【解答】

如果 arr 中没有正数，产生的最大累加和一定是数组中的最大值。

如果 arr 中有正数，从左到右遍历 arr，用变量 cur 记录每一步的累加和，遍历到正数 cur 增加，遍历到负数 cur 减少。当 cur<0 时，说明累加到当前数出现了小于 0 的结果，那么累加的这一部分肯定不能作为产生最大累加和的子数组的左边部分，此时令 cur=0，表示重新从下一个数开始累加。当 cur>=0 时，每一次累加都可能是最大的累加和，所以，用另外一个变量 max 全程跟踪记录 cur 出现的最大值即可。

举例来说明一下，arr=[1,-2,3,5,-2,6,-1]，开始时，max=极小值，cur=0。

遍历到 1，cur=cur+1=1，max 更新成 1。遍历到-2，cur=cur-2=-1，开始出现负的累加和，所以，说明[1,-2]这一部分肯定不会作为产生最大累加和的子数组的左边部分，于是令 cur=0，max 不变。遍历到 3，cur=cur+3=3，max 更新成 3。遍历到 5，cur=cur+5=8，max 更新成 8。遍历到-2，cur=cur-2=6，虽然累加了一个负数，但是 cur 依然大于 0，说明累加的这一部分（也就是[3,5,-2]）仍可能作为最大累加和的子数组的左边部分。max 不更新。遍历到 6，cur=cur+6=12，max 更新成 12。遍历到-1，cur=cur-1=11，max 不更新。最后返回 12。解释得再直白一点，cur 累加成为负数就清零重新累加，max 记录 cur 的最大值即可。

求解最大累加和具体过程请参看如下代码中的 maxSum 方法。

```
public int maxSum(int[] arr) {
        if (arr == null || arr.length == 0) {
                return 0;
        }
        int max = Integer.MIN_VALUE;
        int cur = 0;
        for (int i = 0; i != arr.length; i++) {
                cur += arr[i];
                max = Math.max(max, cur);
                cur = cur < 0 ? 0 : cur;
        }
        return max;
}
```

子矩阵的最大累加和问题

【题目】

给定一个矩阵 matrix，其中的值有正、有负、有 0，返回子矩阵的最大累加和。

例如，矩阵 matrix 为：

-90　48　78

64　-40　64

-81　-7　66

其中，最大累加和的子矩阵为：

48　78

-40　64

-7　66

所以返回累加和 209。

例如，matrix 为：

-1　-1　-1

-1　2　2

-1　-1　-1

其中，最大累加和的子矩阵为：

2　2

所以返回累加和 4。

【难度】

尉 ★★☆☆

【解答】

在阅读本题的解释之前，请先阅读上一道题“子数组的最大累加和问题”，因为本题的最优解深度利用了上一题的解法。首先来看这样一个例子，假设一个 2 行 4 列的矩阵如下：

-2　3　-5　7

1　4　-1　-3

如何求必须含有 2 行元素的子矩阵中的最大累加和？可以把两列的元素累加，然后得到累加数组[-1,7,-6,4]，接下来求这个累加数组的最大累加和，结果是 7。也就是说，必须含有 2 行元素的子矩阵中的最大和为 7，且这个子矩阵是：

3

4

也就是说，如果一个矩阵一共有 k 行且限定必须含有 k 行元素的情况下，我们只要把矩阵中每一列的 k 个元素累加生成一个累加数组，然后求出这个数组的最大累加和，这个

最大累加和就是必须含有 k 行元素的子矩阵中的最大累加和。

请读者务必理解以上解释，下面看原问题如何求解。为了方便讲述，我们用题目的第一个例子来展示求解过程，首先考虑只有一行的矩阵[-90,48,78]，因为只有一行，所以累加数组 arr 就是[-90,48,78]，这个数组的最大累加和为 126。

接下来考虑含有两行的矩阵：

-90　48　78

64　-40　64

这个矩阵的累加数组就是在上一步的累加数组[-90,48,78]的基础上，依次在每个位置上加上矩阵最新一行[64, -40, 64]的结果，即[-26,8,142]，这个数组的最大累加和为 150。

接下来考虑含有三行的矩阵：

-90　48　78

64　-40　64

-81　-7　66

这个矩阵的累加数组就是在上一步累加数组[-26,8,142]的基础上，依次在每个位置上加上矩阵最新一行[-81,-7,66]的结果，即[-107,1,208]，这个数组的最大累加和为 209。

此时，必须从矩阵的第一行元素开始，并往下的所有子矩阵已经查找完毕，接下来从矩阵的第二行开始，继续这样的过程，含有一行矩阵：

64 -40　64

因为只有一行，所以累加数组就是[64,-40,64]，这个数组的最大累加和为 88。

接下来考虑含有两行的矩阵：

64　-40　64

-81　-7　66

这个矩阵的累加数组就是在上一步累加数组[64,-40,64]的基础上，依次在每个位置上加上矩阵最新一行[-81,-7,66]的结果，即[-17,-47,130]，这个数组的最大累加和为 130。

此时，必须从矩阵的第二行元素开始，并往下的所有子矩阵已经查找完毕，接下来从矩阵的第三行开始，继续这样的过程，含有一行矩阵：

-81　-7　66

因为只有一行，所以累加数组就是[-81,-7,66]，这个数组的最大累加和为 66。

全部过程结束，所有的子矩阵都已经考虑到了，结果为以上所有最大累加和中最大的 209。

整个过程最关键的地方有两处：

- 用求累加数组的最大累加和的方式得到每一步的最大子矩阵的累加和。
- 每一步的累加数组可以利用前一步求出的累加数组很方便地更新得到。

如果矩阵大小为 $N \times N$ 的，以上全部过程的时间复杂度为 $O(N^3)$，具体请参看如下代码中的 maxSum 方法。

```
public int maxSum(int[][] m) {
        if (m == null || m.length == 0 || m[0].length == 0) {
                return 0;
        }
        int max = Integer.MIN_VALUE;
        int cur = 0;
        int[] s = null; // 累加数组
        for (int i = 0; i != m.length; i++) {
                s = new int[m[0].length];
                for (int j = i; j != m.length; j++) {
                        cur = 0;
                        for (int k = 0; k != s.length; k++) {
                                s[k] += m[j][k];
                                cur += s[k];
                                max = Math.max(max, cur);
                                cur = cur < 0 ? 0 : cur;
                        }
                }
        }
        return max;
}
```

在数组中找到一个局部最小的位置

【题目】

定义局部最小的概念。arr 长度为 1 时，arr[0]是局部最小。arr 的长度为 $N(N>1)$时，如果 arr[0]<arr[1]，那么 arr[0]是局部最小；如果 arr[N-1]<arr[N-2]，那么 arr[N-1]是局部最小；如果 $0<i<N-1$，既有 arr[i]<arr[i-1]，又有 arr[i]<arr[i+1]，那么 arr[i]是局部最小。

给定无序数组 arr，已知 arr 中任意两个相邻的数都不相等。写一个函数，只需返回 arr 中任意一个局部最小出现的位置即可。

【难度】

尉 ★★☆☆