1，有一个城市需要修建，给你 N 个民居的坐标 X,Y，问把这么多民居全都包进城市的话，城市所需最小面积是多少（注意，城市为平行于坐标轴的正方形）

```
import java.util.Arrays;
import java.util.Scanner;

/**
 * @Author：likui
 * @PacakgeName:城市所需最小面积
 * @Description:
 * @Date:Created in 20:37 2019/8/9
 */
public class Main {
    public static long Way(long x[],long y[]){
        Arrays.sort(x);
        Arrays.sort(y);
        if (x.length<2||y.length<2)
            return 0;
        long x_length=Math.abs(x[x.length-1]-x[0]);
        long y_length=Math.abs(y[y.length-1]-y[0]);
        return Math.max(x_length,y_length);
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        long X[]=new long[n];
        long Y[]=new long[n];
        for (int i = 0; i <n ; i++) {
            X[i]=sc.nextLong();
            Y[i]=sc.nextLong();
        }
        long res=Way(X,Y);
        System.out.println(res*res);
    }
}
```

2，圈地运动，就是用很多木棍摆在地上组成一个面积大于 0 的多边形～

小明喜欢圈地运动，于是他需要去小红店里面买一些木棍，期望圈出一块地来。小红想挑战一下小明，所以给小明设置了一些障碍。障碍分别是：

1.如果小明要买第 i 块木棍的话，他就必须把前 i-1 块木棍都买下来。

2.买了的木棍都必须用在圈地运动中。

那么请问小明最少买多少根木棍，才能使得木棍围成的图形是个面积大于 0 多边形呢？

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
```

```
public class  {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // 木棍的个数
        int n = scanner.nextInt();
        if(n < 3){
            System.out.println(-1);
        }
        List<Integer> lengths = new ArrayList<>();
        for (int i=0;i<n;i++){
            int length = scanner.nextInt();
            lengths.add(length);
        }

        for(int i=2;i<lengths.size();i++){
            int sum = 0;
            int max = 0;
            for(int j=0;j<=i;j++){
                if(lengths.get(j) > max){
                    sum = sum + max;
                    max = lengths.get(j);
                }else{
                    sum = sum + lengths.get(j);
                }

            }
            if(max < sum){
                System.out.println(i+1);
                break;
            }else if(i== lengths.size()-1){
                System.out.println(-1);
            }
        }


    }


}
```

2，现在有 q 个询问，每次询问想问你在[l,r]区间内，k 进制表示中，k-1 的数量最多的数是哪个数。比如当 k=2 时，9 的二进制就是 1001，那么他就有 2 个 1.

```
import java.util.ArrayList;
```

```
import java.util.Scanner;
public class Main {

    public static long minNum(int k, long start, long end) {
        ArrayList<Integer> list = new ArrayList<Integer>();
        long tmp = start;
        while (tmp != 0) {
            long rest = tmp % k;
            list.add((int) rest);
            tmp = tmp / k;
        }
        long sum = 1;
        for (int i = 0; i < list.size(); i++) {
            long num = list.get(i);
            num = k - 1 - num;
            long size = (long) (num * sum);
            if (start + size <= end) {
                start = start + size;
            } else {
                return start;
            }
            sum = sum * k;
        }
        while (start < end) {
            long size = (long) ((k - 1) * sum);
            if (start + size <= end) {
                start = start + size;
            } else {
                return start;
            }
            sum = sum * k;
        }
        return start;
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (in.hasNextInt()) {
            int time = in.nextInt();
            for (int i = 0; i < time; i++) {
                int k = in.nextInt();
                long start = in.nextLong();
                long end = in.nextLong();
                System.out.println(minNum(k, start, end));
```

```
            }
        }
    }
}
```

3，小明有一个花园，花园里面一共有 m 朵花，对于每一朵花，都是不一样的，小明用 1～m 中的一个整数表示每一朵花。

他很喜欢去看这些花，有一天他看了 n 次，并将 n 次他看花的种类是什么按照时间顺序记录下来。

记录用 a[i]表示，表示第 i 次他看了 a[i]这朵花。

小红很好奇，她有 Q 个问题,问[l,r]的时间内，小明一共看了多少朵不同的花儿，小明因为在忙着欣赏他的花儿，所以想请你帮他回答这些问题。

```
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int m = sc.nextInt();
        int []A = new int [n];
        int []B = new int [m];
        for(int i=0;i<n;i++){
            A[i] = sc.nextInt();
        }
        int Q = sc.nextInt();
        int [][]a = new int [Q][2];
        for(int i=0;i<Q;i++){
            a[i][0] = sc.nextInt();
            a[i][1] = sc.nextInt();
        }


        for(int i=0;i<Q;i++){
            int t=0;
            for(int j=a[i][0]-1;j<=a[i][1]-1;j++){
                int dd = A[j];
                if(B[dd-1]!=i+1){
                    B[dd-1]=i+1;
                    t++;
                }

            }
            System.out.println(t);

            }
```

```
        }
    }
```

4，小红有两个长度为 n 的排列 A 和 B。每个排列由[1,n]数组成，且里面的数字都是不同的。现在要找到一个新的序列 C，要求这个新序列中任意两个位置(i,j)满足:
如果在 A 数组中 C[i]这个数在 C[j]的后面,那么在 B 数组中需要 C[i]这个数在 C[j]的前面。请问 C 序列的长度最长为多少呢？

```
import java.util.*;

public class Main
{
    public static void main(String []args)
    {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        if(n<=0)
        {
            System.out.println( 0 );
            return;
        }

        int[] an = new int[n];
        int[] bn = new int[n];


        for(  int i = 0 ; i < n ; i++  )
        {
            an[i] = sc.nextInt();
        }
        for(  int i = n-1 ; i >-1 ; i--  )
        {
            bn[i] = sc.nextInt();
        }

        System.out.println(md( an , bn ,n));
    }

    public static int md(  int[] an , int[] bn , int n  )
    {
        if(n==1)return an[0]==bn[0]?1:0;


        int[] res = new int[n];
```

```
        for( int i = 0 ; i < n ; i++ )
            for(int j = 0 ; j<n ; j++)
            {
                if( i == 0 )
                {
                    if( j==0 )
                    {
                        res[0] = an[0]==bn[0]?1:0;
                    }else{
                        res[j] = an[i]==bn[j]?1:res[j-1];
                    }

                }else if(j == 0 )
                {
                    res[j] =  res[j]==1?1:( an[i]==bn[j]?1:0  );
                }else{
                    if( an[i] == bn[j] )
                    {
                        res[j] = res[j-1]+1;
                    }else{
                        res[j] = Math.max( res[j] ,res[j-1] );
                    }
                }
            }
        return res[n-1];
    }
}
```

5，众所周知，集合{1 2 3 … N}有N!种不同的排列，假设第 i 个排列为Pi 且Pi,j是该排列的第 j 个数。将N 个点放置在 x 轴上，第 i 个点的坐标为xi 且所有点的坐标两两不同。对于每个排列（以Pi 为例），可以将其视为对上述N 个点的一种遍历顺序，即从第Pi,1个点出发，沿直线距离到达第Pi,2个点，再沿直线距离到达第Pi,3个点，以此类推，最后到达第Pi,N个点，将该路线的总长度定义为L(Pi)，那么所有N!种路线的总长度之和是多少，即L(P1)+L(P2)+L(P3)+...+L(PN!)的结果是多少？

```
import java.util.Scanner;

public class Main {

    static long mod=1000000007;
    //时间复杂度o(n^2)(这个时间复杂度不能通过测试 附上为了方便大家理解思路)
    public static int result(int arr[]) {

        long sum=0;
```

```
        long n=factorial(arr.length-1);
        for (int i = 0; i < arr.length; i++) {
            for (int j = i+1; j < arr.length; j++) {
                int tmp=(arr[j]-arr[i])<<1;
                sum=(sum+tmp)%mod;
            }
        }
        sum=(sum*n)%mod;
        return (int)sum;
    }

    public static int factorial(int n) {
        long result=1;
        for (int i = 1; i <=n; i++) {
            result=(result*i)%mod;
        }
        return (int)result;
    }
    //时间复杂度(o(n))
    public static int SumOfDistence(int arr[]) {
        int n=factorial(arr.length-1);
        long sum=0;
        long tmp=0;
        long a=0;
        for (int i = 1; i <arr.length; i++) {
            tmp=(tmp+arr[i-1])%mod;
            a=arr[i];
            a=a*i;
            //这里千万不能用 sum=(sum+a[i]*i-tmp)%mod;
            //原因就是计算顺序中会先算 a[i]*i，这两部分的计算结果是 int，会导
致溢出。所以只能增加一个 long 变量来进行计算
            sum=(sum+a-tmp)%mod;
        }

        sum=(sum<<1)%mod;
        sum=(sum*n)%mod;
        return (int) sum;
    }


    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);
        while (in.hasNextInt()) {
```

```
            int n = in.nextInt();
            int arr[] = new int[n];
            for (int i = 0; i < arr.length; i++) {
                arr[i] = in.nextInt();
            }
            System.out.println(SumOfDistence(arr));
        }
    }
}
```