⇒ Module 2

- Theory
- Problems
  - Explain/Understand Ques
  - Logic / Idea / Approach
  - ☆☆ Dry Run
  - Coding

⇒ **Sorting** (arranging element in a perticular order)

(algorithms)

- → Bubble sorting
- → Selection sorting
- → Insertion sorting

} $O(N^2)$ where N is size of array

⇒ Bubble sort

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

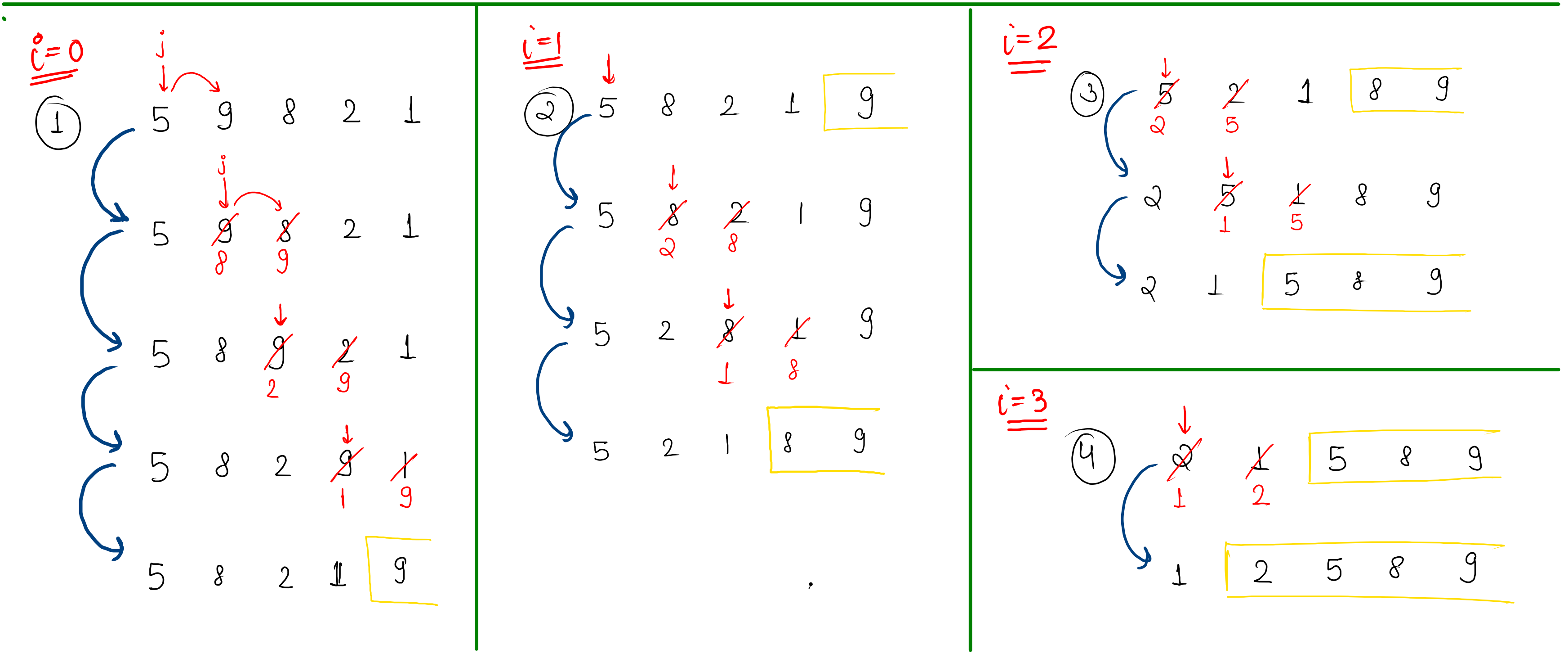
dryrun

arr =

0	1	2	3	4
5	9	8	2	1

Note :- j = myself
j+1 = other

arr[j] > arr[j+1]
swap (j, j+1)



psudo code

		j loop	
i = 0	→	4	times
i = 1	→	3	"
i = 2	→	2	"
i = 3	→	1	"

$n=5$

j loop will be running how many times

=> $n-1-i$

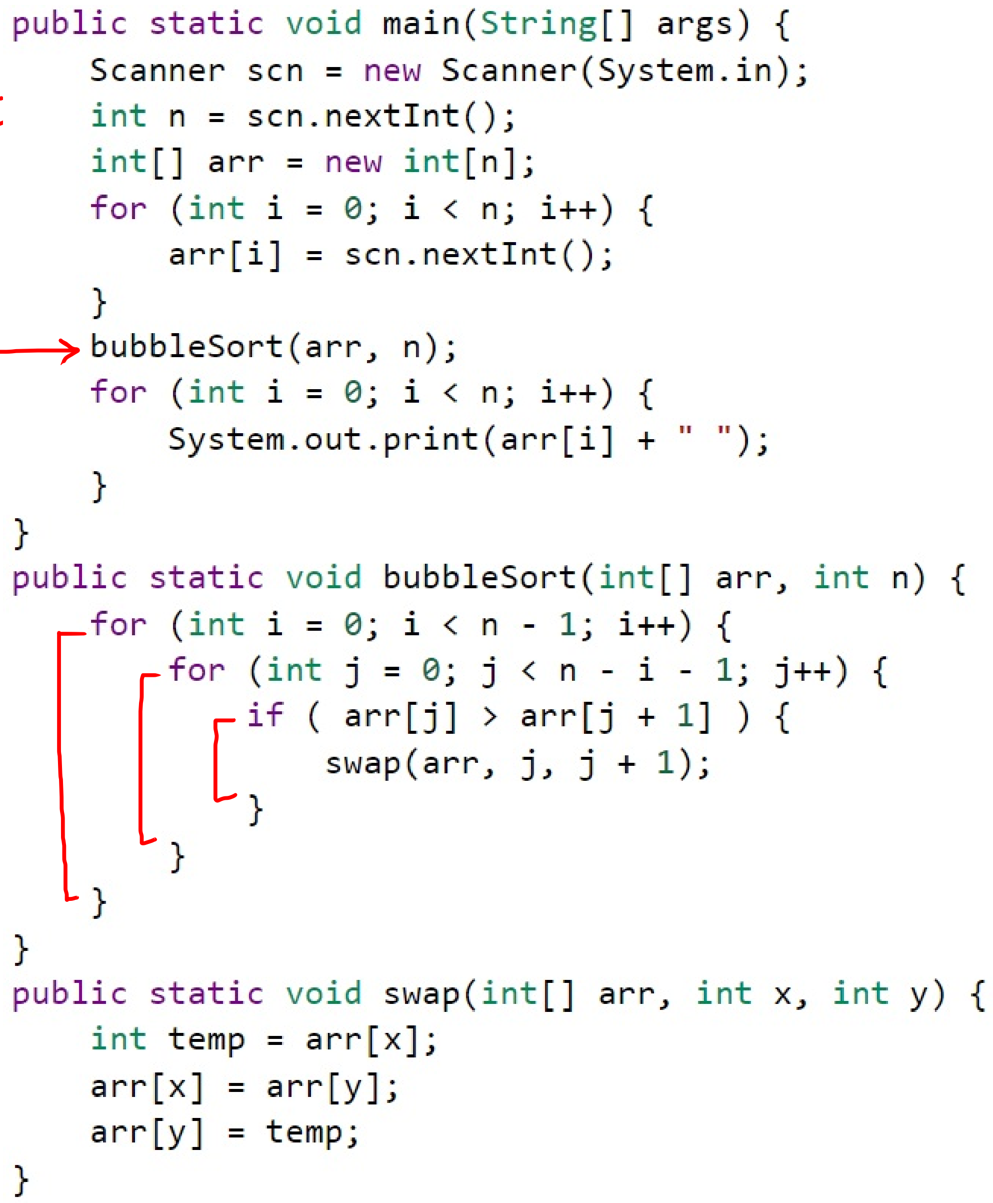
```
for(int i=0; i<n-1 ; i++ ){
    for(int j=0; j<n-1-i ; j++ ){
        if( arr[j] > arr[j+1] ){
            swap (j, j+1);
        }
    }
}
```

Note:-

```
fun( ) {
        int  x = 6;   ← I
        int  y = 5;   ← II
        int  temp = x;
        x = y;
        y = temp;
}
```

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

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);
    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}
public static void bubbleSort(int[] arr, int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if ( arr[j] > arr[j + 1] ) {
                swap(arr, j, j + 1);
            }
        }
    }
}
public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```

T.C = $O(N^2)$ where N is the size of array

Permutation

⇒ Insertion sort

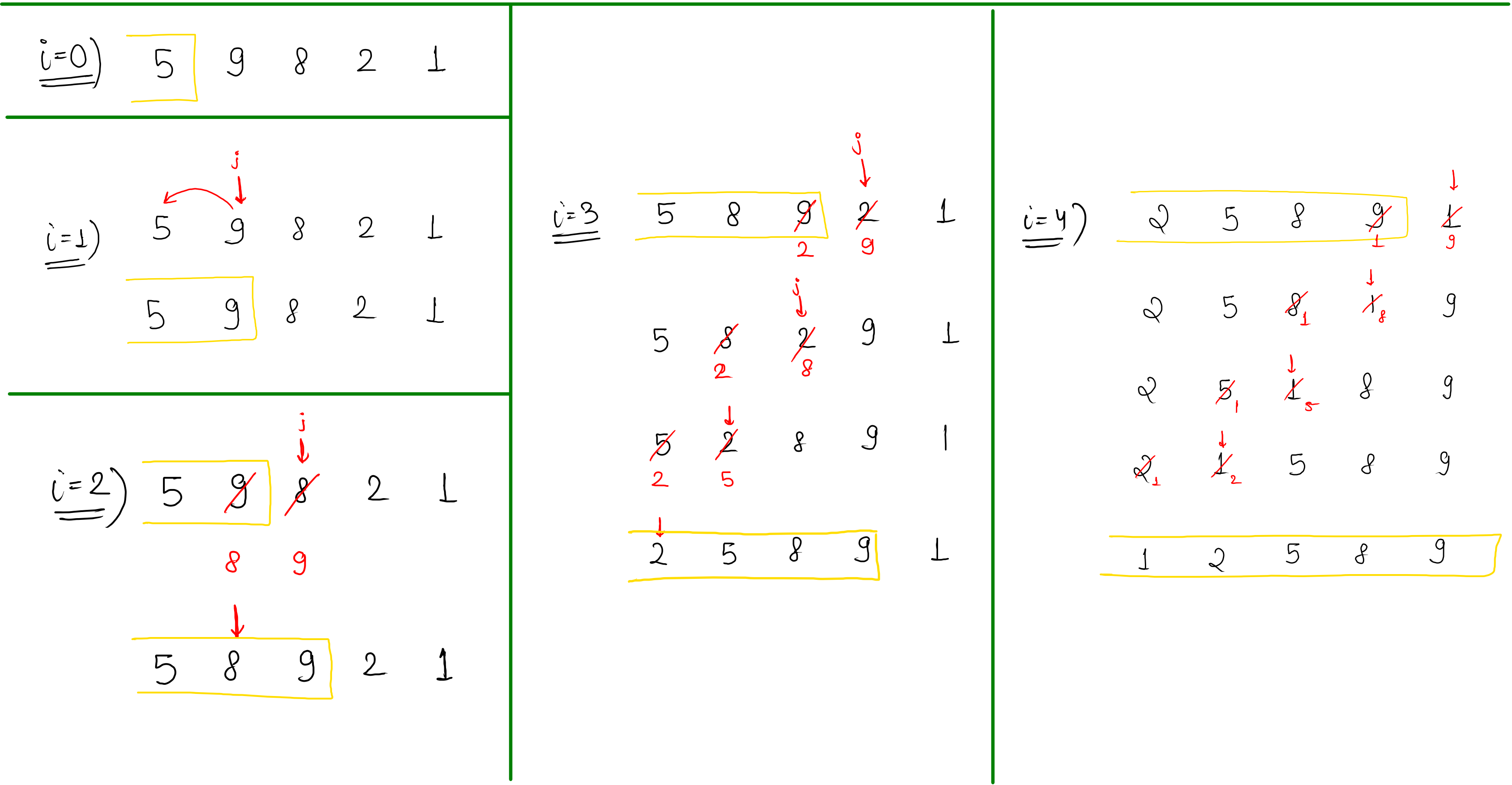
(pick the first element of unsorted array and insert it at the correct position)

arr =

0	1	2	3	4
5	9	8	2	1

j = myself
j-1 = other

arr[j] < arr[j-1]
then swap(j, j-1)



i=0) 5 9 8 2 1

i=1) 5 9 8 2 1
5 9 8 2 1

i=2) 5 ~~9~~ ~~8~~ 2 1
8 9
5 8 9 2 1

i=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

i=4) 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