**2019年**

参考链接：<https://blog.csdn.net/qq_38265137/article/details/88866269>

1.组队：

作为篮球队教练，你需要从以下名单中选出1 号位至5 号位各一名球员，  
组成球队的首发阵容。  
每位球员担任1 号位至5 号位时的评分如下表所示。请你计算首发阵容1  
号位至5 号位的评分之和最大可能是多少？

答案

490

2.不同的子串：

一个字符串的非空子串是指字符串中长度至少为1 的连续的一段字符组成

的串。例如，字符串aaab 有非空子串a, b, aa, ab, aaa, aab, aaab，一共7 个。

注意在计算时，只算本质不同的串的个数。

请问，字符串0100110001010001 有多少个不同的非空子串？

答案：

100

3.数列求值：

给定数列1, 1, 1, 3, 5, 9, 17, …，从第4 项开始，每项都是前3 项的和。求  
第20190324 项的最后4 位数字。

答案：

4659

4.数的分解：

把2019 分解成3 个各不相同的正整数之和，并且要求每个正整数都不包  
含数字2 和4，一共有多少种不同的分解方法？  
注意交换3 个整数的顺序被视为同一种方法，例如1000+1001+18 和  
1001+1000+18 被视为同一种。

答案：

40785

5.迷宫：

下图给出了一个迷宫的平面图，其中标记为1 的为障碍，标记为0 的为可  
以通行的地方。

010000

000100

001001

110000

迷宫的入口为左上角，出口为右下角，在迷宫中，只能从一个位置走到这

个它的上、下、左、右四个方向之一。

对于上面的迷宫，从入口开始，可以按DRRURRDDDR 的顺序通过迷宫，

一共10 步。其中D、U、L、R 分别表示向下、向上、向左、向右走。

对于下面这个更复杂的迷宫（30 行50 列），请找出一种通过迷宫的方式，

其使用的步数最少，在步数最少的前提下，请找出字典序最小的一个作为答案。

请注意在字典序中D<L<R<U。

01010101001011001001010110010110100100001000101010

00001000100000101010010000100000001001100110100101

01111011010010001000001101001011100011000000010000

01000000001010100011010000101000001010101011001011

00011111000000101000010010100010100000101100000000

11001000110101000010101100011010011010101011110111

00011011010101001001001010000001000101001110000000

10100000101000100110101010111110011000010000111010

00111000001010100001100010000001000101001100001001

11000110100001110010001001010101010101010001101000

00010000100100000101001010101110100010101010000101

11100100101001001000010000010101010100100100010100

00000010000000101011001111010001100000101010100011

10101010011100001000011000010110011110110100001000

10101010100001101010100101000010100000111011101001

10000000101100010000101100101101001011100000000100

10101001000000010100100001000100000100011110101001

00101001010101101001010100011010101101110000110101

11001010000100001100000010100101000001000111000010

00001000110000110101101000000100101001001000011101

10100101000101000000001110110010110101101010100001

00101000010000110101010000100010001001000100010101

10100001000110010001000010101001010101011111010010

00000100101000000110010100101001000001000000000010

11010000001001110111001001000011101001011011101000

00000110100010001000100000001000011101000000110011

10101000101000100010001111100010101001010000001000

10000010100101001010110000000100101010001011101000

00111100001000010000000110111000000001000000001011

10000001100111010111010001000110111010101101111000

6.特别数的和：

小明对数位中含有2、0、1、9 的数字很感兴趣（不包括前导0），在1 到

40 中这样的数包括1、2、9、10 至32、39 和40，共28 个，他们的和是574。

请问，在1 到n 中，所有这样的数的和是多少？

输入：

输入一行包含一个整数n。

输出：

输出一行，包含一个整数，表示满足条件的数的和。

【样例输入】

40

【样例输出】

574

7.外卖店优先级：

“饱了么”外卖系统中维护着N 家外卖店，编号1 N。每家外卖店都有

一个优先级，初始时(0 时刻) 优先级都为0。

每经过1 个时间单位，如果外卖店没有订单，则优先级会减少1，最低减

到0；而如果外卖店有订单，则优先级不减反加，每有一单优先级加2。

如果某家外卖店某时刻优先级大于5，则会被系统加入优先缓存中；如果

优先级小于等于3，则会被清除出优先缓存。

给定T 时刻以内的M 条订单信息，请你计算T 时刻时有多少外卖店在优

先缓存中。

【输入格式】

第一行包含3 个整数N、M 和T。

以下M 行每行包含两个整数ts 和id，表示ts 时刻编号id 的外卖店收到

一个订单。

【输出格式】

输出一个整数代表答案。

【样例输入】

2 6 6

1 1

5 2

3 1

6 2

2 1

6 2

【样例输出】

1

样例解释】

6 时刻时，1 号店优先级降到3，被移除出优先缓存；2 号店优先级升到6，

加入优先缓存。所以是有1 家店(2 号) 在优先缓存中。

8.人物相关性分析：

【问题描述】

小明正在分析一本小说中的人物相关性。他想知道在小说中Alice 和Bob

有多少次同时出现。

更准确的说，小明定义Alice 和Bob“同时出现”的意思是：在小说文本

中Alice 和Bob 之间不超过K 个字符。

例如以下文本：

This is a story about Alice and Bob. Alice wants to send a private message to Bob.

假设K = 20，则Alice 和Bob 同时出现了2 次，分别是”Alice and Bob”

和”Bob. Alice”。前者Alice 和Bob 之间有5 个字符，后者有2 个字符。

注意:

Alice 和Bob 是大小写敏感的，alice 或bob 等并不计算在内。

Alice 和Bob 应为单独的单词，前后可以有标点符号和空格，但是不能

有字母。例如Bobbi 並不算出现了Bob。

【输入格式】

第一行包含一个整数K。

第二行包含一行字符串，只包含大小写字母、标点符号和空格。长度不超

过1000000。

【输出格式】

输出一个整数，表示Alice 和Bob 同时出现的次数。

【样例输入】

20

This is a story about Alice and Bob. Alice wants to send a private message to Bob.

【样例输出】

2

9.后缀表达式：

【问题描述】

给定N 个加号、M 个减号以及N + M + 1 个整数A1; A2; ; AN+M+1，小

明想知道在所有由这N 个加号、M 个减号以及N + M +1 个整数凑出的合法的

后缀表达式中，结果最大的是哪一个？

请你输出这个最大的结果。

例如使用1 2 3 + -，则“2 3 + 1 -” 这个后缀表达式结果是4，是最大的。

【输入格式】

第一行包含两个整数N 和M。

第二行包含N + M + 1 个整数A1; A2; ; AN+M+1。

【输出格式】

输出一个整数，代表答案。

【样例输入】

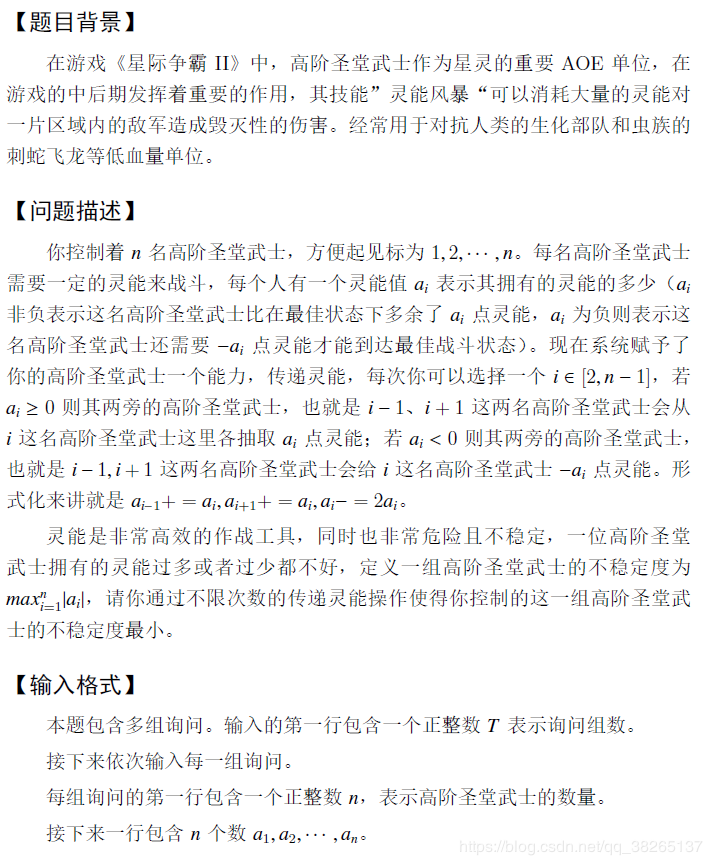
1 1

1 2 3

【样例输出】

4

10. 灵能传输:



【输出格式】

输出T 行。每行一个整数依次表示每组询问的答案。

【样例输入】

3

3

5 -2 3

4

0 0 0 0

3

1 2 3

【样例输出】

3

0

3

【样例说明】

对于第一组询问：

对2 号高阶圣堂武士进行传输操作后a1 = 3，a2 = 2，a3 = 1。答案为3。

对于第二组询问：

这一组高阶圣堂武士拥有的灵能都正好可以让他们达到最佳战斗状态。

【样例输入】

3

4

-1 -2 -3 7

4

2 3 4 -8

5

-1 -1 6 -1 -1

【样例输出】

5

7

4

**2018年**

参考：<https://blog.csdn.net/qq_33245342/article/details/79781855>

1．第几天

2000年的1月1日，是那一年的第1天。

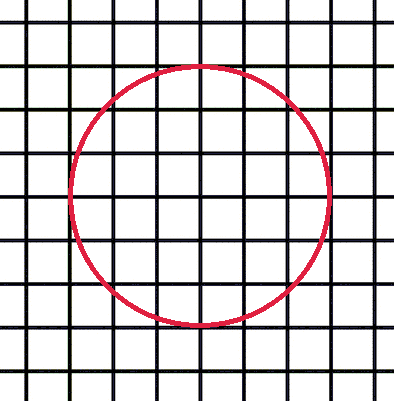
那么，2000年的5月4日，是那一年的第几天？

答案

125

2. 方格计数

在二维平面上有无数个1x1的小方格。



我们以某个小方格的一个顶点为圆心画一个半径为1000的圆。

你能计算出这个圆里有多少个完整的小方格吗？

注意：需要提交的是一个整数

3.复数幂

设i为虚数单位。对于任意正整数n，(2+3i)^n 的实部和虚部都是整数。

求 (2+3i)^123456 等于多少？ 即(2+3i)的123456次幂，这个数字很大，要求精确表示。

答案写成 "实部±虚部i" 的形式，实部和虚部都是整数（不能用科学计数法表示），中间任何地方都不加空格，实部为正时前面不加正号。(2+3i)^2 写成: -5+12i，

(2+3i)^5 的写成: 122-597i

注意：需要提交的是一个很庞大的复数，不要填写任何多余内容。

4.测试次数

x星球的居民脾气不太好，但好在他们生气的时候唯一的异常举动是：摔手机。

各大厂商也就纷纷推出各种耐摔型手机。x星球的质监局规定了手机必须经过耐摔测试，并且评定出一个耐摔指数来，之后才允许上市流通。

x星球有很多高耸入云的高塔，刚好可以用来做耐摔测试。塔的每一层高度都是一样的，与地球上稍有不同的是，他们的第一层不是地面，而是相当于我们的2楼。

如果手机从第7层扔下去没摔坏，但第8层摔坏了，则手机耐摔指数=7。

特别地，如果手机从第1层扔下去就坏了，则耐摔指数=0。

如果到了塔的最高层第n层扔没摔坏，则耐摔指数=n

为了减少测试次数，从每个厂家抽样3部手机参加测试。

某次测试的塔高为1000层，如果我们总是采用最佳策略，在最坏的运气下最多需要测试多少次才能确定手机的耐摔指数呢？

请填写这个最多测试次数。

注意：需要填写的是一个整数，不要填写任何多余内容。

5.快速排序

以下代码可以从数组a[]中找出第k小的元素。

它使用了类似快速排序中的分治算法，期望时间复杂度是O(N)的。

请仔细阅读分析源码，填写划线部分缺失的内容。

import java.util.Random;

public class Main{

public static int quickSelect(int a[], int l, int r, int k) {

Random rand = new Random();

int p = rand.nextInt(r - l + 1) + l;

int x = a[p];

int tmp = a[p]; a[p] = a[r]; a[r] = tmp;

int i = l, j = r;

while(i < j) {

                while(i < j && a[i] < x) i++;

                if(i < j) {

                        a[j] = a[i];

                        j--;

                }

                while(i < j && a[j] > x) j--;

                if(i < j) {

                        a[i] = a[j];

                        i++;

                }

        }

        a[i] = x;

        p = i;

        if(i - l + 1 == k) return a[i];

        if(i - l + 1 < k) return quickSelect( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ); //填空

        else return quickSelect(a, l, i - 1, k);

}

public static void main(String args[]) {

int [] a = {1, 4, 2, 8, 5, 7};

System.out.println(quickSelect(a, 0, 5, 4));

}

}

注意：只提交划线部分缺少的代码，不要抄写任何已经存在的代码或符号。

答案

a,i+1,r,k-(i-l+1)

6.递增三元组

给定三个整数数组

A = [A1, A2, ... AN],

B = [B1, B2, ... BN],

C = [C1, C2, ... CN]，

请你统计有多少个三元组(i, j, k) 满足：

1. 1 <= i, j, k <= N

2. Ai < Bj < Ck

【输入格式】

第一行包含一个整数N。

第二行包含N个整数A1, A2, ... AN。

第三行包含N个整数B1, B2, ... BN。

第四行包含N个整数C1, C2, ... CN。

对于30%的数据，1 <= N <= 100

对于60%的数据，1 <= N <= 1000

对于100%的数据，1 <= N <= 100000 0 <= Ai, Bi, Ci <= 100000

【输出格式】

一个整数表示答案

【输入样例】

3

1 1 1

2 2 2

3 3 3

【输出样例】

27

资源约定：

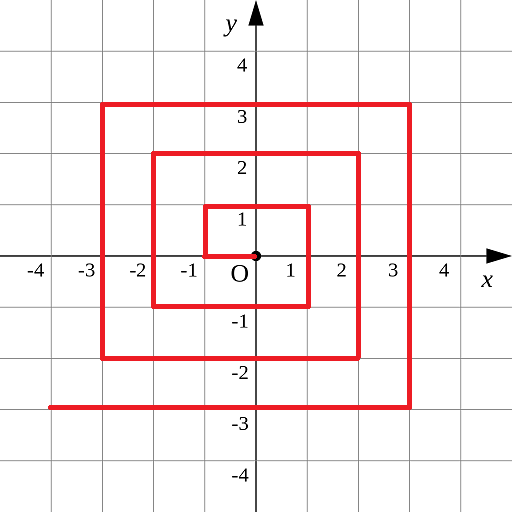
峰值内存消耗（含虚拟机） < 256M

CPU消耗  < 1000ms

注意：不要使用package语句。不要使用jdk1.7及以上版本的特性。

7.螺旋折线

螺旋折线经过平面上所有整点恰好一次。



对于整点(X, Y)，我们定义它到原点的距离dis(X, Y)是从原点到(X, Y)的螺旋折线段的长度。

例如dis(0, 1)=3, dis(-2, -1)=9

给出整点坐标(X, Y)，你能计算出dis(X, Y)吗？

【输入格式】

X和Y

对于40%的数据，-1000 <= X, Y <= 1000

对于70%的数据，-100000 <= X， Y <= 100000

对于100%的数据, -1000000000 <= X, Y <= 1000000000

【输出格式】

输出dis(X, Y)

【输入样例】

0 1

【输出样例】

3

import java.util.Scanner;

class Main

{

public static void main(String args[])

{

int a,b,p,q;

long ans=0;

Scanner sc=new Scanner(System.in);

a=sc.nextInt();

b=sc.nextInt();

if(Math.abs(a)>Math.abs(b))

{

p=a;q=1;

}

else {

p=b;q=2;

}

ans=4\*Math.abs(p)\*(Math.abs(p)-1);

if(q==1&&a<0)

ans+=b-a;

else if(q==1&&a>=0)

ans+=5\*a-b;

else if(q==2&&b<0)

ans=-7\*b-a;

else ans+=3\*b+a;

System.out.println(ans);

}

8.日志统计

小明维护着一个程序员论坛。现在他收集了一份"点赞"日志，日志共有N行。其中每一行的格式是：

ts id

表示在ts时刻编号id的帖子收到一个"赞"。

现在小明想统计有哪些帖子曾经是"热帖"。如果一个帖子曾在任意一个长度为D的时间段内收到不少于K个赞，小明就认为这个帖子曾是"热帖"。

具体来说，如果存在某个时刻T满足该帖在[T, T+D)这段时间内(注意是左闭右开区间)收到不少于K个赞，该帖就曾是"热帖"。

给定日志，请你帮助小明统计出所有曾是"热帖"的帖子编号。

【输入格式】

第一行包含三个整数N、D和K。

以下N行每行一条日志，包含两个整数ts和id。

对于50%的数据，1 <= K <= N <= 1000

对于100%的数据，1 <= K <= N <= 100000 0 <= ts <= 100000 0 <= id <= 100000

【输出格式】

按从小到大的顺序输出热帖id。每个id一行。

【输入样例】

7 10 2

0 1

0 10

10 10

10 1

9 1

100 3

100 3

【输出样例】

1

3

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

class Main

{

public static void main(String args[])

{

int n,d,k;

Scanner sc=new Scanner(System.in);

n=sc.nextInt();

d=sc.nextInt();

k=sc.nextInt();

Node arr[]=new Node[n];

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

arr[i]=new Node(sc.nextInt(),sc.nextInt());

Arrays.sort(arr);

int curid=arr[0].id;

int added=0;

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

{

if(i+k-1<n&&arr[i+k-1].id==curid&&arr[i+k-1].ts-arr[i].ts<d&&added==0)

{System.out.println(curid);added=1;}

else if(arr[i].id!=curid)

{

curid=arr[i].id;

added=0;

i=i-1;

}

}

}

}

class Node implements Comparable<Node>

{

int ts,id;

Node(int a,int b)

{

ts=a;

id=b;

}

@Override

public int compareTo(Node o) {

if(id==o.id)

return ts-o.ts;

else

return id-o.id;

}

}

9.全球变暖

你有一张某海域NxN像素的照片，"."表示海洋、"#"表示陆地，如下所示：

.......

.##....

.##....

....##.

..####.

...###.

.......

其中"上下左右"四个方向上连在一起的一片陆地组成一座岛屿。例如上图就有2座岛屿。

由于全球变暖导致了海面上升，科学家预测未来几十年，岛屿边缘一个像素的范围会被海水淹没。具体来说如果一块陆地像素与海洋相邻(上下左右四个相邻像素中有海洋)，它就会被淹没。

例如上图中的海域未来会变成如下样子：

.......

.......

.......

.......

....#..

.......

.......

请你计算：依照科学家的预测，照片中有多少岛屿会被完全淹没。

【输入格式】

第一行包含一个整数N。  (1 <= N <= 1000)

以下N行N列代表一张海域照片。

照片保证第1行、第1列、第N行、第N列的像素都是海洋。

【输出格式】

一个整数表示答案。

【输入样例】

7

.......

.##....

.##....

....##.

..####.

...###.

.......

【输出样例】

1

import java.util.Scanner;

import org.omg.PortableInterceptor.SYSTEM\_EXCEPTION;

class Main

{

static int n,map[][],vis[][];

static boolean dfs(int i,int j)

{

if(i<0||i>=n||j<0||j>=n||vis[i][j]==1) return false;

vis[i][j]=1;

boolean f=false;

if(map[i][j+1]==1&&map[i][j-1]==1&&map[i-1][j]==1&&map[i+1][j]==1)

f=true;

return dfs(i, j+1)||dfs(i, j-1)||dfs(i+1, j)||dfs(i-1, j)||f;

}

public static void main(String args[])

{

int r=0,t=0;

Scanner sc=new Scanner(System.in);

n=sc.nextInt();

map=new int[n][n];

vis=new int[n][n];

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

{

String s=sc.next();

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

if(s.charAt(j)!='.')

map[i][j]=1;

else

vis[i][j]=1;

}

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

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

if(vis[i][j]!=1)

{

r++;

t+=dfs(i, j)?1:0;

}

System.out.println(r-t);

}

}

10.堆的计数

我们知道包含N个元素的堆可以看成是一棵包含N个节点的完全二叉树。

每个节点有一个权值。对于小根堆来说，父节点的权值一定小于其子节点的权值。

假设N个节点的权值分别是1~N，你能求出一共有多少种不同的小根堆吗？

例如对于N=4有如下3种：

    1

   / \

  2   3

 /

4

    1

   / \

  3   2

 /

4

    1

   / \

  2   4

 /

3

由于数量可能超过整型范围，你只需要输出结果除以1000000009的余数。

【输入格式】

一个整数N。

对于40%的数据，1 <= N <= 1000

对于70%的数据，1 <= N <= 10000

对于100%的数据，1 <= N <= 100000

【输出格式】

一个整数表示答案。

【输入样例】

4

【输出样例】

3

**2017年**

参考：<https://blog.csdn.net/mcp3128/article/details/79583045>

1.购物单

    小明刚刚找到工作，老板人很好，只是老板夫人很爱购物。老板忙的时候经常让小明帮忙到商场代为购物。小明很厌烦，但又不好推辞。

    这不，XX大促销又来了！老板夫人开出了长长的购物单，都是有打折优惠的。

    小明也有个怪癖，不到万不得已，从不刷卡，直接现金搞定。

    现在小明很心烦，请你帮他计算一下，需要从取款机上取多少现金，才能搞定这次购物。

    取款机只能提供100元面额的纸币。小明想尽可能少取些现金，够用就行了。

    你的任务是计算出，小明最少需要取多少现金。

以下是让人头疼的购物单，为了保护隐私，物品名称被隐藏了。

-----------------

\*\*\*\*     180.90       88折

\*\*\*\*      10.25       65折

\*\*\*\*      56.14        9折

\*\*\*\*     104.65        9折

\*\*\*\*     100.30       88折

\*\*\*\*     297.15        半价

\*\*\*\*      26.75       65折

\*\*\*\*     130.62        半价

\*\*\*\*     240.28       58折

\*\*\*\*     270.62        8折

\*\*\*\*     115.87       88折

\*\*\*\*     247.34       95折

\*\*\*\*      73.21        9折

\*\*\*\*     101.00        半价

\*\*\*\*      79.54        半价

\*\*\*\*     278.44        7折

\*\*\*\*     199.26        半价

\*\*\*\*      12.97        9折

\*\*\*\*     166.30       78折

\*\*\*\*     125.50       58折

\*\*\*\*      84.98        9折

\*\*\*\*     113.35       68折

\*\*\*\*     166.57        半价

\*\*\*\*      42.56        9折

\*\*\*\*      81.90       95折

\*\*\*\*     131.78        8折

\*\*\*\*     255.89       78折

\*\*\*\*     109.17        9折

\*\*\*\*     146.69       68折

\*\*\*\*     139.33       65折

\*\*\*\*     141.16       78折

\*\*\*\*     154.74        8折

\*\*\*\*      59.42        8折

\*\*\*\*      85.44       68折

\*\*\*\*     293.70       88折

\*\*\*\*     261.79       65折

\*\*\*\*      11.30       88折

\*\*\*\*     268.27       58折

\*\*\*\*     128.29       88折

\*\*\*\*     251.03        8折

\*\*\*\*     208.39       75折

\*\*\*\*     128.88       75折

\*\*\*\*      62.06        9折

\*\*\*\*     225.87       75折

\*\*\*\*      12.89       75折

\*\*\*\*      34.28       75折

\*\*\*\*      62.16       58折

\*\*\*\*     129.12        半价

\*\*\*\*     218.37        半价

\*\*\*\*     289.69        8折

--------------------

需要说明的是，88折指的是按标价的88%计算，而8折是按80%计算，余者类推。

特别地，半价是按50%计算。

请提交小明要从取款机上提取的金额，单位是元。

答案是一个整数，类似4300的样子，结尾必然是00，不要填写任何多余的内容

【解析】：将上面出现的数字复制进eclipse，然后把\*\*\*\*改成+把文字去掉，半折改成50，在数字与打的折之间加“0.” 最后得出结果。

【程序输出结果】：5136.859500000001

【答案】：5200

【代码】：

public class Main {

public static void main(String[] args) {

double s=180.90 \*0.88+ 10.25\*0.65

+ 56.14 \*0.9

+ 104.65 \*0.9

+ 100.30\*0.88

+ 297.15 \*0.5

+ 26.75 \*0.65

+130.62 \*0.5

+ 240.28 \*0.58

+ 270.62 \*0.8

+ 115.87 \*0.88

+247.34 \*0.95

+ 73.21 \*0.9

+ 101.00 \*0.5

+ 79.54 \*0.5

+ 278.44 \*0.7

+ 199.26 \*0.5

+ 12.97 \*0.9

+ 166.30 \*0.78

+ 125.50 \*0.58

+ 84.98 \*0.9

+ 113.35 \*0.68

+ 166.57 \*0.5

+ 42.56 \*0.9

+ 81.90 \*0.95

+ 131.78 \*0.8

+ 255.89 \*0.78

+ 109.17 \*0.9

+ 146.69 \*0.68

+ 139.33 \*0.65

+ 141.16 \*0.78

+ 154.74 \*0.8

+ 59.42 \*0.8

+ 85.44 \*0.68

+ 293.70 \*0.88

+ 261.79 \*0.65

+ 11.30 \*0.88

+ 268.27 \*0.58

+ 128.29 \*0.88

+ 251.03 \*0.8

+208.39 \*0.75

+ 128.88 \*0.75

+ 62.06 \*0.9

+ 225.87 \*0.75

+ 12.89 \*0.75

+ 34.28 \*0.75

+ 62.16 \*0.58

+ 129.12 \*0.5

+ 218.37 \*0.5

+ 289.69 \*0.8;

System.out.println(s);

}

2.纸牌三角形

        A,2,3,4,5,6,7,8,9 共9张纸牌排成一个正三角形（A按1计算）。要求每个边的和相等。

        下图就是一种排法（如有对齐问题，参看p1.png）。

              A

             9 6

            4   8

           3 7 5 2

        这样的排法可能会有很多。

        如果考虑旋转、镜像后相同的算同一种，一共有多少种不同的排法呢？

        请你计算并提交该数字。

        注意：需要提交的是一个整数，不要提交任何多余内容。

【解析】：穷举即可，需要注意的是考虑旋转、镜像后相同的算同一种，顶点可以有三个位置，并且左右对称的话也是相同的一种，因此最后要除以6.

【代码】：

public class Main {

public static void main(String[] args) {

int a, b, c, d, e, f, g, h, i;

int sum = 0;

for (a = 1; a < 10; a++) {

for (b = 1; b < 10; b++) {

for (c = 1; c < 10; c++) {

for (d = 1; d < 10; d++) {

for (e = 1; e < 10; e++) {

for (f = 1; f < 10; f++) {

for (g = 1; g < 10; g++) {

for (h = 1; h < 10; h++) {

for (i = 1; i < 10; i++) {

if (a + b + d + f == a + c + e + i

&& a + b + d + f == f + g

+ h + i && a != b

&& a != c && a != d

&& a != e && a != f

&& a != g && a != h

&& a != i && b != c

&& b != d && b != e

&& b != f && b != g

&& b != h && b != i

&& c != d && c != e

&& c != f && c != g

&& c != h && c != i

&& d != e && d != f

&& d != g && d != h

&& d != i && e != f

&& e != g && e != h

&& e != i && f != g

&& f != h && f != i

&& g != h && g != i

&& h != i) {

sum++;

}

}

}

}

}

}

}

}

}

}

System.out.println(sum/3/2);

}

}

3.承压计算

X星球的高科技实验室中整齐地堆放着某批珍贵金属原料。

每块金属原料的外形、尺寸完全一致，但重量不同。

金属材料被严格地堆放成金字塔形。

                             7

                            5 8

                           7 8 8

                          9 2 7 2

                         8 1 4 9 1

                        8 1 8 8 4 1

                       7 9 6 1 4 5 4

                      5 6 5 5 6 9 5 6

                     5 5 4 7 9 3 5 5 1

                    7 5 7 9 7 4 7 3 3 1

                   4 6 4 5 5 8 8 3 2 4 3

                  1 1 3 3 1 6 6 5 5 4 4 2

                 9 9 9 2 1 9 1 9 2 9 5 7 9

                4 3 3 7 7 9 3 6 1 3 8 8 3 7

               3 6 8 1 5 3 9 5 8 3 8 1 8 3 3

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

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

            2 8 4 2 2 9 9 2 8 3 4 9 6 3 9 4 6 9

           7 9 7 4 9 7 6 6 2 8 9 4 1 8 1 7 2 1 6

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

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

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

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

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

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

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

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

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

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

X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X

其中的数字代表金属块的重量（计量单位较大）。

最下一层的X代表30台极高精度的电子秤。

假设每块原料的重量都十分精确地平均落在下方的两个金属块上，

最后，所有的金属块的重量都严格精确地平分落在最底层的电子秤上。

电子秤的计量单位很小，所以显示的数字很大。

工作人员发现，其中读数最小的电子秤的示数为：2086458231

请你推算出：读数最大的电子秤的示数为多少？

注意：需要提交的是一个整数，不要填写任何多余的内容。

【解析】：将所有的数存在一个二维数组a[i][j]中，可以先写一写具体的数的排法：

a

a a

a a a

a a a a

......

通过上式可以得出规律：

if(j!=0)

a[i][j] = a[i-1][j-1]/2.0+ a[i-1][j]/2.0+a[i][j];

else

a[i][j] = a[i-1][j]/2.0+a[i][j];

最后用题目中所给的数乘以最后一行数组中的最大值除以最小值。

【代码】：

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

double[][] a = new double[30][30];

double min = 1000000;

double max = 0;

for(int i=0;i<29;i++){

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

a[i][j]=sc.nextDouble();

}

}

for(int i=1;i<30;i++){

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

if(j!=0)

a[i][j] = a[i-1][j-1]/2.0+ a[i-1][j]/2.0+a[i][j];

else

a[i][j] = a[i-1][j]/2.0+a[i][j];

}

}

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

max = Math.max(max, a[29][i]);

min = Math.min(min, a[29][i]);

}

System.out.println(max);

System.out.println(min);

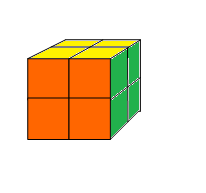
System.out.println((long)(2086458231\*max/min));

}

}

4.魔方状态

二阶魔方就是只有2层的魔方，只由8个小块组成



小明很淘气，他只喜欢3种颜色，所有把家里的二阶魔方重新涂了颜色，如下：

前面：橙色   
右面：绿色   
上面：黄色   
左面：绿色   
下面：橙色   
后面：黄色

请你计算一下，这样的魔方被打乱后，一共有多少种不同的状态。

如果两个状态经过魔方的整体旋转后，各个面的颜色都一致，则认为是同一状态。

5.取数位

 求1个整数的第k位数字有很多种方法。

 以下的方法就是一种。

 对于题目中的测试数据，应该打印5。

public class Main

{

    static int len(int x){

        if(x<10) return 1;

        return len(x/10)+1;

    }

    // 取x的第k位数字

    static int f(int x, int k){

        if(len(x)-k==0)

        return x%10;

        return \_\_\_\_\_\_\_\_\_\_;  //填空

    }

    public static void main(String[] args)

    {

        int x = 23513;

        //System.out.println(len(x));

        System.out.println(f(x,3));

    }

}

 请仔细分析源码，并补充划线部分所缺少的代码。

 注意：只提交缺失的代码，不要填写任何已有内容或说明性的文字。。

【解析】：典型的递归问题

【答案】：f(x/10,k)

【代码】：

public class Main

{

static int len(int x){

if(x<10) return 1;

return len(x/10)+1;

}

// 取x的第k位数字

static int f(int x, int k){

if(len(x)-k==0)

return x%10;

return f(x/10,k); //填空

}

public static void main(String[] args)

{

int x = 23513;

//System.out.println(len(x));

System.out.println(f(x,3));

}

}

6.最大公共子串

最大公共子串长度问题就是：

求两个串的所有子串中能够匹配上的最大长度是多少。

比如："abcdkkk" 和 "baabcdadabc"，

可以找到的最长的公共子串是"abcd",所以最大公共子串长度为4。

下面的程序是采用矩阵法进行求解的，这对串的规模不大的情况还是比较有效的解法。

public class Main

{

    static int f(String s1, String s2)

    {

        char[] c1 = s1.toCharArray();

        char[] c2 = s2.toCharArray();

        int[][] a = new int[c1.length+1][c2.length+1];

        int max = 0;

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

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

                if(c1[i-1]==c2[j-1]) {

                    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  //填空

                    if(a[i][j] > max) max = a[i][j];

                }

            }

        }

        return max;

    }

    public static void main(String[] args){

        int n = f("abcdkkk", "baabcdadabc");

        System.out.println(n);

    }

}

请分析该解法的思路，并补全划线部分缺失的代码。

【答案】： a[i][j]=a[i-1][j-1]+1

【代码】：

public class Main {

static int f(String s1, String s2) {

char[] c1 = s1.toCharArray();

char[] c2 = s2.toCharArray();

int[][] a = new int[c1.length + 1][c2.length + 1];

int max = 0;

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

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

if (c1[i - 1] == c2[j - 1]) {

a[i][j] = a[i - 1][j - 1] + 1; // 填空

if (a[i][j] > max)

max = a[i][j];

}

}

}

return max;

}

public static void main(String[] args) {

int n = f("abcdkkk", "baabcdadabc");

System.out.println(n);

}

}

7.日期问题

小明正在整理一批历史文献。这些历史文献中出现了很多日期。小明知道这些日期都在1960年1月1日至2059年12月31日。令小明头疼的是，这些日期采用的格式非常不统一，有采用年/月/日的，有采用月/日/年的，还有采用日/月/年的。更加麻烦的是，年份也都省略了前两位，使得文献上的一个日期，存在很多可能的日期与其对应。

比如02/03/04，可能是2002年03月04日、2004年02月03日或2004年03月02日。

给出一个文献上的日期，你能帮助小明判断有哪些可能的日期对其对应吗？

输入

----

一个日期，格式是"AA/BB/CC"。  (0 <= A, B, C <= 9)

输出

----

输出若干个不相同的日期，每个日期一行，格式是"yyyy-MM-dd"。多个日期按从早到晚排列。

样例输入

----

02/03/04

样例输出

----

2002-03-04

2004-02-03

2004-03-02

【解析】：在输入的年份是正确的情况下，先判断a是否大于60，若大于则一定是19\*\*年，若小于再判断a是否在12和60之间，若在，则a一定为年份不是月份，再根据b判断b是否为月份。

【代码】：

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s = sc.next();

String a = s.substring(0, 2);

String b = s.substring(3, 5);

String c = s.substring(6, 8);

if (Integer.parseInt(a) >= 60) {

System.out.println(1900 + Integer.parseInt(a) + "-" + b + "-" + c);

}

if (Integer.parseInt(a) > 12 && Integer.parseInt(a) < 60) {

if (Integer.parseInt(b) < 12) {

if (Integer.parseInt(b) == 1 || Integer.parseInt(b) == 3

|| Integer.parseInt(b) == 5 || Integer.parseInt(b) == 7

|| Integer.parseInt(b) == 8

|| Integer.parseInt(b) == 10

|| Integer.parseInt(b) == 12

&& Integer.parseInt(a) <= 31)

System.out.println(2000 + Integer.parseInt(c) + "-" + b

+ "-" + a);

if (Integer.parseInt(b) == 4 || Integer.parseInt(b) == 6

|| Integer.parseInt(b) == 9

|| Integer.parseInt(b) == 11

&& Integer.parseInt(a) < 31)

System.out.println(2000 + Integer.parseInt(c) + "-" + b

+ "-" + a);

if (Integer.parseInt(b) == 2

&& (run(Integer.parseInt(b) + 2000) && Integer

.parseInt(a) <= 29) || Integer.parseInt(a) < 28) {

System.out.println(2000 + Integer.parseInt(c) + "-" + b

+ "-" + a);

}

} else

System.out.println(2000 + Integer.parseInt(a) + "-" + b + "-"

+ c);

}

if (Integer.parseInt(a) < 13) {

if (Integer.parseInt(b) < 13) {

System.out.println(2000 + Integer.parseInt(a) + "-" + b + "-"

+ c);

System.out.println(2000 + Integer.parseInt(c) + "-" + a + "-"

+ b);

System.out.println(2000 + Integer.parseInt(c) + "-" + b + "-"

+ a);

} else {

System.out.println(2000 + Integer.parseInt(c) + "-" + a + "-"

+ b);

}

}

}

private static boolean run(int i) {

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

return true;

} else

return false;

}

}

8.包子凑数

小明几乎每天早晨都会在一家包子铺吃早餐。他发现这家包子铺有N种蒸笼，其中第i种蒸笼恰好能放Ai个包子。每种蒸笼都有非常多笼，可以认为是无限笼。

每当有顾客想买X个包子，卖包子的大叔就会迅速选出若干笼包子来，使得这若干笼中恰好一共有X个包子。比如一共有3种蒸笼，分别能放3、4和5个包子。当顾客想买11个包子时，大叔就会选2笼3个的再加1笼5个的（也可能选出1笼3个的再加2笼4个的）。

当然有时包子大叔无论如何也凑不出顾客想买的数量。比如一共有3种蒸笼，分别能放4、5和6个包子。而顾客想买7个包子时，大叔就凑不出来了。

小明想知道一共有多少种数目是包子大叔凑不出来的。

输入

----

第一行包含一个整数N。(1 <= N <= 100)

以下N行每行包含一个整数Ai。(1 <= Ai <= 100)

输出

----

一个整数代表答案。如果凑不出的数目有无限多个，输出INF。

例如，

输入：

2

4

5

程序应该输出：

6

再例如，

输入：

2

4

6

程序应该输出：

INF

样例解释：

对于样例1，凑不出的数目包括：1, 2, 3, 6, 7, 11。

对于样例2，所有奇数都凑不出来，所以有无限多个。

【解析】：

欧几里德定理：

    对于不完全为 0 的整数 a，b，gcd（a，b）表示 a，b 的最大公约数。那么一定存在整

数 x，y 使得 gcd（a，b）=ax+by。

扩展：如果有的包子种类的最大公约数不是1 那么凑不出来的情况就有无限多种。

剩下的用完全背包解决

dp[i]数组里存放着是数量为i个的包子数能不能被凑出来

【代码】：

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[101];

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

a[i] = sc.nextInt();

int yueshu = a[1];

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

yueshu = yue(yueshu, a[i]);

}

if (yueshu != 1) {

System.out.println("INF");

} else {

boolean dp[] = new boolean[10010];

dp[0] = true;

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

for (int j = 0; j + a[i] <= 10000; j++) {

if (dp[j]) {

dp[j + a[i]] = true;

}

}

}

int sum = 0;

for (int i = 0; i <= 10000; i++)

if (dp[i] == false)

sum++;

System.out.println(sum);

}

}

private static int yue(int x, int y) {

if (y == 0)

return x;

else

return yue(y, x % y);

}

}

9. 分巧克力

儿童节那天有K位小朋友到小明家做客。小明拿出了珍藏的巧克力招待小朋友们。

    小明一共有N块巧克力，其中第i块是Hi x Wi的方格组成的长方形。

    为了公平起见，小明需要从这 N 块巧克力中切出K块巧克力分给小朋友们。切出的巧克力需要满足：

    1. 形状是正方形，边长是整数

    2. 大小相同

例如一块6x5的巧克力可以切出6块2x2的巧克力或者2块3x3的巧克力。

当然小朋友们都希望得到的巧克力尽可能大，你能帮小Hi计算出最大的边长是多少么？

输入

第一行包含两个整数N和K。(1 <= N, K <= 100000)

以下N行每行包含两个整数Hi和Wi。(1 <= Hi, Wi <= 100000)

输入保证每位小朋友至少能获得一块1x1的巧克力。

输出

输出切出的正方形巧克力最大可能的边长。

样例输入：

2 10

6 5

5 6

样例输出：

2

【解析】：从1开始遍历，直到查找到不能分的最小数量，输出最小数量减一即为可以分到的最大值

【代码】：

import java.util.Scanner;

public class Main {

static int h[] = new int[100000];

static int w[] = new int[100000];

static int n, k;

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

n = sc.nextInt();

k = sc.nextInt();

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

h[i] = sc.nextInt();

w[i] = sc.nextInt();

}

int max;

for (max = 1;; max++)//每个人最少可以分1块，最大边从1开始试

if (check(max) < k) {//出现分割不够的情况，则上一次是可以分割的最大数目

System.out.println(max - 1);

break;

}

}

private static long check(int max) {//计算总共可以分为几块

long sum = 0;

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

sum += (long) (h[i] / max) \* (long) (w[i] / max);

return sum;

}

}

10. k倍区间

给定一个长度为N的数列，A1, A2, ... AN，如果其中一段连续的子序列Ai, Ai+1, ... Aj(i <= j)之和是K的倍数，我们就称这个区间[i, j]是K倍区间。

你能求出数列中总共有多少个K倍区间吗？

输入

-----

第一行包含两个整数N和K。(1 <= N, K <= 100000)

以下N行每行包含一个整数Ai。(1 <= Ai <= 100000)

输出

-----

输出一个整数，代表K倍区间的数目。

例如，

输入：

5 2

1

2

3

4

5

程序应该输出：

6

【解析】：求区间[l,r]的和是k的倍数的个数。求区间和，我们可以通过前缀和来求出。我们规定sum[i]表示第1个元素到第i个元素的和。那么sum[r] - sum[l-1]就是区间[l,r]的和。区间[l,r]的和是k的倍数即(sum[r] - sum[l-1])%k == 0 即sum[r]%k == sum[l-1]%k。

　　那么，我们求出每个前缀和，在求的过程中取模，两个相等的前缀和就能组成一个k倍区间。

       数列 1 2 3 4 5   mod = 2

　　对前1个数的和取模， 为1 之前有0个前缀和取模后为1，个数+0

　　对前2个数的和取模， 为1 之前有1个前缀和取模后为1，个数+1

　　对前3个数的和取模， 为0 之前有0个前缀和取模后为0, 个数+0

　　对前4个数的和取模， 为0 之前有1个前缀和取模后为0，个数+1

　　对钱5个数的和取模， 为1 之前有2个前缀和取模后为1，个数+2

　　到目前为止ans = 4。但是ans应该等于6，因为这样计算后，我们漏掉了前i个数的和取模是k的倍数的情况，即[0,i]区间和是k的倍数，因此，我们要在ans = 4 的基础上 加上前缀和取模后为0的个数 即ans+2 = 6;

【代码】：

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int k = sc.nextInt();

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

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

a[i] = sc.nextInt();

a[0] = a[0] % k;

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

a[i] = ((a[i] % k) + a[i - 1]) % k;

long sum = 0;

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

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

sum += b[a[i]]++;

System.out.println(sum + b[0]);

}

}

**2016年**

参考：<https://www.cnblogs.com/wyb666/p/10829163.html>

参考：<https://blog.csdn.net/LEE_FIGHTING_JINGYU/article/details/79144541>

参考：<https://www.cnblogs.com/tanshaoshenghao/p/10542723.html>

1.煤球数目

此题是一个找规律的题，规律比较好发现，第1层就是1，第二层就是1+2，第三层就是1+2+3，第n层就是1+2+3+、、、+n

【代码】:

1 /\*

2 有一堆煤球，堆成三角棱锥形。具体：

3 第一层放1个，

4 第二层3个（排列成三角形），

5 第三层6个（排列成三角形），

6 第四层10个（排列成三角形），

7 ....

8 answer: 171700

9 \*/

10 public class t1 {

11 private static int f(int n){ // 算第n层有多少个

12 int res = 0;

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

14 res += i;

15 }

16

17 return res;

18 }

19

20 private static int cal(int n){

21 int res = 0;

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

23 res += f(i);

24 }

25 return res;

26 }

27

28 public static void main(String[] args) {

29 // test

30 System.out.println(f(1));

31 System.out.println(f(2));

32 System.out.println(f(3));

33 System.out.println(f(4));

34

35 // result

36 System.out.println(cal(100)); // 171700

37 }

38 }

2. 生日蜡烛

/\*生日蜡烛:

某君从某年开始每年都举办一次生日party，并且每次都要吹熄与年龄相同根数的蜡烛。

现在算起来，他一共吹熄了236根蜡烛。

请问，他从多少岁开始过生日party的？

请填写他开始过生日party的年龄数。

注意：你提交的应该是一个整数，不要填写任何多余的内容或说明性文字。

答案: 26\*/

【代码】：

public class t2 {

public static void main(String[] args) {

for(int i=1; i<=100; i++){

int t = 0;

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

t += j;

if(t==236){

System.out.println(i + " " + j); // 26 33

// test:

System.out.println(26+27+28+29+30+31+32+33);

}

if(t>236){

break;

}

}

}

}

}

3.凑算式

        B       DEF  
A + --- + ------- = 10  
        C       GHI  
 这个算式中A~I代表1~9的数字，不同的字母代表不同的数字。  
比如：

6+8/3+952/714 就是一种解法，

5+3/1+972/486 是另一种解法。

这个算式一共有多少种解法？

有两种解法

第一种：暴力法

【代码】：

1 public class t3 {

2 private static boolean check(int i, int j, int k, int l, int a, int b,

3 int c, int d, int e) {

4 // 抽屉原理

5 int[] array = new int[10];

6 array[i] += 1;

7 array[j] += 1;

8 array[k] += 1;

9 array[l] += 1;

10 array[a] += 1;

11 array[b] += 1;

12 array[c] += 1;

13 array[d] += 1;

14 array[e] += 1;

15 for (int t = 1; t <= 9; t++) {

16 if (array[t] != 1) {

17 return false;

18 }

19 }

20

21 return true;

22 }

23

24 public static void main(String[] args) {

25 int res = 0;

26 for (int i = 1; i <= 9; i++)

27 for (int j = 1; j <= 9; j++)

28 for (int k = 1; k <= 9; k++)

29 for (int l = 1; l <= 9; l++)

30 for (int a = 1; a <= 9; a++)

31 for (int b = 1; b <= 9; b++)

32 for (int c = 1; c <= 9; c++)

33 for (int d = 1; d <= 9; d++)

34 for (int e = 1; e <= 9; e++) {

35 int A = j;

36 int B = k;

37 int C = l \* 100 + a \* 10 + b;

38 int D = c \* 100 + d \* 10 + e;

39 // System.out.println(s);

40 if (check(i, j, k, l, a, b, c, d, e)) {

41 if (i \* B \* D + A \* D + B \* C == 10

42 \* B \* D) {

43 res += 1;

44 }

45 }

46 }

47 System.out.println(res);

48 }

49 }

第二种  全排列

【代码】：

1 // 用全排列进行优化

2

3 public class t3 {

4

5 public static int res;

6

7 public static boolean check(int[] arr) {

8 int A = arr[0];

9 int B = arr[1];

10 int C = arr[2];

11 int D = arr[3] \* 100 + arr[4] \* 10 + arr[5];

12 int E = arr[6] \* 100 + arr[7] \* 10 + arr[8];

13 // A + B/C + D/E == 10

14 if(B\*E + C\*D == (10 - A) \* C \* E) {

15 return true;

16 }

17 return false;

18 }

19

20 public static void f(int[] arr, int k) {

21

22 if (k >= arr.length) {

23 if (check(arr)) {

24 res += 1;

25 }

26 return;

27 }

28

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

30 int t = arr[k];

31 arr[k] = arr[i];

32 arr[i] = t;

33 f(arr, k + 1);

34 t = arr[k];

35 arr[k] = arr[i];

36 arr[i] = t;

37 }

38

39 }

40

41 public static void main(String[] args) {

42 f(new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 }, 0);

43 System.out.println(res); // 29

44 }

45 }

【答案】：29

4.分小组

【代码】：

1 /\* 分小组

2 9名运动员参加比赛，需要分3组进行预赛。有哪些分组的方案呢？

3 我们标记运动员为 A,B,C,... I

4 下面的程序列出了所有的分组方法。

5 该程序的正常输出为：

6 ABC DEF GHI

7 ABC DEG FHI

8 ABC DEH FGI

9 ABC DEI FGH

10 ABC DFG EHI

11 ABC DFH EGI

12 ABC DFI EGH

13 ABC DGH EFI

14 ABC DGI EFH

15 ABC DHI EFG

16 ABC EFG DHI

17 ABC EFH DGI

18 ABC EFI DGH

19 ABC EGH DFI

20 ABC EGI DFH

21 ABC EHI DFG

22 ABC FGH DEI

23 ABC FGI DEH

24 ABC FHI DEG

25 ABC GHI DEF

26 ABD CEF GHI

27 ABD CEG FHI

28 ABD CEH FGI

29 ABD CEI FGH

30 ABD CFG EHI

31 ABD CFH EGI

32 ABD CFI EGH

33 ABD CGH EFI

34 ABD CGI EFH

35 ABD CHI EFG

36 ABD EFG CHI

37 ..... (以下省略，总共560行)。

38

39 \*/

40 public class t4 {

41 public static void printArray(int[] arr){

42 if(arr==null){

43 return;

44 }

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

46 System.out.print(arr[i] + " ");

47 }

48 System.out.println();

49 }

50

51 public static String remain(int[] a)

52 {

53 String s = "";

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

55 if(a[i] == 0) s += (char)(i+'A');

56 }

57 return s;

58 }

59

60 public static void f(String s, int[] a)

61 {

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

63 if(a[i]==1) continue;

64 a[i] = 1;

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

66 if(a[j]==1) continue;

67 a[j]=1;

68 for(int k=j+1; k<a.length; k++){

69 if(a[k]==1) continue;

70 a[k]=1;

71 // printArray(a); // print array

72 System.out.println(s + " " + (char)(i+'A') + (char)(j+'A') + (char)(k+'A') + " " +remain(a)); //填空位置

73 a[k]=0;

74 }

75 a[j]=0;

76 }

77 a[i] = 0;

78 }

79 }

80

81 public static void main(String[] args)

82 {

83 int[] a = new int[9];

84 a[0] = 1;

85

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

87 a[b] = 1;

88 for(int c=b+1; c<a.length; c++){

89 a[c] = 1;

90 String s = "A" + (char)(b+'A') + (char)(c+'A');

91 f(s,a);

92 a[c] = 0;

93 }

94 a[b] = 0;

95 }

96

97 }

98

99 }

[复制代码](javascript:void(0);)

5.抽签

递归

【代码】：

1 /\* 抽签

2 X星球要派出一个5人组成的观察团前往W星。

3 其中：

4 A国最多可以派出4人。

5 B国最多可以派出2人。

6 C国最多可以派出2人。

7 、、、

8 那么最终派往W星的观察团会有多少种国别的不同组合呢？

9

10 下面的程序解决了这个问题。

11 数组a[] 中是每个国家可以派出的最多的名额。

12 程序执行结果为：

13 DEFFF

14 CEFFF

15 CDFFF

16 CDEFF

17 CCFFF

18 CCEFF

19 CCDFF

20 CCDEF

21 BEFFF

22 BDFFF

23 BDEFF

24 BCFFF

25 BCEFF

26 BCDFF

27 BCDEF

28 ....

29 (以下省略，总共101行)

30

31 \*/

32

33 public class t5

34 {

35 public static int res;

36 public static void f(int[] a, int k, int n, String s)

37 {

38 // k -> 当前到了第几个国家

39 // n -> 派出的人数总数

40 // s -> 结果字符串

41 if(k==a.length){

42 // 遍历完数组a一遍

43 if(n==0){

44 // 一个解

45 System.out.println(s);

46 res += 1;

47 }

48 return;

49 }

50

51 String s2 = s;

52 for(int i=0; i<=a[k]; i++){

53 f(a, k+1, n-i, s2); //填空位置

54 s2 += (char)(k+'A');

55 }

56 }

57

58 public static void main(String[] args)

59 {

60 res = 0;

61 int[] a = {4,2,2,1,1,3};

62

63 f(a,0,5,"");

64 System.out.println(res);

65 }

66 }

[复制代码](javascript:void(0);)

6.方格填数

经典解法：全排列

【代码】：

1

2

3 /\*

4 方格填数

5 如下的10个格子

6 +--+--+--+

7 | | | |

8 +--+--+--+--+

9 | | | | |

10 +--+--+--+--+

11 | | | |

12 +--+--+--+

13 （如果显示有问题，也可以参看【图1.jpg】）

14

15 填入0~9的数字。要求：连续的两个数字不能相邻。

16 （左右、上下、对角都算相邻）

17

18 一共有多少种可能的填数方案？请填写表示方案数目的整数。

19 \*/

20 public class t6 {

21

22 public static int res = 0;

23

24 public static boolean check(int[] arr) {

25 if (Math.abs(arr[0] - arr[1]) == 1 || Math.abs(arr[0] - arr[3]) == 1 || Math.abs(arr[0] - arr[4]) == 1

26 || Math.abs(arr[0] - arr[5]) == 1) {

27 return false;

28 }

29

30 if (Math.abs(arr[1] - arr[2]) == 1 || Math.abs(arr[1] - arr[4]) == 1 || Math.abs(arr[1] - arr[5]) == 1

31 || Math.abs(arr[1] - arr[6]) == 1) {

32 return false;

33 }

34

35 if(Math.abs(arr[2] - arr[5]) == 1 ||Math.abs(arr[2] - arr[6]) == 1) {

36 return false;

37 }

38

39 if(Math.abs(arr[3] - arr[4]) == 1 ||Math.abs(arr[3] - arr[7]) == 1 ||Math.abs(arr[3] - arr[8]) == 1) {

40 return false;

41 }

42

43 if(Math.abs(arr[4] - arr[5]) == 1 || Math.abs(arr[4] - arr[7]) == 1 || Math.abs(arr[4] - arr[8]) == 1 ||

44 Math.abs(arr[4] - arr[9]) == 1) {

45 return false;

46 }

47

48 if(Math.abs(arr[5] - arr[6]) == 1 ||Math.abs(arr[5] - arr[8]) == 1 ||Math.abs(arr[5] - arr[9]) == 1) {

49 return false;

50 }

51

52 if(Math.abs(arr[6] - arr[9]) == 1) {

53 return false;

54 }

55

56 if(Math.abs(arr[7] - arr[8]) == 1) {

57 return false;

58 }

59

60 if(Math.abs(arr[8] - arr[9]) == 1) {

61 return false;

62 }

63

64 return true;

65 }

66

67 public static void f(int[] arr, int n) {

68 // 全排列经典代码

69 if (n == arr.length) {

70 if (check(arr)) {

71 res++;

72 }

73 return;

74 }

75

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

77 int temp = arr[i];

78 arr[i] = arr[n];

79 arr[n] = temp;

80 f(arr, n + 1);

81 temp = arr[i];

82 arr[i] = arr[n];

83 arr[n] = temp;

84 }

85

86 }

87

88 public static void main(String[] args) {

89 int[] arr = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

90 f(arr, 0);

91 System.out.println(res);

92 }

93

94 }

[复制代码](javascript:void(0);)

**另外一种解法：剪枝**

**【代码】：**

[复制代码](javascript:void(0);)

1 // 将方格补成五行六列

2 // 补充的默认填-10

3 // 然后递归一个一个填 不符合规则的就回溯

4 // 填到最后res++

5 public class t6\_2 {

6

7 public static int res = 0;

8 public static boolean[] vis = new boolean[10]; // 标记每个数是否已经使用

9

10 public static boolean check(int[][] arr, int i, int j) {

11 // 检查当前元素的相邻元素是否连续

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

13 for (int y = j - 1; y <= j + 1; y++) {

14 if (Math.abs(arr[x][y] - arr[i][j]) == 1) {

15 // 相邻元素是连续的

16 return false;

17 }

18 }

19 }

20

21 return true;

22 }

23

24 public static void f(int[][] arr, int x, int y) {

25 if (x == 3 && y == 4) {

26 // 之前所有都满足就是一种情况

27 res++;

28 return;

29 }

30

31 for(int i=0; i<10; i++) {

32 // 从0到9中抽一个放到当前位置

33 if(!vis[i]) { // i没有被用过

34 arr[x][y] = i;

35 if(!check(arr, x, y)) {

36 // 不满足条件 恢复并continue

37 arr[x][y] = -10;

38 continue;

39 }

40 vis[i] = true; // 标记为已经访问

41 if(y==4) {

42 f(arr, x+1, 1); // 当达每一行的边界 换行

43 }else {

44 f(arr, x, y+1); // 继续填右边的格子

45 }

46 // 回溯

47 vis[i] = false;

48 arr[x][y] = -10;

49 }

50 }

51 }

52

53 public static void main(String[] args) {

54

55 int[][] arr = { { -10, -10, -10, -10, -10, -10 }, { -10, -10, 0, 0, 0, -10 }, { -10, 0, 0, 0, 0, -10 },

56 { -10, 0, 0, 0, -10, -10 }, { -10, -10, -10, -10, -10, -10 }, };

57 f(arr, 1, 2);

58 System.out.println(res);

59

60 }

61

62 }

7.剪邮票

有12张连在一起的12生肖的邮票

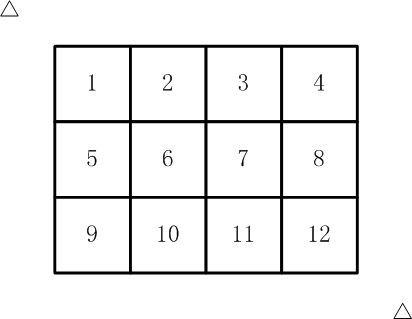
现在你要从中剪下5张来，要求必须是连着的

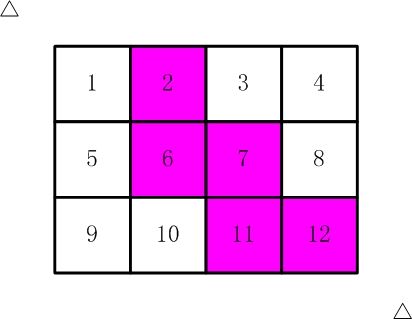
（仅仅连接一个角不算相连）

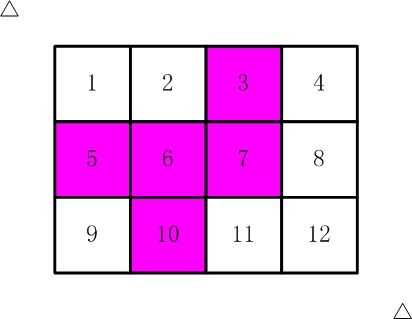
粉红色所示部分就是合格的剪取。

请你计算，一共有多少种不同的剪取方法。

请填写表示方案数目的整数。







【思路】：

首先通过5层for循环枚举选出5数, 难点在于判断这5个数是否是相连的, 可以利用图中标有编号的特性进行判断. 可以直观地看到一个数上面联通的数可-4得到, 下面的数可+4得到, 同理左边是-1, 右边是+1. 但是, 有特殊情况, 比如4+1为5, 理论上4的右边是5, 但是由于换行的原因, 4和5不是联通的. 但是这个方法是好的, 可以通过修改图中的编号继续使用这个方法.

可以把图改成:

1 2 3 4

6 7 8 9

10 11 12 13

【代码】：

public class Seven {

public static int res = 0;

public static int[] arr = {1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14};

public static int[] p = new int[5];

public static boolean[] visit = new boolean[5];

public static int[] c = {5, -5, 1, -1};

public static void main(String[] args) {

for(int a = 0; a < 12; a++){

for(int b = a + 1; b < 12; b++){

for(int c = b + 1; c < 12; c++){

for(int d = c + 1; d < 12; d++){

for(int e = d + 1; e < 12; e++){

p[0] = arr[a];

p[1] = arr[b];

p[2] = arr[c];

p[3] = arr[d];

p[4] = arr[e];

process(0);

boolean flag = true;

for(int i = 0; i < 5; i++){

if(!visit[i]){

flag = false;

}

}

if(flag){

res++;

}

cleanVisit();

}

}

}

}

}

System.out.println(res);

}

public static void process(int n){

for(int i = 0; i < 4; i++){

int t = p[n] + c[i];

if(t < 1 || t > 14 || t == 5 || t == 10){

continue;

}

for(int j = 0; j < 5; j++){

if(!visit[j] && p[j] == t){

visit[j] = true;

process(j);

}

}

}

}

public static void cleanVisit(){

for(int i = 0; i < 5; i++){

visit[i] = false;

}

}

}

【答案】116

8.四平方和

四平方和定理，又称为拉格朗日定理：

每个正整数都可以表示为至多4个正整数的平方和。

如果把0包括进去，就正好可以表示为4个数的平方和。

比如：

5 = 0^2 + 0^2 + 1^2 + 2^2

7 = 1^2 + 1^2 + 1^2 + 2^2

（^符号表示乘方的意思）

对于一个给定的正整数，可能存在多种平方和的表示法。

要求你对4个数排序：

0 <= a <= b <= c <= d

并对所有的可能表示法按 a,b,c,d 为联合主键升序排列，最后输出第一个表示法

程序输入为一个正整数N (N<5000000)

要求输出4个非负整数，按从小到大排序，中间用空格分开

例如，输入：

5

则程序应该输出：

0 0 1 2

再例如，输入：

12

则程序应该输出：

0 2 2 2

再例如，输入：

773535

则程序应该输出：

1 1 267 838

【思路】

这道题是常规的暴力枚举题, 难在如何降低时间复杂度跑过所有的用例, 如果a, b, c, d每个数都穷举以便, 按照最大的输入5000000, 每个数要枚举的次数为10^3级, 一共就是10^12. 太大了, 会超时. 为了能减少次数, 把枚举分成两半, 一半的数据是a \* a + b \* b, 这一半的数据用哈希表存储, 另外一半的数据是c \* c + d \* d = n - a \* a + b \* b. 这样能把枚举的规模缩小一半.

public class Eight {

static HashMap<Integer, Integer> map = new HashMap<Integer, Integer>();

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

for(int c = 0; c \* c <= n / 2; c++){

for(int d = c; c \* c + d \* d <= n; d++){

if(!map.containsKey(c \* c + d \* d)){

map.put(c \* c + d \* d, c);

}

}

}

for(int a = 0; a \* a <= n / 4; a++){

for(int b = a; a \* a + b \* b <= n; b++){

if(map.containsKey(n - a \* a - b \* b)){

int c = map.get(n - a \* a - b \* b);

int d = (int) Math.sqrt(n - a \* a - b \* b - c \* c);

System.out.println(a + " " + b + " " + c + " " + d);

return;

}

}

}

}}

9.取球博弈

两个人玩取球的游戏。

一共有N个球，每人轮流取球，每次可取集合{n1,n2,n3}中的任何一个数目。

如果无法继续取球，则游戏结束。

此时，持有奇数个球的一方获胜。

如果两人都是奇数，则为平局。

假设双方都采用最聪明的取法，

第一个取球的人一定能赢吗？

试编程解决这个问题。

输入格式：

第一行3个正整数n1 n2 n3，空格分开，表示每次可取的数目 (0<n1,n2,n3<100)

第二行5个正整数x1 x2 ... x5，空格分开，表示5局的初始球数(0<xi<1000)

输出格式：

一行5个字符，空格分开。分别表示每局先取球的人能否获胜。

能获胜则输出+，

次之，如有办法逼平对手，输出0，

无论如何都会输，则输出-

例如，输入：

1 2 3

1 2 3 4 5

程序应该输出：

+ 0 + 0 -

再例如，输入：

1 4 5

10 11 12 13 15

程序应该输出：

0 - 0 + +

再例如，输入：

2 3 5

7 8 9 10 11

程序应该输出：

+ 0 0 0 0

【思路】

首先要明确每一局的情况, 假设一局有x个球, 取球情况的数组为n.

在每一局中, 某一方取球都会根据取球情况数组进行试探, 如上图.

在每次递归中, 需要的参数有当前剩下的球, 我摸的球数, 对手摸的球数.

如何才能判断一局的胜负? 在所有情况中, 只要对手输过, 结果就是赢; 如果对手没输过, 但是平过, 结果就是平; 否则, 结果就是输. (因为题目条件: 双方都采取最聪明的办法)

最大的难点在于在于递归的过程中, 不是每次都是我摸, 而是我摸一次, 对手摸一次, 这里恰好就是这种类型递归的魅力所在.

具体落实到代码中

【代码】

public class QuQiuBoYi {

public static int[] n = new int[3];

public static int[] m = new int[5];

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for(int i = 0; i < 3; i++){

n[i] = sc.nextInt();

}

Arrays.sort(n);

for(int i = 0; i < 5; i++){

m[i] = sc.nextInt();

char res = process(m[i], 0, 0);//每输入一局的球数, 就计算一局

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

}

}

public static char process(int num, int me, int you){

if(num < n[0]){//没球可摸,进行结算

if((me & 1) != 0 && (you & 1) == 0){//我奇数, 对手偶数, 赢

return '+';

}else if((me & 1) == 0 && (you & 1) != 0){//我偶数, 对手奇数, 输

return '-';

}else{

return '0';//平

}

}

boolean draw = false;

for(int i = 0; i < 3; i++){

if(num >= n[i]){

//重点!假设当前这轮是我摸球, 在我摸完球me+n[i]后, 就到对手摸, me和you的位置转换, 设定中间的参数是摸球的一方

char res = process(num - n[i], you, me + n[i]);

if(res == '-'){//由于下一轮是对手摸球, 如果他输了

return '+';//我就赢了

}else if(res == '0'){//如果有平局, 要记录下来

draw = true;

}

}

}

return draw ? '0' : '-';

}

}

【优化】

作为编程大题, 如果不优化是过不了极端的数据样本的.

这题可以用数组记录重复出现的情况, 进行记忆型递归, 用空间换时间.

问题来了: me和you是不断交换的, 怎样才算是一种情况?

在base case中, 最后决定输赢的, 不是你我手上有多少个球, 而是你我的球数的奇偶情况, 所以我们把每一种情况记录为: 剩余的球数和双方球数的奇偶情况.

public class Nine {

public static int[] n = new int[3];

public static int[] m = new int[5];

public static char[][][] cache = new char[1000][2][2];

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for(int i = 0; i < 3; i++){

n[i] = sc.nextInt();

}

Arrays.sort(n);

for(int i = 0; i < 5; i++){

m[i] = sc.nextInt();

char res = process(m[i], 0, 0);

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

}

}

public static char process(int num, int me, int you){

if(num < n[0]){

if((me & 1) != 0 && (you & 1) == 0){

return '+';

}else if((me & 1) == 0 && (you & 1) != 0){

return '-';

}else{

return '0';//平

}

}

if(cache[num][me][you] != '\0'){

return cache[num][me][you];

}

boolean draw = false;

for(int i = 0; i < 3; i++){

if(num >= n[i]){

char res = process(num - n[i], you, ((me + n[i]) & 1) == 0 ? 0 : 1);//只记录奇偶情况

if(res == '-'){

cache[num][me][you] = '+';

return '+';

}else if(res == '0'){

draw = true;

}

}

}

if(draw){

cache[num][me][you] = '0';

return '0';

}else{

cache[num][me][you] = '-';

return '-';

}

}

}

10.压缩变换

小明最近在研究压缩算法。

他知道，压缩的时候如果能够使得数值很小，就能通过熵编码得到较高的压缩比。

然而，要使数值很小是一个挑战。

最近，小明需要压缩一些正整数的序列，这些序列的特点是，后面出现的数字很大可能是刚出现过不久的数字。对于这种特殊的序列，小明准备对序列做一个变换来减小数字的值。

变换的过程如下：

从左到右枚举序列，每枚举到一个数字，如果这个数字没有出现过，刚将数字变换成它的相反数，如果数字出现过，则看它在原序列中最后的一次出现后面（且在当前数前面）出现了几种数字，用这个种类数替换原来的数字。

比如，序列(a1, a2, a3, a4, a5)=(1, 2, 2, 1, 2)在变换过程为：

a1: 1未出现过，所以a1变为-1；

a2: 2未出现过，所以a2变为-2；

a3: 2出现过，最后一次为原序列的a2，在a2后、a3前有0种数字，所以a3变为0；

a4: 1出现过，最后一次为原序列的a1，在a1后、a4前有1种数字，所以a4变为1；

a5: 2出现过，最后一次为原序列的a3，在a3后、a5前有1种数字，所以a5变为1。

现在，给出原序列，请问，按这种变换规则变换后的序列是什么。

输入格式：

输入第一行包含一个整数n，表示序列的长度。

第二行包含n个正整数，表示输入序列。

输出格式：

输出一行，包含n个数，表示变换后的序列。

例如，输入：

5

1 2 2 1 2

程序应该输出：

-1 -2 0 1 1

再例如，输入：

12

1 1 2 3 2 3 1 2 2 2 3 1

程序应该输出：

-1 0 -2 -3 1 1 2 2 0 0 2 2

数据规模与约定

对于30%的数据，n<=1000；

对于50%的数据，n<=30000；

对于100%的数据，1 <=n<=100000，1<=ai<=10^9

【思路】

按照题目的思路, 使用HashMap记录数的下标, 使用HashSet记录不同种的的数字, 时间复杂度O(N^2), 暴力做法能得30分.

【代码】

public class Ten {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

long[] arr = new long[n];

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

arr[i] = sc.nextInt();

}

long[] former = Arrays.copyOf(arr, arr.length);

HashMap<Long, Integer> map = new HashMap<Long, Integer>();

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

if(!map.containsKey(arr[i])){

map.put(arr[i], i);

arr[i] = -arr[i];

}else{

int types = getTypes(map.get(arr[i]), i, former);

map.put(arr[i], i);

arr[i] = types;

}

}

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

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

}

}

public static int getTypes(int begin, int last, long[] former){

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

int index = begin + 1;

while(index < last){

set.add(former[index++]);

}

return set.size();

}

}