## Sorting:-

Sorting is a process of arranging the data/element in some given order, such as ascending or descending order.

There are multiple sorting algorithms or techniques.

1. Bubble Sort.
2. Insertion Sort.
3. Selection Sort.
4. Quick Sort.
5. Two-Way Merge Sort.
6. Heap Sort.

Bubble Sort:-Bubble Sort is the simplest sorting technique or algorithm, in this sorting technique compares two adjacent elements and swaps them, if a lower index value is greater than higher index value.

Example:-

| | | | | |
|---|---|---|---|---|
| -2 | 45 | 0 | 11 | -9 |

Step:-1 i=0

| | | | | | |
|---|---|---|---|---|---|
| j=0 | -2 | 45 | 0 | 11 | -9 |
| j=1 | -2 | 45 | 0 | 11 | -9 |
| j=2 | -2 | 0 | 45 | 11 | -9 |
| j=3 | -2 | 0 | 11 | 45 | -9 |
| | -2 | 0 | 11 | -9 | 45 |

Step:-1 i=1

| | | | | | |
|---|---|---|---|---|---|
| j=0 | -2 | 0 | 11 | -9 | 45 |
| j=1 | -2 | 0 | 11 | -9 | 45 |
| j=2 | -2 | 0 | 11 | -9 | 45 |
| | -2 | 0 | -9 | 11 | 45 |

**Step:-1 i=2**

**j=0**

**j=1**

| | | | | |
|---|---|---|---|---|
| -2 | 0 | -9 | 11 | 45 |
| -2 | 0 | -9 | 11 | 45 |
| -2 | -9 | 0 | 11 | 45 |

**Step:-1 i=2**

**j=0**

| | | | | |
|---|---|---|---|---|
| -2 | -9 | 0 | 11 | 45 |
| -9 | -2 | 0 | 11 | 45 |

**Example:-1 Bubble Sort by using loop.**

```
package com.ds;
public class BubbleSortDemo1
{
int temp;
public void sort(int[] arr)
{
for(int i=0;i<arr.length-1;i++)
{
for(int j=0;j<arr.length-i-1;j++)
{
if(arr[j]>arr[j+1])
{
temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
}
}
}
}
public static void main(String[] args)
{
int[]arr= {-2,45,0,11,-9};
System.out.println("Array Before Sorting");
for(int arr1:arr)
{
System.out.print(arr1+" ");
}
BubbleSortDemo1 bsm=new BubbleSortDemo1();
bsm.sort(arr);
System.out.println();
System.out.println("Array After Sorting");
for(int arr1:arr)
{
```

```
System.out.print(arr1+" ");
}
}
}
Result:- -9 -2 0 11 45
```

Example:-2 Bubble Sort by using recursion.

```
package com.ds;
public class BubbleSortDemo2
{
public static void sort(int arr[],int size){
if (size == 1)
return;
for (int i=0; i<size-1; i++)
if (arr[i] > arr[i+1])
{
int temp = arr[i];
arr[i] = arr[i+1];
arr[i+1] = temp;
}
sort(arr,size-1);
}
public static void main(String[] args){
int arr[] = {45, 67, 89, 31, 63, 0, 21, 12};
System.out.println("Array Before Sorting");
for(int arr1:arr)
{
System.out.print(arr1+" ");
}

sort(arr,arr.length);
System.out.println();
System.out.println("Array After Sorting");
for(int arr1:arr)
{
System.out.print(arr1+" ");
}
}
}
```

Result:-

```
Array Before Sorting
45 67 89 31 63 0 21 12
Array After Sorting
0 12 21 31 45 63 67 89
```

Insertion Sort:-In this sorting technique we compare higher index values with all lower index values.

If the higher index value is smaller than the adjacent lower index value, it will not perform swapping immediately this process will continue until the higher index value got the right place.

Example:-

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 9 | 16 | 5 |

Step:-1 i=1

| | | | | | |
|---|---|---|---|---|---|
| 3 | 2 | 1 | 9 | 16 | 5 |
| 2 | 3 | 1 | 9 | 16 | 5 |

Step:-2 i=2

| | | | | | |
|---|---|---|---|---|---|
| 2 | 3 | 1 | 9 | 16 | 5 |
| 2 | ? | 3 | 9 | 16 | 5 |
| 1 | 2 | 3 | 9 | 16 | 5 |

Step:-3 i=3

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 9 | 16 | 5 |

Step:-4 i=4

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 9 | 16 | 5 |

Step:-5 i=5

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 9 | 16 | 5 |
| 1 | 2 | 3 | 9 | ? | 16 |
| 1 | 2 | 3 | ? | 9 | 16 |
| 1 | 2 | 3 | 5 | 9 | 16 |

Example:-1 Insertion sorting by using looping.

```
package com.ds;

public class InsertionSortDemo
{
public void sort(int[]arr)
{
for(int i=1;i<arr.length;i++)
{
int key=arr[i];
int j=i-1;
while(j>=0 && arr[j]>key)
{
arr[j+1]=arr[j];
j=j-1;
}
arr[j+1]=key;
}
}
public static void main(String[] args)
{
int[]arr= {3,2,1,9,16,5};
System.out.println("Array Before Sorting");
for(int arr1:arr)
{
System.out.print(arr1+" ");
}
BubbleSortDemo1 bs=new BubbleSortDemo1();
bs.sort(arr);
System.out.println();
System.out.println("Array After Sorting");
for(int arr1:arr)
{
System.out.print(arr1+" ");
}
}
}
```

**Result:-**

```
Array Before Sorting
3 2 1 9 16 5
Array After Sorting
1 2 3 5 9 16
```

**Example:-1 Insertion sorting by using recursion.**

```
package com.ds;
public class InsertionSortDemo1
{
void sort(int arr[], int n)
{
if (n <= 1)
{
return;
}
else
{
sort( arr, n-1 );
int last = arr[n-1];
int j = n-2;
while (j >= 0 && arr[j] > last)
{
arr[j+1] = arr[j];
j--;
}
arr[j+1] = last;
}
}

public static void main(String[] args)
{
int[]arr= {3,2,1,9,16,5};
System.out.println("Array Before Sorting");
for(int arr1:arr)
{
System.out.print(arr1+" ");
}
InsertionSortDemo1 isd=new InsertionSortDemo1();
isd.sort(arr,arr.length);
System.out.println();
System.out.println("Array After Sorting");
for(int arr1:arr)
{
System.out.print(arr1+" ");
}
}

}
```

**Result:-**

```
Array Before Sorting
3 2 1 9 16 5
Array After Sorting
1 2 3 5 9 16
```

**Selection Sort:-In this sorting technique we find the smallest element from an unsorted array in each iteration and place that element at the beginning of the unsorted element.**

**Example:-**

| | | | | |
|---|---|---|---|---|
| 15 | 12 | 10 | 20 | 2 |

**Step:-1 i=0**

| | | | | | |
|---|---|---|---|---|---|
| j=1 | 15 | 12 | 10 | 20 | 2 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| j=2 | 15 | 12 | 10 | 20 | 2 |
| j=3 | 15 | 12 | 10 | 20 | 2 |
| j=4 | 15 | 12 | 10 | 20 | 2 |
|  | 2 | 12 | 10 | 20 | 15 |

**Step:-2 i=1**

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| j=2 | 2 | 12 | 10 | 20 | 15 |
| j=3 | 2 | 12 | 10 | 20 | 15 |
| J=4 | 2 | 12 | 10 | 20 | 15 |
|  | 2 | 10 | 12 | 20 | 15 |

**Step:-3 i=2**

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| j=3 | 2 | 10 | 12 | 20 | 15 |
| j=4 | 2 | 10 | 12 | 20 | 15 |
|  | 2 | 10 | 12 | 20 | 15 |

**Step:-4 i=3**

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| j=4 | 2 | 10 | 12 | 20 | 15 |
|  | 2 | 10 | 12 | 15 | 20 |

**Example:-1 Selection Sort by using looping.**

```
package com.ds;

public class SelectionSortDemo1
{
public void sort(int[]arr)
{
int temp=0;
int min=0;
for(int i=0;i<arr.length-1;i++)
{
min=i;
for(int j=i+1;j<arr.length;j++)
{
if(arr[j]<arr[min])
{
min=j;
}

}
temp=arr[i];
arr[i]=arr[min];
arr[min]=temp;
}
}
public static void main(String[] args)
{
int arr[] = {15,12,10,20,2};
System.out.println("Array Before Sorting");
for(int arr1:arr)
{
System.out.print(arr1+" ");
}
SelectionSortDemo1 ssd=new SelectionSortDemo1();
ssd.sort(arr);
System.out.println();
System.out.println("Array After Sorting");
for(int arr1:arr)
{
System.out.print(arr1+" ");
}
}
}
```

**Result:-**

```
Array Before Sorting
15 12 10 20 2
Array After Sorting
2 10 12 15 20
```

**Example:-2 Selection sort by using recursion.**