第一题签到题，充电桩 charger 类的 distance 属性表示横纵距离之和，sort 后取前 k 个输出。

第二题树，力扣 监控二叉树

后序遍历，左右中

空节点默认有覆盖，return2

左右都覆盖，父节点一定没覆盖，return0

左右至少一个无覆盖，放摄像头 res++，return1

左右至少一个摄像头，父节点覆盖，return2

最后看看 root 返回，如果是 0，放摄像头 res++

第三题动态规划

从左上角构建一遍 dp 数组：左和上的较小值+当前值

从右下角构建一遍 dp 数组

不处理不可达的话过了 15%，处理了后 95%

华为 1

```
// 100 分 95%
/*

public class Main {
    static class Charger {
        int id;
        int x, y;
        int distance;
        public Charger(int id, int x, int y, int carX, int carY) {
            this.id = id;
            this.x = x;
            this.y = y;
            this.distance = Math.abs(x-carX) + Math.abs(y-carY);
        }
    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        int k = sc.nextInt();
        int n = sc.nextInt();
        if (k == 0 || k > n) {
            System.out.println("null");
            return;
        }
        int carX = sc.nextInt();
        int carY = sc.nextInt();
        List<Charger> chargers = new ArrayList<>();

        for (int i = 1; i <= n; i++) {
            int x = sc.nextInt();
            int y = sc.nextInt();
            chargers.add(new Charger(i, x, y, carX, carY));
        }

        chargers.sort((a,b) -> {
           if (a.distance != b.distance)
               return Integer.compare(a.distance, b.distance);
           return Integer.compare(a.id, b.id);
        });
```

```
        for (int i = 0; i < k; i++) {
            Charger c = chargers.get(i);
            System.out.print(c.id +" "+ c.x +" "+ c.y +" "+ c.distance +
"\n");
            // System.out.println();
        }

        sc.close();
    }
    }
```

华为 2

```java
// 200 分 100%

public class Main {
    public static class TreeNode {
        int val;
        TreeNode left, right;
        TreeNode() {}
        TreeNode(int val) {
            this.val = val;
        }
        TreeNode(int val, TreeNode left, TreeNode right) {
            this.val = val;
            this.left = left;
            this.right = right;
        }
    }

    static int res = 0;

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        String line = sc.nextLine().trim();
        if (line.isEmpty())
            return;
        String[] nodes = line.split("\\s+");
        if (nodes[0].equals("N"))
            return;
        // for (String node : nodes) {
        //     System.out.println(node);
        // }

        TreeNode root = new TreeNode(Integer.parseInt(nodes[0]));
        Queue<TreeNode> queue = new LinkedList<>();
        queue.offer(root);

        int i = 1;
        while (!queue.isEmpty() && i < nodes.length) {
            TreeNode cur = queue.poll();
            if(!nodes[i].equals("N")) {
```

```
                TreeNode leftNode = new
TreeNode(Integer.parseInt(nodes[i]));
                cur.left = leftNode;
                queue.offer(leftNode);
                // System.out.println(leftNode.val);
            }
            i++;
            if(i >= nodes.length)
                break;
            if(!nodes[i].equals("N")) {
                TreeNode rightNode = new
TreeNode(Integer.parseInt(nodes[i]));
                cur.right = rightNode;
                queue.offer(rightNode);
            }
            i++;
        }


        // inorder(root);
        // root
        if (process(root) == 0) {    // 头节点没有覆盖，需要摄像头，res++
            res++;
        }
        System.out.println(res);

    }


    public static void inorder(TreeNode node) {
        if(node == null)    return;
        inorder(node.left);
        System.out.println(node.val+" ");
        inorder(node.right);
    }

    // 0 左右孩子都覆盖，父节点是无覆盖
    // 1 左右孩子至少一个没覆盖，父节点要放摄像头，res++
    // 2 左右孩子至少有一个摄像头，父节点覆盖
    // 3 头节点没有覆盖，需要摄像头，res++

    public static int process(TreeNode root) {
        if (root == null) { // 空节点默认覆盖
            return 2;
```

```
        }
        int left = process(root.left);
        int right = process(root.right);

        if (left == 2 && right == 2) {    // 左右孩子都覆盖
            return 0;
        } else if (left == 0 || right == 0) {    // 左右孩子至少一个没覆盖
            res++;
            return 1;
        } else {    // 左右孩子至少有一个摄像头
            return 2;
        }
    }
    }
```

华为 3

```java
// 300 分 95%

public class Main {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        int[][] grid = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                grid[i][j] = sc.nextInt();
            }
        }

        int[][] a = new int[n][n];
        a[0][0] = grid[0][0];
        for (int i = 1; i < n; i++) {
            if (a[i-1][0]!=Integer.MAX_VALUE && grid[i][0] != 0)
                a[i][0] = a[i-1][0] + grid[i][0];
            else
                a[i][0] = Integer.MAX_VALUE;
        }
        for (int j = 1; j < n; j++) {
            if (a[0][j-1]!=Integer.MAX_VALUE && grid[0][j] != 0)
                a[0][j] = a[0][j-1] + grid[0][j];
            else
                a[0][j] = Integer.MAX_VALUE;
        }
        for (int i = 1; i < n; i++) {
            for (int j = 1; j < n; j++) {
                if (Math.min(a[i-1][j], a[i][j-1])!=Integer.MAX_VALUE &&
grid[i][j] != 0)
                    a[i][j] = Math.min(a[i-1][j], a[i][j-1]) +
grid[i][j];
                else
                    a[i][j] = Integer.MAX_VALUE;
            }
        }
```

```
        int[][] b = new int[n][n];
        b[n-1][n-1] = grid[n-1][n-1];
        for (int i = n-2; i >= 0; i--) {
            if (b[i+1][n-1]!=Integer.MAX_VALUE && grid[i][n-1] != 0)
                b[i][n-1] = b[i+1][n-1] + grid[i][n-1];
            else
                b[i][n-1] = Integer.MAX_VALUE;
        }
        for (int j = n-2; j >= 0; j--) {
            if (b[n-1][j+1]!=Integer.MAX_VALUE && grid[n-1][j] != 0)
                b[n-1][j] = b[n-1][j+1] + grid[n-1][j];
            else
                b[n-1][j] = Integer.MAX_VALUE;
        }
        for (int i = n-2; i >= 0; i--) {
            for (int j = n-2; j >= 0; j--) {
                if (Math.min(b[i+1][j], b[i][j+1])!=Integer.MAX_VALUE &&
grid[i][j] != 0)
                    b[i][j] = Math.min(b[i+1][j], b[i][j+1]) +
grid[i][j];
                else
                    b[i][j] = Integer.MAX_VALUE;
            }
        }

        int res = Integer.MAX_VALUE;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (j+1 < n && grid[i][j]!=0 && grid[i][j+1]!=0) {
                    res = Math.min(res, Math.max(a[i][j], b[i][j+1]));
                }
                if (i+1 < n && grid[i][j]!=0 && grid[i+1][j]!=0) {
                    res = Math.min(res, Math.max(a[i][j], b[i+1][j]));
                }
            }
        }
        if (res != Integer.MAX_VALUE)
            System.out.println(res);
        else
            System.out.println(-1);
    }
    }
```

// 双线程交替打印 1-100（奇偶分开）

```
public class Main {
    private static int number = 1;
    private static final int MAX = 100;
    private static final Object lock = new Object();

    public static void main(String[] args) {
        Thread oddThread = new Thread(() -> {
            while (true) {
                synchronized (lock) {
                    if (number > MAX) {
                        lock.notifyAll(); // 防止另一个线程死锁
                        break;
                    }
                    if (number % 2 == 1) {
                        System.out.println("奇数线程打印: " + number);
                        number++;
                        lock.notifyAll();
                    } else {
                        try {
                            lock.wait();
                        } catch (InterruptedException e) {
                            Thread.currentThread().interrupt();
                        }
                    }
                }
            }
        });

        Thread evenThread = new Thread(() -> {
            while (true) {
                synchronized (lock) {
                    if (number > MAX) {
                        lock.notifyAll(); // 防止另一个线程死锁
                        break;
                    }
                    if (number % 2 == 0) {
                        System.out.println("偶数线程打印: " + number);
                        number++;
                        lock.notifyAll();
```

```
                    } else {
                        try {
                            lock.wait();
                        } catch (InterruptedException e) {
                            Thread.currentThread().interrupt();
                        }
                    }
                }
            }
        });

        oddThread.start();
        evenThread.start();
    }
}
```

Comparator

```java
// 数组拼凑最大数的字符串
/*
    输入：nums = [3, 10]
    输出："310"

    输入：nums = [5, 9, 30, 3, 34]
    输出："9534330"
*/

import java.util.Arrays;
import java.util.Comparator;

public class MaxNumberString {
    public static String largestNumber(int[] nums) {
        // 将 int 数组转换成 String 数组
        String[] strNums = new String[nums.length];
        for (int i = 0; i < nums.length; i++) {
            strNums[i] = String.valueOf(nums[i]);
        }

        // 自定义排序：比较两个字符串拼接后哪个更大
        Arrays.sort(strNums, new Comparator<String>() {
            @Override
            public int compare(String a, String b) {
                String order1 = a + b;
                String order2 = b + a;
                return order2.compareTo(order1); // 降序排列
            }
        });

        // 如果排序后第一个是 "0"，说明所有数都是 0
        if (strNums[0].equals("0")) {
            return "0";
        }

        // 拼接结果
        StringBuilder result = new StringBuilder();
        for (String str : strNums) {
            result.append(str);
        }
```

```
        return result.toString();
    }

    public static void main(String[] args) {
        int[] nums1 = {3, 10};
        System.out.println(largestNumber(nums1)); // 输出 "310"

        int[] nums2 = {5, 9, 30, 3, 34};
        System.out.println(largestNumber(nums2)); // 输出 "9534330"
    }
}
```