Active Learning of Various Sorting Algorithm

Nirav Patel[1], Vaishali Patel[2]

[1]Computer Engineering, Alpha College of Engineering & Technology

[2]Computer Engineering, Universal College of Engineering & Technology

Abstract- Sorting algorithm are widely used for arranging objects in some order like ascending order or descending order. In this paper various types of sorting algorithm will be explained with respect to their time and space complexity. Sorting is ordering a list of objects.

Index Terms- Sorting Algorithm, Quick Sort, Heap Sort, Merge sort, Selection sort, Bubble sort, Insertion sort

I. INTRODUCTION

A sorting problem has attracted a great interest of research due to the complexity of solving it efficiently[3]. Efficient sorting is very important for optimizing the use of other algorithms which needed input data to be in sorted lists. We can classify sorting techniques in two different classes. If the number of substance is sufficient to fits into the main memory, then this type of sorting is called *internal sorting*. If the number of items is so large that some of them reside on external storage during the sorting process, it is known as external *sorting*.

Each sorting algorithm is better in some situation and has its own merits and demerits. Likewise the insertion sort is preferable to the quick sort for small files and for almost-sorted files [1].

There are various types of sorting algorithms. Some of them are enlisted as follows:

(1) Quick sort
(2) Merge sort
(3) Selection sort
(4) Bubble sort
(5) Insertion sort
(6) Heap sort

II. SORTING ALGORITHM AND TIME COMPLEXITY

➢ Quick sort:

The steps are:

1. Pick an element from all of the array elements. This element is known as pivot
2. **partition operation:** Rearrange the array so that all elements with values less than the pivot come prior to the pivot, while all elements with values greater than the pivot come after it After this partitioning, the pivot is in its final location.
3. Recursively apply the above two steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.

Time complexity:

- Best case : n log(n)
- Average case: n log(n)
- Worst case: n^2

➢ Merge sort:

Merge sort works as follows:

1. Partition the unsorted list into *n* sub parts, each subpart contain one element in it.
2. Repeatedly merge sublists to produce new sorted sublists until there is only 1 sublist remain. This will be the sorted list.

Time complexity:

- Best case : n log(n)
- Average case: n log(n)
- Worst case: n log(n)

➢ Selection sort:

This algorithm divides the input list into two parts where each part is the sublist of items already sorted, which is developed from left to right at the front (left) of the list, and the sublist of items left behind to be sorted that occupy the rest of the list.

At the initial stage, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by discovering the smallest element in the unsorted sublist, interchanging it with the leftmost unsorted element and moving the sublist boundaries one element to the right.

Time complexity:

- Best case : n^2
- Average case:n^2
- Worst case: n^2

➢ Bubble sort:

Bobble sort is also known as sinking sort. It works by repeatedly stepping through the list to be sorted, comparing each pair of nearby items and swapping them if they are in the wrong order[2]. The pass throughout the list is repeated until no swaps are required, which point to that the list is sorted.

Time complexity

- Best case : n
- Average case: n^2
- Worst case: n^2

➢ Insertion sort:

Insertion sort algorithm consumes one input element for each iteration and grows a sorted output list. In each iteration, insertion sort removes one element from the input list, finds the exact position it belongs within the sorted list, and inserts it there. This process is repeated until no input elements remain

Time complexity

- Best case : n
- Average case: n^2
- Worst case: n^2

➢ Heap sort:

Heap sort can be treated as an improved selection sort. This algorithm divides its input into a sorted and an unsorted area, and it repeatedly shrinks the unsorted region by extracting the smallest element and moving that to the sorted region.

The improvement consists of the use of a heap data structure rather than a linear-time search to find the minimum.

Time complexity

- Best case : n log(n)
- Average case: n log(n)
- Worst case: n log(n)

➢ Comparison

Sort	**n=100**	**n=1000**	**n=10000**
Bubble sort	**6**	**49**	**184**
Insertion sort	**8**	**26**	**163**
Selection sort	**9**	**37**	**572**
Quick sort	**2**	**13**	**43**

Table 1: Average case runtime of some sorting techniques for different size of array.

Sort	**n=100**	**n=1000**	**n=10000**
Bubble sort	**8**	**73**	**697**
Insertion sort	**11**	**42**	**227**
Selection sort	**12**	**63**	**940**
Quick sort	**4**	**8**	**65**

Table 2: The worst case runtime of some sorting techniques for different size of array.

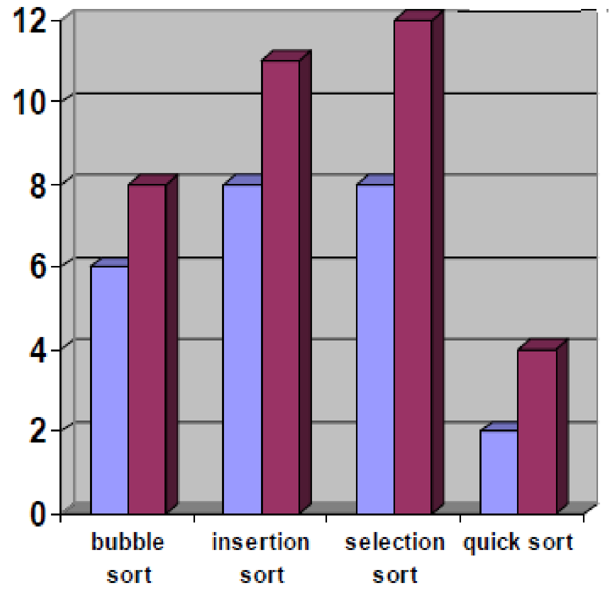

Fig. 1: Run time in average and worst case for some sorting techniques Integer (n) =100

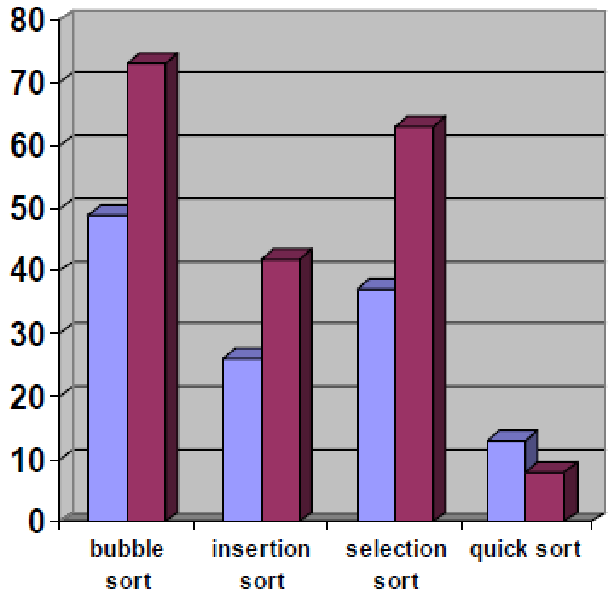

Fig.2: Run time in average and worst case for some sorting techniques Integer (n) =1000

III. CONCLUSION

In this paper we are discuss about sorting algorithms which are useful to sort the activity or the data in different manners.

REFERENCES

[1] Introduction to algorithm,Thomas H. Coreman

[2] https://www.cs.cmu.edu/~adamchik/15121/lectures/Sorting%20Algorithms/sorting.html

[3] http://en.wikipedia.org/wiki/Bubble_sort

[4] http://en.wikipedia.org/wiki/Sorting_algorithm

[5] http://www.eportfolio.lagcc.cuny.edu/scholars/doc_fa09/eP_fa09/Jariya.%20Phongsai/documents/mac%20286/sorting%20algorithms%20research.pdf

[6] www.ijarcsse.com/docs/papers/Volume_3/11.../V3I10-0312.pdf

[7] http://www.expertsmind.com/questions/sorting-algorithm-30133656.aspx

[8] http://www.personal.umich.edu/~rlsmith/Solution%20and%20Forecast%20Horizons.pdf