最多等和不相交连续子序列

知识点贪心
时间限制：1s 空间限制：256MB 限定语言：不限

题目描述：

给定一个数组，我们称其中连续的元素为连续子序列，称这些元素的和为连续子序列的和。数组中可能存在几组连续子序列，组内的连续子序列互不相交且有相同的和。求一组连续子序列，组内子序列的数目最多。输出这个数目。

输入描述：

第一行输入为数组长度 N，1 <= N <= 10^3。
第二行为 N 个用空格分开的整数 Ci，-10^5 <= Ci <= 10^5。

输出描述：

第一行是一个整数 M，表示满足要求的最多的组内子序列的数目。

示例1

输入：

10
8 8 9 1 9 6 3 9 1 0

输出：

4

说明：

四个子序列的第一个元素和最后一个元素的下标分别为：
2 2
4 4
5 6
7 7

示例2

输入：

10
-1 0 4 -3 6 5 -6 5 -7 -3

输出：

3

说明：

三个子序列的第一个元素和最后一个元素的下标分别为：
3 3
5 8
9 9

解题思路：

使用map来放置子序列和和其首位坐标集合
Key：连续子序列和
Value：子序列的首位坐标集合
因为坐标集合中会存在相交的集合所以还必须进行处理
如例2：
子序列和为-3的坐标集合为[3，3]，[3，9]，[5，8][9，9]
[3，9]与[3，3]，[5，8]，[9，9]都存在交集。
[3，3]，[5，8]，[9，9]不存在交集，最长子序列数目为3。

```
public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int N = sc.nextInt();
        sc.nextLine();
        String[] strings = sc.nextLine().split(" ");

        int[] ints = new int[N];

        for(int i=0; i<N; i++){
            ints[i] = Integer.valueOf(strings[i]);
        }

        /**
         * key：连续子序列和
         * value：子序列的首尾坐标数组
         */
        Map<Integer, List<int[]>> map = new HashMap<>();
        for(int i=0; i<N; i++){
            int count = ints[i];
            for (int j=i; j<N; j++){
                int[] temp = { i, j};     //首坐标 i，尾座标 j
                if( i != j){       //单独的序列不需要进行求和
                    count += ints[j];
                }
                if(map.containsKey(count)){
                    map.get(count).add(temp);
                }else {
                    List<int[]> tempList = new ArrayList<>();
                    tempList.add(temp);
                    map.put( count, tempList);
                }
            }
        }

        int res = 0;
        for (List<int[]> list : map.values()){
            res = Math.max( res, removeIntersect(list));
        }

        System.out.println(res);
```

```
    }

    /**
     * 求出子序列中互不相交的最长子序列
     * @param list    和相同的子序列集合
     * @return          最长子序列的长度
     */
    public static int removeIntersect(List<int[]> list){

        int max = 0;
        for(int i=0; i<list.size(); i++){
            List<int[]> tempList = new ArrayList<>();
            tempList.add(list.get(i));
            int right = list.get(i)[1];         //第一个序列的最后一个元素下标
            for(int j=0; j<list.size(); j++){
                if(i != j){
                    int left = list.get(j)[0];
                    if(left > right) {    //没有交集
                        tempList.add(list.get(j));
                        right = list.get(j)[1];
                    }
                }
            }
            max = Math.max( max, tempList.size());
        }

        return max;
    }
}
```