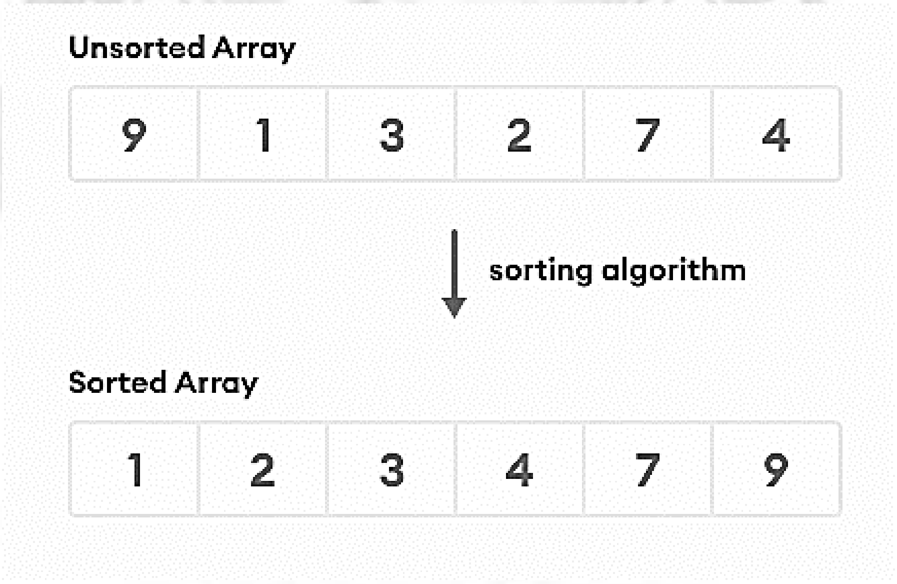
INTRODUCTION TO SORTING

- Sorting is the process of arranging the elements of an array so that they can be placed either in ascending or descending order.
- Efficient sorting is important to optimizing the use of other algorithms that require sorted lists to work correctly and for producing human – read able input

For example,



There are many techniques by using which, sorting can be performed

Sl. No.	Sorting Algorithms	Description
1	Insertion Sort	Insertion sort inserts each element of the array to its proper place. It is a very simple sort method which is used to arrange the deck of cards while playing bridge
2	Quick Sort	Quick sort follows the divide and conquer approach in which the algorithm is breaking down into sub problems, then solving the sub problems, and combining the results back together to solve the original problem.
3	Heap Sort	In the heap sort, Min heap or max heap is maintained from the array elements depending upon the choice and the elements are sorted by deleting the root element of the heap.
4	Merge Sort	Merge sort follows divide and conquer approach in which, the list is first divided into the sets of equal elements and then each half of the list is sorted by using merge sort. The sorted list is combined again to form an elementary sorted array

INSERTION SORT

- Insertion sort is a simple sorting algorithm.
- This sorting method sorts the array by shifting elements one by one.
- It builds the final sorted array one item at a time.
- Insertion sort has one of the simplest implementation.
- This sort is efficient for smaller data sets but it is insufficient for larger lists.
- It has less space complexity like bubble sort.
- It requires single additional memory space.
- Insertion sort does not change the relative order of elements with equal keys because it is stable.

Algorithm:

Step 1 − If the element is the first one, it is already sorted.

Step 2 – Move to next element

Step 3 − Compare the current element with all elements in the sorted array

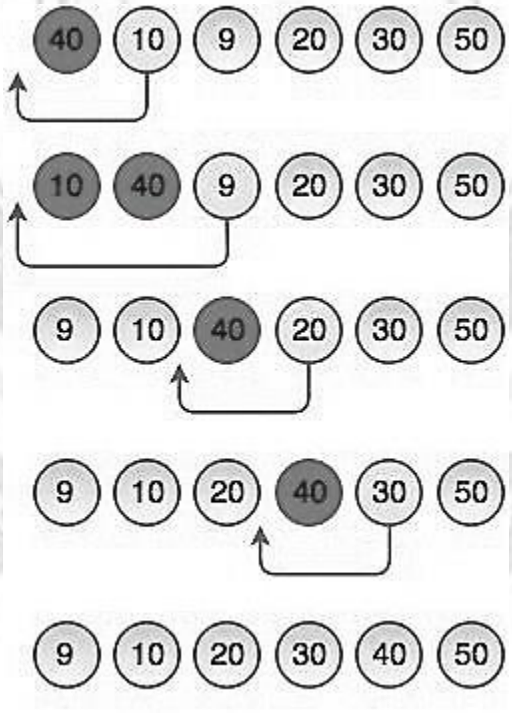
Step 4 – If the element in the sorted array is smaller than the current element, iterate to the next element. Otherwise, shift all the greater element in the array by one position towards right

Step 5 − Insert the value at the correct position

Step 6 − Repeat until the complete list is sorted

Working of Insertion sort Algorithm

Consider an unsorted array of elements 40, 10, 9, 20, 30, 50



- The above steps represents how insertion sort works. Insertion sort works like the way we sort playing cards in our hands. It always starts with the second element as key. The key is compared with the elements ahead of it and is put it in the right place.

- At the first step, 40 has nothing before it. Element 10 is compared to 40 and is inserted before 40. Element 9 is smaller than 40 and 10, so it is inserted before 10 and this operation continues until the array is sorted in ascending order.

Analysis of Insertion Sort:

Time Complexity	
Best	O(n)
Worst	O(n^2)
Average	O(n^2)
Space Complexity	O(1)
Stability	Yes

Applications

- The insertion sort is used when:
  - The array is has a small number of elements
  - There are only a few elements left to be sorted

Example Program 5.1

```
#include <stdio.h>
int main()
{
 int n, array[1000], c, d, t;
 printf("Enter number of elements\n");
 scanf("%d", &n);
 printf("Enter %d integers\n", n);
 for (c = 0; c < n; c++)
 { scanf("%d", &array[c]);
 }
 for (c = 1 ; c <= n - 1; c++)
   { d = c;
```

```
      while ( d > 0 && array[d] < array[d-1])
      {
         t = array[d];
         array[d]    = array[d-1];
         array[d-1] = t;
         d--;
      }
      }
    printf("Sorted list in ascending order:\n");
    for (c = 0; c <= n - 1; c++)
    {
        printf("%d\n", array[c]);
    } return 0;
    }
```

Output

```
Enter the number of elements
5
Enter 5 integers
40
30
20
10
40
Sorted list in ascending order
10
20
30
40
40
```