Question-1

What is the Big-O Time Complexity Analysis of BubbleSort? LC

- Process
  - Step 1: Please use Loop Analysis method to analyze the function

    void bubbleSort(int arr[])

    Please explain your answer.

```
class BubbleSort
{
    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+1] and arr[i]
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
    }

    /* Prints the array */
    void printArray(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    // Driver method to test above
    public static void main(String args[])
    {
        BubbleSort ob = new BubbleSort();
        int arr[] = {64, 34, 25, 12, 22, 11, 90};
        ob.bubbleSort(arr);
        System.out.println("Sorted array");
        ob.printArray(arr);
    }
}
```

The outer loop of the given code iterates from i = 0 to n-2. This means that n-1 iterations are performed (where n is the number of elements in the array).

The inner loop iterates from j = 0 to n-i-2. The first iteration of the outer loop performs n-1 iterations. The second iteration performs n-2 iterations.

This pattern continues until the last iteration of the outer loop, where one iteration is executed.

First loop:

```
                    for (int i = 0; i < n-1; i++) => O(n) * O(n) = O(n^2)
                                  for (int j = 0; j < n-i-1; j++) => O(n)
```

Second loop:

```
                    for (int i=0; i<n; ++i) => O(n)
```

Big(O) = $O(n^2) + O(n) = O(n^2)$