Lab Manual - 02

Course Title: Algorithm Lab

Course Code: CSE 223

Semester: Spring 2020

Topic: Sorting Algorithm

(Bubble and Insertion Sort)

::::Course Teacher::::
Masud Rabbani
Lecturer
Dept. of CSE, DIU

::::Prefect::::

Muhaiminul Islam
Student of DIU
Id: 163-15-8473
Mail: muhaiminul15-8473@diu.edu.bd

Mohiyminul Islam
Student of DIU
Id: 172-15-10187
Mail: muhiyminul15-10187@diu.edu.bd

Hasan Imam Bijoy
Student of DIU
Id: 182-15-11743
Mail: hasan15-11743@diu.edu.bd

Md Mahbubur Rahman
Student of DIU
Id: 182-15-11742
Mail: mahbubur15-11742@diu.edu.bd

Bubble Sort

Introduction: With no obvious definitive **origin** of the name ``**bubble sort**'', researchers investigated its origins by consulting early journal articles as well as professional and pedagogical texts. An early (1959) book on programming [21] devotes a chapter to **sorting**, but uses the term exchange **sorting** rather than **bubble sort**.

An early (1959) book on programming devotes a chapter to sorting, but uses the term *exchange sorting* rather than bubble sort. The same term is used in a 1962 JACM article as well as in the earlier (1961, submitted 1959) JACM article referenced as the definitive source. Iverson uses the name ``bubble sort'' in 1962, this appears to be the first use of the term in print.

Applications: Due to its simplicity, bubble sort is often used to introduce the concept of an algorithm, or a sorting algorithm, to introductory computer science students.
Let's give an example of real life application, in physical education classes or sports classes back in school. Lined up in a random order in front of the teacher, who's put to the task of lining you all up in an ascending order of height. The bubble sort algorithm comes in handy here. In this case every person's height is an element of the list. With every pass that the teacher goes over the students, they slowly start standing in a more orderly fashion till all of them stand according to height.

Complexity:

- Best case: O(n)
- Worst case: $O(n^2)$
- Average case: $O(n^2)$

Outcome:

The algorithm is very simple and easy to implement. After performing it we can have a sorted list / array or dataset. For its simplicity programmers likes to implement it.

Advantages and disadvantages:

Advantages are –

- Easy to understand.
- Easy to implement.
- In-place, no external memory is needed.
- Performs greatly when the array is almost sorted.

Disadvantages are –

- Very expensive, O($\mathbf{n^2}$)O($\mathbf{n^2}$)in worst case and average case.
- It does more element assignments than its counterpart, insertion sort.

Implementation:

Basic Code (using C)

```
#include<stdio.h>
int main()
{
   int n,i,j, temp;
   printf("Enter Array Size : ");
   scanf("%d",&n);
   int a[n];
   printf("Enter Elements of Array : \n");
   for(i=0; i<n; i++)
   {
      scanf("%d",&a[i]);
   }

   for(i=0; i<n; i++)
   {
      for(j=0; j<n-1; j++)
      {
         if(a[j]>a[j+1])
         {
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
         }
      }
   }

   printf("Bubble Sorted list in Ascending  : ");
   for(i=0; i<n; i++)
   {
      printf("%d ",a[i]);
   }
}
```

Practice Problem:

1. You have some of random data, then apply the bubble sort and print the data in descending order

Input: Enter Array Size: 6 Enter Data: 10 5 20 15 40 30	**Input**: Enter Array Size: 5 Enter Data: 50 5 7 49 34
Output: Descending Order: 40 30 20 15 10 5	**Output:** Descending Order: 50 49 34 7 5

2. Suppose, you have some of various data, then apply bubble sort and print the data ascending order and also print the sum of data.

Input: Enter Array Size: 6 Enter Data: 9 1 5 3 7 8	**Input**: Enter Array Size: 3 Enter Data: 50 5 7
Output: Ascending Order: 1 3 5 7 8 9 Sum = 33	**Output:** Ascending Order: 5 7 50 Sum = 62

Insertion Sort

History: It might be hard to find the first person who came up with the idea behind insertion sort, because the simple version is one of the basic way's humans would sort a list of items.

Knuth (TAOCP 3, p. 82) writes that the variant of using binary insertion "was mentioned by John Muchly as early as 1946, in the first published discussion of computer sorting."

Applications:

1. **Sorting short lists.** If you know your lists are never going to contain more than say 25 elements, then insertion sort is an excellent choice.
2. **Sorting "almost sorted" lists.** If you know that no element is more than say 25 locations from its final location in sorted order (the data could have been produced from another program) or that at most say 25 elements are out of order (the remaining elements are already in sorted order), then insertion sort is again an excellent choice.

Complexity:

Best case: O(n)

Worst case: $O(n^2)$

Average case: $O(n^2)$

Outcome: If the list is small then it will be an outstanding working method for sorting.

After performing it we can have a sorted array / list or dataset.

Advantage and disadvantage:

Advantage are -

- The pure simplicity of the algorithm.
- The relative order of items with equal keys does not change.
- The ability to sort a list as it is being received.

- Efficient for small data sets, especially in practice than other quadratic algorithms—i.e. $O(n^2)$.
- It only requires a constant amount of additional memory space—O(1)

Disadvantage are -

- For unsorted/reverse-sorted data, it's slow for large N.
- With n-squared steps required for every n element to be sorted, the insertion sort does not deal well with a huge list.
- Its performance is very poor when the data set is huge.

Implementation:

Basic Code (using C)

```
#include<stdio.h>
int main()
{
   int n,i;
   printf("Enter Array Size : ");
   scanf("%d",&n);
   int a[n];
   printf("Enter Elements of Array : \n");
   for(i=0; i<n; i++)
   {
      scanf("%d",&a[i]);
   }
   int key,j;
   for(j=0; j<n; j++)
   {
      key=a[j];
      i=j-1;
      while(i>=0 && key<a[i])
      {
         a[i+1]=a[i];
         i--;
      }
      a[i+1]=key;
   }
   printf("Insertion Sorted list in Ascending  : ");
   for(i=0; i<n; i++)
   {
      printf("%d ",a[i]);
   }
   return 0;
}
```

- **Practice Problem:**

1. You have some random data your job is that you have to sort them in descending order using Insertion Sort.

Input: Enter Array Size: 6 Enter Data: 10 5 20 15 40 30	**Input**: Enter Array Size: 5 Enter Data: 50 5 7 49 34
Output: Descending Order: 40 30 20 15 10 5	**Output:** Descending Order: 50 49 34 7 5

2. Suppose, you have some of various data, then apply insertion sort & print the data ascending order. Now, you have sorted data, Find the difference between maximum data and minimum data and also print the Difference of max and min.

Input: Enter Array Size: 6 Enter Data: 9 1 5 3 7 8	**Input**: Enter Array Size: 3 Enter Data: 50 5 7
Output: Ascending Order: 1 3 5 7 8 9 Difference btn max-min = 8	**Output:** Ascending Order: 5 7 50 Difference btn max-min = 45

3. Suppose, you know two sorting algorithms one is Bubble sort & second is Insertion sort. Then applying the better algorithm in random data , print sorted data and also find the average of data.

Input: Enter Array Size: 6 Enter Data: 9 1 5 3 7 8	**Input**: Enter Array Size: 3 Enter Data: 50 5 7
Output: Ascending Order: 1 3 5 7 8 9 Avg = 5.50	**Output:** Ascending Order: 5 7 50 Avg = 20.67

“Happy Coding”