整的全部过程请参看如下代码中的 modify 方法。

```
public void modify(int[] arr) {
        if (arr == null || arr.length < 2) {
                return;
        }
        int even = 0;
        int odd = 1;
        int end = arr.length - 1;
        while (even <= end && odd <= end) {
                if ((arr[end] & 1) == 0) {
                        swap(arr, end, even);
                        even += 2;
                } else {
                        swap(arr, end, odd);
                        odd += 2;
                }
        }
}

public void swap(int[] arr, int index1, int index2) {
        int tmp = arr[index1];
        arr[index1] = arr[index2];
        arr[index2] = tmp;
}
```

子数组的最大累加和问题

【题目】

给定一个数组 arr，返回子数组的最大累加和。

例如，arr=[1,-2,3,5,-2,6,-1]，所有的子数组中，[3,5,-2,6]可以累加出最大的和 12，所以返回 12。

【要求】

如果 arr 长度为 N，要求时间复杂度为 $O(N)$，额外空间复杂度为 $O(1)$。

【难度】

士 ★☆☆☆

【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：