求最大数字

知识点 单调栈

时间限制：1s 空间限制：256MB 限定语言：不限

题目描述：

给定一个由纯数字组成以字符串表示的数值，现要求字符串中的每个数字最多只能出现2次，超过的需要进行删除；删除某个重复的数字后，其它数字相对位置保持不变。

如"34533"，数字3重复超过2次，需要删除其中一个3，删除第一个3后获得最大数值"4533"

请返回经过删除操作后的最大的数值，以字符串表示。

输入描述：

第一行为一个纯数字组成的字符串，长度范围：[1,100000]

输出描述：

输出经过删除操作后的最大的数值

示例1

输入：

34533

输出：

4533

示例2

输入：

5445795045

输出：

5479504

解题思路：

1. 根据题意，所有数字不超过2个，对输入的字符串进行去重处理，所有数字最多保留2个
2. 对步骤1处理完的数组进行全排列，同时需要判断排列后的字符串是否为原始字符串的子串，最后求出其中最大值

```
public class Main{

    public static List<Integer> resList = new ArrayList<>();
    public static String[] inputStrs;

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        inputStrs = sc.nextLine().split("");

        Map<String, Integer> map = new HashMap<>();

        for(int i=0; i<inputStrs.length; i++){    //求出所有数字的出现次数
            map.put( inputStrs[i], map.getOrDefault(inputStrs[i], 0) + 1);
        }

        List<String> list = new ArrayList<>();
        for(Map.Entry<String,Integer> m : map.entrySet()){
```

```
                String num = m.getKey();
                list.add(num);
                if(m.getValue() >= 2){    //次数大于等于 2 只记两次
                    list.add(num);
                }
            }

            String[] strsNum = new String[list.size()];
            list.toArray(strsNum);          //集合转化为数组
            combine( strsNum,0, strsNum.length-1);
            Collections.sort(resList);

            System.out.println(resList.get(resList.size()-1));

        }

        public static void swap(String[] strings, int indexa, int indexb){

            String temp = strings[indexa];
            strings[indexa] = strings[indexb];
            strings[indexb] = temp;

        }

        /**
         * 对存在的数字进行全排列，找出其中最大的值
         * @param strs          存在数字数组
         * @param start         进行交换的 index
         * @param end           数组的最后一个 index
         */
        public static void combine(String[] strs, int start, int end){

            if(start == end){
                String numStr = "";
                for (String s : strs){
                    numStr += s;       //将数组中的数字进行拼接
                }
                if(!resList.contains(Integer.valueOf(numStr)) && isChild(strs)){        //剔除满足子
串且存在的数字
                    resList.add(Integer.valueOf(numStr));
                }
            }else {
                for(int i=start; i<strs.length; i++){
                    if(i != start && strs[i].equals(strs[start])){    //用来交换的索引不同且数字相
```

等则重复

```
                        continue;
                    }
                    swap(strs, i, start);
                    if(strs[0].equals("0")){
                        continue;     //首位为 0 不满足
                    }
                    combine(strs,start+1, end);
                    swap(strs, i, start);
                }
            }
        }

        /**
         * 判断数组是否为输入数组的子串
         * @param child
         * @return
         */
        public static boolean isChild(String[] child){

            int index = 0;   //父数组索引
            boolean isOver = false;         //父数组是否遍历结束
            int count = 0;
            for(int i=0; i<child.length; i++){
                String str = child[i];
                for(int j = index; j<inputStrs.length; j++){
                    String temp = inputStrs[j];
                    if(j == inputStrs.length-1){
                        isOver = true;   //父数组遍历结束了
                    }
                    if(temp.equals(str)){
                        index = j + 1;   //存在该字符，则更新 count
                        count ++;
                        break;
                    }
                }
                if(isOver){       //父数组遍历完成且 index 没有刷新
                    break;
                }
            }

            return count == child.length;
        }
}
```