# Module 02

topic :- xyz

→ Theory

→ Problems

- Question
- Logic building
- Dry run
- Coding

⇒ Sorting (arranging the elements in a particular order)

(algorithms)

- Bubble sort
- Selection sort
- Insertion sort

} $O(N^2)$

where N is the size of array

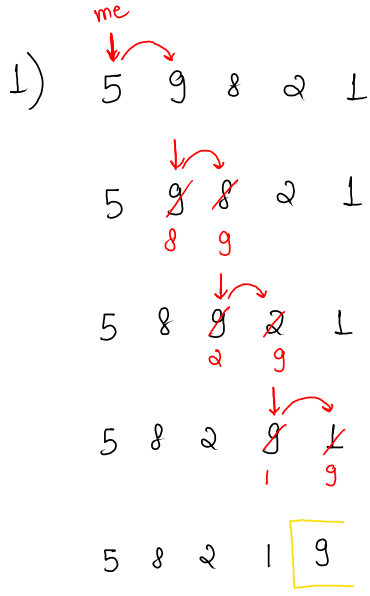
# ⇒ Bubble sort

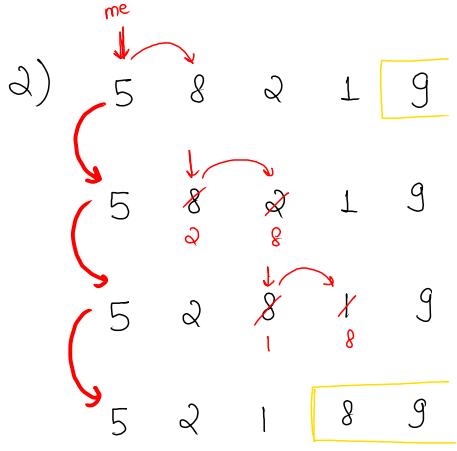
↳ pick the largest element and place it to the rightmost side of unsorted part of array

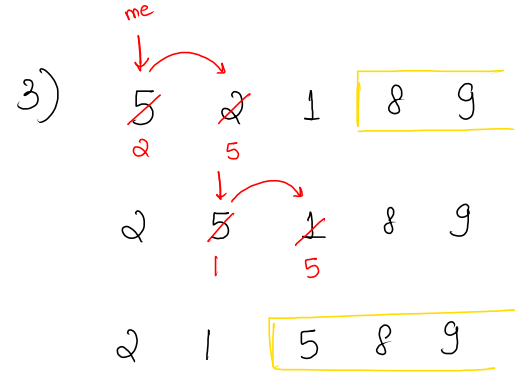
ex:- arr =

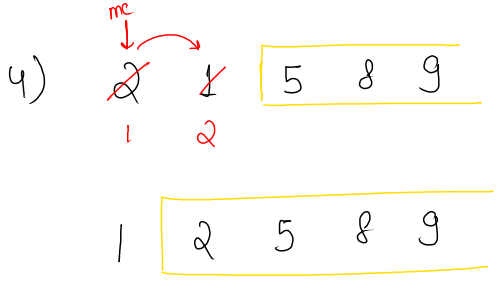
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 9 | 8 | 2 | 1 |

(n = 5)

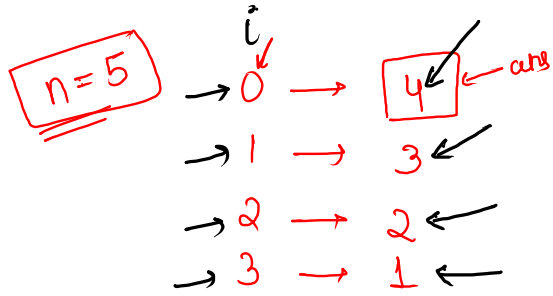


Note:-




relation : n-1-i

me → j
other → j+1

Psudo code

```
int n = arr.length;
for (int i=0; i < n-1 ; i++) {
    for (int j=0; j < n-1-i ; j++) {
        if ( arr[j] > arr[j+1]) {
            swap(arr, j, j+1);
        }
    }
}
```

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    bubbleSort(arr, n);
}
// main logic
public static void bubbleSort(int[] arr, int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - 1 - i; j++) {
            if ( arr[j] > arr[j + 1] ) {
                swap(arr, j, j + 1);
            }
        }
    }

    // printing
    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}

public static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

$T.C = O(N^2)$

$S.C = O(1)$

# ⇒ Insertion sort

( pick the first element of unsorted array and place it at the correct position )

arr =

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 9 | 8 | 2 | 1 |

( n = 5 )

a) 5 9 8 2 1

1) 5 9 8 2 1

5 9 8 2 1

2) 5 9 8 2 1

8 9

5 8 9 2 1

3) 5 8 9 2 1

2 9

5 8 2 9 1

2 8

5 2 8 9 1

2 5

2 5 8 9 1

5) 2 5 8 9 1

1 9

2 5 8 1 9

1 8

2 5 1 8 9

1 5

2 1 5 8 9

1 2

1 2 5 8 9

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    insertionSort(arr, n);
}
// main logic
public static void insertionSort(int[] arr, int n) {

    for (int i = 1; i < n; i++) {   // (n - 1)
        for (int j = i; j >= 1; j--) {
            if ( arr[j] < arr[j - 1] ) {
                swap(arr, j, j - 1);
            } else {
                break;
            }
        }
    }

    // print
    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }

}

public static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```