ASSIGNMENT 3

DCIT 204

11245046

11047613

11302223

11249336

11083926

11090094

11117313

TIME COMPLEXITIES.

The following are the time complexities of the algorithms implemented in our code.

**Bubble Sort**

Bubble Sort is a simple sorting algorithm with a time complexity of O(n^2). It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process is repeated until the list is sorted.

**Quick Sort**

Quick Sort is an efficient, divide-and-conquer sorting algorithm with an average time complexity of O(nlogn). It works by selecting a 'pivot' element and partitioning the array into two sub-arrays, according to whether they are less than or greater than the pivot.

**Heap Sort**

Heap Sort is a comparison-based sorting technique based on Binary Heap data structure. It has a time complexity of O(nlogn) for all cases (best, average, and worst). It first builds a max heap and then repeatedly extracts the maximum element from the heap and reduces the heap size.

**Insertion Sort**

Insertion Sort is a simple sorting algorithm that builds the final sorted array one item at a time. It has an average and worst-case time complexity of)O(n^2), but it performs well on small or nearly sorted datasets.

**Selection Sort**

Selection Sort is a simple comparison-based sorting algorithm. It has a time complexity of O(n^2) for all cases. It repeatedly selects the minimum element from the unsorted portion of the list and swaps it with the first unsorted element.

**Shell Sort**

Shell Sort is an in-place comparison sort. It generalizes insertion sort by comparing elements separated by a gap of several positions. It has an average time complexity of O(nlogn) and a worst-case time complexity between O(n^{3/2}) and O(n^2).

**Binary Search**

Binary Search is a fast search algorithm with a time complexity of O(logn). It works on sorted arrays by repeatedly dividing the search interval in half. It efficiently locates the target value by discarding half of the remaining elements at each step.

**Linear Search**

Linear Search is a simple search algorithm that sequentially checks each element in a list until the target element is found or the end of the list is reached. It has a time complexity of O(n), where n is the number of elements in the list.

**Exhaustive Search/ Brute Force Search**

Exhaustive Search, also known as Sequential Search or Brute Force Search, is a simple search algorithm that systematically checks all possible solutions until the desired solution is found. It has a time complexity of O(n), where n is the number of elements in the list.