题号 题目 知识点 难度 通过率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BD1 | 罪犯转移 | 动态规划 | 较难 | 15.94% |
| BD2 | 裁减网格纸 | 贪心 | 中等 | 25.14% |
| BD3 | 钓鱼比赛 | 数学 | 困难 | 14.79% |
| BD4 | 蘑菇阵 | 动态规划 | 较难 | 14.64% |
| BD5 | 字符串匹配 | 字符串模拟穷举 | 中等 | 28.26% |
| BD6 | 表格排序 | 排序模拟 | 中等 | 24.81% |
| BD7 | 替换链接 | 字符串模拟 | 中等 | 3.09% |
| BD8 | 完成括号匹配 | 字符串模拟栈 | 中等 | 34.55% |
| BD9 | 倒计时 | 模拟 | 中等 | 12.35% |
| BD10 | 字符覆盖 | 排序字符串模拟 | 中等 | 36.66% |
| BD11 | 分页组件 | 模拟 | 中等 | 14.07% |
| BD12 | 最大子序列 | 字符串贪心 | 中等 | 32.90% |
| BD13 | 正三角形的顶点位置 |  | 简单 | 40.79% |
| BD14 | 双素数 | 数学 | 中等 | 25.24% |
| BD15 | 序列合并 | 排序 | 中等 | 34.08% |
| BD16 | 浇花 | 排序贪心穷举 | 较难 | 13.44% |
| BD17 | 智能提示 | 字符串模拟 | 中等 | 0.00% |
| BD18 | 猜数 | 穷举 | 较难 | 21.30% |
| BD19 | 文字输出 | 字符串模拟 | 中等 | 7.33% |

# ### BD1 罪犯转移

####题目描述

```

C市现在要转移一批罪犯到D市，C市有n名罪犯，按照入狱时间有顺序，另外每个罪犯有一个罪行值，值越大罪越重。现在为了方便管理，市长决定转移入狱时间连续的c名犯人，同时要求转移犯人的罪行值之和不超过t，问有多少种选择的方式（一组测试用例可能包含多组数据，请注意处理）？

```

####输入描述:

```

第一行数据三个整数:n，t，c(1≤n≤2e5,0≤t≤1e9,1≤c≤n)，第二行按入狱时间给出每个犯人的罪行值ai(0≤ai≤1e9)

```

```

####输出描述:

```

一行输出答案。

```

####示例1

```

####输入

```

3 100 2

1 2 3

```

####输出

```

2

import java.io.\*;

import java.util.\*;

public class Main {

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

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

String str;

while((str=br.readLine())!=null){

String[] s = str.trim().split(" ");

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

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

int c = Integer.parseInt(s[2]);

int[] value = new int[n];

if((str=br.readLine())!=null){

String[] s2 = str.trim().split(" ");

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

value[i] = Integer.parseInt(s2[i]);

}

}

System.out.println(carryMan(n,t,c,value));

}

br.close();

}

public static int carryMan(int n,int t,int c,int[] value){

int count=0;

int sum=0;

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

sum+=value[j];

}

if(sum<=t){

count++;

}

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

sum=sum+value[i]-value[i-c];

if(sum<=t){

count++;

}

}

return count;

}

}

# ```

# ### BD2 裁减网格纸

####题目描述

```

度度熊有一张网格纸，但是纸上有一些点过的点，每个点都在网格点上，若把网格看成一个坐标轴平行于网格线的坐标系的话，每个点可以用一对整数x，y来表示。度度熊必须沿着网格线画一个正方形，使所有点在正方形的内部或者边界。然后把这个正方形剪下来。问剪掉正方形的最小面积是多少。

```

####输入描述:

```

第一行一个数n(2≤n≤1000)表示点数，接下来每行一对整数xi,yi(－1e9<=xi,yi<=1e9)表示网格上的点

```

```

####输出描述:

```

一行输出最小面积

```

####示例1

```

####输入

```

2

0 0

0 3

```

####输出

```

9

import java.util.\*;

import java.io.\*;

public class Main {

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

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

String str;

int i;

int maxx,minx,maxy,miny;

while((str=br.readLine())!=null){

int n=Integer.parseInt(str);

if((str=br.readLine())!=null){

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

maxx=Integer.parseInt(data[0]);

minx=Integer.parseInt(data[0]);

maxy=Integer.parseInt(data[1]);

miny=Integer.parseInt(data[1]);

for(i=1;i<n&&((str=br.readLine())!=null);i++){

data=str.split(" ");

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

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

if(x>maxx) maxx=x;

if(x<minx) minx=x;

if(y>maxy) maxy=y;

if(y<miny) miny=y;

}

long area;

int p=maxx-minx;

int q=maxy-miny;

if(p>q) area=(long)p\*(long)p;

else area=(long)q\*(long)q;

System.out.println(area);

}

}

br.close();

}

}

# ```

# ### BD3 钓鱼比赛

####题目描述

```

ss请cc来家里钓鱼，鱼塘可划分为n＊m的格子，每个格子有不同的概率钓上鱼，cc一直在坐标(x,y)的格子钓鱼，而ss每分钟随机钓一个格子。问t分钟后他们谁至少钓到一条鱼的概率大？为多少？

```

####输入描述:

```

第一行五个整数n,m,x,y,t(1≤n,m,t≤1000,1≤x≤n,1≤y≤m);

接下来为一个n＊m的矩阵，每行m个一位小数，共n行，第i行第j个数代表坐标为(i,j)的格子钓到鱼的概率为p(0≤p≤1)

```

```

####输出描述:

```

输出两行。第一行为概率大的人的名字(cc/ss/equal),第二行为这个概率(保留2位小数)

```

####示例1

```

####输入

```

2 2 1 1 1

0.2 0.1

0.1 0.4

```

####输出

```

equal

0.20

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

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

// TODO Auto-generated method stub

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

String s="";

while((s=br.readLine())!=null){

String[] strs = s.trim().split(" ");

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

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

int x = Integer.parseInt(strs[2]);

int y = Integer.parseInt(strs[3]);

int t = Integer.parseInt(strs[4]);

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

double sum = 0;

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

s = br.readLine();

strs = s.trim().split(" ");

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

matrix[i][j] = Double.parseDouble(strs[j]);

sum+=matrix[i][j];

}

}

double cc = 1-Math.pow((1-matrix[x-1][y-1]), t);

double ss = 1-Math.pow((1-sum/(n\*m)), t);

if(cc>ss){

System.out.println("cc");

System.out.printf("%.2f\n", cc);

}else if(cc<ss){

System.out.println("ss");

System.out.printf("%.2f\n", ss);

}else{

System.out.println("equal");

System.out.printf("%.2f\n", ss);

}

}

br.close();

}

}

# ```

# ### BD4 蘑菇阵

####题目描述

```

现在有两个好友A和B，住在一片长有蘑菇的由n＊m个方格组成的草地，A在(1,1),B在(n,m)。现在A想要拜访B，由于她只想去B的家，所以每次她只会走(i,j+1)或(i+1,j)这样的路线，在草地上有k个蘑菇种在格子里(多个蘑菇可能在同一方格),问：A如果每一步随机选择的话(若她在边界上，则只有一种选择)，那么她不碰到蘑菇走到B的家的概率是多少？

```

####输入描述:

```

第一行N，M，K(1 ≤ N,M ≤ 20, k ≤ 100),N,M为草地大小，接下来K行，每行两个整数x，y，代表(x,y)处有一个蘑菇。

```

```

####输出描述:

```

输出一行，代表所求概率(保留到2位小数)

```

####示例1

```

####输入

```

2 2 1

2 1

```

####输出

```

0.50

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

public class Main {

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

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

String line;

while((line = br.readLine()) != null) {

String[] params = line.trim().split(" ");

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

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

int k = Integer.parseInt(params[2]);

int[][] grid = new int[n + 1][m + 1];

int x, y;

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

params = br.readLine().trim().split(" ");

x = Integer.parseInt(params[0]);

y = Integer.parseInt(params[1]);

grid[x][y] = 1;

}

double[][] dp = new double[n + 1][m + 1];

dp[1][1] = 1.0; // 初始化起点的概率

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

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

if(i == 1 && j == 1) continue;

// 只要当前位置没有蘑菇，就可以从上面或者左边移动到当前位置

if(grid[i][j] == 0)

dp[i][j] = (j == m? 1: 0.5)\*dp[i - 1][j] + (i == n? 1: 0.5)\*dp[i][j - 1];

}

}

System.out.printf("%.2f\n", dp[n][m]);

}

}

}

# ```

# ### BD5 字符串匹配

####题目描述

```

牛牛有两个字符串A和B,其中A串是一个01串,B串中除了可能有0和1,还可能有'?',B中的'?'可以确定为0或者1。 寻找一个字符串T是否在字符串S中出现的过程,称为字符串匹配。牛牛现在考虑所有可能的字符串B,有多少种可以在字符串A中完成匹配。

例如:A = "00010001", B = "??"

字符串B可能的字符串是"00","01","10","11",只有"11"没有出现在字符串A中,所以输出3

```

####输入描述:

```

输入包括两行,第一行一个字符串A,字符串A长度length(1 ≤ length ≤ 50),A中每个字符都是'0'或者'1'。

第二行一个字符串B,字符串B长度length(1 ≤ length ≤ 50),B中的字符包括'0','1'和'?'。

```

```

####输出描述:

```

输出一个整数,表示能完成匹配的字符串种数。

```

####示例1

```

####输入

```

00010001

??

```

####输出

```

3

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.HashSet;

public class Main {

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

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

char[] a = br.readLine().toCharArray();

char[] b = br.readLine().toCharArray();

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

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

StringBuilder sb = new StringBuilder();

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

if (b[j] == '?' || b[j] == a[i + j]) {

sb.append(a[i + j]);

}

}

if (sb.toString().length() == b.length) set.add(sb.toString());

}

System.out.println(set.size());

}

}

# ```

# ### BD6 表格排序

####题目描述

```

系统会在tbody中随机生成一份产品信息表单，如html所示。

请完成 sort 函数，根据参数的要求对表单所有行进行重新排序。

1、type为id、price或者sales，分别对应第1 ~ 3列

2、order为asc或者desc，asc表示升序，desc为降序

3、例如 sort('price', 'asc') 表示按照price列从低到高排序

4、所有表格内容均为数字，每一列数字均不会重复

5、不要使用第三方插件

题解：略（前端的算法题）

# ```

# ### BD7 替换链接

略

# ```

# ### BD8 完成括号匹配

####题目描述

```

合法的括号匹配序列被定义为:

1. 空串""是合法的括号序列

2. 如果"X"和"Y"是合法的序列,那么"XY"也是一个合法的括号序列

3. 如果"X"是一个合法的序列,那么"[X]"也是一个合法的括号序列

4. 每个合法的括号序列都可以由上面的规则生成

例如"", "[]", "[][][]", "[[][]]", "[[[[]]]]"都是合法的。

牛牛现在给出一个括号序列s,牛牛允许你执行的操作是:在s的开始和结尾处添加一定数量的左括号('[')或者右括号(']')使其变为一个合法的括号匹配序列。牛牛希望你能求出添加最少的括号之后的合法的括号匹配序列是什么。

```

####输入描述:

```

输入包括一个字符串s,s的长度length(1 ≤ length ≤ 50),s中只包含'['和']'。

```

```

####输出描述:

```

输出一个字符串,表示括号完全匹配的序列。

```

####示例1

```

####输入

```

][

```

####输出

```

[][]

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

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

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

char[] str = br.readLine().toCharArray();

int count = 0;

StringBuilder sb = new StringBuilder();

for (char ch : str) {

if (ch == '[') count++;

else {

if (count == 0) sb.insert(0, '[');

else count--;

}

sb.append(ch);

}

for (int i = 0; i < count; i++) sb.append(']');

System.out.println(sb);

}

}

# ```

# ### BD9 倒计时

略

# ```

# ### BD10 字符覆盖

####题目描述

```

小度有一个小写字母组成的字符串s.字符串s已经被写在墙上了.

小度还有很多卡片,每个卡片上有一个小写字母,组成一个字符串t。小度可以选择字符串t中任意一个字符,然后覆盖在字符串s的一个字符之上。小度想知道在选取一些卡片覆盖s的一些字符之后,可以得到的字典序最大的字符串是什么。

```

####输入描述:

```

输入包括两行,第一行一个字符串s,字符串s长度length(1 ≤ length ≤ 50),s中每个字符都是小写字母

第二行一个字符串t,字符串t长度length(1 ≤ length ≤ 50),t中每个字符都是小写字母

```

```

####输出描述:

```

输出一个字符串,即可以得到的字典序最大字符串

```

####示例1

```

####输入

```

fedcba

ee

```

####输出

```

feeeba

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Arrays;

public class Main {

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

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

StringBuilder s1 = new StringBuilder(br.readLine());

String s2 = br.readLine();

char[] chars = s2.toCharArray();

Arrays.sort(chars);

int n = s2.length();

for (int i=0;i<s1.length();i++){

if (s1.charAt(i)<chars[n-1]){

s1.replace(i,i+1,chars[n-1]+"");

n--;

}

if (n<=0)

break;

}

System.out.println(s1);

}

}

# ```

# ### BD11 分页组件

略

# ```

# ### BD12 最大子序列

####题目描述

```

对于字符串x和y, 如果擦除x中的某些字母(有可能全擦掉或者都不擦)能够得到y,我们就称y是x的子序列。例如."ncd"是"nowcoder"的子序列,而"xt"不是。

现在对于给定的一个字符串s,请计算出字典序最大的s的子序列。

```

####输入描述:

```

输入包括一行,一个字符串s,字符串s长度length(1 ≤ length ≤ 50).

s中每个字符都是小写字母

```

```

####输出描述:

```

输出一个字符串,即字典序最大的s的子序列。

```

####示例1

```

####输入

```

test

```

####输出

```

tt

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

/\*\*

\* @author wylu

\*/

public class Main {

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

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

char[] s = br.readLine().toCharArray();

StringBuilder sb = new StringBuilder();

char ch = s[s.length - 1];

for (int i = s.length - 1; i >= 0; i--) {

if (s[i] >= ch) {

sb.append(s[i]);

ch = s[i];

}

}

System.out.println(sb.reverse());

}

}

# ```

# ### BD13 正三角形的顶点位置

####题目描述

```

给出直角三角坐标平面上三角形其中两个顶点的坐标，求第三个顶点的坐标，要求保留小数点后两位小数

```

####输入描述:

```

有多组测试数据，输入的第一行是整数T(1≤T≤200)表示随后测试数据的组数。

每组测试数据占一行，由4个带两位小数由一个空格隔开的实数构成，表示已知的两个顶点的横纵坐标。

```

```

####输出描述:

```

对应每组测试数据，输出对应的第三个顶点（两组解），如果两组解的横坐标不相等，则先输出横坐标较小的顶点，否则输出纵坐标较小的顶点，每组输出占一行，输出保留两位小数

```

####示例1

```

####输入

```

3

12.00 3.00 12.00 9.00

12.00 3.00 24.00 3.00

1.00 2.00 3.00 4.00

```

####输出

```

6.80 6.00 17.20 6.00

18.00 -7.39 18.00 13.39

0.27 4.73 3.73 1.27

import java.io.\*;

import java.util.\*;

public class Main {

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

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

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

while (t-- != 0) {

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

double x1 = Double.parseDouble(strs[0]), y1 = Double.parseDouble(strs[1]);

double x2 = Double.parseDouble(strs[2]), y2 = Double.parseDouble(strs[3]);

double dx = x2 - x1, dy = y2 - y1;

double edgeLen = Math.sqrt(dx \* dx + dy \* dy);

double height = Math.sqrt(3) / 2.0 \* edgeLen;

double resX1, resY1, resX2, resY2;

if (dx == 0) {

resX1 = x1 - height;

resX2 = x1 + height;

resY1 = (y1 + y2) / 2.0;

resY2 = resY1;

} else if (dy == 0) {

resX1 = (x1 + x2) / 2.0;

resX2 = resX1;

resY1 = y1 - height;

resY2 = y1 + height;

} else {

double midX = (x1 + x2) / 2.0, midY = (y1 + y2) / 2.0;

double sine = Math.abs(dy / edgeLen), k = dy / dx;

double ox = height \* sine, oy = ox / k;

resX1 = midX - ox;

resY1 = midY + oy;

resX2 = midX + ox;

resY2 = midY - oy;

}

System.out.printf("%.2f %.2f %.2f %.2f\n", resX1, resY1, resX2, resY2);

}

}

}

# ```

# ### BD14 双素数

####题目描述

```

一个正整数是素数当且仅当它除了1和自身以外没有其他因子，现在我们定义双素数；一个正整数是双素数当且仅当它本身是个素数，并且将他的十进制表示反转后得到数不等于它自身且也是个素数，如13就是一个双素数，因为13和31不相等且都是素数，现给出一个整数k,你需要找到第k小的双素数

```

####输入描述:

```

第一行包含一个整数k,1≤k≤10000

```

```

####输出描述:

```

若第k小的素数不超过10^6则输出它，否则输出-1

```

####示例1

```

####输入

```

1

```

####输出

```

13

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

/\*\*

\* @author wylu

\*/

public class Main {

//元素值为false表示该元素下标值为素数

static boolean[] mark = eulerSieve(1000000);

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

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

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

int res = -1;

for (int i = 13, j = 0; j < k && i < mark.length; i += 2) {

if (!mark[i]) {

int ri = reverse(i);

if (ri != i && !mark[ri]) j++;

if (j == k) res = i;

}

}

System.out.println(res);

}

private static int reverse(int x) {

int s = 0;

while (x != 0) {

s = s \* 10 + x % 10;

x /= 10;

}

return s;

}

//欧拉线性筛法 O(n)

private static boolean[] eulerSieve(int n) {

boolean[] mark = new boolean[n + 1];

int[] primes = new int[n / 2];

for (int i = 2, count = 0; i <= n; i++) {

if (!mark[i]) primes[count++] = i;

for (int j = 0; j < count && i \* primes[j] <= n; j++) {

mark[i \* primes[j]] = true;

if (i % primes[j] == 0) break;

}

}

return mark;

}

}

# ```

# ### BD15 序列合并

####题目描述

```

F(n)=a7n7+a6n6+a5n5+a4n4+a3n3+a2n2+a1n+a0

其中系数aj都是整数满足0≤aj≤1000且至少有两个系数严格大于0，分别将n=1,n=2,n=3n...代入以上函数可以得到一个无穷长度的整数序列，即用8个系数a7,a6...a0可以唯一确定一个无穷长度的整数序列，现在给出k个通过以上方法定义的无穷序列，你需要求出将这些序列所有数字放在一起后，第n小的数字是多少？

```

####输入描述:

```

第一行包含一个整数k,1≤k≤104

接下来k行，每行包含8个整数a7,a6,.....a0,表示一个函数的系数，0≤aj≤1000

最后一行包含一个整数n,1≤n≤105

```

```

####输出描述:

```

输出对应的答案，保证最后的答案不超过1017

```

####示例1

```

####输入

```

3

0 0 0 0 1 2 0 0

0 0 0 0 0 0 10 6

0 0 0 0 0 0 25 1

9

```

####输出

```

51

import java.io.\*;

import java.util.\*;

public class Main{

static class Node{

long val;

int cnt;

int id;

public Node(long val, int cnt, int id){this.val = val; this.cnt = cnt; this.id = id;}

}

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

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

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

int[][] seq = new int[k][8];

String[] strs = null;

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

strs = br.readLine().split(" ");

for(int j = 0; j < 8; j++)

seq[i][7-j] = Integer.parseInt(strs[j]);

}

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

br.close(); strs = null;

// 为了更快可以不用lambda

//PriorityQueue<Node> queue = new PriorityQueue<>(k, (x,y)-> x.val-y.val>0?1:(x.val-y.val==0?0:-1));

PriorityQueue<Node> queue = new PriorityQueue<>(k, new Comparator<Node>(){

@Override

public int compare(Node x, Node y){

return x.val-y.val>0?1:(x.val-y.val==0?0:-1); // 最好不要将x.val-y.val强转为int,可能溢出

}

});

int cnt = 0; long res = 0;

for(int i = 0; i < k; i++) queue.offer(new Node(cal(seq[i],1),1,i));

while(cnt++ != n){

Node tmp = queue.poll();

res = tmp.val;

tmp.val = cal(seq[tmp.id],++(tmp.cnt));

queue.offer(tmp);

}

System.out.println(res);

}

private static long cal(int[] par, int n){

long val = 0; long tmp = 1;

for(int i = 0; i < 8; i++){

val += par[i]\*tmp;

tmp \*= n;

}

return val;

}

}

# ```

# ### BD16 浇花

####题目描述

```

一个花坛中有很多花和两个喷泉。

喷泉可以浇到以自己为中心，半径为r的圆内的所有范围的花。

现在给出这些花的坐标和两个喷泉的坐标，要求你安排两个喷泉浇花的半径r12和r22，使得所有的花都能被浇到的同时, r12 + r22 的值最小。

```

####输入描述:

```

第一行5个整数n，x1，y1，x2，y2表示花的数量和两个喷泉的坐标。

接下来n行，每行两个整数xi, yi表示第i朵花的坐标。

满足1 <= n <= 2000，花和喷泉的坐标满足-107<= x, y <= 107。

```

```

####输出描述:

```

一个整数，r12 + r22 的最小值。

```

####示例1

```

####输入

```

2 -1 0 5 3

0 2

5 2

```

####输出

```

6

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Arrays;

import java.util.Comparator;

public class Main {

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

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

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

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

int x1 = Integer.parseInt(s[1]);

int y1 = Integer.parseInt(s[2]);

int x2 = Integer.parseInt(s[3]);

int y2 = Integer.parseInt(s[4]);

long[][] r = new long[n][2];

long d2 = 0, min = Long.MAX\_VALUE;

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

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

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

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

r[i][0] = (long) (Math.pow(x - x1, 2) + Math.pow(y - y1, 2));

r[i][1] = (long) (Math.pow(x - x2, 2) + Math.pow(y - y2, 2));

}

Arrays.sort(r, new Comparator<long[]>() {

@Override

public int compare(long[] o1, long[] o2) {

return o1[0] > o2[0] ? -1 : 1;

}

});

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

min = Math.min(min, r[i][0] + d2);

d2 = Math.max(d2, r[i][1]);

}

System.out.println(Math.min(min, d2));

}

}

# ```

# ### BD17 智能提示

略

# ```

# ### BD18 猜数

####题目描述

```

牛牛和妞妞正在玩一个猜数游戏，妞妞心里想两个不相等的正数，把这两个正数的和y告诉牛牛。

妞妞声称这两个数都不超过x，让牛牛猜这两个数是多少。

牛牛每猜一次，妞妞会告诉他猜对了还是猜错了，猜对了就停止游戏，猜错了就直到牛牛猜对为止。

妞妞为了加大难度，有时会误报x的大小，如果牛牛可以判断出了这个x是错误的，就会直接询问妞妞答案。

牛牛最坏情况下要猜多少次才能猜到妞妞想的数呢？

```

####输入描述:

```

两个整数x，y。1<=x,y<=1014。

```

```

####输出描述:

```

一个数n，表示牛牛在最坏情况下猜测的次数。

```

####示例1

```

####输入

```

7 10

```

####输出

```

2

```

####示例2

```

####输入

```

4 10

```

####输出

```

0

import java.io.\*;

public class Main {

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

InputStream in = System.in;

BufferedInputStream is=new BufferedInputStream(in);

String s="";

int i;

while ((i=is.read())!=10){

char a= (char) i;

s+=a;

}

String[] s1 = s.split(" ");

Long x= Long.valueOf(s1[0]);

Long y= Long.valueOf(s1[1]);

if(2\*x<=y){

System.out.println(0);

}else if(x>=y) {

System.out.println(y/2);

}else{

System.out.println(x-(y/2));

}

}

}

# ```

# ### BD19 文字```

####输出

```

略