Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2022-2023

Name – Prerna Sunil Jadhav **SAP ID - 60004220127**

Experiment No – 07

AIM: Implementation of Insertion and Selection Sort.

Insertion Sort

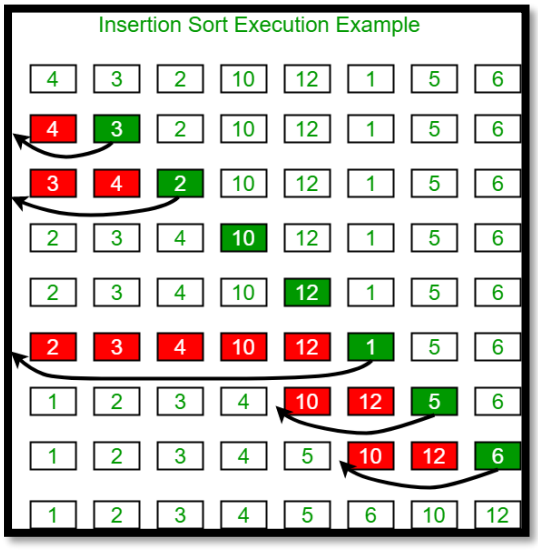
Theory:

- Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.
- Characteristics of Insertion Sort:
  - This algorithm is one of the simplest algorithms with simple implementation
  - Basically, Insertion sort is efficient for small data values
  - Insertion sort is adaptive in nature, i.e., it is appropriate for data sets which are already partially sorted.

Algorithm:

```
1. If the element is the first one, it is already sorted.
2. Move to next element
3. Compare the current element with all elements in the sorted array
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 the right
5. Insert the value at the correct position
6. Repeat until the complete list is sorted
```

Example:



Program:

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

void ins(int a[], int n)
{
    int no, i, j;
    for (i = 1; i < n; i++)
    {
        no = a[i];
        for (j = i - 1; j >= 0; j--)
        {
            if (a[j] > no)
                a[j + 1] = a[j];
            else
                break;
        }
        a[j + 1] = no;
    }
}

void main()
{
    printf("Prerna Sunil Jadhav - 60004220127\n");

    int a[100], n, r;
    printf("Enter the number of elements: \n");
    scanf("%d", &n);

    printf("Enter the elements: \n");
    for (r = 0; r < n; r++)
        scanf("%d", &a[r]);

    ins(a, n);
    printf("The sorted elements are as follows: \n");
    for (r = 0; r < n; r++)
    {
        printf("%d\n", a[r]);
    }

    getch();
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
Enter the number of elements:
5
Enter the elements:
187
3
24
76
89
The sorted elements are as follows:
3
24
76
89
187
```

Selection Sort

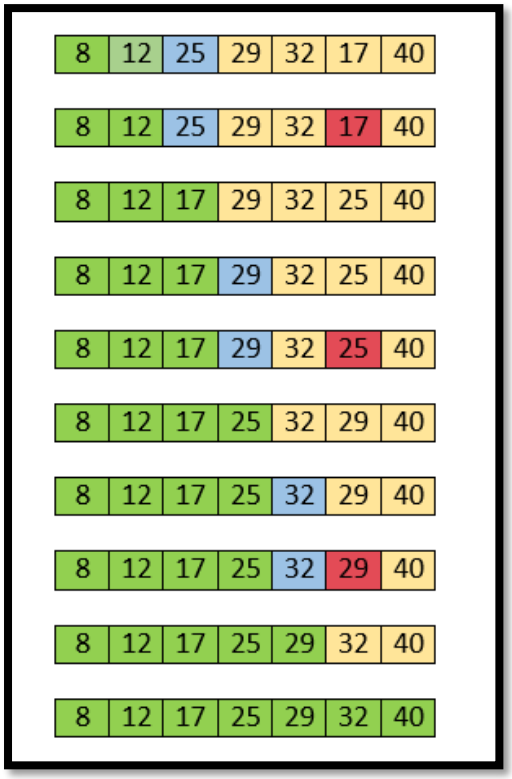
Theory:

- The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning.
- The algorithm maintains two subarrays in a given array.
    - The subarray which already sorted.
    - The remaining subarray was unsorted.
- In every iteration of the selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the beginning of unsorted subarray.
- After every iteration sorted subarray size increase by one and unsorted subarray size decrease by one.
- After N (size of array) iteration we will get sorted array.

Algorithm:

```
1. Set min to the first location
2. Search the minimum element in the array
3. swap the first location with the minimum value in the array
4. assign the second element as min.
5. Repeat the process until we get a sorted array.
```

Example:



Program:

```c
#include <stdio.h>
#include <conio.h>
void select(int a[], int n)
{
    int i, r, j, min, index;
    for (i = 0; i < n - 1; i++)
    {
        min = a[i];
        index = i;
        for (j = i + 1; j < n; j++)
        {
            if (min > a[j])
            {
                index = j;
                min = a[j];
            }
        }

        a[index] = a[i];
        a[i] = min;
    }
    printf("Sorted elements are as follows:\n");
```

```
    for (r = 0; r < n; r++)
    {
        printf("%d\n", a[r]);
    }
}
void main()
{
    printf("Prerna Sunil Jadhav - 60004220127\n");

    int a[100], n, r;
    printf("Enter the number of elements: \n");
    scanf("%d", &n);
    printf("Enter the elements: \n");
    for (r = 0; r < n; r++)
        scanf("%d", &a[r]);
    select(a, n);
    getch();
}
```

OUTPUT:

```
Prerna Sunil Jadhav - 60004220127
Enter the number of elements:
5
Enter the elements:
65
87
9
90
23
Sorted elements are as follows:
9
23
65
87
90
```

Conclusion:

Among both of the sorting algorithm, the insertion sort is fast, efficient, stable while selection sort only works efficiently when the small set of elements is involved or the list is partially previously sorted. The number of comparisons made by selection sort is greater than the movements performed whereas in insertion sort the number of times an element is moved or swapped is greater than the comparisons made.