Code Implementation:

The selection sort algorithm is implemented in Python to sort an array of integers in ascending order. The algorithm iterates through the array, selecting the smallest element from the unsorted portion and swapping it with the first unsorted element. This process is repeated until the entire array is sorted.  
  
A computer screen shot of a program code

Description automatically generated

Time and Space Complexity:

Time Complexity: The time complexity of the selection sort algorithm is O(n^2) due to the nested loops. For each element in the array, the algorithm searches through the unsorted part of the array to find the minimum element. This results in a quadratic number of comparisons and swaps, making the algorithm inefficient for large datasets.

Space Complexity: The space complexity of selection sort is O(1) because it is an in-place sorting algorithm. It requires a constant amount of additional memory space, regardless of the size of the input array.

Enhancement Changes:

Descending Order Sort: The selection sort algorithm was modified to sort an array in descending order by changing the comparison operator in the inner loop.

A computer screen with white text

Description automatically generated

Stable Selection Sort: To make the selection sort algorithm stable, the approach was modified to shift elements instead of swapping them, ensuring that equal elements maintain their relative order.

A computer screen shot of a program code

Description automatically generated

Test Cases:

The test cases cover a range of scenarios, including randomly generated arrays, arrays already sorted in ascending and descending order, arrays with all elements being the same, empty arrays, and arrays with a single element. All test cases pass.