Heaven's Light is Our Guide

Computer Science & Engineering

Rajshahi University of Engineering & Technology

Lab Manual

Module- 01

Course Title : Sessional based on CSE 2201

Course No. : CSE 2202

Experiment No. 1

Name of the Experiment: Complexity analysis of various sorting algorithms.

Date: 1st Cycle

Algorithms:

- **Bubble Sort**
- **Selection Sort**
- **Insertion Sort**

Bubble Sort:

It works by repeatedly stepping through the list to be sorted, comparing two items at a time and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.

Example:

Let us take the array of numbers "5 1 4 2 8", and sort the array from lowest number to greatest number using bubble sort algorithm. In each step, elements written in bold are being compared.

First Pass:

(**5 1** 4 2 8) (**1 5** 4 2 8) Here, algorithm compares the first two elements, and swaps them.
(1 **5 4** 2 8) (1 **4 5** 2 8)
(1 4 **5 2** 8) (1 4 **2 5** 8)
(1 4 2 **5 8**) (1 4 2 **5 8**) Now, since these elements are already in order, algorithm does not swap them.

Second Pass:

(**1 4** 2 5 8) (**1 4** 2 5 8)
(1 **4 2** 5 8) (1 **2 4** 5 8)
(1 2 **4 5** 8) (1 2 **4 5** 8)
(1 2 4 **5 8**) (1 2 4 **5 8**)

Now, the array is already sorted, but our algorithm does not know if it is completed. Algorithm needs one whole pass without any swap to know it is sorted.

Third Pass:

(**1 2** 4 5 8) (**1 2** 4 5 8)
(1 **2 4** 5 8) (1 **2 4** 5 8)
(1 2 **4 5** 8) (1 2 **4 5** 8)
(1 2 4 **5 8**) (1 2 4 **5 8**)

Finally, the array is sorted, and the algorithm can terminate.

Pseudo-code:

A simple way to express bubble sort in pseudocode is as follows:

```
procedure bubbleSort( A : list of sortable items ) defined as:
  do
    swapped := false
    for each i in 0 to length( A ) - 1 do:
      if A[ i ] > A[ i + 1 ] then
        swap( A[ i ], A[ i + 1 ] )
        swapped := true
      end if
    end for
  while swapped
end procedure
```

Complexity: $O(n^2)$

Selection Sort:

The algorithm works as follows:

1. Find the minimum value in the list
2. Swap it with the value in the first position
3. Repeat the steps above for remainder of the list (starting at the second position)

Example:

Here is an example of this sort algorithm sorting five elements:

64 25 12 22 11
11 25 12 22 64
11 12 25 22 64
11 12 22 25 64

Pseudo-code:

A is the set of elements to sort, n is the number of elements in A (the array starts at index 0)

```
for i ← 0 to n-2 do
   min ← i
   for j ← (i + 1) to n-1 do
      if A[j] < A[min]
         min ← j
   swap A[i] and A[min]
```

Complexity: $O(n^2)$

Insertion Sort:

In abstract terms, every iteration of an insertion sort removes an element from the input data, inserting it at the correct position in the already sorted list, until no elements are left in the input. The choice of which element to remove from the input is arbitrary and can be made using almost any choice algorithm.

Sorting is typically done in-place. The resulting array after k iterations contains the first k entries of the input array and is sorted. In each step, the first remaining entry of the input is removed, inserted into the result at the right position, thus extending the result.

Example:

The following table shows the steps for sorting the sequence 5 7 0 3 4 2 6 1. On the left side the sorted part of the sequence is shown in red. For each iteration, the number of positions the inserted element has moved is shown in brackets. Altogether this amounts to 17 steps.

5	7	0	3	4	2	6	1	(0)
5	7	0	3	4	2	6	1	(0)
0	5	7	3	4	2	6	1	(2)
0	3	5	7	4	2	6	1	(2)
0	3	4	5	7	2	6	1	(2)
0	2	3	4	5	7	6	1	(4)
0	2	3	4	5	6	7	1	(1)
0	1	2	3	4	5	6	7	(6)

Pseudo-code:

```
insertionSort(array A)
   for i = 1 to length[A]-1 do
   begin
      value = A[i]
      j = i-1
      while j >= 0 and A[j] > value do
      begin
         A[j + 1] = A[j]
         j = j-1
      end
      A[j+1] =
   value end
```

Complexity: $O(n^2)$

Report should contain:

1. Machine configuration.
2. Complete Table 1.
3. Plot graphical output [x-axis=number of data, y-axis=Time complexity][Using any software or Language].
4. Plot bar graph [using Microsoft Office Excel].
5. Properly analysis of **table** and **graph**.

[**N.B.** Pick randomly 1000 numbers ranging from 1 to 30,000 then write them in a file. Read them from the file and sort them by the described three sorting algorithms. Then write the sorted numbers in that file.

Example:

Generate 5 numbers randomly ranging 1 to 20.

2 4 1 12 18

Write them in a file. Then sort them by those three sorting algorithms and write the sorted numbers in that file.

File content should like as follows:

Input [5] numbers [Ranging from 1 to 20]

2 4 1 12 18

After sorting by Bubble

sort 1 2 4 12 18

After sorting by Selection sort

1 2 4 12 18

After sorting by Insertion

sort 1 2 4 12 18

]

Table 1

	Required Time		
No. of Data	**Bubble Sort**	**Selection Sort**	**Insertion Sort**
1000			
5000			
10000			
15000			
20000			

Recommended Exercise:

Programming Exercises of Chapter 1: "Introduction" of "Fundamentals of Computer Algorithm", Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran.