First we sort the array with the java library in the following code

```
Arrays.sort(a);
```

Then, judge the length of the array, if the length is an even number, we split the array to two array, and the every element in the first array is smaller than the second array, the insert the element in the second array to the first array by the following way

```
int[] temp1 = Arrays.copyOf(a,arraySize/2);
int[] temp2 = Arrays.copyOfRange(a, arraySize/2 , arraySize );
for(int i = 0; i <  arraySize/2; i++)
{
    a[2*i] = temp2[i];
    a[2*i + 1] = temp1[i];
}
```

If the length is an odd number, we split the array into two arrays, and the length of first array is less one element than the second array. And replace the array with the following code.

```
int[] temp1 = Arrays.copyOf(a, arraySize/2);
int[] temp2 = Arrays.copyOfRange(a, arraySize/2, arraySize );
for(int i = 0; i <  arraySize/2; i++)
{
    a[2*i] = temp2[i];
    a[2*i + 1] = temp1[i];
}
```

Then we get an array in terms of local maxima and local minima.

The overall complexity is the complexity of the sort in java library add O(n), because first we sort the array, and then we replace the array, for this step we just make one loop.

there is't an linear time solution for this problem, because, we should know the order in the array and then replace the array, for the first step, with the best case, it may be O(n), but we also need to do the second step. So there is't an linear solation.