**Experiment No. 1**

**Aim:** To implement the (a)Selection sort (b)Insertion sort.

**Theory:**

### Insertion sorting works on the premise that first take one element, iterate it through the sorted array. Although it is simple to use, it is not appropriate for large data sets as the time complexity of insertion sort in the average case and worst case is O(n2), where n is the number of items. Insertion sort is less efficient than the other sorting algorithms like heap sort, quick sort, merge sort, etc.

**Complexity of Insertion sort**

* Average Time complexity - O(N2)
* Best case - O(N)
* Worst case - O(N2)
* Space complexity - O(N)

## What is Insertion Sort Algorithm?

#### The simple steps of achieving the insertion sort are listed as follows -

#### Step 1 - If the element is the first element, assume that it is already sorted. Return 1.

#### Step2 - Pick the next element, and store it separately in a key.

#### Step3 - Now, compare the key with all elements in the sorted array.

#### Step 4 - If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.

#### Step 5 - Insert the value.

#### Step 6 - Repeat until the array is sorted.

#### Using a selection sort algorithm, you can sort data simply and efficiently that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list. In selection sort, the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. In this algorithm, the array is divided into two parts, first is sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and unsorted part is the given array. Sorted part is placed at the left, while the unsorted part is placed at the right. In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted.

## Complexity of Selection sort

* Average Time complexity - O(N2)
* Best case - O(N2)
* Worst case - O(N2)
* Space complexity - O(N)

## 

## What is Selection Sort Algorithm?

* Initialize minimum value (**minimum index**) to location 0.
* Traverse the array to find the minimum element in the array.
* While traversing if any element smaller than **minimum index** is found then swap both the values.
* Then, increment **minimum index** to point to the next element.
* Repeat until the array is sorted.

**Selection Sort Code:**

import java.util.\*;

public class Main

{

public static void selectionSort(int []arr){

int n = arr.length;

for(int i = 0; i<n-1; i++){

int min = i;

for (int j=i+1; j<n ;j++ ) {

if(arr[j] < arr[min]){

min = j;

}

}

int temp = arr[i];

arr[i] = arr[min];

arr[min] = temp;

}

return;

}

public static void printArray(int []arr){

for(int i : arr){

System.out.print(i+" ");

}

System.out.println();

return;

}

public static void main(String[] args) {

// int []arr = {2,1,7,3,8};

Scanner sc = new Scanner(System.in);

int []arr = new int[5];

System.out.println("Enter array elements");

for(int i = 0; i<5; i++){

arr[i] = sc.nextInt();

}

System.out.println();

printArray(arr);

System.out.println("selectionSort in ascending order");

selectionSort(arr);

printArray(arr);

}

}

**Output:**

Enter array elements

1

5

4

8

6

1 5 4 8 6

selectionSort in ascending order

1 4 5 6 8

**Insertion Sort Code:**

import java.util.\*;

public class Main

{

public static void insertionSort(int []arr){

int n = arr.length-1;

for(int i = 1; i <= n; i++){

int key = arr[i],j;

j = i-1;

while(j >= 0 && arr[j] < key){

arr[j+1] = arr[j];

j--;

}

arr[j+1] = key;

}

return;

}

public static void printArray(int []arr){

for(int i : arr){

System.out.print(i+" ");

}

System.out.println();

return;

}

public static void main(String[] args) {

// int []arr = {2,1,7,3,8};

Scanner sc = new Scanner(System.in);

int []arr = new int[5];

System.out.println("Enter array elements");

for(int i = 0; i<5; i++){

arr[i] = sc.nextInt();

}

System.out.println();

printArray(arr);

System.out.println("selectionSort in ascending order");

selectionSort(arr);

printArray(arr);

System.out.println("insertionSort in descending order");

insertionSort(arr);

printArray(arr);

}

}

**Output:**

Enter array elements

2

5

3

8

6

2 5 3 8 6

selectionSort in ascending order

2 3 5 6 8

insertionSort in descending order

8 6 5 3 2

**Conclusion:**

The purpose of this experiment was to demonstrate how Selection Sort and Insertion Sort work and how to implement them. In addition, we learned about their time complexity.