**CSE 5311 Design and Analysis of Algorithms**

**Project Assignment-**1

**Submitted to**

**Dr.** Negin Fraidouni

**Computer Science and Engineering Department**

**University of Texas , Arlington**

Submitted By

Sunil Joshi

1001789901

Department of Civil Engineering

University of Texas, Arlington

**Table of Content: page**

1. Statement of problem………………………………….3
2. Introduction………………………………………………….3
3. Data structures explanation …………………………5
4. Method used………………………………………………..7
5. Result……………………………………………………………8
6. Graphical Representation…………………….........9
7. Conclusion………………………………………………….13
8. Discussion…………………………………………………..13

References ……………………………………………………14

1. **Statement of problem:**

Implement and compare the following sorting algorithm:

* Merge sort
* Heap sort
* Quicksort (Regular quick sort\* and quick sort using 3 medians)
* Insertion sort
* Selection sort
* Bubble sort

1. **Introduction**

Efficiency of an algorithm depends on two parameters:

1. Time Complexity

2. Space Complexity

**Time Complexity:** Time Complexity is defined as the number of times a particular instruction set is executed rather than the total time is taken. It is because the total time taken also depends on some external factors like the compiler used, processor’s speed, etc.

The time complexity of different sorting algorithms:

| **ALGORITHM** | **TIME COMPLEXITY** | | |  |
| --- | --- | --- | --- | --- |
|  | **BEST** | **AVERAGE** | **WORST** |  |
| [Selection Sort](http://geeksquiz.com/selection-sort/) | Ω(n^2) | θ(n^2) | O(n^2) |  |
| [Bubble Sort](http://geeksquiz.com/bubble-sort/) | Ω(n) | θ(n^2) | O(n^2) |  |
| [Insertion Sort](http://geeksquiz.com/insertion-sort/) | Ω(n) | θ(n^2) | O(n^2) |  |
| [Heap Sort](http://geeksquiz.com/heap-sort/) | Ω(n log(n)) | θ(n log(n)) | O(n log(n)) |  |
| [Quick Sort](http://geeksquiz.com/quick-sort/) | Ω(n log(n)) | θ(n log(n)) | O(n^2) |  |
| [Merge Sort](http://geeksquiz.com/merge-sort/) | Ω(n log(n)) | θ(n log(n)) | O(n log(n)) |  |

**3.Data Structures explanation:**

I created the functions for each algorithm and designed it with following concepts. Time of execution of each algorithm is calculated calling each function and passes different size of array into it.

**Sorting Algorithms**: A sorting algorithm is an algorithm that puts elements of a list in a certain order. Efficient sorting is important for optimizing the efficiency of other algorithms (such as search and merge algorithms) that require input data to be in sorted lists.

**Insertion Sort**: Insertion sort is based on the idea that one element from the input elements is consumed in each iteration to find its correct position

**Approach**:

The input is an unsorted array

◦ Find the correct position of xi in the list. Move other elements to Insert xi in the correct place.

**Merge Sort** :Merge sort is a divide-and-conquer algorithm based on the idea of breaking down a list into several sub-lists until each sub list consists of a single element and merging those sub lists in a manner that results into a sorted list.

**Approach**: ◦ Divide the unsorted list into 2 sub lists each time ◦ Do this until each sub list has 1 element ◦ Take adjacent pairs of two singleton lists and merge them to form a list of 2 elements. N will now convert into N/2 lists of size 2. ◦ Repeat the process till a single sorted list of obtained.

**Selection Sort**: Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts:

1) The subarray which is already sorted.

2) Remaining subarray which is unsorted.

Initially, the sorted part is empty, and the unsorted part is the entire list. The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

**Bubble sort**: Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

**Heap Sort**: Heap sort algorithm is divided into two basic parts: Creating a Heap of the unsorted list/array. Then a sorted array is created by repeatedly removing the largest/smallest element from the heap and inserting it into the array. The heap is reconstructed after each removal.

**4.Method used**:

Python was used for implementing the sorting algorithms. Different libraries such as random, matplotlib, time were used. The different sizes of data were generated using random library of python. Four sets of data (1000,2000,3000,4000) were generated and used as input for sorting. Using date and time library of python, time complexity (execution time) was calculated. To show data visualization properly, graphs were drawn using matplotlib.

**5.Result**: Execution time for different size of array for randomly generated array was found as follows:

|  |  |  |
| --- | --- | --- |
| **Sorting Algorithms** | **Array Size** | **Time taken to execute in MS(millisec)** |
| Merge Sort | [1000, 2000, 3000, 4000] | [1020.607, 1057.570, 1014.09390, 1041.441] |
| Bubble Sort | [1000, 2000, 3000, 4000] | [1094.4290161132812, 1334.601640701294, 1748.6822605133057, 2390.2077674865723] |
| Selection Sort | [1000, 2000, 3000, 4000] | [1041.0058498382568, 1125.2837181091309, 1410.3705883026123, 1627.4232864379883] |
| Heap Sort | [1000, 2000, 3000, 4000] | [1007.594108581543, 1014.246940612793, 1030.1635265350342, 1050.1925945281982] |
| Insertion Sort | [1000, 2000, 3000, 4000] | [1050.2727031707764, 1157.9911708831787, 1456.8092823028564, 1649.6515274047852] |
| Quick Sort | [1000, 2000, 3000, 4000] | [1084.094524383545, 1320.5039501190186, 1791.595220565796, 2586.719751358032] |

**6.The Graphical representation for execution time of different algorithm is mentioned as follows:**

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Rx = [1,2,5,3,9,7] #Rx is a random list

Text

Description automatically generated

**7.Conclusion:**

Each sorting algorithms was implemented using python. The execution time for the algorithms was found in the following order (Increasing time for execution): merge sort, heapsort, selection sort, insertion sort, bubble sort, quick sort.

Chart, line chart

Description automatically generated

Since our model is generating data randomly, the output result might be varied. It will depend on the best case, average case or worst case for each algorithm.

**8.Discussion:**

The run time for merge sort and heap sort was found be similar, selection sort and insertion sort were similar, quick sort and bubble sort was found to be similar. Running time of these algorithm fluctuate when data generated are different.

**References**: Classnote, Thomas H. Cormen, Charles E. Leiserson, Ronald L.  Rivest and Clifford Stein, **Introduction to  Algorithms**