Remove Covered Intervals

題目

- Given a list of intervals, remove all intervals that are covered by another interval in the list.

- Interval [a,b) is covered by interval [c,d) if and only if **c <= a and b <= d.**

- After doing so, return **the number of remaining intervals**.

題目範例

Example 1:
Input: intervals = [[1,4],[3,6],[2,8]]
Output: 2
Explanation: Interval [3,6] is covered by [2,8], therefore it is removed.

[1,4]	[2,8]	[3,6]	[1,4]	[3,6]	[2,8]
0	2	3	0	3	2
1	2	~~3~~	4	~~6~~	8

Return 2

Example 2:
Input: intervals = [[1,4],[2,3]]
Output: 1

[1,4]	[2,3]	[2,3]	[1,4]
0	2	2	0
1	~~2~~	~~3~~	4

Return 1

Example 3:
Input: intervals = [[0,10],[5,12]]
Output: 2

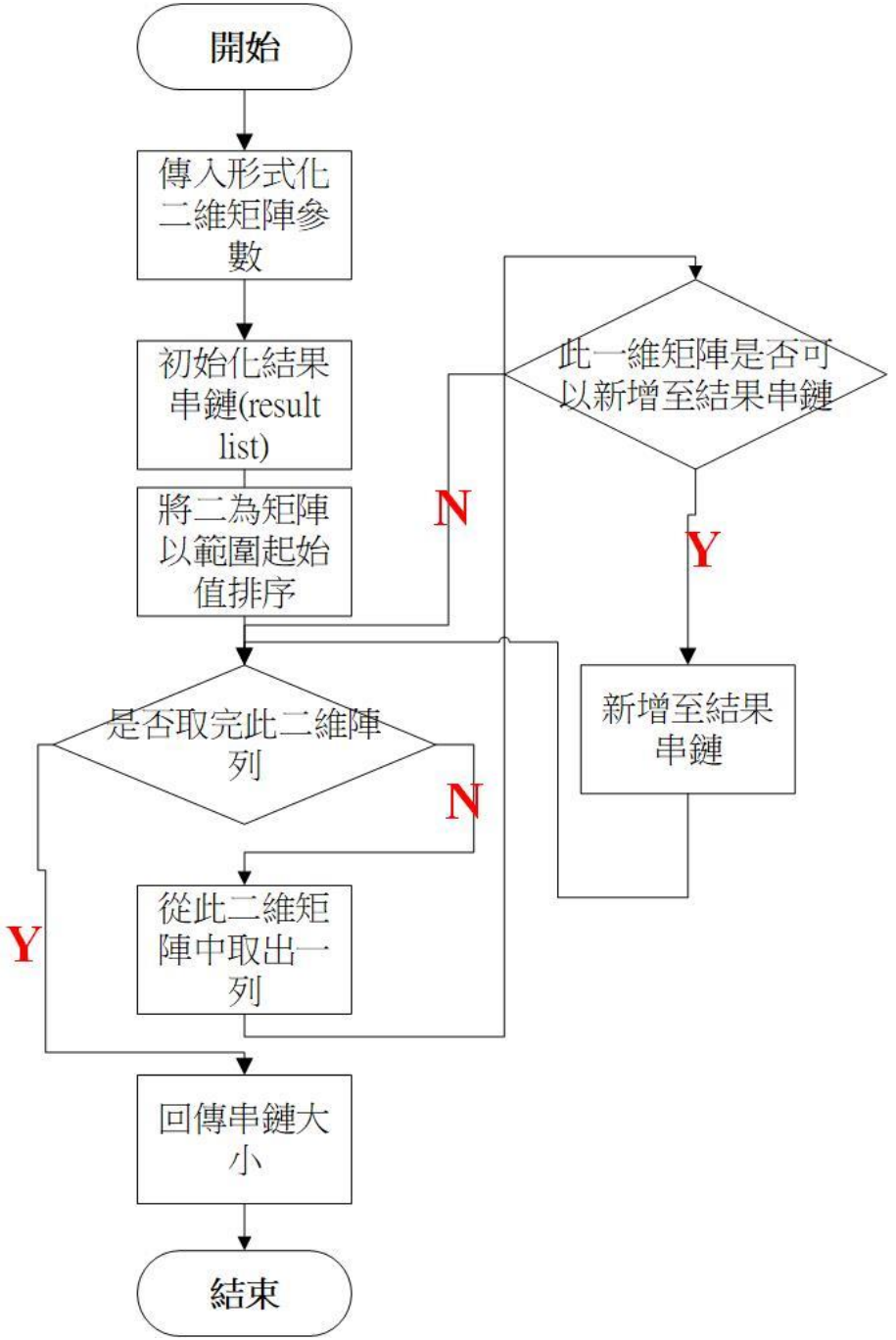
[0,10]	[5,12]	[0,10]	[5,12]
0	1	0	1
0	5	10	12

Return 2

題目觀察

- 題目給予明確的條件，所以我想應要往如何限制儲存或過濾數列的方向做處理。
- 我使用ArrayList為主要的資料結構，並以測試資料的成員是否能add於此List作為依據，此寫法的好處在於不僅找到解的串鏈，同時在return List的大小即為答案。

程式流程


開始
傳入形式化二維矩陣參數
初始化結果串鏈(result list)
將二為矩陣以範圍起始值排序
此一維矩陣是否可以新增至結果串鏈
N
Y
新增至結果串鏈
是否取完此二維陣列
N
Y
從此二維矩陣中取出一列
回傳串鏈大小
結束

程式碼

```
        List group=new ArrayList<int[]>() {
            @Override
           public boolean contains(Object o) {
           // TODO Auto-generated method stub
                  int[] e =(int[])o;

                   for(int i=0;i<this.size();i++) {
                       //被覆蓋的人不add
                       //28覆蓋3,6
                       if(this.get(i)[0]==e[0]&&this.get(i)[1]<=e[1]) {//換頭
                           this.remove(i);
                           this.add(e);
                           return false;
                       }
                       if(this.get(i)[0]<=e[0]&&this.get(i)[1]>=e[1]) { //包覆
                           return false;
                       }
                   }
                   return true;
           }

           };
        Arrays.sort(intervals,(a,b)->a[0]-b[0]);
//      for(int i=0;i<intervals.length;i++) {
//          System.out.print(intervals[i][0]+","+intervals[i][1]);
//          System.out.println();
//      }

        for(int i=0;i<intervals.length;i++) {
            if(group.contains(intervals[i])) {
                group.add(intervals[i]);
            }

        }
//      System.out.println(group.size());
    }
```