Sort of Sort.

When running this method, the best case scenario would be if the array were of empty of size 1.

The reasoning behind that it would run 1 time. It would go through the method and would not have any elements to compare so it would just print out the empty array or the element which exist.

Worst case scenario is n^2 given that there two for loops the, non-repeat elements which are all the variables before the for loop will all be worth 1 and then in order to continue the j loop must be also considered that will n times before the loop is finished. The I loop will also run n times given the n^2.   
When an array is provided the method will check each index of the array verify if it is larger than the first element if so then that index becomes the largest element. Once that is complete the largest elements will be swapped to the last index given the counter which tracks how many times to conduct this. Same goes for the left side of the array once the counter is reached the elements will swap in opposite direction. This is the max elements being swapped. If the code is incorrectly written when swapping this will cause a bug in the code and continue to leave errors. In a case like this time complexity will be hard to determine. So in this case where the current code has a bug that has not been rectified the time complexity for this method is probably n since it a certain point it will break from the loop and return an error.