

Rajan Gautam

19BCP101

Div. II, CE 19

SOT, PDPU

Pandit Deendayal Petroleum University

School of Technology

Design & Analysis of Algorithm (20CP209P)

B. Tech - Computer Science & Engineering (Sem-IV)

Table of Contents

Lab 1 Assignment: Comparison of Insertion Sort & Bubble Sort	1
AIM: To write a C/C++ Program to implement Insertion Sort & Bubble Sort.....	1
ALGORITHMS:.....	1
1. Insertion Sort Algorithm (Pseudocode)	1
2. Bubble Sort Algorithm (Pseudocode)	1
CODE:.....	1
OUTPUT:	4
COMPARISON:.....	4

Lab 1 Assignment: Comparison of Insertion Sort & Bubble Sort

AIM: To write a C/C++ Program to implement Insertion Sort & Bubble Sort.

ALGORITHMS:

1. Insertion Sort Algorithm (Pseudocode)

```
INSERTION-SORT (A, n)
  for j ← 2 to n
    Do key ← A[j]
      i ← j - 1
      while i > 0 and A[i] > key
        Do A[i + 1] ← A[i]
          i ← i - 1
      A[i + 1] = key
```

2. Bubble Sort Algorithm (Pseudocode)

```
BUBBLE-SORT (A, n)
  For i ← 1 to N do
    For j = 0 to N - 1 do
      If A[j] > A[j+1] then
        Temp = A[j]
        A[j] = A[j+1]
        A[j+1] = temp
```

CODE:

```
1. /* ----- 19BCP101 ----- */
2. /* ----- Rajan Gautam ----- */
```

```
3.
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <time.h>           // For Time Calculation
7.
8. // Function for Insertion Algorithm
9. void insertion_sort(int A[], int n)
10. {
11.     int i, j, key;
12.     for(i = 1; i < n; i++)
13.     {
14.         key = A[i];
15.         j = i - 1;
16.         while (j >= 0 && A[j] > key)
17.         {
18.
19.             A[j+1] = A[j];
20.             j = j - 1;
21.         }
22.         A[j+1] = key;
23.     }
24. }
25.
26. // Function for Bubble Sort Algorithm
27. void bubble_sort(int A[], int n)
28. {
29.     int i, j, temp;
30.     for (i = 0; i < n-1; i++)
31.     {
32.         for (j = 0; j < n-i-1; j++)
33.         {
34.             if (A[j] > A[j+1])
35.             {
36.                 temp=A[j];
37.                 A[j]=A[j+1];
38.                 A[j+1]=temp;
39.             }
40.         }
41.     }
42. }
43.
44.
45. int main()
46. {
47.     printf("<----- Sorting ----->\n\n");
48.
49.     int n = 1000, it = 0;
```

```
50.     double time1[20], time2[20];           // To store the time
       values
51.
52.     printf(" Array      | Bubble(s)      | Insertion(s) \n\n");
53.
54.     while(it++ < 10)
55.     {
56.         long int a[n], b[n];
57.         for (int i = 0; i < n ; i++)
58.         {
59.             // Generating Random Integer Array for each algorithm
60.
61.             a[i] = (rand() % n);
62.             b[i] = (rand() % n);
63.         }
64.
65.
66.         // For time calculation
67.         clock_t start, end;
68.
69.
70.         // For Bubble Sort Algorithm
71.         start = clock();
72.         bubble_sort(a, n);
73.         end = clock();
74.
75.         time1[it] = ((double)(end - start)/CLOCKS_PER_SEC);
76.
77.
78.         // For Insertion Algorithm
79.         start = clock();
80.         insertion_sort(b, n);
81.         end = clock();
82.
83.         time2[it] = ((double)(end - start)/CLOCKS_PER_SEC);
84.
85.         // Printing the table of array size, time taken by Bubble
           Sort and Insertion Algorithm
86.         printf(" %d      |    %f    |    %f\n", n, time1[it],
           time2[it]);
87.
88.
89.         // Incrementing the value of n by 1000
90.         n += 1000;
91.     }
92.     return 0;
93. }
```

OUTPUT:

```

1  /* ----- 19BCP101 ----- */
2  /* ----- Rajan Gautam ----- */
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <time.h>           // For Time
7
8  // Function for Insertion Algorithm
9  void insertion_sort(int A[], int n)
10 {
11     int i, j, key;
12     for(i = 1; i < n; i++)
13     {
14         key = A[i];
15         j = i - 1;
16         while (j >= 0 && A[j] > key)
17         {
18             A[j+1] = A[j];
19             j = j - 1;
20         }
21         A[j+1] = key;
22     }
23 }
24

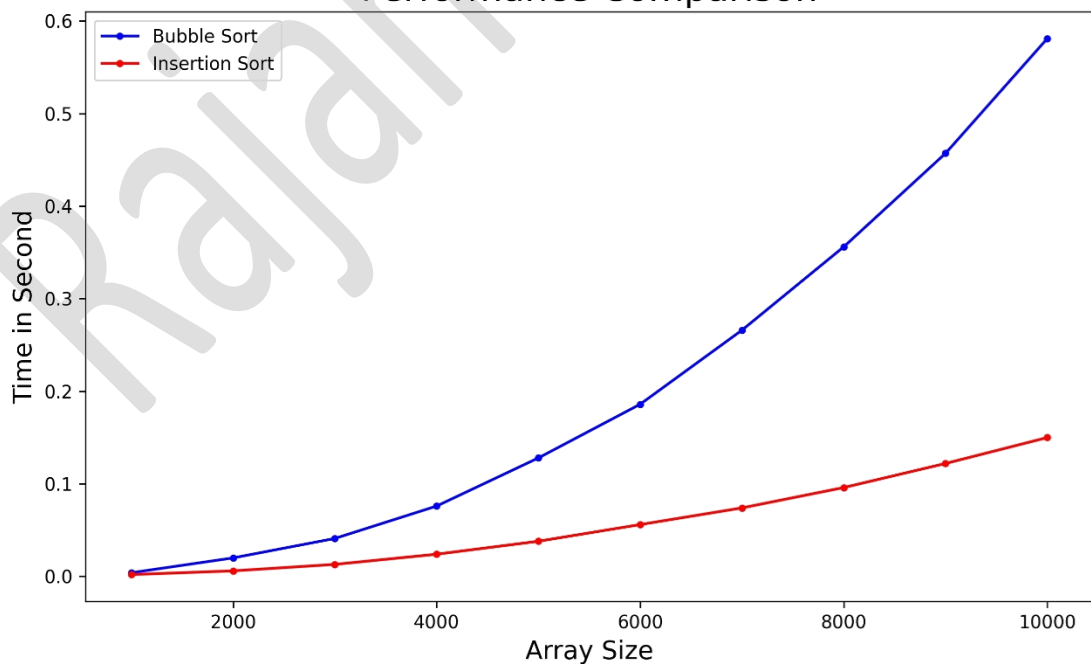
```

Array	Bubble(s)	Insertion(s)
1000	0.004000	0.002000
2000	0.020000	0.006000
3000	0.041000	0.013000
4000	0.076000	0.024000
5000	0.120000	0.038000
6000	0.186000	0.056000
7000	0.266000	0.074000
8000	0.356000	0.096000
9000	0.457000	0.122000
10000	0.581000	0.150000

Process returned 0 (0x0) execution time : 3.006 s
Press any key to continue.

COMPARISON:

Performance Comparison



Link: https://github.com/rgautam320/Design-and-Analysis-of-Algorithm-Lab/tree/master/Lab_1_Sorting