sort the following elements using merge sort divide-and conquer startegy [38,27,43,3] and analyze-complexity of the algorithm.

split into two halves: [38,27] and [43,3].

Recursively sort each half:

* sort [38,27]
* split into [38] and [27].
* Both lists are of size 1, so they are already sorted.
* merge [38] and [27]
    * compare 38 and 27, take 27.
    * remaining: 38.
    * merged result: [27,28].

sort [43,3].

* split into: [43] and [3]
* Both lists are size of 1, so, they are already sorted.
* merge [43] and [3]:

  compare 43 and 3, take 3.

    * remaining: 43.
    * merged result: [3,43].

merged the two sorted halves:

merge [27, 38] and [3,43]:

* compare 27 and 3, take 3.
* compare 27 and 43, take 27.
* compare 38 and 43, take 38.
* remaining: 43.

merged result: [3,27,38,43].

2. sort the array 64, 34, 25, 12 using bubble sort. what is the complexity of selection sort.

A.) First pass:

* compare 64 and 34, swap: [34, 64, 25, 12].
* compare 64 and 25, swap: [34, 25, 64, 12].
* compare 64 and 12, swap: [34, 25, 12, 64].

second pass:

* compare 34 and 25, swap: [25, 34, 12, 64].
* compare 34 and 12, swap: [25, 12, 34, 64].
* compare 34 and 64, no swap: [25, 12, 34, 64].
* array after second pass: [25, 12, 34, 64].

third pass:

* compare 25 and 12, swap: [12, 25, 34, 64].
* compare 25 and 34, no swap: [12, 25, 34, 64].
* compare 34 and 64, no swap: [12, 25, 34, 64].

fourth pass:

* compare 12 and 25, no swap: [12, 25, 34, 64].
* compare 25 and 34, no swap: [12, 25, 34, 64].
* compare 34 and 64, no swap: [12, 25, 34, 64].

sorted array: [12, 25, 34, 64].

18. sort the array 64, 25, 12, 22 using selection sort. what is the time complexity of selection sort.

A. find the minimum element in the list [64, 25, 12, 22]:

* minimum element is 12.

* swap 12 with the first element (64):

Array after first pass: [12, 25, 64, 22].

second pass:

* find the minimum element in the list [25, 64, 22]
  * minimum element is 22.

* swap 22 with the first element of the unsorted part (25).

  * array after sorted pass: [12, 22, 64, 25].

third pass:

* find the minimum element in the list [64, 25].
  * minimum element is 25.

* swap 25 with the first element of the unsorted part (64).

  Array after third pass [12, 22, 25, 64].

∴ sorted array: [12, 22, 25, 64].

4. sort the following elements using insertion sort using brute force approach by [38, 27, 43, 3] and analyze complexity of the algorithm.

A.) Intital array : [38, 27, 43, 3]

first pass (i=1):

* Key: 27
* compare 43 with 38
* 43 is greater than 38.
* 27 is less than 38, so moves 38 to the right.
* Insert 27 at the begining.
* Array: [27, 38, 43, 3].

second pass (i=2):

* Key: 43.
* compare 43 with 38.

* 43 is greater than 38, so it stays in place.

* Array: [27, 38, 43, 3].

* third class:

compare 3 with 43, 38 and 27.

* 3 is less than 43, move 43 to the right.

* 3 is less than 38, move 38 to the right.

* 3 is less than 27, move 27 to the right.

* Insert 3 at the beginning.

* Array: [3, 27, 38, 43].

Time complexity : $f(n) = O(n^2)$.

5. Given an array of [4, -2, 5, 3, 10] integers, sort the elements using insertion sort using brute force approach startegy analyze complexity of the algorithm.

first pass (i=1):

* key: -2
* compare -2 with 4
* since -2 less than 4, move 4 to the right.
* insert -2 at the begining.
* Array after the first pass: [-2, 4, 5, 3, 10].

second pass (i=2):

* Key : 5
* compare 5 with 4.
* since 5 is greater than 4, it stays in place.
* Array after the second pass: [-2, 4, 5, 3, 10].

third pass (i=3):

* Key : 3.

* compare 3 with 5 and 4.

* since 3 is less than 5, move 5 to the right.

* since 3 is less than 4, move 4 to the right.

* Insert 3 after -2.

* Array after the third pass : [-2, 3, 4, 5, 10].

fourth pass (i=4)

* Key : 10.

* compare 10 with 5, 4, 3, and -2.

* since 10 is greater than 5, it stays in place.

* Array after the fourth pass: [-2, 3, 4, 5, 10].

∴ sorted array: [-2, 3, 4, 5, 10].