**PSEUDO CODE FOR FIRST ALGORITHM**

1. Read input n from the user  
2. Declare an array arr of size n  
3. Read n integers and store them in arr  
4. For i=1 to n-1 (incrementing by 2)  
a. If arr[i] is greater than arr[i-1]  
i. If arr[i] is less than arr[i+1]  
1. Swap arr[i] with arr[i+1]  
b. Else  
i. Swap arr[i] with arr[i-1]  
ii. If arr[i] is less than arr[i+1]  
1. Swap arr[i] with arr[i+1]  
5. Print the sorted array arr

**ANALYSIS AND TIME COMPLEXITY**

Reads an integer n from the user using scanf().

Creates an integer array of size n.

Reads n integers from the user and stores them in the array using a for loop and scanf().

Sorts the elements of the array in a specific way using another for loop.

Prints the sorted array using a for loop and printf().

Time complexity is O(n)

**Pseudo code for second algorithm**

wiggleSort(array, Size):  
if Size <= 1:  
return  
mid = Size / 2  
wiggleSort(array, mid)  
wiggleSort(array + mid, Size - mid)  
for i = 0 to mid - 1:  
j = i + mid  
if array[i] < array[j]:  
swap(array[i], array[j])

**Analysis and time complexity**

The algorithm works by recursively sorting the first and second halves of the array. Then, it compares each pair of adjacent elements from the two halves and swaps them if the first element is less than the second element. This ensures that the final result satisfies the desired property.  
The base case of the recursion is when the size of the array is 1 or less, in which case there is nothing to do.

Time complexity is O(nlogn)