★ SORTING

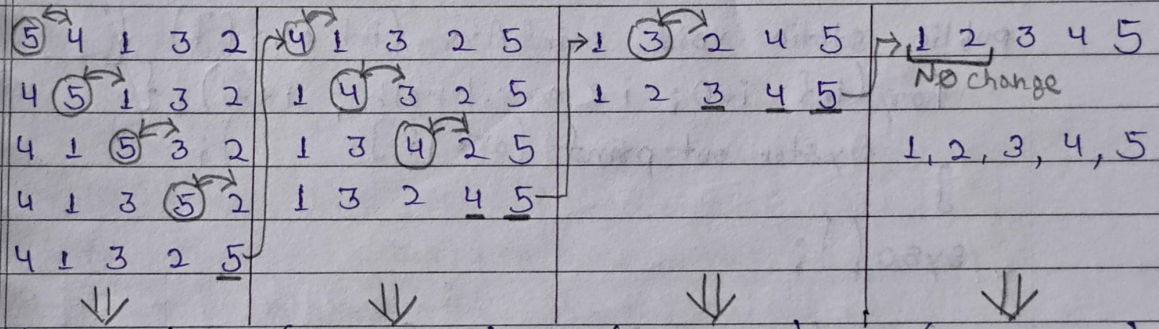
Page No. 60
Date: / /

* Sorting is arranging anything in order.
* It can be increasing or decreasing.

* Basic Algorithms :
  1. Bubble Sort
  2. Selection Sort
  3. Insertion sort
  4. Counting sort

1. **Bubble Sort :** large elements come to the end of an array by swapping with adjacent elements.

Ex: {5, 4, 1, 3, 2} n = 5



$\leq [(n-2)]$ $< [(n-1)]$

⇒ for (0 to n-2 turns)
  for (j = 0 to n-2-turns)

→ turn 0 j=0 to (5-2-0) ⇒ 3
→ turn 1 j=0 to (5-2-1) ⇒ 2

* Bubble Sort - Code $O(n^2)$

```
public static void bubble_sort (int arr[]) {
    for (int turn=0; turn < arr.length-1; turn++) {
        for (int j=0; j < arr.length-1-turn; j++) {
            int t if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
public static void printArr (int arr[]) {
    for (int i=0; i<arr.length; i++) {
        System.out.print (arr[i]+" ");
    }
    sysO();
}
psvm (String[] args) {
    int arr[] = {5, 4, 1, 3, 2};
    bubble_sort (arr);
    printArr (arr);
}
```

Output: 1 2 3 4 5

for (j = i+1 to n)
unsorted array का start.

2. Selection Sort : pick the smallest (from unsorted), put it at the beginning.

$O(n^2)$

Ex: [5, 4, (1), 3, 2] ⇒ n = 5

1, [5, 4, 3, (2)] * 0th indexing ⇒ (n-2)
1, 2, [5, 4, (3)]
1, 2, 3, [5, (4)] * 1st indexing ⇒ (n-1)
1, 2, 3, 4, [5]

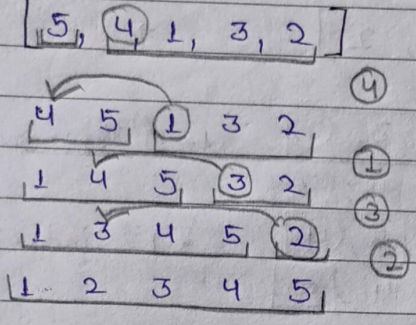
⇒
```
public static void printArr (int arr[]) {
    for (i=0; i < arr.length ; i++) {
        System.out.print (arr[i] + " ");
    }
    Syso ();
}
public static void selection_sort (int arr[]) {
    for (i=0; i < arr.length -1 ; i++) {
        int minPos = i;
        for (j = i+1; j < arr.length ; j++) {
            if (arr[minPos] > arr[j]) {
                minPos = j;
            }
        }
        int temp = arr[minPos];
        arr[minPos] = arr[i];
        arr[i] = temp;
    }
}
```

> change to <.
// If we have to print in decreasing order.

$O(n^2)$

3. **Insertion Sort :** Pick an element (from unsorted part) & place in the right pos in sorted part.

Ex:



$\Rightarrow n = 5$

$(n-1)$.

* Unsorted Array में से Sorted Array में एक-एक element insert करते जायेंगे।

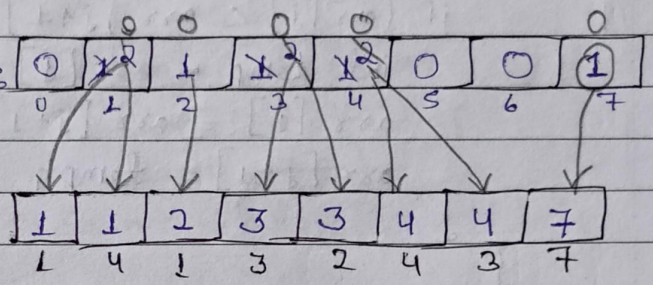
⇒
```
public static void insertion_sort (int arr[]) {
    for (int i = 1; i < arr.length; i++) {
        int curr = arr[i];
        int prev = i - 1;
// finding the right position to do insertion.
        while (prev >= 0 && arr[prev] > curr) {
            arr[prev + 1] = arr[prev];
            prev--;
        }
// insertion
        arr[prev + 1] = curr;
    }
}
```

O(n + range)

4. Counting Sort : This is only used for positive numbers. Counting sort is only applied on short range of numbers.

Ex: Unsorted Array : | 1 | 4 | 1 | 3 | 2 | 4 | 3 | 7 |

(counts frequency) Count ~~sorted~~ Array :



∴ Sorted Array : 1, 1, 2, 3, 3, 4, 4, 7

⇒
```
public static void counting_sort (int arr[]) {
    int largest = Integer.MIN_VALUE; // -∞
    for (int i=0; i<arr.length; i++) {
        largest = Math.max (largest, arr[i]);
    }
    int count[] = new int [largest+1]; // start from 0.
    for (int i=0; i<arr.length; i++) {
        count [arr[i]] ++;
    }
    int j=0;
    for (int i=0; i<count.length; i++) {
        while (count [i] > 0) {
            arr[j] = i;
            j++;
            count [i] --;
        }
    }
}
```

⇒ [3, 6, 2, 1, 8, 7, 4, 5, 3, 1]

Ques. Write all sorting program in descending order.

Bubble:
```
public static void Bubble_sort (int arr[]) {
    for (int turn=0; turn < arr.length-1; turn++) {
        for (int j= 0; j < arr.length-1- turn; j++) {
            if (arr[j] < arr [j+1] ) {
                int temp = arr [j];
                arr [j] = arr [j+1];
                arr [j+1] = temp;
            }
        }
    }
}
```

Insertion:
```
public static void Insertion_sort (int arr[]) {
    for (int i=1; i < arr.length; i++) {
        int curr = arr [i];
        int prev = i-1;
        while (prev >=0 && arr[prev] < curr) {
            arr[prev+1] = arr[prev];
            prev--;
        }
        arr [prev+1] = curr;
    }
}
```

Selection:
```
public static void Selection_sort (int arr[]) {
    for (int i=0; i<arr.length; i++) {
        int minPos = i;
        for (int j=i+1; j<arr.length; j++) {
            if (arr [minPos] < arr[j]) {
                minPos =j;
            }
        }
    }
```

```
        int temp = arr[minPos];
        arr[minPos] = arr[i];
        arr[i] = temp;
    }
}
```

Counting:

```
public static void Counting_sort(int arr[]) {
    int largest = Integer.MIN_VALUE;
    for(int i=0; i<arr.length; i++) {
        largest = Math.max(largest, arr[i]);
    }
    int count[] = new int[largest+1];
    for(int i=0; i<arr.length; i++) {
        count[arr[i]]++;
    }
    int j=0;
    for(int i=count.length-1; i>=0; i--) {
        while(count[i]>0) {
            arr[j] = i;
            j++;
            count[i]--;
        }
    }
}
```