Java-猜密码-小杨申请了一个保密柜

题目描述：

小杨申请了一个保密柜，但是他忘记了密码。只记得密码都是数字，而且所有数字都是不重复的。请你根据他记住的数字范围和密码的最小数字数量，帮他算下有哪些可能的组合，规则如下：

1、输出的组合都是从可选的数字范围中选取的，且不能重复；

2、输出的密码数字要按照从小到大的顺序排列，密码组合需要按照字母顺序，从小到大的顺序排序。

3、输出的每一个组合的数字的数量要大于等于密码最小数字数量；

4、如果可能的组合为空，则返回“None”

输入描述：

1、输入的第一行是可能的密码数字列表，数字间以半角逗号分隔

2、输入的第二行是密码最小数字数量

输出描述：

可能的密码组合，每种组合显示成一行，每个组合内部的数字以半角逗号分隔，从小到大的顺序排列。

输出的组合间需要按照字典序排序。

比如：

2,3,4 放到 2,4 的前面

补充说明：

字典序是指按照单词出现在字典的顺序进行排序的方法，比如：

a 排在 b 前

a 排在 ab 前

ab 排在 ac 前

ac 排在 aca 前

示例1
输入：2,3,4
2
输出：2,3
2,3,4
2,4
3,4
说明：最小密码数量是两个，可能有三种组合：
2,3
2,4
3,4
三个密码有一种：
2,3,4

示例2
输入：2,0
1
输出：0
0,2
2
说明：可能的密码组合，一个的有两种：
0
2
两个的有一个：
0,2

```
import java.util.*;
import java.util.stream.Collectors;

public class Main {
    public static void main(String[] args) {
        // 获取输入
        Scanner sc = new Scanner(System.in);
        String digitalString = sc.nextLine();
        List<Integer> digitalList = Arrays.stream(digitalString.split(",")).map(

Integer::parseInt).distinct().sorted(Integer::compareTo).collect(
                                                    Collectors.toList());
        int minSize = sc.nextInt();
        // 可选数字数量小于最小长度，直接返回 false
        if (digitalList.size() < minSize) {
            System.out.println("None");
            return;
        }
        // 可选数字数量等于最小长度，直接返回全部
```

```
        if (digitalList.size() == minSize) {
            System.out.println(digitalList.stream().map(String::valueOf).collect(
                                        Collectors.joining(",")));
            return;
        }
        // 广度优先搜索
        ArrayList<String> res = new ArrayList<>();
        Queue<PwdInfo> queue = new LinkedList<>();
        // 初始化一维元素
        for (int i = 0; i < digitalList.size(); i++) {
            PwdInfo pwdInfo = new PwdInfo(String.valueOf(digitalList.get(i)), 1, i);
            queue.add(pwdInfo);
        }
        // 遍历数组
        while (queue.size() > 0) {
            PwdInfo pwdInfo = queue.poll();
            // 如果密码长度大于等于最小长度，加入结果集
            if (pwdInfo.size >= minSize) {
                res.add(pwdInfo.pwd);
            }
            // 获取密码当前下标(用于不重复和判断结束)
            int index = pwdInfo.getIndex();
            // 下标是否是最后一位
            if (index == digitalList.size() - 1) {
                continue;
            }
            // 否则，遍历全部，新密码组装，大小+1，下标+1
            for (int i = index + 1; i < digitalList.size(); i++) {
                PwdInfo nextPwd = new PwdInfo(pwdInfo.getPwd() + "," + digitalList.get(i),
                                                pwdInfo.getSize() + 1, i);
                queue.add(nextPwd);
            }
        }
        // 对结果进行排序，分行打印
        res.sort(String::compareTo);
        for (String result : res) {
            System.out.println(result);
        }
    }

    public static class PwdInfo {
        private String pwd;

        private int size;
```

```
        private int index;

        public PwdInfo(String pwd, int size, int index) {
            this.pwd = pwd;
            this.size = size;
            this.index = index;
        }

        public String getPwd() {
            return pwd;
        }

        public void setPwd(String pwd) {
            this.pwd = pwd;
        }

        public int getSize() {
            return size;
        }

        public void setSize(int size) {
            this.size = size;
        }

        public int getIndex() {
            return index;
        }

        public void setIndex(int index) {
            this.index = index;
        }
    }
}
```