INDEX

Ex. No.: 10

Date :

Implementation of Insertion Sort

Aim: To write a C-program to implement Insertion Sort.

Insertion Sort:

Insertion sort is based on the idea that one element from the input elements is consumed in each iteration to find its correct position i.e., the position to which it belongs in a sorted array.

It iterates the input elements by growing the sorted array at each iteration. It compares the current element with the largest value in the sorted array. If the current element is greater, then it leaves the element in its place and moves on to the next element else it finds its correct position in the sorted array and moves it to that position. This is done by shifting all the elements, which are larger than the current element, in the sorted array to one position ahead.

- Efficient for (quite) small data sets, much like other quadratic sorting algorithms
- More efficient in practice than most other simple quadratic (i.e., O(n2)) algorithms such as selection sort or bubble sort
- Adaptive, i.e., efficient for data sets that are already substantially sorted: the time complexity is O(kn) when each element in the input is no more than k places away from its sorted position
- Stable; i.e., does not change the relative order of elements with equal keys
- In-place; i.e., only requires a constant amount O(1) of additional memory space
- Online; i.e., can sort a list as it receives it.

Algorithm

Step 1 − If it is the first element, it is already sorted. return 1;
Step 2 − Pick next element
Step 3 − Compare with all elements in the sorted sub-list
Step 4 − Shift all the elements in the sorted sub-list that is greater than the value to be sorted
Step 5 − Insert the value
Step 6 − Repeat until list is sorted

Program

/*This program demonstrate Insertion Sort */

/* Insertion sort code */

```
#include<stdio.h>
#include<conio.h>

void main(){

   int size, i, j, temp, list[100];

   printf("Enter the size of the list: ");
   scanf("%d", &size);

   printf("Enter %d integer values: ", size);
   for (i = 0; i < size; i++)
      scanf("%d", &list[i]);

   //Insertion sort logic
   for (i = 1; i < size; i++) {
      temp = list[i];
      j = i - 1;
      while ((temp < list[j]) && (j >= 0)) {
         list[j + 1] = list[j];
         j = j - 1;
      }
      list[j + 1] = temp;
   }

   printf("List after Sorting is: ");
   for (i = 0; i < size; i++)
      printf(" %d", list[i]);

  getch();
}
```

Worst and Average Case Time Complexity: O(n^2) comparisons and swaps
Best Case Time Complexity: O(n) comparisons, O(*1*) swaps
Space Complexity: O(n) total, O(*1*) auxiliary
Sorting In Place: Yes
Stable: Yes
Adaptive: Yes
Online: Yes