Aiman Abdullah

1001472426

Project- 01

**Abstract**:

The main idea of the project is to show run time and efficiency of each Sorting Algorithm by applying same and different input size of data. By, apply Bubble Sort, Insert Sort, Select Sort, Quick Sort, Merge Sort and Heap Sort to our data we can figure which Sort algorithm is good to our data set. In the graphs below you can see that by increase the size of input data the Time Complexity will increase regarding the size.

**Components Project:**

1. Data Structure Is Array of integers.
2. Sorting Algorithms Bubble, Insertion, Selection, Merge, Quick and Heap.
3. Switch to select one of the Algorithms
4. Random Class to generate Random Values
5. currentTimeMillis which use to measure the execution time of Algorithm

**Data Size and Execution Time:**

Input Data represent the size of the array(input)

![A screenshot of a cell phone

Description automatically generated]()

Algorithms Results:

1. Bubble Sort: Depending on Excel Sheet I have provided. Different input Values have been chosen to test the Algorithm. Bubble Algorithm is working fast if data set is small and work slow if data set is big as we can notice in the graph below every time the input increase the graph move up toward y-axis. Bubble sort has nested loops which need to check every element from left to right with it neighbors which will cause to low efficiency. Worse case is O(N^2). Notice Time represented in Milliseconds

![A screenshot of a social media post

Description automatically generated]()

1. Insertion Sort: Depending on Excel Sheet I have provided. Different input Values have been chosen to test the Algorithm. Insertion Algorithm is working fast if data set is small and work slow if data set is big as we can notice in the graph below every time the input increase the graph move up toward y-axis depending on results sheet we can see Insertion is better than Bubble and selection algorithms. However, insertion sort still has nested loop which will cause to low efficiency with huge data set. Worse case is O(N^2). Notice Time represented in Milliseconds

![A screenshot of a cell phone

Description automatically generated]()

1. Selection Sort: Depending on Excel Sheet I have provided. Different input Values have been chosen to test the Algorithm. Selection Algorithm is working fast if data set is small and work slow if data set is big as we can notice in the graph below every time the input increase the graph move up toward y-axis depending on result we can see Selection is better than Bubble but worse than insertion. However, selection sort still has nested loop which will cause to low efficiency with huge data set. Selection sort Worse case is O(N^2). Notice Time represented in Milliseconds.

![A close up of a map

Description automatically generated]()

1. Merge Sort: Depending on Excel Sheet I have provided. Different input Values have been chosen to test the Algorithm. Merge Algorithm is working faster than Bubble, Insert and Select regarding Big data set. Its divide set to subsets and do recursion calls for that job then will loop back to merge the elements from bottom to up and comparing elements to find the small value and make it at the left side of the array. So, make this algorithm fast. As you can see, The time complexity is O(n log n ) because dividing take log n time and n is number of elements in the set. Notice Time represented in Milliseconds.

![A screenshot of a social media post

Description automatically generated]()

1. Heap Sort: Depending on Excel Sheet I have provided. Different input Values have been chosen to test the Algorithm. Heap Algorithm is working faster than Bubble, Insert, Select and Merge sorting algorithms regarding Big data set. Heap use heapify calls which check and manage the correct position for left, right child’s and parent. As you can see, the time complexity is O (log n). Notice Time represented in Milliseconds.

![A close up of a map

Description automatically generated]()

1. Quick Sort: Depending on Excel Sheet I have provided. Different input Values have been chosen to test the Algorithm. Quick Algorithm is working faster than Bubble, Insert, Select, Heap and Merge sorting algorithms regarding Big data set.

As you can see, the time complexity is O (n log n) when pivot is middle, last, or first. Quick sort work as Merge it divides set to subsets every time select pivot and range values depending on pivot. Notice Time represented in Milliseconds.

![A close up of text on a white background

Description automatically generated]()

1. 3- Median Quick Sort:

In this part we are using 3-median approach. To select the pivot we will choice the first, the last and the middle of the array and the pivot will select as median between those three values. This approach is not only used to select the pivot but also to put the three items in their sorted place in the array. The worst case for this algorithm will be O(N^2)

First of all, I will list all functions and explain functionality of each one.

1. quick\_3way\_sort : From here the algorithm will start. This is recursive function mean it will call it self many time until reach the condition to stop. This function will take 3 parameters the array, the begin and the end of the array. When this function start will check if the length less than or equal to 3 if yes then, it will call another function (Sortquick\_3\_way) but if the size is greater than 3 then there are two functions will called (quick\_3\_medians, quicksort\_divition) and function will keep call itself recursively.
2. quicksort\_divition: Here will take 4 inputs (array, begin , pivot, end)

this will do swapping and at the end will return partition index.

1. quick\_3\_medians: This also will take 3 input( array, begin, end) first of all will find median by divide begin +end /2

and will do some checking and swaps to return the median pivot by checking the first, last and middle elements of array.

1. Sortquick\_3\_way: This will take 3 inputs (array, begin, end)

Here the function will do swap when the size is less than input of the array by comparing places of the elements.

![A close up of a map

Description automatically generated]()

**Conclusion:**

Depending to our study of Sorting algorithms we found that with small data set we can use one of these bubble, insertion, and selection algorithms. With big data set its efficient to use one of these algorithms Quick sort, Merge Sort,3 median quick sort and Heap Sort. As a prefer decision we prefer to use Heap Sort in terms of huge data and for small data set we prefer to use insertion algorithm.