18/11/24

1) Bubble sort :

```
public class BubbleSort {
  public static void bubbleSort(int[] arr) {
    int n = arr.length;
    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] and arr[j+1]
          int temp = arr[j];
          arr[j] = arr[j + 1];
          arr[j + 1] = temp;
        }
      }
    }
  }
```

Output:

Input: arr[] = [4, 1, 3, 9, 7]
Output: [1, 3, 4, 7, 9]
Input: arr[] = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
Output: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Time complexity : O(N^2)
Space complexity: O(1)

2) QuickSort

```
public class QuickSort {
  public static void quickSort(int[] arr, int low, int high) {
    if (low < high) {
      int pivotIndex = partition(arr, low, high);
      quickSort(arr, low, pivotIndex - 1);
      quickSort(arr, pivotIndex + 1, high);
    }
  }

  public static int partition(int[] arr, int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
      if (arr[j] <= pivot) {
        i++;
        int temp = arr[i];
```

```
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        int temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;
        return i + 1;
    }
```

Output:

Input:

```
4 1 3 9 7
```

Output:

```
1 3 4 7 9
```

Time complexity : O(n log n)
Space complexity: O(log n)