Fundamentals of
Data Structures using C

Insertion Sort

B.Bhuvaneswaran, AP (SG) / CSE

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI
ENGINEERING COLLEGE

Insertion Sort

- Insertion sorts works by taking elements from the list one by one and inserting them in their current position into a new sorted list.
- Insertion sort consists of N - 1 passes, where N is the number of elements to be sorted.
- The i^{th} pass of insertion sort will insert the i^{th} element A[i] into its rightful place among A[1], A[2], . . . A[i-1].
- After doing this insertion the records occupying A[1] . . . A[i] are in sorted order.

Insertion Sort

- To understand the insertion sorting method, consider a scenario where an array A containing n elements needs to be sorted.
- Now, each pass of the insertion sorting method will insert the element A[i] into its appropriate position in the previously sorted subarray, i.e., A[1], A[2], …, A[i–1].

Insertion Sort

- Pass 1: A[2] is compared with A[1] and inserted either before or after A[1]. This makes A[1], A[2] a sorted sub array.
- Pass 2: A[3] is compared with both A[1] and A[2] and inserted at an appropriate place. This makes A[1], A[2], A[3] a sorted sub array.
- Pass n–1: A[n] is compared with each element in the sub array A[1], A[2], A[3], … A[n-1] and inserted at an appropriate position.
- This eventually makes the entire array A sorted.

Routine

```
void InsertionSort(int a[], int n)
{
        int i, j, temp;
        for (i = 1; i < n; i++)
        {
                temp = a[i];
                j = i;
                while (j > 0 && a[j - 1] > temp)
                {
                        a[j] = a[j - 1];
                        j = j - 1;
                }
                a[j] = temp;
        }
}
```

Example

Original	20	10	60	40	30	15	Positions Moved
After i = 1	10	20	60	40	30	15	1
After i = 2	10	20	60	40	30	15	0
After i = 3	10	20	40	60	30	15	1
After i = 4	10	20	30	40	60	15	2
After i = 5	10	15	20	30	40	60	4
Sorted Array	10	15	20	30	40	60	

Efficiency of Insertion Sort

- Assume that an array containing n elements is sorted using insertion sort technique.
  - The minimum number of elements that must be scanned = n – 1.
  - For each of the elements the maximum number of shifts possible = n – 1.
- Thus, efficiency of insertion sort = $O(n^2)$

Analysis of Insertion Sort

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

Advantages of Insertion Sort

- It is one of the simplest sorting techniques that is easy to implement.
- It performs well in case of smaller lists.
- It leverages the presence of any existing sort pattern in the list, thus resulting in better efficiency.

Limitations of Insertion Sort

- The efficiency of $O(n^2)$ is not well suited for large sized lists.
- It is expensive because of shifting all following elements by one.

Queries?

Thank You!