1：区间问题：

例题：

给定 N 个闭区间 $[a_i, b_i]$，请你在数轴上选择尽量少的点，使得每个区间内至少包含一个选出的点。

输出选择的点的最小数量。

位于区间端点上的点也算作区间内。

输入格式

第一行包含整数 N，表示区间数。

接下来 N 行，每行包含两个整数 a_i, b_i，表示一个区间的两个端点。

输出格式

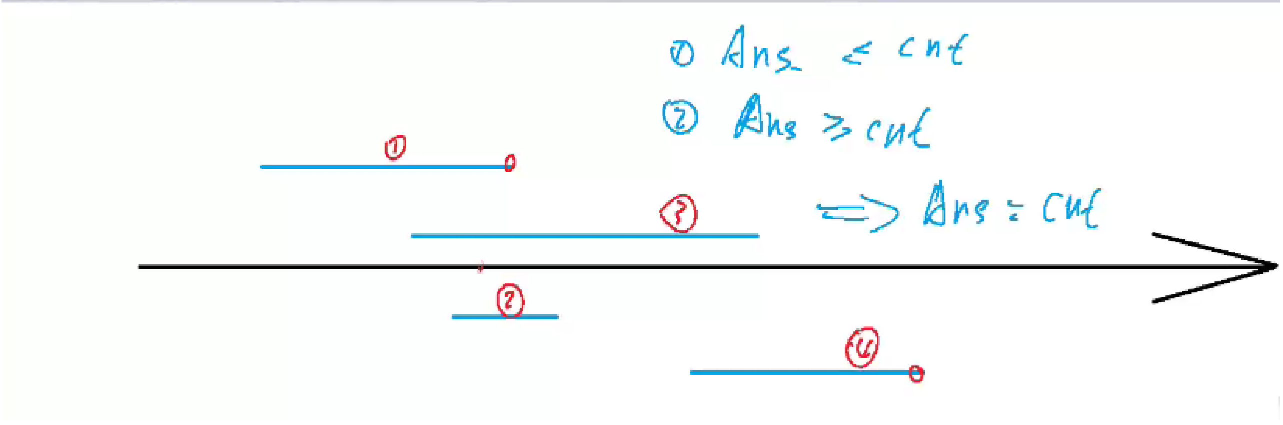
输出一个整数，表示所需的点的最小数量。

输入样例：

```
3
-1 1
2 4
3 5
```

输出样例：

```
2
```



```
思路：
1：将每个区间按右端点从小到大排序
2：从前往后依次枚举每个区间
如果当前区间中已经包含该点，则直接pass，否则，选择当前区间的右端点
```

```java
import java.util.*;
public class Main{
    public static int N = 100010;
    public static int n;
    public static int INF = 0x3f3f3f3f;
    public static Range[] range = new Range[N];
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        for(int i = 0; i < n; i++){
            int l = sc.nextInt();
            int r = sc.nextInt();
            range[i] = new Range(l, r);
        }
        Arrays.sort(range, 0, n);
        int res = 0;
        int ed = -INF;
        for(int i = 0; i < n; i++){
            if(range[i].l > ed){
                res ++;
                ed = range[i].r;
            }
        }
        System.out.println(res);
    }
}
class Range implements Comparable<Range>{
    int l, r;
    public Range(int l, int r){
        this.l = l;
        this.r = r;
    }
    public int compareTo(Range o){
        return Integer.compare(r, o.r);
    }
}
```

例题：

```
思路：
1:将所有区间按左端点从小到大排序
2：从前往后处理每个区间：
```

```
判断能否将其放到某个现有的组中 l[i] > Max_r
3: 如果存在这样的组，将其放进去，并更新当前组的Max_r，如果不存在这样的组，则开新组，然后再将其放进去
```

```java
import java.util.*;
public class Main{
    public static int N = 100010;
    public static int n;
    public static Range[] range = new Range[N];
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        for(int i = 0; i < n; i++){
            int l = sc.nextInt();
            int r = sc.nextInt();
            range[i] = new Range(l, r);
        }
        Arrays.sort(range, 0, n);
        Queue<Integer> minheap = new PriorityQueue<>();
        for(int i = 0; i < n; i++){
            if(minheap.isEmpty() || minheap.peek() >= range[i].l){
                minheap.add(range[i].r);
            }else{
                minheap.poll();
                minheap.add(range[i].r);
            }
        }
        System.out.println(minheap.size());
    }
}
class Range implements Comparable<Range>{
    int l, r;
    public Range(int l, int r){
        this.l = l;
        this.r = r;
    }
    public int compareTo(Range o){
        return Integer.compare(l, o.l);
    }
}
```

2：Huffman 树：

例题：

```java
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int N = 100010;
        int n = sc.nextInt();
        Queue<Integer> minheap = new PriorityQueue<>();
        for(int i = 0; i < n; i++){
            int x = sc.nextInt();
            minheap.add(x);
        }
        int res = 0;
        for(int i = 0; i < n; i++){
            if(minheap.size() > 1){
                int a = minheap.poll();
                int b = minheap.poll();
                res += a + b;
                minheap.add(a + b);
            }
        }
        System.out.println(res);
    }
}
```

3：排序不等式：

例题：

```java
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int N = 100010;
        int[] w = new int[N];
        int n = sc.nextInt();
        for(int i = 0; i < n; i++){
            w[i] = sc.nextInt();
        }
        Arrays.sort(w, 0, n);
        long res = 0;
        for(int i = 0; i < n; i++){
            res += w[i] * (n - 1 - i);
        }
        System.out.println(res);
    }
}
```

4：绝对值不等式：

| x - a | + | x - b | >= | a - b |

例题：

```java
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int N = 100010;
        int[] a = new int[N];
        int n = sc.nextInt();
        for(int i = 0; i < n; i++){
            a[i] = sc.nextInt();
        }
        Arrays.sort(a, 0, n);
        int sum = 0;
        for(int i = 0; i < n; i++){
            sum += Math.abs(a[i] - a[n / 2]);
        }
        System.out.println(sum);
    }
}
```