

BABCDEF

CBABCDE

DCBABCD

EDCBABC

这是一个5行7列的图形，请找出这个图形的规律，并输出一个n行m列的图形。

输入格式

输入一行，包含两个整数n和m，分别表示你要输出的图形的行数的列数。

输出格式

输出n行，每个m个字符，为你的图形。

样例输入

5 7

样例输出

ABCDEFG

BABCDEF

CBABCDE

DCBABCD

EDCBABC

数据规模与约定

1 <= n, m <= 26。

import java.util.HashMap;

import java.util.Scanner;

public class Main

{

    public static void main(String[] args)

    {

        Scanner scanner = new Scanner(System.in);

        while(scanner.hasNext())

        {

            char[] c = {'A','B','C','D','E','F','G','H','I','J',

                    'K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};

            int a = scanner.nextInt();

            int b = scanner.nextInt();

            int j = 0;

            for(int i = 0; i < a; i++)

            {

                int l = j;

                for(int k = 0; k < b; k++)

                {

                    System.out.print(c[Math.abs(l++)]);

                }

                j--;

                System.out.println();

            }

        }

    }

}

数列特征

问题描述

给出n个数，找出这n个数的最大值，最小值，和。

输入格式

第一行为整数n，表示数的个数。

第二行有n个数，为给定的n个数，每个数的绝对值都小于10000。

输出格式

输出三行，每行一个整数。第一行表示这些数中的最大值，第二行表示这些数中的最小值，第三行表示这些数的和。

样例输入

5

1 3 -2 4 5

样例输出

5

-2

11

数据规模与约定

1 <= n <= 10000。

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        int[] a=new int [n];

        int max,min,total=0;

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

            a[i]=sc.nextInt();

        }

        max=a[0];min=a[0];total+=a[0];

        for (int i = 1; i < a.length; i++) {

            if(max<a[i])

                max=a[i];

            if(min>a[i])

                min=a[i];

            total+=a[i];

        }

        System.out.println(max);

        System.out.println(min);

        System.out.println(total);

    }

}

查找整数

问题描述

给出一个包含n个整数的数列，问整数a在数列中的第一次出现是第几个。

输入格式

第一行包含一个整数n。

第二行包含n个非负整数，为给定的数列，数列中的每个数都不大于10000。

第三行包含一个整数a，为待查找的数。

输出格式

如果a在数列中出现了，输出它第一次出现的位置(位置从1开始编号)，否则输出-1。

样例输入

6

1 9 4 8 3 9

9

样例输出

2

数据规模与约定

1 <= n <= 1000。

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        int [] a=new int [n];

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

            a[i]=sc.nextInt();

        }

        int f=0;

        int b=sc.nextInt();

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

            if(a[i]==b){

                f++;

                System.out.println(i+1);

            break;

            }

        }

        if(f==0)

            System.out.println(-1);

    }

}

杨辉三角形

问题描述

杨辉三角形又称Pascal三角形，它的第i+1行是(a+b)i的展开式的系数。

它的一个重要性质是：三角形中的每个数字等于它两肩上的数字相加。

下面给出了杨辉三角形的前4行：

1

1 1

1 2 1

1 3 3 1

给出n，输出它的前n行。

输入格式

输入包含一个数n。

输出格式

输出杨辉三角形的前n行。每一行从这一行的第一个数开始依次输出，中间使用一个空格分隔。请不要在前面输出多余的空格。

样例输入

4

样例输出

1

1 1

1 2 1

1 3 3 1

数据规模与约定

1 <= n <= 34。

import java.util.Scanner;

public class Main

{

    public int[] a = new int[35];

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        while(scanner.hasNext())

        {

            int n = scanner.nextInt();

            Main m = new Main();

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

            {

                m.a[1] = 1;

                m.a[i] = 1;

                for(int j = i - 1; j > 1; j--)

                {

                    m.a[j] = m.a[j] + m.a[j - 1];

                }

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

                {

                    System.out.print(m.a[j] + " ");

                }

                System.out.println();

            }

        }

    }

}

特殊的数字

问题描述

　　153是一个非常特殊的数，它等于它的每位数字的立方和，即153=1\*1\*1+5\*5\*5+3\*3\*3。编程求所有满足这种条件的三位十进制数。

输出格式

　　按从小到大的顺序输出满足条件的三位十进制数，每个数占一行。

public class Main {

    public static void main(String[] args) {

        int a,b,c;

        for (int i = 100; i < 1000; i++) {

            a=i/100;

            b=i/10-a\*10;

            c=i-a\*100-b\*10;

            if(i==a\*a\*a+b\*b\*b+c\*c\*c)

                System.out.println(i);

        }

    }

}

回文数

问题描述

　　1221是一个非常特殊的数，它从左边读和从右边读是一样的，编程求所有这样的四位十进制数。

输出格式

　　按从小到大的顺序输出满足条件的四位十进制数。

public class Main {

    public static void main(String[] args) {

        int a,b,c,d;

        for (int i = 1000; i < 10000; i++) {

            a=i/1000;

            b=i/100-a\*10;

            c=i/10-a\*100-b\*10;

            d=i-a\*1000-b\*100-c\*10;

            if(a==d&&c==b){

                System.out.println(i);

            }

        }

    }

}

特殊回文数

问题描述

　　123321是一个非常特殊的数，它从左边读和从右边读是一样的。

　　输入一个正整数n， 编程求所有这样的五位和六位十进制数，满足各位数字之和等于n 。

输入格式

　　输入一行，包含一个正整数n。

输出格式

　　按从小到大的顺序输出满足条件的整数，每个整数占一行。

样例输入

52

样例输出

899998

989989

998899

数据规模和约定

　　1<=n<=54。

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int m=sc.nextInt();

        int a,b,c,d,e,f;

        for(int n=10000;n<100000;n++)

        {

            a = n/10000;

            b = (n - a\*10000)/1000;

            c = (n - a\*10000 - b\*1000)/100;

            d = (n - a\*10000 - b\*1000 - c\*100)/10;

            e = (n - a\*10000 - b\*1000 - c\*100)%10;

            if(a == e && b == d)

            {

                if(a+b+c+d+e == m)

                {

                    System.out.println(n);

                }

            }

        }

        for(int n=100000;n<1000000;n++)

        {

            a = n/100000;

            b = (n - a\*100000)/10000;

            c = (n - a\*100000 - b\*10000)/1000;

            d = (n - a\*100000 - b\*10000 - c\*1000)/100;

            e = (n - a\*100000 - b\*10000 - c\*1000 - d\*100)/10;

            f = (n - a\*100000 - b\*10000 - c\*1000 - d\*100)%10;

            if(a ==f && b ==e && c == d)

            {

                if((a+b+c+d+e+f) == m)

                {

                   System.out.println(n);

                }

            }

        }

    }

}

十进制转十六进制

问题描述

　　十六进制数是在程序设计时经常要使用到的一种整数的表示方式。它有0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F共16个符号，分别表示十进制数的0至15。十六进制的计数方法是满16进1，所以十进制数16在十六进制中是10，而十进制的17在十六进制中是11，以此类推，十进制的30在十六进制中是1E。

　　给出一个非负整数，将它表示成十六进制的形式。

输入格式

　　输入包含一个非负整数a，表示要转换的数。0<=a<=2147483647

输出格式

　　输出这个整数的16进制表示

样例输入

30

样例输出

1E

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int c=sc.nextInt();

        System.out.println(Integer.toHexString(c).toUpperCase());

    }

}

十六进制转十进制

问题描述

　　从键盘输入一个不超过8位的正的十六进制数字符串，将它转换为正的十进制数后输出。

　　注：十六进制数中的10~15分别用大写的英文字母A、B、C、D、E、F表示。

样例输入

FFFF

样例输出

65535

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        String s=sc.next();

        System.out.println(Long.parseLong(s,16));

    }

}

十六进制转八进制

问题描述

　　给定n个十六进制正整数，输出它们对应的八进制数。

输入格式

　　输入的第一行为一个正整数n （1<=n<=10）。

　　接下来n行，每行一个由0~9、大写字母A~F组成的字符串，表示要转换的十六进制正整数，每个十六进制数长度不超过100000。

输出格式

　　输出n行，每行为输入对应的八进制正整数。

注意

　　输入的十六进制数不会有前导0，比如012A。

　　输出的八进制数也不能有前导0。

样例输入

2

39

123ABC

样例输出

71

4435274

提示

　　先将十六进制数转换成某进制数，再由某进制数转换成八进制。

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

       String c=sc.next();

      //  System.out.println(Integer.toHexString(c).toUpperCase());//十进制转换成十六进制

       // System.out.println(Integer.toBinaryString(c).toUpperCase());//十进制转换成2进制

      //  System.out.println(Integer.toOctalString(c).toUpperCase());//十进制转换成八进制

      // System.out.println(Integer.valueOf(c,16).toString());//十六进制转换成十进制

       String t=Integer.valueOf(c, 16).toString();

       int n=Integer.parseInt(t);//先将十六进制转换成字符串然后将字符串转换成整数

       System.out.println(Integer.toOctalString(n).toUpperCase());//整数转换成八进制

    }

}

数列排序

问题描述

　　给定一个长度为n的数列，将这个数列按从小到大的顺序排列。1<=n<=200

输入格式

　　第一行为一个整数n。

　　第二行包含n个整数，为待排序的数，每个整数的绝对值小于10000。

输出格式

　　输出一行，按从小到大的顺序输出排序后的数列。

样例输入

5

8 3 6 4 9

样例输出

3 4 6 8 9

import java.text.Format;

import java.util.Arrays;

import java.util.Scanner;

public class Main

{

    public static void main(String[] args)

    {

        Scanner scanner = new Scanner(System.in);

        while(scanner.hasNext())

        {

            int n = scanner.nextInt();

            int[] num = new int[n];

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

            {

                num[i] = scanner.nextInt();

            }

            Arrays.sort(num);

            for(int a : num)

            {

                System.out.print(a + " ");

            }

            System.out.println();

        }

    }

}

***算法训练***

摆动序列 (vip)

问题描述

　　如果一个序列满足下面的性质，我们就将它称为摆动序列：

　　1. 序列中的所有数都是不大于k的正整数；

　　2. 序列中至少有两个数。

　　3. 序列中的数两两不相等；

　　4. 如果第i – 1个数比第i – 2个数大，则第i个数比第i – 2个数小；如果第i – 1个数比第i – 2个数小，则第i个数比第i – 2个数大。

　　比如，当k = 3时，有下面几个这样的序列：

　　1 2

　　1 3

　　2 1

　　2 1 3

　　2 3

　　2 3 1

　　3 1

　　3 2

　　一共有8种，给定k，请求出满足上面要求的序列的个数。

输入格式

　　输入包含了一个整数k。（k<=20）

输出格式

　　输出一个整数，表示满足要求的序列个数。

样例输入

3

样例输出

8

集合运算 (vip)

问题描述

　　给出两个整数集合A、B，求出他们的交集、并集以及B在A中的余集。

输入格式

　　第一行为一个整数n，表示集合A中的元素个数。

　　第二行有n个互不相同的用空格隔开的整数，表示集合A中的元素。

　　第三行为一个整数m，表示集合B中的元素个数。

　　第四行有m个互不相同的用空格隔开的整数，表示集合B中的元素。

　　集合中的所有元素均为int范围内的整数，n、m<=1000。

输出格式

　　第一行按从小到大的顺序输出A、B交集中的所有元素。

　　第二行按从小到大的顺序输出A、B并集中的所有元素。

　　第三行按从小到大的顺序输出B在A中的余集中的所有元素。

样例输入

5

1 2 3 4 5

5

2 4 6 8 10

样例输出

2 4

1 2 3 4 5 6 8 10

1 3 5

样例输入

4

1 2 3 4

3

5 6 7

样例输出

1 2 3 4 5 6 7

1 2 3 4

瓷砖铺放 (vip)

问题描述

　　有一长度为N(1<=Ｎ<=10)的地板，给定两种不同瓷砖：一种长度为1，另一种长度为2，数目不限。要将这个长度为N的地板铺满，一共有多少种不同的铺法？

　　例如，长度为4的地面一共有如下5种铺法：

　　4=1+1+1+1

　　4=2+1+1

　　4=1+2+1

　　4=1+1+2

　　4=2+2

　　编程用递归的方法求解上述问题。

输入格式

　　只有一个数N，代表地板的长度

输出格式

　　输出一个数，代表所有不同的瓷砖铺放方法的总数

样例输入

4

样例输出

5

import java.util.Scanner;

public class Main

{

    public static void main(String[] args)

    {

        Scanner scanner = new Scanner(System.in);

        while(scanner.hasNext())

        {

            int n = scanner.nextInt();

            int c = 0;

            int a = 1;

            int b = 1;

            if(n == 1 || n == 2)

                System.out.println(1);

            else

            {

                for(int i = 3; i <= n + 1; i++)

                {

                    c = a + b;

                    a = b;

                    b = c;

                }

                System.out.println(c);

            }

        }

    }

}

幂方分解 (vip)

问题描述

　　任何一个正整数都可以用2的幂次方表示。例如：

　　137=27+23+20

　　同时约定方次用括号来表示，即ab 可表示为a（b）。

　　由此可知，137可表示为：

　　2（7）+2（3）+2（0）

　　进一步：7= 22+2+20 （21用2表示）

　　3=2+20

　　所以最后137可表示为：

　　2（2（2）+2+2（0））+2（2+2（0））+2（0）

　　又如：

　　1315=210 +28 +25 +2+1

　　所以1315最后可表示为：

　　2（2（2+2（0））+2）+2（2（2+2（0）））+2（2（2）+2（0））+2+2（0）

输入格式

　　输入包含一个正整数N（N<=20000），为要求分解的整数。

输出格式

　　程序输出包含一行字符串，为符合约定的n的0，2表示（在表示中不能有空格）

拦截导弹 (vip)

问题描述

　　某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。某天，雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。

　　输入导弹依次飞来的高度（雷达给出的高度数据是不大于30000的正整数），计算这套系统最多能拦截多少导弹，如果要拦截所有导弹最少要配备多少套这种导弹拦截系统。

输入格式

　　一行，为导弹依次飞来的高度

输出格式

　　两行，分别是最多能拦截的导弹数与要拦截所有导弹最少要配备的系统数

样例输入

389 207 155 300 299 170 158 65

样例输出

6

2

回文数 (vip)

问题描述

　　若一个数（首位不为零）从左向右读与从右向左读都一样，我们就将其称之为回文数。

　　例如：给定一个10进制数56，将56加65（即把56从右向左读），得到121是一个回文数。

　　又如：对于10进制数87：

　　STEP1：87+78 = 165 STEP2：165+561 = 726

　　STEP3：726+627 = 1353 STEP4：1353+3531 = 4884

　　在这里的一步是指进行了一次N进制的加法，上例最少用了4步得到回文数4884。

　　写一个程序，给定一个N（2<=N<=10或N=16）进制数M（其中16进制数字为0-9与A-F），求最少经过几步可以得到回文数。

　　如果在30步以内（包含30步）不可能得到回文数，则输出“Impossible!”

输入格式

　　两行，N与M

输出格式

　　如果能在30步以内得到回文数，输出“STEP=xx”（不含引号），其中xx是步数；否则输出一行”Impossible!”（不含引号）

样例输入

9

87

样例输出

STEP=6

import java.io.\*;

public class Main {

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

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

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

        String str=bf.readLine();

        for(int count=0;count<=30;count++){

            char c1[]=str.toCharArray();

            int k=c1.length;

            String s1=new StringBuffer(str).reverse().toString();

                if(s1.equals(str)){

                    System.out.println("STEP="+count);

                    return;

                }else{

                        int a=0,b=0;

                        StringBuffer s2=new StringBuffer();

                for(int j=0;j<k;j++){

                    if(c1[j]<='9'&&c1[k-1-j]<='9'){

                           a=(c1[j]-'0'+c1[k-1-j]-'0'+b)%n;

                            b=(c1[j]-'0'+c1[k-1-j]-'0'+b)/n;

                               if(a>9)

                                   s2.append((char)(a+55));

                                   else

                                       s2.append(a);

                            }

                    if(c1[j]>'9'&&c1[k-1-j]>'9'){

                           a=(c1[j]-'A'+10+c1[k-1-j]-'A'+10+b)%n;

                            b=(c1[j]-'A'+10+c1[k-1-j]-'A'+10+b)/n;

                             if(a>9)

                                   s2.append((char)(a+55));

                                   else

                                       s2.append(a);

                            }

                    if(c1[j]>'9'&&c1[k-1-j]<='9'){

                           a=(c1[j]-'A'+10+c1[k-1-j]-'0'+b)%n;

                            b=(c1[j]-'A'+10+c1[k-1-j]-'0'+b)/n;

                             if(a>9)

                                   s2.append((char)(a+55));

                                   else

                                       s2.append(a);

                            }

                    if(c1[j]<='9'&&c1[k-1-j]>'9'){

                           a=(c1[j]-'0'+c1[k-1-j]-'A'+10+b)%n;

                            b=(c1[j]-'0'+c1[k-1-j]-'A'+10+b)/n;

                             if(a>9)

                                   s2.append((char)(a+55));

                                   else

                                       s2.append(a);

                            }

                    if(j==k-1&&b!=0)

                        s2.append(b);

                }

                str=s2.reverse().toString();

            }

                }

        System.out.println("Impossible!");

}

}

旅行家的预算 (vip)

问题描述

　　一个旅行家想驾驶汽车以最少的费用从一个城市到另一个城市（假设出发时油箱是空的）。给定两个城市之间的距离D1、汽车油箱的容量C（以升为单位）、每升汽油能行驶的距离D2、出发点每升汽油价格P和沿途油站数N（N可以为零），油站i离出发点的距离Di、每升汽油价格Pi（i=1，2，……N）。计算结果四舍五入至小数点后两位。如果无法到达目的地，则输出“No Solution”。

输入格式

　　第一行为4个实数D1、C、D2、P与一个非负整数N；

　　接下来N行，每行两个实数Di、Pi。

输出格式

　　如果可以到达目的地，输出一个实数（四舍五入至小数点后两位），表示最小费用；否则输出“No Solution”（不含引号）。

样例输入

275.6 11.9 27.4 2.8 2

102.0 2.9

220.0 2.2

样例输出

26.95

进制转换 (vip)

问题描述

　　我们可以用这样的方式来表示一个十进制数： 将每个阿拉伯数字乘以一个以该数字所处位置的（值减１）为指数，以１０为底数的幂之和的形式。例如：１２３可表示为 １＊１０２＋２＊１０１＋３＊１００这样的形式。

　　与之相似的，对二进制数来说，也可表示成每个二进制数码乘以一个以该数字所处位置的（值－１）为指数，以２为底数的幂之和的形式。一般说来，任何一个正整数Ｒ或一个负整数－Ｒ都可以被选来作为一个数制系统的基数。如果是以Ｒ或－Ｒ为基数，则需要用到的数码为 ０，１，．．．．Ｒ－１。例如，当Ｒ＝７时，所需用到的数码是０，１，２，３，４，５和６，这与其是Ｒ或－Ｒ无关。如果作为基数的数绝对值超过１０，则为了表示这些数码，通常使用英文字母来表示那些大于９的数码。例如对１６进制数来说，用Ａ表示１０，用Ｂ表示１１，用Ｃ表示１２，用Ｄ表示１３，用Ｅ表示１４，用Ｆ表示１５。

　　在负进制数中是用－Ｒ 作为基数，例如－１５（十进制）相当于１１０００１（－２进制），并且它可以被表示为２的幂级数的和数：

　　１１０００１＝１＊（－２）５＋１＊（－２）４＋０＊（－２）３＋０＊（－２）２＋

　　０＊（－２）１ ＋１＊（－２）０

　　 设计一个程序，读入一个十进制数和一个负进制数的基数, 并将此十进制数转换为此负进制下的数： －Ｒ∈｛－２，－３，－４，．．．，－２０｝

输入格式

　　一行两个数，第一个是十进制数Ｎ（－32768＜＝Ｎ＜＝32767）， 第二个是负进制数的基数－Ｒ。

输出格式

　　输出所求负进制数及其基数，若此基数超过１０，则参照１６进制的方式处理。（格式参照样例）

样例输入

30000 -2

样例输出

30000=11011010101110000(base-2)

样例输入

-20000 -2

样例输出

-20000=1111011000100000(base-2)

样例输入

28800 -16

样例输出

28800=19180(base-16)

样例输入

-25000 -16

样例输出

-25000=7FB8(base-16)

乘积最大 (vip)

问题描述

　　今年是国际数学联盟确定的“2000——世界数学年”，又恰逢我国著名数学家华罗庚先生诞辰90周年。在华罗庚先生的家乡江苏金坛，组织了一场别开生面的数学智力竞赛的活动，你的一个好朋友XZ也有幸得以参加。活动中，主持人给所有参加活动的选手出了这样一道题目：

　　设有一个长度为N的数字串，要求选手使用K个乘号将它分成K+1个部分，找出一种分法，使得这K+1个部分的乘积能够为最大。

　　同时，为了帮助选手能够正确理解题意，主持人还举了如下的一个例子：

　　有一个数字串：312， 当N=3，K=1时会有以下两种分法：

　　3\*12=36

　　31\*2=62

　　这时，符合题目要求的结果是：31\*2=62

　　现在，请你帮助你的好朋友XZ设计一个程序，求得正确的答案。

输入格式

　　程序的输入共有两行：

　　第一行共有2个自然数N，K（6≤N≤40，1≤K≤6）

　　第二行是一个长度为N的数字串。

输出格式

　　输出所求得的最大乘积（一个自然数）。

样例输入

4 2

1231

样例输出

62

单词接龙 (vip)

问题描述

　　单词接龙是一个与我们经常玩的成语接龙相类似的游戏，现在我们已知一组单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”（每个单词都最多在“龙”中出现两次），在两个单词相连时，其重合部分合为一部分，例如 beast和astonish，如果接成一条龙则变为beastonish，另外相邻的两部分不能存在包含关系，例如at 和 atide 间不能相连。

输入格式

　　输入的第一行为一个单独的整数n (n<=20)表示单词数，以下n 行每行有一个单词，输入的最后一行为一个单个字符，表示“龙”开头的字母。你可以假定以此字母开头的“龙”一定存在.

输出格式

　　只需输出以此字母开头的最长的“龙”的长度

样例输入

5

at

touch

cheat

choose

tact

a

样例输出

23

样例说明

　　连成的“龙”为atoucheatactactouchoose

方格取数 (vip)

问题描述

　　设有N\*N的方格图(N<=10),我们将其中的某些方格中填入正整数,而其他的方格中则放入数字0。

　　某人从图的左上角的A 点(1,1)出发，可以向下行走，也可以向右走，直到到达右下角的B点(N,N)。在走过的路上，他可以取走方格中的数（取走后的方格中将变为数字0）。

　　此人从A点到B 点共走两次，试找出2条这样的路径，使得取得的数之和为最大。

输入格式

　　输入的第一行为一个整数N（表示N\*N的方格图），接下来的每行有三个整数，前两个表示位置，第三个数为该位置上所放的数。一行单独的0表示输入结束。

输出格式

　　只需输出一个整数，表示2条路径上取得的最大的和。

样例输入

8

2 3 13

2 6 6

3 5 7

4 4 14

5 2 21

5 6 4

6 3 15

7 2 14

0 0 0

样例输出

67

求先序排列 (vip)

问题描述

　　给出一棵二叉树的中序与后序排列。求出它的先序排列。（约定树结点用不同的大写字母表示，长度<=8）。

输入格式

　　两行，每行一个字符串，分别表示中序和后序排列

输出格式

　　一个字符串，表示所求先序排列

样例输入

BADC

BDCA

样例输出

ABCD

装箱问题 (vip)

问题描述

　　有一个箱子容量为V（正整数，0＜＝V＜＝20000），同时有n个物品（0＜n＜＝30），每个物品有一个体积（正整数）。

　　要求n个物品中，任取若干个装入箱内，使箱子的剩余空间为最小。

输入格式

　　第一行为一个整数，表示箱子容量；

　　第二行为一个整数，表示有n个物品；

　　接下来n行，每行一个整数表示这n个物品的各自体积。

输出格式

　　一个整数，表示箱子剩余空间。

样例输入

24

6

8

3

12

7

9

7

样例输出

0

import java.io.\*;

import java.util.\*;

public class Main {

    private static int v[][]=new int[31][20001];

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

    //Scanner sc=new Scanner(System.in);

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

    //int n=sc.nextInt();

    //int m=sc.nextInt();

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

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

    int a[]=new int[m+1];

    for(int i=1;i<=m;i++)

        a[i]=Integer.parseInt(bf.readLine());

        //a[i]=sc.nextInt();

    for(int i=1;i<=m;i++){

        for(int j=1;j<=n;j++){

            if(j<a[i])

                v[i][j]=v[i-1][j];

            else

                v[i][j]=Math.max(v[i-1][j], v[i-1][j-a[i]]+a[i]);

                    }

            }

    System.out.println(n-v[m][n]);

}

}

数的划分 (vip)

问题描述

　　将整数n分成k份，且每份不能为空，任意两份不能相同(不考虑顺序)。

　　例如：n=7，k=3，下面三种分法被认为是相同的。

　　1，1，5; 1，5，1; 5，1，1;

　　问有多少种不同的分法。

输入格式

　　n，k

输出格式

　　一个整数，即不同的分法

样例输入

7 3

样例输出

4 {四种分法为：1，1，5;1，2，4;1，3，3;2，2，3;}

数据规模和约定

　　6<n<=200，2<=k<=6

一元三次方程求解 (vip)

问题描述

　　有形如：ax3+bx2+cx+d=0 这样的一个一元三次方程。给出该方程中各项的系数(a，b，c，d 均为实数)，并约定该方程存在三个不同实根(根的范围在-100至100之间)，且根与根之差的绝对值>=1。要求三个实根。。

输入格式

　　四个实数：a，b，c，d

输出格式

　　由小到大依次在同一行输出这三个实根(根与根之间留有空格)，并精确到小数点后2位

样例输入

1 -5 -4 20

样例输出

-2.00 2.00 5.00

数据规模和约定

　　|a|，|b|，|c|，|d|<=10

统计单词个数 (vip)

问题描述

　　给出一个长度不超过200的由小写英文字母组成的字母串(约定;该字串以每行20个字母的方式输入，且保证每行一定为20个)。要求将此字母串分成k份 (1<k<=40)，且每份中包含的单词个数加起来总数最大(每份中包含的单词可以部分重叠。当选用一个单词之后，其第一个字母不能再用。例 如字符串this中可包含this和is，选用this之后就不能包含th)。

　　单词在给出的一个不超过6个单词的字典中。

　　要求输出最大的个数。

输入格式

　　第一行有二个正整数(p，k)

　　p表示字串的行数;

　　k表示分为k个部分。

　　接下来的p行，每行均有20个字符。

　　再接下来有一个正整数s，表示字典中单词个数。(1<=s<=6)

　　接下来的s行，每行均有一个单词。

输出格式

　　每行一个整数，分别对应每组测试数据的相应结果。

样例输入

1 3

thisisabookyouareaoh

4

is

a

ok

sab

样例输出

7

数据规模和约定

　　长度不超过200，1<k<=40，字典中的单词数不超过6。

Car的旅行路线 (vip)

问题描述

　　又到暑假了，住在城市A的Car想和朋友一起去城市B旅游。她知道每个城市都有四个飞机场，分别位于一个矩形的四个顶点上，同一个城市中两个机场之间有一 条笔直的高速铁路，第I个城市中高速铁路了的单位里程价格为Ti，任意两个不同城市的机场之间均有航线，所有航线单位里程的价格均为t。

　　那么Car应如何安排到城市B的路线才能尽可能的节省花费呢?她发现这并不是一个简单的问题，于是她来向你请教。

　　找出一条从城市A到B的旅游路线，出发和到达城市中的机场可以任意选取，要求总的花费最少。

输入格式

　　的第一行有四个正整数s，t，A，B。

　　S表示城市的个数，t表示飞机单位里程的价格，A，B分别为城市A，B的序号，(1<=A，B<=S)。

　　接下来有S行，其中第I行均有7个正整数xi1，yi1，xi2，yi2，xi3，yi3，Ti，这当中的(xi1，yi1)，(xi2，yi2)，(xi3，yi3)分别是第I个城市中任意三个机场的坐标，T I为第I个城市高速铁路单位里程的价格。

输出格式

　　共有n行，每行一个数据对应测试数据，保留一位小数。

样例输入

1

1 10 1 3

1 1 1 3 3 1 30

2 5 7 4 5 2 1

8 6 8 8 11 6 3

样例输出

47.55

数据规模和约定

　　0<S<=100，

麦森数 (vip)

问题描述

　　形如2P-1的素数称为麦森数，这时P一定也是个素数。但反过来不一定，即如果P是个素数，2P-1不一定也是素数。到1998年底，人们已找到了37个麦森数。最大的一个是P=3021377，它有909526位。麦森数有许多重要应用，它与完全数密切相关。

　　任务：从文件中输入P（1000<P<3100000），计算2P-1的位数和最后500位数字（用十进制高精度数表示）

输入格式

　　文件中只包含一个整数P（1000<P<3100000）

输出格式

　　第一行：十进制高精度数2P-1的位数。

　　第2-11行：十进制高精度数2P-1的最后500位数字。（每行输出50位，共输出10行，不足500位时高位补0）

　　不必验证2P-1与P是否为素数。

样例输入

1279

样例输出

386

00000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000

00000000000000104079321946643990819252403273640855

38615262247266704805319112350403608059673360298012

23944173232418484242161395428100779138356624832346

49081399066056773207629241295093892203457731833496

61583550472959420547689811211693677147548478866962

50138443826029173234888531116082853841658502825560

46662248318909188018470682222031405210266984354887

32958028878050869736186900714720710555703168729087

FBI树 (vip)

问题描述

　　我们可以把由“0”和“1”组成的字符串分为三类：全“0”串称为B串，全“1”串称为I串，既含“0”又含“1”的串则称为F串。

　　FBI树是一种二叉树，它的结点类型也包括F结点，B结点和I结点三种。由一个长度为2N的“01”串S可以构造出一棵FBI树T，递归的构造方法如下：

　　1)T的根结点为R，其类型与串S的类型相同；

　　2)若串S的长度大于1，将串S从中间分开，分为等长的左右子串S1和S2；由左子串S1构造R的左子树T1，由右子串S2构造R的右子树T2。

　　现在给定一个长度为2N的“01”串，请用上述构造方法构造出一棵FBI树，并输出它的后序遍历序列。

输入格式

　　第一行是一个整数N（0 <= N <= 10），第二行是一个长度为2N的“01”串。

输出格式

　　包括一行，这一行只包含一个字符串，即FBI树的后序遍历序列。

样例输入

3

10001011

样例输出

IBFBBBFIBFIIIFF

数据规模和约定

　　对于40%的数据，N <= 2；

　　对于全部的数据，N <= 10。

　　注：

　　[1] 二叉树：二叉树是结点的有限集合，这个集合或为空集，或由一个根结点和两棵不相交的二叉树组成。这两棵不相交的二叉树分别称为这个根结点的左子树和右子树。

　　[2] 后序遍历：后序遍历是深度优先遍历二叉树的一种方法，它的递归定义是：先后序遍历左子树，再后序遍历右子树，最后访问根。

星际交流 (vip)

问题描述

　　人类终于登上了火星的土地并且见到了神秘的火星人。人类和火星人都无法理解对方的语言，但是我们的科学家发明了一种用数字交流的方法。这种交流方法是这样 的，首先，火星人把一个非常大的数字告诉人类科学家，科学家破解这个数字的含义后，再把一个很小的数字加到这个大数上面，把结果告诉火星人，作为人类的回 答。

　　火星人用一种非常简单的方式来表示数字——掰手指。火星人只有一只手，但这只手上有成千上万的手指，这些手指排成一列，分别编号为1，2，3……。火星人的任意两根手指都能随意交换位置，他们就是通过这方法计数的。

　　一个火星人用一个人类的手演示了如何用手指计数。如果把五根手指——拇指、食指、中指、无名指和小指分别编号为1，2，3，4和5，当它们按正常顺序排列 时，形成了5位数12345，当你交换无名指和小指的位置时，会形成5位数12354，当你把五个手指的顺序完全颠倒时，会形成54321，在所有能够形 成的120个5位数中，12345最小，它表示1；12354第二小，它表示2；54321最大，它表示120。下表展示了只有3根手指时能够形成的6个 3位数和它们代表的数字：

　　三进制数

　　123

　　132

　　213

　　231

　　312

　　321

　　代表的数字

　　1

　　2

　　3

　　4

　　5

　　6

　　现在你有幸成为了第一个和火星人交流的地球人。一个火星人会让你看他的手指，科学家会告诉你要加上去的很小的数。你的任务是，把火星人用手指表示的数与科 学家告诉你的数相加，并根据相加的结果改变火星人手指的排列顺序。输入数据保证这个结果不会超出火星人手指能表示的范围。

输入格式

　　包括三行，第一行有一个正整数N，表示火星人手指的数目（1 <= N <= 10000）。第二行是一个正整数M，表示要加上去的小整数（1 <= M <= 100）。下一行是1到N这N个整数的一个排列，用空格隔开，表示火星人手指的排列顺序。

输出格式

　　只有一行，这一行含有N个整数，表示改变后的火星人手指的排列顺序。每两个相邻的数中间用一个空格分开，不能有多余的空格。

样例输入

5

3

1 2 3 4 5

样例输出

1 2 4 5 3

数据规模和约定

　　对于30%的数据，N<=15；

　　对于60%的数据，N<=50；

　　对于全部的数据，N<=10000；

校门外的树 (vip)

问题描述

　　某校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置；数 轴上的每个整数点，即0，1，2，……，L，都种有一棵树。

　　由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已 知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树 都移走后，马路上还有多少棵树。

输入格式

　　输入文件的第一行有两个整数L（1 <= L <= 10000）和 M（1 <= M <= 100），L代表马路的长度，M代表区域的数目，L和M之间用一个空格隔开。接下来的M行每行包含两个不同的整数，用一个空格隔开，表示一个区域的起始点 和终止点的坐标。

输出格式

　　输出文件包括一行，这一行只包含一个整数，表示马路上剩余的树的数目。

样例输入

500 3

150 300

100 200

470 471

样例输出

298

数据规模和约定

　　对于20%的数据，区域之间没有重合的部分；

　　对于其它的数据，区域之间有重合的情况。

import java.util.\*;

public class Main {

public static void main(String[] args) {

    Scanner sc=new Scanner(System.in);

    int l=sc.nextInt();

    int m=sc.nextInt();

    int sum=0;

    int a[]=new int[10001];

    int x=0,y=0;

    for(int i=0;i<m;i++){

         x=sc.nextInt();

        y=sc.nextInt();

        for(int j=x;j<=y;j++)

            a[j]=1;

    }

        for(int i=0;i<10001;i++)

            if(a[i]==1)

                sum+=1;

        System.out.println(l-sum+1);

    }

}

入学考试 (vip)

问题描述

　　辰辰是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”

　　如果你是辰辰，你能完成这个任务吗？

输入格式

　　第一行有两个整数T（1 <= T <= 1000）和M（1 <= M <= 100），用一个空格隔开，T代表总共能够用来采药的时间，M代表山洞里的草药的数目。接下来的M行每行包括两个在1到100之间（包括1和100）的整数，分别表示采摘某株草药的时间和这株草药的价值。

输出格式

　　包括一行，这一行只包含一个整数，表示在规定的时间内，可以采到的草药的最大总价值。

样例输入

70 3

71 100

69 1

1 2

样例输出

3

数据规模和约定

　　对于30%的数据，M <= 10；

　　对于全部的数据，M <= 100。

开心的金明 (vip)

问题描述

　　金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间他自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎 么布置，你说了算，只要不超过N元钱就行”。今天一早金明就开始做预算，但是他想买的东西太多了，肯定会超过妈妈限定的N元。于是，他把每件物品规定了一 个重要度，分为5等：用整数1~5表示，第5等最重要。他还从因特网上查到了每件物品的价格（都是整数元）。他希望在不超过N元（可以等于N元）的前提 下，使每件物品的价格与重要度的乘积的总和最大。

　　设第j件物品的价格为v[j]，重要度为w[j]，共选中了k件物品，编号依次为 j1，j2，……，jk，则所求的总和为：

　　v[j1]\*w[j1]+v[j2]\*w[j2]+ …+v[jk]\*w[jk]。（其中\*为乘号）

　　请 你帮助金明设计一个满足要求的购物单。

输入格式

　　输入文件 的第1行，为两个正整数，用一个空格隔开：

　　N m

　　（其中N（<30000）表示总钱 数，m（<25）为希望购买物品的个数。）

　　从第2行到第m+1行，第j行给出了编号为j-1的物品的基本数据，每行有2个非负整数

　　v p

　　（其中v表示该物品的价格(v<=10000)，p表示该物品的重要度(1~5)）

输出格式

　　输出文件只有一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（<100000000）。

样例输入

1000 5

800 2

400 5

300 5

400 3

200 2

样例输出

3900

import java.util.\*;

public class Main {

    private static int t[]=new int[30000];

    private static int v[]=new int[30000];

    private static int s[]=new int[30000];

public static void main(String[] args) {

    Scanner sc=new Scanner(System.in);

    int time=sc.nextInt();

    int m=sc.nextInt();

    for(int i=1;i<=m;i++){

        t[i]=sc.nextInt();

        v[i]=sc.nextInt();

    }

    for(int i=1;i<=m;i++){

        for(int j=time;j>=t[i];j--){

            if(s[j-t[i]]+v[i]\*t[i]>s[j])

                s[j]=s[j-t[i]]+v[i]\*t[i];

        }

    }

    System.out.println(s[time]);

}

}

JAM计数法 (vip)

问题描述

　　Jam是个喜欢标新立异的科学怪人。他不使用阿拉伯数字计数，而是使用小写英文字母计数，他觉得这样做，会使世界更加丰富多彩。在他的计数法中，每个数字的位数都是相同的（使用相同个数的字母），英文字母按原先的顺序，排在前面的字母小于排在它后面的字母。我们把这样的“数字”称为Jam数字。在Jam数字中，每个字母互不相同，而且从左到右是严格递增的。每次，Jam还指定使用字母的范围，例如，从2到10，表示只能使用{b,c,d,e,f,g,h,i,j}这些字母。如果再规定位数为5，那么，紧接在Jam数字“bdfij”之后的数字应该是“bdghi”。（如果我们用U、V依次表示Jam数字“bdfij”与“bdghi”，则U<V< span>，且不存在Jam数字P，使U<P<V< span>）。你的任务是：对于从文件读入的一个Jam数字，按顺序输出紧接在后面的5个Jam数字，如果后面没有那么多Jam数字，那么有几个就输出几个。

输入格式

　　有2行，第1行为3个正整数，用一个空格隔开：

　　s t w

　　（其中s为所使用的最小的字母的序号，t为所使用的最大的字母的序号。w为数字的位数，这3个数满足：1≤s<T≤26, 2≤w≤t-s ）

　　第2行为具有w个小写字母的字符串，为一个符合要求的Jam数字。

　　所给的数据都是正确的，不必验证。

输出格式

　　最多为5行，为紧接在输入的Jam数字后面的5个Jam数字，如果后面没有那么多Jam数字，那么有几个就输出几个。每行只输出一个Jam数字，是由w个小写字母组成的字符串，不要有多余的空格。

样例输入

2 10 5

bdfij

样例输出

bdghi

bdghj

bdgij

bdhij

befgh

数列 (vip)

问题描述

　　给定一个正整数k(3≤k≤15),把所有k的方幂及所有有限个互不相等的k的方幂之和构成一个递增的序列，例如，当k=3时，这个序列是：

　　1，3，4，9，10，12，13，…

　　（该序列实际上就是：30，31，30+31，32，30+32，31+32，30+31+32，…）

　　请你求出这个序列的第N项的值（用10进制数表示）。

　　例如，对于k=3，N=100，正确答案应该是981。

输入格式

　　只有1行，为2个正整数，用一个空格隔开：

　　k N

　　（k、N的含义与上述的问题描述一致，且3≤k≤15，10≤N≤1000）。

输出格式

　　计算结果，是一个正整数（在所有的测试数据中，结果均不超过2.1\*109）。（整数前不要有空格和其他符号）。

样例输入

3 100

样例输出

981

import java.io.\*;

public class Main {

    static long a[]=new long[1001];

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

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

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

        int k=Integer.parseInt(s[0]);

        int n=Integer.parseInt(s[1]);

        int m=1;

        int sum=0;

        while(n>0){

            a[m++]=n%2;

            n=n/2;

        }

        m--;

     for(int i=1;i<=m;i++){

        sum+=a[i]\*add(k,i-1);

     }

        System.out.println(sum);

    }

    static int add(int k,int n){

        int sum=1;

        for(int i=1;i<=n;i++)

            sum\*=k;

            return sum;

    }

}

纪念品分组 (vip)

问题描述

　　元旦快到了，校学生会让乐乐负责新年晚会的纪念品发放工作。为使得参加晚会的同学所获得的纪念品价值 相对均衡，他要把购来的纪念品根据价格进行分组，但每组最多只能包括两件纪念品，并且每组纪念品的价格之和不能超过一个给定的整数。为了保证在尽量短的时 间内发完所有纪念品，乐乐希望分组的数目最少。

　　你的任务是写一个程序，找出所有分组方案中分组数最少的一种，输出最少的分组数目。

输入格式

　　输入包含n+2行：

　　第1行包括一个整数w，为每组纪念品价格之和的上限。

　　第2行为一个整数n，表示购来的纪念品的总件数。

　　第3~n+2行每行包含一个正整数pi (5 <= pi <= w)，表示所对应纪念品的价格。

输出格式

　　输出仅一行，包含一个整数，即最少的分组数目。

样例输入

100

9

90

20

20

30

50

60

70

80

90

样例输出

6

数据规模和约定

　　50%的数据满足：1 <= n <= 15

　　100%的数据满足：1 <= n <= 30000, 80 <= w <= 200

import java.io.\*;

import java.util.Arrays;

public class Main {

    static int a[]=new int[30001];

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

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

        int w=Integer.parseInt(bf.readLine());

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

        for(int i=1;i<30001;i++)

            a[i]=999999;

        int k=n;

        int sum=0;

        for(int i=1;i<=n;i++)

        a[i]=Integer.parseInt(bf.readLine());

        Arrays.sort(a);

        oter:for(int i=1;i<=n;i++){

            for(int j=n;j>0;j--){

                if(k==i){

                    sum++;

                    break oter;

                }

                if(k-i==1&&a[i]+a[k]>w){

                    sum+=2;

                    break oter;

                }

                if(k-i==1&&a[i]+a[k]<=w){

                    sum++;

                    break oter;

                }

                if(a[i]+a[k]>w){

                    k--;

                    sum++;

                    }

                else{

                    sum++;

                    k--;

                    break;

                }

            }

        }

        System.out.println(sum);

    }

}

传球游戏 (vip)

问题描述

　　上体育课的时候，小蛮的老师经常带着同学们一起做游戏。这次，老师带着同学们一起做传球游戏。

　　游戏规则是这样的：n个同学站成一个圆圈，其中的一个同学手里拿着一个球，当老师吹哨子时开始传球，每个同学可以把球传给自己左右的两个同学中的一个（左右任意），当老师再次吹哨子时，传球停止，此时，拿着球没传出去的那个同学就是败者，要给大家表演一个节目。

　　聪明的小蛮提出一个有趣的问题：有多少种不同的传球方法可以使得从小蛮手里开始传的球，传了m次以后，又回到小蛮手里。两种传球的方法被视作不同的方法，当且仅当这两种方法中，接到球的同学按接球顺序组成的序列是不同的。比如有3个同学1号、2号、3号，并假设小蛮为1号，球传了3次回到小蛮手里的方式有1->2->3->1和1->3->2->1，共2种。

输入格式

　　共一行，有两个用空格隔开的整数n，m（3<=n<=30，1<=m<=30）。

输出格式

　　t共一行，有一个整数，表示符合题意的方法数。

样例输入

3 3

样例输出

2

数据规模和约定

　　40%的数据满足：3<=n<=30，1<=m<=20

　　100%的数据满足：3<=n<=30，1<=m<=30

传纸条 (vip)

问题描述

　　小渊和小轩是好朋友也是同班同学，他们在一起总有谈不完的话题。一次素质拓展活动中，班上同学安排做成一个m行n列的矩阵，而小渊和小轩被安排在矩阵对角线的两端，因此，他们就无法直接交谈了。幸运的是，他们可以通过传纸条来进行交流。纸条要经由许多同学传到对方手里，小渊坐在矩阵的左上角，坐标(1,1)，小轩坐在矩阵的右下角，坐标(m,n)。从小渊传到小轩的纸条只可以向下或者向右传递，从小轩传给小渊的纸条只可以向上或者向左传递。

　　在活动进行中，小渊希望给小轩传递一张纸条，同时希望小轩给他回复。班里每个同学都可以帮他们传递，但只会帮他们一次，也就是说如果此人在小渊递给小轩纸条的时候帮忙，那么在小轩递给小渊的时候就不会再帮忙。反之亦然。

　　还有一件事情需要注意，全班每个同学愿意帮忙的好感度有高有低（注意：小渊和小轩的好心程度没有定义，输入时用0表示），可以用一个0-100的自然数来表示，数越大表示越好心。小渊和小轩希望尽可能找好心程度高的同学来帮忙传纸条，即找到来回两条传递路径，使得这两条路径上同学的好心程度只和最大。现在，请你帮助小渊和小轩找到这样的两条路径。

输入格式

　　输入第一行有2个用空格隔开的整数m和n，表示班里有m行n列（1<=m,n<=50）。

　　接下来的m行是一个m\*n的矩阵，矩阵中第i行j列的整数表示坐在第i行j列的学生的好心程度。每行的n个整数之间用空格隔开。

输出格式

　　输出一行，包含一个整数，表示来回两条路上参与传递纸条的学生的好心程度之和的最大值。

样例输入

3 3

0 3 9

2 8 5

5 7 0

样例输出

34

数据规模和约定

　　30%的数据满足：1<=m,n<=10

　　100%的数据满足：1<=m,n<=50

Hankson的趣味题 (vip)

问题描述

　　Hanks 博士是BT (Bio-Tech，生物技术) 领域的知名专家，他的儿子名叫Hankson。现 在，刚刚放学回家的Hankson 正在思考一个有趣的问题。 今天在课堂上，老师讲解了如何求两个正整数c1 和c2 的最大公约数和最小公倍数。现 在Hankson 认为自己已经熟练地掌握了这些知识，他开始思考一个“求公约数”和“求公 倍数”之类问题的“逆问题”，这个问题是这样的：已知正整数a0,a1,b0,b1，设某未知正整 数x 满足： 1． x 和a0 的最大公约数是a1； 2． x 和b0 的最小公倍数是b1。 Hankson 的“逆问题”就是求出满足条件的正整数x。但稍加思索之后，他发现这样的 x 并不唯一，甚至可能不存在。因此他转而开始考虑如何求解满足条件的x 的个数。请你帮 助他编程求解这个问题。

输入格式

　　输入第一行为一个正整数n，表示有n 组输入数据。

　　接下来的n 行每 行一组输入数据，为四个正整数a0，a1，b0，b1，每两个整数之间用一个空格隔开。输入 数据保证a0 能被a1 整除，b1 能被b0 整除。

输出格式

　　输出共n 行。每组输入数据的输出结果占一行，为一个整数。

　　对于每组数据：若不存在这样的 x，请输出0； 若存在这样的 x，请输出满足条件的x 的个数；

样例输入

2

41 1 96 288

95 1 37 1776

样例输出

6

2

样例说明

　　第一组输入数据，x 可以是9、18、36、72、144、288，共有6 个。

　　第二组输入数据，x 可以是48、1776，共有2 个。

数据规模和约定

　　对于 50%的数据，保证有1≤a0，a1，b0，b1≤10000 且n≤100。

　　对于 100%的数据，保证有1≤a0，a1，b0，b1≤2,000,000,000 且n≤2000。

接水问题 (vip)

问题描述

　　学校里有一个水房，水房里一共装有m 个龙头可供同学们打开水，每个龙头每秒钟的 供水量相等，均为1。 现在有n 名同学准备接水，他们的初始接水顺序已经确定。将这些同学按接水顺序从1 到n 编号，i 号同学的接水量为wi。接水开始时，1 到m 号同学各占一个水龙头，并同时打 开水龙头接水。当其中某名同学j 完成其接水量要求wj 后，下一名排队等候接水的同学k 马上接替j 同学的位置开始接水。这个换人的过程是瞬间完成的，且没有任何水的浪费。即 j 同学第x 秒结束时完成接水，则k 同学第x+1 秒立刻开始接水。若当前接水人数n’不足m， 则只有n’个龙头供水，其它m−n’个龙头关闭。 现在给出n 名同学的接水量，按照上述接水规则，问所有同学都接完水需要多少秒。

输入格式

　　第1 行2 个整数n 和m，用一个空格隔开，分别表示接水人数和龙头个数。 第2 行n 个整数w1、w2、……、wn，每两个整数之间用一个空格隔开，wi 表示i 号同 学的接水量。

输出格式

　　输出只有一行，1 个整数，表示接水所需的总时间。

样例输入

5 3

4 4 1 2 1

样例输出

4

样例输入

8 4

23 71 87 32 70 93 80 76

样例输出

163

输入输出样例 1 说明

　　第1 秒，3 人接水。第1 秒结束时，1、2、3 号同学每人的已接水量为1，3 号同学接完

　　水，4 号同学接替3 号同学开始接水。

　　第2 秒，3 人接水。第2 秒结束时，1、2 号同学每人的已接水量为2，4 号同学的已接

　　水量为1。

　　第3 秒，3 人接水。第3 秒结束时，1、2 号同学每人的已接水量为3，4 号同学的已接

　　水量为2。4 号同学接完水，5 号同学接替4 号同学开始接水。

　　第4 秒，3 人接水。第4 秒结束时，1、2 号同学每人的已接水量为4，5 号同学的已接

　　水量为1。1、2、5 号同学接完水，即所有人完成接水。

　　总接水时间为4 秒。

数据规模和约定

　　1 ≤ n ≤ 10000，1 ≤m≤ 100 且m≤ n；

　　1 ≤ wi ≤ 100。

区间k大数查询

问题描述

给定一个序列，每次询问序列中第l个数到第r个数中第K大的数是哪个。

输入格式

第一行包含一个数n，表示序列长度。

第二行包含n个正整数，表示给定的序列。

第三个包含一个正整数m，表示询问个数。

接下来m行，每行三个数l,r,K，表示询问序列从左往右第l个数到第r个数中，从大往小第K大的数是哪个。序列元素从1开始标号。

输出格式

总共输出m行，每行一个数，表示询问的答案。

样例输入

5

1 2 3 4 5

2

1 5 2

2 3 2

样例输出

4

2

数据规模与约定

对于30%的数据，n,m<=100；

对于100%的数据，n,m<=1000；

保证k<=(r-l+1)，序列中的数<=106。

import java.util.Arrays;

import java.util.Scanner;

public class Main {

public static void main(String[] args){

       Scanner in=new Scanner(System.in);

       int t;

       int n=in.nextInt();

       int arr[]=new int[n];

       int b[]=new int[1000];

       for (int i=0;i<n;i++)

           arr[i]=in.nextInt();

       int m=in.nextInt();

       int head[]=new int [m];

       int tail[]=new int[m];

       int key[]=new int[m];

       for (int i=0;i<m;i++){

           head[i]=in.nextInt();

           tail[i]=in.nextInt();

           key[i]=in.nextInt();

       }

       for (int i=0;i<m;i++){

            t=tail[i]-head[i]+1;

            for(int j=head[i]-1;j<tail[i];j++)

             b[j-head[i]+1]=arr[j];

            Arrays.sort(b,0,t);

            System.out.println(b[t-key[i]]);

            }

   }

}

最大最小公倍数

问题描述

已知一个正整数N，问从1~N中任选出三个数，他们的最小公倍数最大可以为多少。

输入格式

输入一个正整数N。

输出格式

输出一个整数，表示你找到的最小公倍数。

样例输入

9

样例输出

504

数据规模与约定

1 <= N <= 106。

import java.util.Scanner;

/\*\*

 \* 最大最小公倍数

 \* @author 简玄冰

 \* @当前日期 2014年3月12日下午4:46:13

 \*/

public class Test\_27 {

    public static void main(String args[]){

        Scanner in=new Scanner(System.in);

        long n=in.nextLong();

        long m;

        if(n%2!=0){

            //因为N不能整除，所以倒数3个数都是不能相约数，所以最后三个数相乘即为最大的公倍数

            m=n\*(n-1)\*(n-2);

        }else{

            //能整除的情况

            m=getMax(n);

        }

        System.out.println(m);

        in.close();

    }

    private static long getCommon(long a,long b){

        long n=a\*b;

        long c=a%b;

        while(c>0){

            a=b;

            b=c;

            c=a%b;

        }

        n/=b;

        return n;

    }

    private static long getMax(long n){

        long max=(n-1)\*(n-2)\*(n-3);

        int i=1;

        while(n\*(n-i)\*(n-i-1)>max){

            int j=i+1;

            while(n\*(n-i)\*(n-j)>max){

                long res=getCommon(getCommon(n, n-i), n-j);

                if(res>max){

                    max=res;

                }

                j++;

            }

            i++;

        }

        return max;

    }

}

K好数

问题描述

如果一个自然数N的K进制表示中任意的相邻的两位都不是相邻的数字，那么我们就说这个数是K好数。求L位K进制数中K好数的数目。例如K = 4，L = 2的时候，所有K好数为11、13、20、22、30、31、33 共7个。由于这个数目很大，请你输出它对1000000007取模后的值。

输入格式

输入包含两个正整数，K和L。

输出格式

输出一个整数，表示答案对1000000007取模后的值。

样例输入

4 2

样例输出

7

数据规模与约定

对于30%的数据，KL <= 106；

对于50%的数据，K <= 16， L <= 10；

对于100%的数据，1 <= K,L <= 100。

import java.util.\*;

public class Main {

    public static void main(String args[])

    {

        Scanner sc=new Scanner(System.in);

        while(sc.hasNext())

        {

            int k=sc.nextInt();

            int l=sc.nextInt();

            int x[][]=new int[l+1][k];

            int count=0;

            for(int i=0;i<k;i++)

                x[1][i]=1;

            for(int i=2;i<l+1;i++)

            {

                for(int j=0;j<k;j++)

                {

                    for(int p=0;p<k;p++)

                    {

                        if(p!=j-1&&p!=j+1)

                            x[i][j]=(x[i][j]+x[i-1][p])%1000000007;

                    }

                }

            }

            for(int j=1;j<k;j++)

                count=(count+x[l][j])%1000000007;

            System.out.println(count);

        }

    }

}

结点选择

问题描述

有一棵 n 个节点的树，树上每个节点都有一个正整数权值。如果一个点被选择了，那么在树上和它相邻的点都不能被选择。求选出的点的权值和最大是多少？

输入格式

第一行包含一个整数 n 。

接下来的一行包含 n 个正整数，第 i 个正整数代表点 i 的权值。

接下来一共 n-1 行，每行描述树上的一条边。

输出格式

输出一个整数，代表选出的点的权值和的最大值。

样例输入

5

1 2 3 4 5

1 2

1 3

2 4

2 5

样例输出

12

样例说明

选择3、4、5号点，权值和为 3+4+5 = 12 。

数据规模与约定

对于20%的数据， n <= 20。

对于50%的数据， n <= 1000。

对于100%的数据， n <= 100000。

权值均为不超过1000的正整数。

最短路

问题描述

给定一个n个顶点，m条边的有向图（其中某些边权可能为负，但保证没有负环）。请你计算从1号点到其他点的最短路（顶点从1到n编号）。

输入格式

第一行两个整数n, m。

接下来的m行，每行有三个整数u, v, l，表示u到v有一条长度为l的边。

输出格式

共n-1行，第i行表示1号点到i+1号点的最短路。

样例输入

3 3

1 2 -1

2 3 -1

3 1 2

样例输出

-1

-2

数据规模与约定

对于10%的数据，n = 2，m = 2。

对于30%的数据，n <= 5，m <= 10。

对于100%的数据，1 <= n <= 20000，1 <= m <= 200000，-10000 <= l <= 10000，保证从任意顶点都能到达其他所有顶点。

import java.util.\*;

import java.io.\*;

public class Main

{

    static int[] u;

    static int[] v;

    static int[] w;

    static int[] d;

    static int[] first;

    static int[] next;

    static Queue<Integer> q=new LinkedList<Integer>();

    static boolean[] inq;

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

    {

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        int m=sc.nextInt();

        int i,j;

        u=new int[m+1];

        v=new int[m+1];

        w=new int[m+1];

        first=new int[n+1];

        next=new int[m+1];

        d=new int[n+1];

        inq=new boolean[n+1];

        for(i=1;i<=n;i++) first[i]=-1;

        for( i=1;i<=m;i++){

               u[i]=sc.nextInt();

               v[i]=sc.nextInt();

               w[i]=sc.nextInt();

               next[i]=first[u[i]];

               first[u[i]]=i;

        }

        for(i=2;i<=n;i++) d[i]=Integer.MAX\_VALUE;

        q.offer(1);

        while(!q.isEmpty())

        {

            int x=q.poll();

            inq[x]=false;

            for(int e=first[x];e!=-1;e=next[e])

                if(d[v[e]]>d[x]+w[e]&&v[e]!=1)

                {

                    d[v[e]]=d[x]+w[e];

                    if(!inq[v[e]])

                    {

                        inq[v[e]]=true;

                        q.offer(v[e]);

                    }

                }

        }

        for(i=2;i<=n;i++) System.out.println(d[i]);

    }

}

安慰奶牛

问题描述

Farmer John变得非常懒，他不想再继续维护供奶牛之间供通行的道路。道路被用来连接N个牧场，牧场被连续地编号为1到N。每一个牧场都是一个奶牛的家。FJ计划除去P条道路中尽可能多的道路，但是还要保持牧场之间 的连通性。你首先要决定那些道路是需要保留的N-1条道路。第j条双向道路连接了牧场Sj和Ej(1 <= Sj <= N; 1 <= Ej <= N; Sj != Ej)，而且走完它需要Lj的时间。没有两个牧场是被一条以上的道路所连接。奶牛们非常伤心，因为她们的交通系统被削减了。你需要到每一个奶牛的住处去安慰她们。每次你到达第i个牧场的时候(即使你已经到过)，你必须花去Ci的时间和奶牛交谈。你每个晚上都会在同一个牧场(这是供你选择的)过夜，直到奶牛们都从悲伤中缓过神来。在早上 起来和晚上回去睡觉的时候，你都需要和在你睡觉的牧场的奶牛交谈一次。这样你才能完成你的 交谈任务。假设Farmer John采纳了你的建议，请计算出使所有奶牛都被安慰的最少时间。

输入格式

第1行包含两个整数N和P。

接下来N行，每行包含一个整数Ci。

接下来P行，每行包含三个整数Sj, Ej和Lj。

输出格式

输出一个整数, 所需要的总时间(包含和在你所在的牧场的奶牛的两次谈话时间)。

样例输入

5 7

10

10

20

6

30

1 2 5

2 3 5

2 4 12

3 4 17

2 5 15

3 5 6

样例输出

176

数据规模与约定

5 <= N <= 10000，N-1 <= P <= 100000，0 <= Lj <= 1000，1 <= Ci <= 1,000。

import java.util.\*;

import java.io.\*;

class Bian implements Comparable {

    private int first, second;

    private int value;

    public Bian(int first, int second, int value) {

        this.first = first;

        this.second = second;

        this.value = value;

    }

    public int getFirst() {

        return first;

    }

    public int getSecond() {

        return second;

    }

    public int getValue() {

        return value;

    }

    public int compareTo(Object arg0) {

        return value > ((Bian) arg0).value ? 1

                : (value == ((Bian) arg0).value ? 0 : -1);

    }

}

public class Main{

    static int min = Integer.MAX\_VALUE;

    static int father[];

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

        ArrayList<Bian> l = new ArrayList<Bian>();

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

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

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

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

        int d[] = new int[n + 1];

            father=new int[n+1];

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

            father[i]=-1;

            d[i] = Integer.parseInt(bf.readLine());

            if (min > d[i])

                min = d[i];

        }

         String s1[]=new String[3];

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

            s1=bf.readLine().split(" ");

            int x = Integer.parseInt(s1[0]);

            int y = Integer.parseInt(s1[1]);

            int w = 2 \* Integer.parseInt(s1[2]) + d[x] + d[y];

            l.add(new Bian(x, y, w));

        }

        Collections.sort(l);

        int count=0,sum=0;

        for (int i = 0; i <m; i++) {

             if(Union(l.get(i).getFirst(),l.get(i).getSecond())){

                 sum+=l.get(i).getValue();

                 count++;

             }

             if(count==n-1)

                 break;

        }

        System.out.println(sum+min);

    }

    public static int Find(int x){

        if(father[x]==-1) return x;

        return father[x]=Find(father[x]);

    }

    public static boolean Union(int x,int y){

        x=Find(x);

        y=Find(y);

        if(x==y) return false;

        if(x<y)  father[y]=x;

        if(y<x)  father[x]=y;

        return true;

    }

}

逆序对

问题描述

Alice是一个让人非常愉跃的人!他总是去学习一些他不懂的问题，然后再想出许多稀奇古怪的题目。这几天，Alice又沉浸在逆序对的快乐当中，他已近学会了如何求逆序对对数，动态维护逆序对对数等等题目，他认为把这些题让你做简直是太没追求了，于是，经过一天的思考和完善，Alice终于拿出了一道他认为差不多的题目：

有一颗2n-1个节点的二叉树，它有恰好n个叶子节点，每个节点上写了一个整数。如果将这棵树的所有叶子节点上的数从左到右写下来，便得到一个序列a[1]…a[n]。现在想让这个序列中的逆序对数量最少，但唯一的操作就是选树上一个非叶子节点，将它的左右两颗子树交换。他可以做任意多次这个操作。求在最优方案下，该序列的逆序对数最少有多少。

Alice自己已近想出了题目的正解，他打算拿来和你分享，他要求你在最短的时间内完成。

输入格式

第一行一个整数n。

下面每行，一个数x。

如果x=0，表示这个节点非叶子节点，递归地向下读入其左孩子和右孩子的信息，如果x≠0，表示这个节点是叶子节点，权值为x。

输出格式

输出一个整数，表示最少有多少逆序对。

样例输入

3

0

0

3

1

2

样例输出

1

数据规模与约定

对于20%的数据，n <= 5000。

对于100%的数据，1 <= n <= 200000，0 <= a[i]<2^31。

操作格子

问题描述

有n个格子，从左到右放成一排，编号为1-n。

共有m次操作，有3种操作类型：

1.修改一个格子的权值，

2.求连续一段格子权值和，

3.求连续一段格子的最大值。

对于每个2、3操作输出你所求出的结果。

输入格式

第一行2个整数n，m。

接下来一行n个整数表示n个格子的初始权值。

接下来m行，每行3个整数p,x,y，p表示操作类型，p=1时表示修改格子x的权值为y，p=2时表示求区间[x,y]内格子权值和，p=3时表示求区间[x,y]内格子最大的权值。

输出格式

有若干行，行数等于p=2或3的操作总数。

每行1个整数，对应了每个p=2或3操作的结果。

样例输入

4 3

1 2 3 4

2 1 3

1 4 3

3 1 4

样例输出

6

3

数据规模与约定

对于20%的数据n <= 100，m <= 200。

对于50%的数据n <= 5000，m <= 5000。

对于100%的数据1 <= n <= 100000，m <= 100000，0 <= 格子权值 <= 10000。

import java.util.Scanner;

public class Num4 {

    public static void main(String[] args) {

        // TODO Auto-generated method stub

        System.out.println("请输入几个格子和多少行");

        Scanner s=new Scanner(System.in);

        int d1=s.nextInt();

        int d2=s.nextInt();

        int[] d=new int[d1];

        int[][] d3=new int[d2][3];

        for(int i=0;i<d1;i++)

        {

            d[i]=s.nextInt();

        }

        for(int i=0;i<d2;i++)

        {

            for(int j=0;j<3;j++)

            {

                d3[i][j]=s.nextInt();

            }

        }

        for(int i=0;i<d2;i++)

        {

            int m=0;

            int t1=0;

            switch(d3[i][0])

            {

            case 1:d[d3[i][1]-1]=d[d3[i][2]-1];break;

            case 2:for(int j=d3[i][1]-1;j<d3[i][2];j++)

            {

                m=d[j]+m;

            }

            System.out.println(m);break;

            case 3:for(int j=d3[i][1]-1;j<d3[i][2]-1;j++)

            {

                t1=d[j];

                if(d[j]<d[j+1])

                {

                    t1=d[j+1];

                }

                else

                {

                    t1=d[j];

                }

            }

            System.out.println(t1);break;

            }

        }

    }

}

***算法提高***

邮票面值设计 (vip)

问题描述

　　给定一个信封，最多只允许粘贴N张邮票，计算在给定K（N+K≤13）种邮票的情况下（假定所有的邮票数量都足够），如何设计邮票的面值，能得到最大值MAX，使在1～MAX之间的每一个邮资值都能得到。

　　例如，N=3，K=2，如果面值分别为1分、4分，则在1分～6分之间的每一个邮资值都能得到（当然还有8分、9分和12分）；如果面值分别为1分、3分，则在1分～7分之间的每一个邮资值都能得到。可以验证当N=3，K=2时，7分就是可以得到的连续的邮资最大值，所以MAX=7，面值分别为1分、3分。

输入格式

　　一行，两个数N、K

输出格式

　　两行，第一行升序输出设计的邮票面值，第二行输出“MAX=xx”（不含引号），其中xx为所求的能得到的连续邮资最大值。

样例输入

3 2

样例输出

1 3

MAX=7

子集选取 (vip)

问题描述

　　一个有N个元素的集合有2^N个不同子集（包含空集），现在要在这2^N个集合中取出若干集合（至少一个），使得它们的交集的元素个数为K，求取法的方案数，答案模1000000007。

输入格式

　　输入一行两个整数N，K。

输出格式

　　输出一个整数表示答案。

样例输入

3 2

样例输出

6

数据规模和约定

　　1 <= K <= N <= 10 ^ 6。

冒泡排序计数 (vip)

问题描述

　　考虑冒泡排序的一种实现。

　　bubble-sort (A[], n)

　　> round = 0

　　> while A is not sorted

　　> > round := round + 1

　　> > for i := 1 to n - 1

　　> > > if (A[i] > A[i + 1])

　　> > > > swap(A[i], A[i + 1])

　　求1 .. n的排列中，有多少个排列使得A被扫描了K遍，亦即算法结束时round == K。

　　答案模20100713输出。

输入格式

　　输入包含多组数据。每组数据为一行两个整数N，K。

输出格式

　　对每组数据，输出一行一个整数表示答案。

样例输入

3

3 0

3 1

3 2

样例输出

1

3

2

数据规模和约定

　　T <= 10 ^ 5。

　　1 <= K < N < 10 ^ 6。

两条直线

问题描述

给定平面上n个点。

求两条直线，这两条直线互相垂直，而且它们与x轴的夹角为45度，并且n个点中离这两条直线的曼哈顿距离的最大值最小。

两点之间的曼哈顿距离定义为横坐标的差的绝对值与纵坐标的差的绝对值之和，一个点到两条直线的曼哈顿距离是指该点到两条直线上的所有点的曼哈顿距离中的最小值。

输入格式

第一行包含一个数n。

接下来n行，每行包含两个整数，表示n个点的坐标（横纵坐标的绝对值小于109）。

输出格式

输出一个值，表示最小的最大曼哈顿距离的值，保留一位小数。

样例输入

4

1 0

0 1

2 1

1 2

样例输出

1.0

数据规模与约定

对于30%的数据，n<=100。

对于另外30%的数据，坐标范的绝对值小于100。

对于100%的数据，n<=105。

矩阵翻转

问题描述

Ciel有一个N\*N的矩阵，每个格子里都有一个整数。

N是一个奇数，设X = (N+1)/2。Ciel每次都可以做这样的一次操作：他从矩阵选出一个X\*X的子矩阵，并将这个子矩阵中的所有整数都乘以-1。

现在问你经过一些操作之后，矩阵中所有数的和最大可以为多少。

输入格式

第一行为一个正整数N。

接下来N行每行有N个整数，表示初始矩阵中的数字。每个数的绝对值不超过1000。

输出格式

输出一个整数，表示操作后矩阵中所有数之和的最大值。

样例输入

3

-1 -1 1

-1 1 -1

1 -1 -1

样例输出

9

数据规模与约定

1 <= N <= 33，且N为奇数。

金属采集

问题描述

人类在火星上发现了一种新的金属！这些金属分布在一些奇怪的地方，不妨叫它节点好了。一些节点之间有道路相连，所有的节点和道路形成了一棵树。一共有 n 个节点，这些节点被编号为 1~n 。人类将 k 个机器人送上了火星，目的是采集这些金属。这些机器人都被送到了一个指定的着落点， S 号节点。每个机器人在着落之后，必须沿着道路行走。当机器人到达一个节点时，它会采集这个节点蕴藏的所有金属矿。当机器人完成自己的任务之后，可以从任意一个节点返回地球。当然，回到地球的机器人就无法再到火星去了。我们已经提前测量出了每条道路的信息，包括它的两个端点 x 和 y，以及通过这条道路需要花费的能量 w 。我们想花费尽量少的能量采集所有节点的金属，这个任务就交给你了。

输入格式

第一行包含三个整数 n, S 和 k ，分别代表节点个数、着落点编号，和机器人个数。

接下来一共 n-1 行，每行描述一条道路。一行含有三个整数 x, y 和 w ，代表在 x 号节点和 y 号节点之间有一条道路，通过需要花费 w 个单位的能量。所有道路都可以双向通行。

输出格式

输出一个整数，代表采集所有节点的金属所需要的最少能量。

样例输入

6 1 3

1 2 1

2 3 1

2 4 1000

2 5 1000

1 6 1000

样例输出

3004

样例说明

所有机器人在 1 号节点着陆。

第一个机器人的行走路径为 1->6 ，在 6 号节点返回地球，花费能量为1000。

第二个机器人的行走路径为 1->2->3->2->4 ，在 4 号节点返回地球，花费能量为1003。

第一个机器人的行走路径为 1->2->5 ，在 5 号节点返回地球，花费能量为1001。

数据规模与约定

本题有10个测试点。

对于测试点 1~2 ， n <= 10 ， k <= 5 。

对于测试点 3 ， n <= 100000 ， k = 1 。

对于测试点 4 ， n <= 1000 ， k = 2 。

对于测试点 5~6 ， n <= 1000 ， k <= 10 。

对于测试点 7~10 ， n <= 100000 ， k <= 10 。

道路的能量 w 均为不超过 1000 的正整数。

道路和航路

问题描述

农夫约翰正在针对一个新区域的牛奶配送合同进行研究。他打算分发牛奶到T个城镇（标号为1..T），这些城镇通过R条标号为（1..R）的道路和P条标号为（1..P）的航路相连。

每一条公路i或者航路i表示成连接城镇Ai（1<=A\_i<=T）和Bi（1<=Bi<=T）代价为Ci。每一条公路，Ci的范围为0<=Ci<=10,000；由于奇怪的运营策略，每一条航路的Ci可能为负的，也就是-10,000<=Ci<=10,000。

每一条公路都是双向的，正向和反向的花费是一样的，都是非负的。

每一条航路都根据输入的Ai和Bi进行从Ai->Bi的单向通行。实际上，如果现在有一条航路是从Ai到Bi的话，那么意味着肯定没有通行方案从Bi回到Ai。

农夫约翰想把他那优良的牛奶从配送中心送到各个城镇，当然希望代价越小越好，你可以帮助他嘛？配送中心位于城镇S中（1<=S<=T）。

输入格式

输入的第一行包含四个用空格隔开的整数T，R，P，S。

接下来R行，描述公路信息，每行包含三个整数，分别表示Ai，Bi和Ci。

接下来P行，描述航路信息，每行包含三个整数，分别表示Ai，Bi和Ci。

输出格式

输出T行，分别表示从城镇S到每个城市的最小花费，如果到不了的话输出NO PATH。

样例输入

6 3 3 4

1 2 5

3 4 5

5 6 10

3 5 -100

4 6 -100

1 3 -10

样例输出

NO PATH

NO PATH

5

0

-95

-100

数据规模与约定

对于20%的数据，T<=100，R<=500，P<=500；

对于30%的数据，R<=1000，R<=10000，P<=3000；

对于100%的数据，1<=T<=25000，1<=R<=50000，1<=P<=50000。

最小方差生成树

问题描述

给定带权无向图，求出一颗方差最小的生成树。

输入格式

输入多组测试数据。第一行为N,M，依次是点数和边数。接下来M行，每行三个整数U,V,W，代表连接U,V的边，和权值W。保证图连通。n=m=0标志着测试文件的结束。

输出格式

对于每组数据，输出最小方差，四舍五入到0.01。输出格式按照样例。

样例输入

4 5

1 2 1

2 3 2

3 4 2

4 1 1

2 4 3

4 6

1 2 1

2 3 2

3 4 3

4 1 1

2 4 3

1 3 3

0 0

样例输出

Case 1: 0.22

Case 2: 0.00

数据规模与约定

1<=U,V<=N<=50,N-1<=M<=1000,0<=W<=50。数据不超过5组。

***历届试题***

核桃的数量

问题描述

小张是软件项目经理，他带领3个开发组。工期紧，今天都在加班呢。为鼓舞士气，小张打算给每个组发一袋核桃（据传言能补脑）。他的要求是：

1. 各组的核桃数量必须相同

2. 各组内必须能平分核桃（当然是不能打碎的）

3. 尽量提供满足1,2条件的最小数量（节约闹革命嘛）

输入格式

输入包含三个正整数a, b, c，表示每个组正在加班的人数，用空格分开（a,b,c<30）

输出格式

输出一个正整数，表示每袋核桃的数量。

样例输入1

2 4 5

样例输出1

20

样例输入2

3 1 1

样例输出2

3

import java.util.Scanner;

/\*\*

 \* @author Flyingfish

 \*

 \*/

public class Main {

    /\*\*

     \* @param args

     \*/

    public static void main(String[] args) {

        Scanner reader = new Scanner(System.in);

        int a,b,c;

        a=reader.nextInt();

        b=reader.nextInt();

        c=reader.nextInt();

        int d=a\*b/fun(a,b);

        System.out.println(d\*c/fun(d,c));

    }

    static int fun(int num1,int num2){

        if(num1%num2==0){

            return num2;

        }

        return fun(num2,num1%num2);

    }

}

import java.util.\*;

public class Main {

    public static void main(String[] args) {

        // TODO Auto-generated method stub

        Scanner sc=new Scanner(System.in);

        int i,a=sc.nextInt(),b=sc.nextInt(),c=sc.nextInt();

        for(i=1;;i++){

            if(i%a==0&&i%b==0&&i%c==0)

                break;

        }

        System.out.println(i);

    }

}

打印十字图

问题描述

小明为某机构设计了一个十字型的徽标（并非红十字会啊），如下所示：

..$$$$$$$$$$$$$..

..$...........$..

$$$.$$$$$$$$$.$$$

$...$.......$...$

$.$$$.$$$$$.$$$.$

$.$...$...$...$.$

$.$.$$$.$.$$$.$.$

$.$.$...$...$.$.$

$.$.$.$$$$$.$.$.$

$.$.$...$...$.$.$

$.$.$$$.$.$$$.$.$

$.$...$...$...$.$

$.$$$.$$$$$.$$$.$

$...$.......$...$

$$$.$$$$$$$$$.$$$

..$...........$..

..$$$$$$$$$$$$$..

对方同时也需要在电脑dos窗口中以字符的形式输出该标志，并能任意控制层数。

输入格式

一个正整数 n (n<30) 表示要求打印图形的层数。

输出格式

对应包围层数的该标志。

样例输入1

1

样例输出1

..$$$$$..

..$...$..

$$$.$.$$$

$...$...$

$.$$$$$.$

$...$...$

$$$.$.$$$

..$...$..

..$$$$$..

样例输入2

3

样例输出2

..$$$$$$$$$$$$$..

..$...........$..

$$$.$$$$$$$$$.$$$

$...$.......$...$

$.$$$.$$$$$.$$$.$

$.$...$...$...$.$

$.$.$$$.$.$$$.$.$

$.$.$...$...$.$.$

$.$.$.$$$$$.$.$.$

$.$.$...$...$.$.$

$.$.$$$.$.$$$.$.$

$.$...$...$...$.$

$.$$$.$$$$$.$$$.$

$...$.......$...$

$$$.$$$$$$$$$.$$$

..$...........$..

..$$$$$$$$$$$$$..

提示

请仔细观察样例，尤其要注意句点的数量和输出位置。

import java.util.Scanner;

public class Main{

    public static void main(String args[])

    {

        Scanner sc=new Scanner(System.in);

        while(sc.hasNext())

        {

            int n=sc.nextInt();

            int [][]a=new int[4\*n+5+1][4\*n+5+1];

            Outprint(n,a);

            for(int i=1;i<=4\*n+5;i++)

            {

                for(int j=1;j<=4\*n+5;j++)

                {

                    if(a[i][j]==1)

                        System.out.print('$');

                    else

                        System.out.print('.');

                }

                System.out.println();

            }

        }

    }

    public static void Outprint(int n,int[][] a)

    {

        for(int i=1;i<=(4\*n+5+1)/2;i++){

            for(int j=1;j<=(4\*n+5+1)/2;j++){

                if(i!=1&&i==j&&i%2==1){

                    a[i][j]=1;a[i-1][j]=1;a[i][j-1]=1;

                }

                else if(i>=3||j>=3){

                    if(j%2==1&&i>j+1)

                        a[i][j]=1;

                    if(i%2==1&&j>i+1)

                        a[i][j]=1;

                }

            }

        }

        for(int i=1;i<=(4\*n+5+1)/2;i++){

            for(int j=(4\*n+5+1)/2+1;j<=4\*n+5;j++){

                 a[i][j]=a[i][4\*n+5+1-j];

            }

        }

        for(int i=(4\*n+5+1)/2+1;i<=4\*n+5;i++){

            for(int j=1;j<=4\*n+5;j++){

                a[i][j]=a[4\*n+5+1-i][j];

            }

        }

    }

}

带分数

问题描述

100 可以表示为带分数的形式：100 = 3 + 69258 / 714。

还可以表示为：100 = 82 + 3546 / 197。

注意特征：带分数中，数字1~9分别出现且只出现一次（不包含0）。

类似这样的带分数，100 有 11 种表示法。

输入格式

从标准输入读入一个正整数N (N<1000\*1000)

输出格式

程序输出该数字用数码1~9不重复不遗漏地组成带分数表示的全部种数。

注意：不要求输出每个表示，只统计有多少表示法！

样例输入1

100

样例输出1

11

样例输入2

105

样例输出2

6

import java.util.HashSet;

import java.util.Scanner;

public class Main{

    static HashSet<String> set=new HashSet<String>();

    static HashSet<String> set1=new HashSet<String>();

    public static boolean f(int x){

        String s=x+"";

        if(s.indexOf('0')>0)return false;

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

            String ch=s.charAt(i)+"";

            if(!set.add(ch)){

                return false;

            }

            set1.add(ch);

        }

        return true;

    }

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int n=Integer.parseInt(sc.nextLine());

        int num=0;

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

            set.clear();

            if(!f(i)) continue;

            set1.clear();

            double jj=Math.pow(10, (9-Math.log10(i)-Math.log10(n-i))/2);

            for (int j = 1; j <jj; j++) {

                if(i==j) continue;

                set.removeAll(set1);

                set1.clear();

                if(!f(j)) continue;

                int m=(n-i)\*j;

                if(!f(m)) continue;

                if(set.size()==9){

                    num++;

                }

            }

        }

        System.out.println(num);

    }

}

剪格子

问题描述

如下图所示，3 x 3 的格子中填写了一些整数。

+--\*--+--+

|10\* 1|52|

+--\*\*\*\*--+

|20|30\* 1|

\*\*\*\*\*\*\*--+

| 1| 2| 3|

+--+--+--+

我们沿着图中的星号线剪开，得到两个部分，每个部分的数字和都是60。

本题的要求就是请你编程判定：对给定的m x n 的格子中的整数，是否可以分割为两个部分，使得这两个区域的数字和相等。

如果存在多种解答，请输出包含左上角格子的那个区域包含的格子的最小数目。

如果无法分割，则输出 0。

输入格式

程序先读入两个整数 m n 用空格分割 (m,n<10)。

表示表格的宽度和高度。

接下来是n行，每行m个正整数，用空格分开。每个整数不大于10000。

输出格式

输出一个整数，表示在所有解中，包含左上角的分割区可能包含的最小的格子数目。

样例输入1

3 3

10 1 52

20 30 1

1 2 3

样例输出1

3

样例输入2

4 3

1 1 1 1

1 30 80 2

1 1 1 100

样例输出2

10

import java.util.\*;

public class Main {

   static int[][] num=null;

   static boolean[][] is=null;

   static int n;

   static int m;

   static int sum;

   static int count;

   static int min=Integer.MAX\_VALUE;

   static int zong;

   public static void main(String args[]){

       Scanner scan=new Scanner(System.in);

       m=scan.nextInt();

       n=scan.nextInt();

        num=new int[n][m];

        is=new boolean[n][m];

       for(int i=0;i<n;i++){

           for(int j=0;j<m;j++){

               num[i][j]=scan.nextInt();

               zong+=num[i][j];

           }

       }

       serach(0,0);

       System.out.println(min);

   }

   public static boolean serach(int i,int j){

       if(i>=n||j>=m||i<0||j<0) return false;

       if(is[i][j]) return false;

       if(sum>zong/2) return false;

       if(sum==zong/2){

           min=Math.min(min, count);

           return true;

       }

       is[i][j]=true;

       sum+=num[i][j];

       count++;

       if(serach(i-1,j)) return true;

       else if(serach(i+1,j)) return true;

       else if(serach(i,j-1)) return true;

       else if(serach(i,j+1)) return true;

       else{

           count--;

           is[i][j]=false;

           sum-=num[i][j];

           return false;

       }

   }

}

错误票据

问题描述

某涉密单位下发了某种票据，并要在年终全部收回。

每张票据有唯一的ID号。全年所有票据的ID号是连续的，但ID的开始数码是随机选定的。

因为工作人员疏忽，在录入ID号的时候发生了一处错误，造成了某个ID断号，另外一个ID重号。

你的任务是通过编程，找出断号的ID和重号的ID。

假设断号不可能发生在最大和最小号。

输入格式

要求程序首先输入一个整数N(N<100)表示后面数据行数。

接着读入N行数据。

每行数据长度不等，是用空格分开的若干个（不大于100个）正整数（不大于100000），请注意行内和行末可能有多余的空格，你的程序需要能处理这些空格。

每个整数代表一个ID号。

输出格式

要求程序输出1行，含两个整数m n，用空格分隔。

其中，m表示断号ID，n表示重号ID

样例输入1

2

5 6 8 11 9

10 12 9

样例输出1

7 9

样例输入2

6

164 178 108 109 180 155 141 159 104 182 179 118 137 184 115 124 125 129 168 196

172 189 127 107 112 192 103 131 133 169 158

128 102 110 148 139 157 140 195 197

185 152 135 106 123 173 122 136 174 191 145 116 151 143 175 120 161 134 162 190

149 138 142 146 199 126 165 156 153 193 144 166 170 121 171 132 101 194 187 188

113 130 176 154 177 120 117 150 114 183 186 181 100 163 160 167 147 198 111 119

样例输出2

105 120

import java.util.Iterator;

import java.util.Scanner;

import java.util.TreeSet;

public class Main

{

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        while(scanner.hasNext())

        {

            int error = 0;

            int doubleNum= 0;

            int n = Integer.valueOf(scanner.nextLine());

            StringBuilder build = new StringBuilder();

            TreeSet<Integer> set = new TreeSet<Integer>();

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

            {

                build = build.append(scanner.nextLine()+ " ");

            }

            String[] c = build.toString().split("[\\s]+");

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

            {

                if(!set.contains(Integer.parseInt(c[i])))

                {

                    set.add(new Integer(c[i]));

                }

                else

                {

                    doubleNum = Integer.parseInt(c[i]);

                }

            }

            Iterator<Integer> it = set.iterator();

            int d = it.next();

            while(it.hasNext())

            {

                int f = it.next();

                if(f - d == 2)

                {

                    error = d + 1;

                }

                else

                {

                    d = f;

                }

            }

            System.out.println(error + " " + doubleNum);

        }

    }

}

翻硬币

问题描述

小明正在玩一个“翻硬币”的游戏。

桌上放着排成一排的若干硬币。我们用 \* 表示正面，用 o 表示反面（是小写字母，不是零）。

比如，可能情形是：\*\*oo\*\*\*oooo

如果同时翻转左边的两个硬币，则变为：oooo\*\*\*oooo

现在小明的问题是：如果已知了初始状态和要达到的目标状态，每次只能同时翻转相邻的两个硬币,那么对特定的局面，最少要翻动多少次呢？

我们约定：把翻动相邻的两个硬币叫做一步操作，那么要求：

输入格式

两行等长的字符串，分别表示初始状态和要达到的目标状态。每行的长度<1000

输出格式

一个整数，表示最小操作步数。

样例输入1

\*\*\*\*\*\*\*\*\*\*

o\*\*\*\*o\*\*\*\*

样例输出1

5

样例输入2

\*o\*\*o\*\*\*o\*\*\*

\*o\*\*\*o\*\*o\*\*\*

样例输出2

1

import java.util.Scanner;

public class Main {

    /\*\*

     \* @param args

     \*/

    static char a[];

    static char b[];

    static int n=0;

    public static void main(String[] args) {

        // TODO Auto-generated method stub

        Scanner reader=new Scanner(System.in);

        String str1=reader.nextLine(),str2=reader.nextLine();

        a=str1.toCharArray();

        b=str2.toCharArray();

        for(int i=0;i<a.length;i++){

            if(a[i]!=b[i]){

                fun(i);

                fun(i+1);

                n++;

            }

        }

        System.out.println(n);

    }

    static void fun(int i){

        if(a[i]=='\*'){

            a[i]='o';

        }else{

            a[i]='\*';

        }

    }

}

连号区间数

问题描述

小明这些天一直在思考这样一个奇怪而有趣的问题：

在1~N的某个全排列中有多少个连号区间呢？这里所说的连号区间的定义是：

如果区间[L, R] 里的所有元素（即此排列的第L个到第R个元素）递增排序后能得到一个长度为R-L+1的“连续”数列，则称这个区间连号区间。

当N很小的时候，小明可以很快地算出答案，但是当N变大的时候，问题就不是那么简单了，现在小明需要你的帮助。

输入格式

第一行是一个正整数N (1 <= N <= 50000), 表示全排列的规模。

第二行是N个不同的数字Pi(1 <= Pi <= N)， 表示这N个数字的某一全排列。

输出格式

输出一个整数，表示不同连号区间的数目。

样例输入1

4

3 2 4 1

样例输出1

7

样例输入2

5

3 4 2 5 1

样例输出2

9

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        // TODO Auto-generated method stub

        Scanner sc = new Scanner(System.in);

        while (sc.hasNext()) {

            int N = sc.nextInt();

            int[] arr = new int[N];

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

                arr[i] = sc.nextInt();

            int count = N;

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

                int min = arr[i];

                int max = arr[i];

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

                    if (min > arr[j])

                        min = arr[j];

                    if (max < arr[j])

                        max = arr[j];

                    if (max - min == j - i)

                        count++;

                }

            }

            System.out.println(count);

        }

    }

}

买不到的数目

问题描述

小明开了一家糖果店。他别出心裁：把水果糖包成4颗一包和7颗一包的两种。糖果不能拆包卖。

小朋友来买糖的时候，他就用这两种包装来组合。当然有些糖果数目是无法组合出来的，比如要买 10 颗糖。

你可以用计算机测试一下，在这种包装情况下，最大不能买到的数量是17。大于17的任何数字都可以用4和7组合出来。

本题的要求就是在已知两个包装的数量时，求最大不能组合出的数字。

输入格式

两个正整数，表示每种包装中糖的颗数(都不多于1000)

输出格式

一个正整数，表示最大不能买到的糖数

样例输入1

4 7

样例输出1

17

样例输入2

3 5

样例输出2

7

import java.util.Scanner;

public class Main

{

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        while(scanner.hasNext())

        {

            int n = scanner.nextInt();

            int m = scanner.nextInt();

            System.out.println(n \* m - n - m);

        }

    }

}

大臣的旅费

问题描述

很久以前，T王国空前繁荣。为了更好地管理国家，王国修建了大量的快速路，用于连接首都和王国内的各大城市。

为节省经费，T国的大臣们经过思考，制定了一套优秀的修建方案，使得任何一个大城市都能从首都直接或者通过其他大城市间接到达。同时，如果不重复经过大城市，从首都到达每个大城市的方案都是唯一的。

J是T国重要大臣，他巡查于各大城市之间，体察民情。所以，从一个城市马不停蹄地到另一个城市成了J最常做的事情。他有一个钱袋，用于存放往来城市间的路费。

聪明的J发现，如果不在某个城市停下来修整，在连续行进过程中，他所花的路费与他已走过的距离有关，在走第x千米到第x+1千米这一千米中（x是整数），他花费的路费是x+10这么多。也就是说走1千米花费11，走2千米要花费23。

J大臣想知道：他从某一个城市出发，中间不休息，到达另一个城市，所有可能花费的路费中最多是多少呢？

输入格式

输入的第一行包含一个整数n，表示包括首都在内的T王国的城市数

城市从1开始依次编号，1号城市为首都。

接下来n-1行，描述T国的高速路（T国的高速路一定是n-1条）

每行三个整数Pi, Qi, Di，表示城市Pi和城市Qi之间有一条高速路，长度为Di千米。

输出格式

输出一个整数，表示大臣J最多花费的路费是多少。

输出格式

大臣J从城市4到城市5要花费135的路费。

样例输入1

5

1 2 2

1 3 1

2 4 5

2 5 4

样例输出1

135

import java.util.\*;

public class Main {

    public static void main(String args[]){

        Scanner scan=new Scanner (System.in);

        int n=scan.nextInt();

        QiDian[] QiDians=new QiDian[n];

        for(int i=0;i<n;i++){

            QiDians[i]=new QiDian(i,new ArrayList<ZhongDian>());

        }

        int tem1=0;

        int tem2=0;

        int quanzhong=0;

        for(int i=0;i<n-1;i++){

            tem1=scan.nextInt();

            tem2=scan.nextInt();

            quanzhong=scan.nextInt();

            QiDians[tem1-1].list.add(new ZhongDian(quanzhong,QiDians[tem2-1]));

            QiDians[tem2-1].list.add(new ZhongDian(quanzhong,QiDians[tem1-1]));

        }

        int[] data=search(QiDians[0],null);

        int[] data2=search(QiDians[data[1]],null);

        int sum=0;

        for(int i=1;i<=data2[0];i++){

            sum+=i+10;

        }

        System.out.println(sum);

      }

     public static int[] search(QiDian q,QiDian p){

        int[] data=new int[]{0,q.index};

        for(int i=0;i<q.list.size();i++){

            if(q.list.get(i).qidian.equals(p)==false){

                int [] data2=search(q.list.get(i).qidian,q);

                int tem=q.list.get(i).quanzhong+data2[0];

                if(tem>data[0]){

                    data[0]=tem;

                    data[1]=data2[1];

                }

            }

        }

        return data;

     }

    }

class QiDian{

    int index;

    ArrayList<ZhongDian> list=new ArrayList<ZhongDian>();

    public QiDian(int index, ArrayList<ZhongDian> list) {

        super();

        this.index = index;

        this.list = list;

    }

}

class ZhongDian{

    int quanzhong;

    QiDian qidian;

    public ZhongDian(int quanzhong, QiDian qidian) {

        super();

        this.quanzhong = quanzhong;

        this.qidian = qidian;

    }

}

幸运数

问题描述

幸运数是波兰数学家乌拉姆命名的。它采用与生成素数类似的“筛法”生成

。

首先从1开始写出自然数1,2,3,4,5,6,....

1 就是第一个幸运数。

我们从2这个数开始。把所有序号能被2整除的项删除，变为：

1 \_ 3 \_ 5 \_ 7 \_ 9 ....

把它们缩紧，重新记序，为：

1 3 5 7 9 .... 。这时，3为第2个幸运数，然后把所有能被3整除的序号位置的数删去。注意，是序号位置，不是那个数本身能否被3整除!! 删除的应该是5，11, 17, ...

此时7为第3个幸运数，然后再删去序号位置能被7整除的(19,39,...)

最后剩下的序列类似：

1, 3, 7, 9, 13, 15, 21, 25, 31, 33, 37, 43, 49, 51, 63, 67, 69, 73, 75, 79, ...

输入格式

输入两个正整数m n, 用空格分开 (m < n < 1000\*1000)

输出格式

程序输出 位于m和n之间的幸运数的个数（不包含m和n）。

样例输入1

1 20

样例输出1

5

样例输入2

30 69

样例输出2

8

import java.util.\*;

public class Main {

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        int m = sc.nextInt();

        int n = sc.nextInt();

        int e = n / 2 + 10;

        int a[] = new int[e];

        int num = 0;

        for (int i = 0; i < e; i++) {

            a[i] = i \* 2 + 1;

        }

        check(1, a, e);

        for (int i = 0; i < e; i++) {

            if (a[i] > n)

                break;

            if (a[i] > m && a[i] < n) {

                num++;

                // System.out.print(a[i]+" ");

            }

        }

        System.out.println(num);

    }

    public static int[] check(int s, int[] a, int e) {

        int k = s;

        int c = a[s];

        for (int i = s; i < e; i++) {

            if ((i + 1) % c != 0) {

                a[k] = a[i];

                k++;

            }

        }

        if (c < e)

            return check(s + 1, a, e);

        else

            return a;

    }

}

横向打印二叉树

问题描述

二叉树可以用于排序。其原理很简单：对于一个排序二叉树添加新节点时，先与根节点比较，若小则交给左子树继续处理，否则交给右子树。

当遇到空子树时，则把该节点放入那个位置。

比如，10 8 5 7 12 4 的输入顺序，应该建成二叉树如下图所示，其中.表示空白。

...|-12

10-|

...|-8-|

.......|...|-7

.......|-5-|

...........|-4

本题目要求：根据已知的数字，建立排序二叉树，并在标准输出中横向打印该二叉树。

输入格式

输入数据为一行空格分开的N个整数。 N<100，每个数字不超过10000。

输入数据中没有重复的数字。

输出格式

输出该排序二叉树的横向表示。为了便于评卷程序比对空格的数目，请把空格用句点代替：

样例输入1

10 5 20

样例输出1

...|-20

10-|

...|-5

样例输入2

5 10 20 8 4 7

样例输出2

.......|-20

..|-10-|

..|....|-8-|

..|........|-7

5-|

..|-4

危险系数

问题描述

抗日战争时期，冀中平原的地道战曾发挥重要作用。

地道的多个站点间有通道连接，形成了庞大的网络。但也有隐患，当敌人发现了某个站点后，其它站点间可能因此会失去联系。

我们来定义一个危险系数DF(x,y)：

对于两个站点x和y (x != y), 如果能找到一个站点z，当z被敌人破坏后，x和y不连通，那么我们称z为关于x,y的关键点。相应的，对于任意一对站点x和y，危险系数DF(x,y)就表示为这两点之间的关键点个数。

本题的任务是：已知网络结构，求两站点之间的危险系数。

输入格式

输入数据第一行包含2个整数n(2 <= n <= 1000), m(0 <= m <= 2000),分别代表站点数，通道数；

接下来m行，每行两个整数 u,v (1 <= u, v <= n; u != v)代表一条通道；

最后1行，两个数u,v，代表询问两点之间的危险系数DF(u, v)。

输出格式

一个整数，如果询问的两点不连通则输出-1.

样例输入

7 6

1 3

2 3

3 4

3 5

4 5

5 6

1 6

样例输出

2

网络寻路

问题描述

X 国的一个网络使用若干条线路连接若干个节点。节点间的通信是双向的。某重要数据包，为了安全起见，必须恰好被转发两次到达目的地。该包可能在任意一个节点产生，我们需要知道该网络中一共有多少种不同的转发路径。

源地址和目标地址可以相同，但中间节点必须不同。

如下图所示的网络。

1 -> 2 -> 3 -> 1 是允许的

1 -> 2 -> 1 -> 2 或者 1 -> 2 -> 3 -> 2 都是非法的。

输入格式

输入数据的第一行为两个整数N M，分别表示节点个数和连接线路的条数(1<=N<=10000; 0<=M<=100000)。

接下去有M行，每行为两个整数 u 和 v，表示节点u 和 v 联通(1<=u,v<=N , u!=v)。

输入数据保证任意两点最多只有一条边连接，并且没有自己连自己的边，即不存在重边和自环。

输出格式

输出一个整数，表示满足要求的路径条数。

样例输入1

3 3

1 2

2 3

1 3

样例输出1

6

样例输入2

4 4

1 2

2 3

3 1

1 4

样例输出2

10

高僧斗法

问题描述

　　古时丧葬活动中经常请高僧做法事。仪式结束后，有时会有“高僧斗法”的趣味节目，以舒缓压抑的气氛。

　　节目大略步骤为：先用粮食（一般是稻米）在地上“画”出若干级台阶（表示N级浮屠）。又有若干小和尚随机地“站”在某个台阶上。最高一级台阶必须站人，其它任意。(如图1所示)

　　两位参加游戏的法师分别指挥某个小和尚向上走任意多级的台阶，但会被站在高级台阶上的小和尚阻挡，不能越过。两个小和尚也不能站在同一台阶，也不能向低级台阶移动。

　　两法师轮流发出指令，最后所有小和尚必然会都挤在高段台阶，再也不能向上移动。轮到哪个法师指挥时无法继续移动，则游戏结束，该法师认输。

　　对于已知的台阶数和小和尚的分布位置，请你计算先发指令的法师该如何决策才能保证胜出。

输入格式

　　输入数据为一行用空格分开的N个整数，表示小和尚的位置。台阶序号从1算起，所以最后一个小和尚的位置即是台阶的总数。（N<100, 台阶总数<1000）

输出格式

　　输出为一行用空格分开的两个整数: A B, 表示把A位置的小和尚移动到B位置。若有多个解，输出A值较小的解，若无解则输出-1。

样例输入

1 5 9

样例输出

1 4

样例输入

1 5 8 10

样例输出

1 3

格子刷油漆

问题描述

　　X国的一段古城墙的顶端可以看成 2\*N个格子组成的矩形（如下图所示），现需要把这些格子刷上保护漆。

　　你可以从任意一个格子刷起，刷完一格，可以移动到和它相邻的格子（对角相邻也算数），但不能移动到较远的格子（因为油漆未干不能踩！）

　　比如：a d b c e f 就是合格的刷漆顺序。

　　c e f d a b 是另一种合适的方案。

　　当已知 N 时，求总的方案数。当N较大时，结果会迅速增大，请把结果对 1000000007 (十亿零七) 取模。

输入格式

　　输入数据为一个正整数（不大于1000）

输出格式

　　输出数据为一个正整数。

样例输入

2

样例输出

24

样例输入

3

样例输出

96

样例输入

22

样例输出

359635897

农场阳光

问题描述

　　X星球十分特殊，它的自转速度与公转速度相同，所以阳光总是以固定的角度照射。

　　最近，X星球为发展星际旅游业，把空间位置出租给Y国游客来晒太阳。每个租位是漂浮在空中的圆盘形彩云（圆盘与地面平行）。当然，这会遮挡住部分阳光，被遮挡的土地植物无法生长。

　　本题的任务是计算某个农场宜于作物生长的土地面积有多大。

输入格式

　　输入数据的第一行包含两个整数a, b，表示某农场的长和宽分别是a和b，此时，该农场的范围是由坐标(0, 0, 0), (a, 0, 0), (a, b, 0), (0, b, 0)围成的矩形区域。

　　第二行包含一个实数g，表示阳光照射的角度。简单起见，我们假设阳光光线是垂直于农场的宽的，此时正好和农场的长的夹角是g度，此时，空间中的一点(x, y, z)在地面的投影点应该是(x + z \* ctg(g度), y, 0)，其中ctg(g度)表示g度对应的余切值。

　　第三行包含一个非负整数n，表示空中租位个数。

　　接下来 n 行，描述每个租位。其中第i行包含4个整数xi, yi, zi, ri，表示第i个租位彩云的圆心在(xi, yi, zi)位置，圆半径为ri。

输出格式

　　要求输出一个实数，四舍五入保留两位有效数字，表示农场里能长庄稼的土地的面积。

样例输入

10 10

90.0

1

5 5 10 5

样例输出

21.46

样例输入

8 8

90.0

1

4 4 10 5

样例输出

1.81

样例输入

20 10

45.0

2

5 0 5 5

8 6 14 6

样例输出

130.15

约数倍数选卡片

问题描述

　　闲暇时，福尔摩斯和华生玩一个游戏：

　　在N张卡片上写有N个整数。两人轮流拿走一张卡片。要求下一个人拿的数字一定是前一个人拿的数字的约数或倍数。例如，某次福尔摩斯拿走的卡片上写着数字“6”，则接下来华生可以拿的数字包括：

　　1，2，3, 6，12，18，24 ....

　　当轮到某一方拿卡片时，没有满足要求的卡片可选，则该方为输方。

　　请你利用计算机的优势计算一下，在已知所有卡片上的数字和可选哪些数字的条件下，怎样选择才能保证必胜！

　　当选多个数字都可以必胜时，输出其中最小的数字。如果无论如何都会输，则输出-1。

输入格式

　　输入数据为2行。第一行是若干空格分开的整数（每个整数介于1~100间），表示当前剩余的所有卡片。

　　第二行也是若干空格分开的整数，表示可以选的数字。当然，第二行的数字必须完全包含在第一行的数字中。

输出格式

　　程序则输出必胜的招法！

样例输入

2 3 6

3 6

样例输出

3

样例输入

1 2 2 3 3 4 5

3 4 5

样例输出

4

车轮轴迹

问题描述

　　栋栋每天骑自行车回家需要经过一条狭长的林荫道。道路由于年久失修，变得非常不平整。虽然栋栋每次都很颠簸，但他仍把骑车经过林荫道当成一种乐趣。

　　由于颠簸，栋栋骑车回家的路径是一条上下起伏的曲线，栋栋想知道，他回家的这条曲线的长度究竟是多长呢？更准确的，栋栋想知道从林荫道的起点到林荫道的终点，他的车前轮的轴（圆心）经过的路径的长度。

　　栋栋对路面进行了测量。他把道路简化成一条条长短不等的直线段，这些直线段首尾相连，且位于同一平面内。并在该平面内建立了一个直角坐标系，把所有线段的端点坐标都计算好。

　　假设栋栋的自行车在行进的过程中前轮一直是贴着路面前进的。

　　上图给出了一个简单的路面的例子，其中蓝色实线为路面，红色虚线为车轮轴经过的路径。在这个例子中，栋栋的前轮轴从A点出发，水平走到B点，然后绕着地面的F点到C点（绕出一个圆弧），再沿直线下坡到D点，最后水平走到E点，在这个图中地面的坐标依次为：(0, 0), (2, 0), (4, -1), (6, -1)，前轮半径为1.50，前轮轴前进的距离依次为：

　　AB=2.0000；弧长BC=0.6955；CD=1.8820；DE=1.6459。

　　总长度为6.2233。

　　下图给出了一个较为复杂的路面的例子，在这个例子中，车轮在第一个下坡还没下完时（D点）就开始上坡了，之后在坡的顶点要从E绕一个较大的圆弧到F点。这个图中前轮的半径为1，每一段的长度依次为：

　　AB=3.0000；弧长BC=0.9828；CD=1.1913；DE=2.6848；弧长EF=2.6224； FG=2.4415；GH=2.2792。

　　总长度为15.2021。

　　现在给出了车轮的半径和路面的描述，请求出车轮轴轨迹的总长度。

输入格式

　　输入的第一行包含一个整数n和一个实数r，用一个空格分隔，表示描述路面的坐标点数和车轮的半径。

　　接下来n行，每个包含两个实数，其中第i行的两个实数x[i], y[i]表示描述路面的第i个点的坐标。

　　路面定义为所有路面坐标点顺次连接起来的折线。给定的路面的一定满足以下性质：

　　\*第一个坐标点一定是(0, 0)；

　　\*第一个点和第二个点的纵坐标相同；

　　\*倒数第一个点和倒数第二个点的纵坐标相同；

　　\*第一个点和第二个点的距离不少于车轮半径；

　　\*倒数第一个点和倒数第二个点的的距离不少于车轮半径；

　　\*后一个坐标点的横坐标大于前一个坐标点的横坐标，即对于所有的i，x[i+1]>x[i]。

输出格式

　　输出一个实数，四舍五入保留两个小数，表示车轮轴经过的总长度。

　　你的结果必须和参考答案一模一样才能得分。数据保证答案精确值的小数点后第三位不是4或5。

样例输入

4 1.50

0.00 0.00

2.00 0.00

4.00 -1.00

6.00 -1.00

样例输出

6.22

样例说明

　　这个样例对应第一个图。

样例输入

6 1.00

0.00 0.00

3.00 0.00

5.00 -3.00

6.00 2.00

7.00 -1.00

10.00 -1.00

样例输出

15.20

样例说明

　　这个样例对应第二个图

数据规模和约定

　　对于20%的数据，n=4；

　　对于40%的数据，n≤10；

　　对于100%的数据，4≤n≤100，0.5≤r≤20.0，x[i] ≤2000.0，-2000.0≤y[i] ≤2000.0。

九宫重排

问题描述

　　如下面第一个图的九宫格中，放着 1~8 的数字卡片，还有一个格子空着。与空格子相邻的格子中的卡片可以移动到空格中。经过若干次移动，可以形成第二个图所示的局面。

　　我们把第一个图的局面记为：12345678.

　　把第二个图的局面记为：123.46758

　　显然是按从上到下，从左到右的顺序记录数字，空格记为句点。

　　本题目的任务是已知九宫的初态和终态，求最少经过多少步的移动可以到达。如果无论多少步都无法到达，则输出-1。

输入格式

　　输入第一行包含九宫的初态，第二行包含九宫的终态。

输出格式

　　输出最少的步数，如果不存在方案，则输出-1。

样例输入

12345678.

123.46758

样例输出

3

样例输入

13524678.

46758123.

样例输出

22

公式求值

问题描述

　　输入n, m, k，输出下面公式的值。

　　其中C\_n^m是组合数，表示在n个人的集合中选出m个人组成一个集合的方案数。组合数的计算公式如下。

输入格式

　　输入的第一行包含一个整数n；第二行包含一个整数m，第三行包含一个整数k。

输出格式

　　计算上面公式的值，由于答案非常大，请输出这个值除以999101的余数。

样例输入

3

1

3

样例输出

162

样例输入

20

10

10

样例输出

359316

数据规模和约定

　　对于10%的数据，n≤10，k≤3；

　　对于20%的数据，n≤20，k≤3；

　　对于30%的数据，n≤1000，k≤5；

　　对于40%的数据，n≤10^7，k≤10；

　　对于60%的数据，n≤10^15，k ≤100；

　　对于70%的数据，n≤10^100，k≤200；

　　对于80%的数据，n≤10^500，k ≤500；

　　对于100%的数据，n在十进制下不超过1000位，即1≤n<10^1000，1≤k≤1000，同时0≤m≤n，k≤n。

提示

　　999101是一个质数；

　　当n位数比较多时，绝大多数情况下答案都是0，但评测的时候会选取一些答案不是0的数据；

import java.math.BigInteger;

import java.util.Scanner;

public class Main{

    /\*\*\*

     \* @author 林梵

     \*/

    public static BigInteger lucas(BigInteger n,BigInteger m,BigInteger p){

        if(m.equals(BigInteger.ZERO)) return BigInteger.ONE;

        return BigInteger.valueOf(f(n.mod(p).longValue(),m.mod(p).longValue())).multiply(lucas(n.divide(p),m.divide(p),p)).mod(p);

    }

    public static long f(long n,long m){

        if(m>n) return 1;

        if(n==m|| m==0) return 1;

        if(m>n-m) m=n-m;

        long tmpi=1,tmpn=1,s1=1,s2=1,ans=1;

        for (int i = 1; i<=m; i++) {

            tmpi=i;

            tmpn=n-i+1;

            s1=s1\*tmpi%999101;

            s2=s2\*tmpn%999101;

        }

        ans = s2\*pow1(s1,999099)%999101;

        return ans%999101;

    }

    public static long pow1(long x,long n) {

        if(x==1) return 1;

        if (n==0)

            return 1;

        else {

            while ((n & 1)==0) {

                n>>=1;

                x=(x \*x)%999101;

            }

        }

        long result = x%999101;

        n>>=1;

        while (n!=0) {

            x=(x \*x)%999101;;

            if ((n & 1)!=0)

                result =result\*x%999101;

            n>>=1;

        }

        return  result;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        BigInteger n = new BigInteger(sc.nextLine());

        BigInteger m = new BigInteger(sc.nextLine());

        int k = Integer.parseInt(sc.nextLine());

        BigInteger md = new BigInteger("999101");

        long Cnm=lucas(n, m,md).longValue()%999101;

        long sum = 0;

        if(Cnm!=0){

            int[][] a = new int[k][k];

            int h = 1;

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

                for (int j = 0; j < k; j++) {

                    if (j >= h)

                        a[i][j] =0;

                    else {

                        if (j == 0 || j == h - 1)

                            a[i][j] = 1;

                        else {

                            a[i][j] = (a[i - 1][j - 1]\*(h - j)+a[i - 1][j])%999101;

                        }

                    }

                }

                h++;

            }

            long m1 = 1,n1 =1;

            long x=n.subtract(new BigInteger(k+"")).mod(md.subtract(BigInteger.ONE)).longValue();

            long n3 = pow1(2,x);

            for (int i = k - 1; i >= 0; i--) {

                n1=n3\*pow1(2,i)%999101;

                m1 = m1\*(n.subtract(new BigInteger((k - 1 - i) + "")).mod(md).longValue())%999101;

                sum = (sum+m1\*a[k - 1][i]\*n1)%999101;

            }

            sum = sum\*Cnm%999101;

        }

        System.out.println(sum);

    }

}