**Array: [4, 77, 98, 30, 20, 50, 77, 22, 49, 2]**

* **Bubble Sort (Non-Recursive)**
* This approach uses an embedded for-loop to manually compare every element with the other, making swaps where necessary.

**Example**

[4, 77] => No swap.

[77, 98] => No swap.

[98, 30] => Swap. (resultant array) -> **[4, 77, 30, 98, 20, 50, 77, 22, 49, 2]**

[98, 20] => Swamp. (resultant array) -> **[4, 77, 30, 20, 98, 50, 77, 22, 49, 2]**

This will continue until **[4, 77, 30, 20, 50, 77, 22, 49, 2, 98].**

* **2nd Pass:** This will repeat until no more swaps are necessary. The program will end when the algorithm runs without swapping any value, ensuring that there are no redundant passes.
* **Bubble Sort (Recursive)** 
  + **Same logic applies here,** but instead of manually swapping values through loops, the function will recursively call itself to process the array step by step.

**Process**

* **Base Case:** if no swaps are needed in a pass/only one element is in the array, return (sorted array)
* **Recursive Cases:** 
  + Process the first pass by comparing the nearest elements. **Once a pass is completed,** reduce the examined array size by **ignoring the last element (which is already correctly placed).**
  + **Call the method on the rest of the array.**

**Example**

**[4, 77, 30, 98, 20, 50, 77, 22, 49, 2] - 1st Recursive Call**

* [98, 30] => Swap => **[4, 77, 30, 98, 20, 50, 77, 22, 49, 2]**
* [98, 20] => Swap => **[4, 77, 30, 20, 98, 50, 77, 22, 49, 2]**
* [98, 50] => Swap => **[4, 77, 30, 20, 50, 98, 77, 22, 49, 2]**
* [98, 77] **=>** Swap **=> [4, 77, 30, 20, 50, 77, 98, 22, 49, 2]**
* [98, 22] **=>** Swap **=> [4, 77, 30, 20, 50, 77, 22, 98, 49, 2]**
* [98, 49] **=>** Swap **=> [4, 77, 30, 20, 50, 77, 22, 49, 98, 2]**
* [98, 2] => Swap => **[4, 77, 30, 20, 50, 77, 22, 49, 2, 98]**

**[4, 77, 30, 20, 50, 77, 22, 49, 2, 98] - 2nd Recursive Call**

* This will compare 77 with every value to the right of it **(after comparing 4 and 77)** and swap with every single one of them except 77.
* This will result in **[4, 30, 20, 50, 77, 22, 49, 2, 77, 98]**

**Further calls will proceed, until each element is positioned correctly.**

* **Selection Sort (Non-Recursive)**
  + **Array: [4, 77, 98, 30, 20, 50, 77, 22, 49, 2]**
* 2 => Swap for 4 => **[2, 77, 98, 30, 20, 50, 77, 22, 49, 4]**
* 4 => Swap for 77 => **[2, 4, 98, 30, 20, 50, 77, 22, 49, 77]**
* 20 => Swap for 98 => **[2, 4, 20, 30, 98, 50, 77, 22, 49, 77]**
* 22 => Swap for 30 => **[2, 4, 20, 22, 98, 50, 77, 30, 49, 77]**
* 30 => Swap for 98 => **[2, 4, 20, 22, 30, 50, 77, 98, 49, 77]**
* 49 => Swap for 50 => **[2, 4, 20, 22, 30, 49, 77, 98, 50, 77]**
* 50 => Swap for 77 => **[2, 4, 20, 22, 30, 49, 50, 77, 98, 77]**
* 77 => Swap for 98 => **[2, 4, 20, 22, 30, 49, 50, 77, 77, 98]**

**Final Sorted Array: [2, 4, 20, 22, 30, 49, 50, 77, 77, 98]**

* **Insertion Sort (Non-Recursive)**
  + **Array: [4, 77, 98, 30, 20, 50, 77, 22, 49, 2]**
* Compare 77 with 4 (No Change)
* Compare 98 with 77 (No Change)
* Compare 30 with 98 and 77 - **Shift 98 and 77 to the right** and **Insert 30 at Index 1.** 
  + **[4, 30, 77, 98, 20, 50, 77, 22, 49, 2]**
* Compare 20 with 98, 77, 30, and 4 - **Shift 98, 77, and 30 to the right** and **Insert 20 at Index 1.** 
  + **[4, 20, 30, 77, 98, 50, 77, 22, 49, 2]**
* Compare 50 with 98, 77, and 30 - **Shift 98 and 77 to the right and Insert 50 at Index 3.** 
  + **[4, 20, 30, 50, 77, 98, 77, 22, 49, 2]**
* Compare 77 **(currently at index 6)** with 98 - **Shift 98 to the right and Insert 77 at Index 5.** 
  + **[4, 20, 30, 50, 77, 77, 98, 22, 49, 2]**
* Compare 22 with all the values before it - **Shift 30, 50, 77, 77, 98 to the right and Insert 22 at Index 2.** 
  + **[4, 20, 22, 30, 50, 77, 77, 98, 49, 2]**
* Compare 49 with 98, 77, 77, and 50 - **Shift 50, 77, 77, and 98 to the right and Insert 49 at Index 4.** 
  + **[4, 20, 22, 30, 49, 50, 77, 77, 98, 2]**
* **Compare 2 with all the values before it - Shift all the elements to the right and Insert 2 at Index 0.** 
  + **[2, 4, 20, 22, 30, 49, 50, 77, 77, 98]**
* **Merge Sort (Recursive)**
  + **1) First, we split into two halves:**
    - **[4, 77, 98, 30, 20] and [50, 77, 22, 49, 2]**
  + **2) Then you split the first half.** 
    - **[4, 77] and [98, 30, 20]**
  + **3) Then you split each of these sub-halves.** 
    - **[4, 77] => [4] and [77]**
    - **[98, 30, 20] => 98 and [30, 20] => [30] and [20]**
  + **4) Now the first half is [4, 77, 98, 30, 20]**
  + **5) Following the same steps with the second half => [50, 77, 98, 30, 20]**

**Merging**

* **Merge 4 and 77 => [4, 77]**
* **Merge 30 and 20 => [20, 30]**
* **Merge 98 and [20, 30] => [20, 30, 98]**
* **Merge [4, 77] and [20, 30, 98] => [4, 20, 30, 77, 98]**
* **Merge 50 and 77 => [50, 77]**
* **Merge 49 and 2 => [2, 49]**
* **Merge 22 and [2, 49] => [2, 22, 49]**
* **Merge [50, 77] and [2, 22, 49] => [2, 22, 49, 50, 77]**
* **Now, merge both arrays: Array 1[4, 20, 30, 77, 98] and Array 2[2, 22, 49, 50, 77]**
  + Compare each element **one by one and merge.**
  + **2 < 4 => Add 2 | 4 < 22 => Add 4 | 20 < 22 => Add 20 | 22 < 30 => Add 22 | 30 < 49 => Add 30.. and**
* **Final Array: [2, 4, 20, 22, 30, 49, 50, 77, 77, 98]**
* **Quick Sort (Recursive)**
  + **1st step is to choose a pivot.**
  + Then, rearrange the array **so that 1) all elements less than the pivot come before it** and **2) all elements greater than or equal to the pivot come after it.**
  + **Ex: Pivot = 20**
    - **Partion the array around the pivot.** 
      * [4, 2] - elements < 20
      * [77, 98, 30, 50, 77, 22, 49] - elements >= 20
    - **Recursively Sort the left subarray.**
      * [4, 2] - **4 is the middle element (pivot)**
      * [2, 4] - sorted left subarray.
    - **Recursively Sort the right subarray.**
      * [77, 98, 30, 50, 77, 22, 49] - **50 is the pivot.**
      * Partition: Left: [30, 22, 49], Right: [77, 98, 77]
      * Continue sorting until both sides are sorted.
    - **Merge sorted left and right subarrays**
      * Combine the sorted subarrays to make the final sorted array ( **[2, 4, 20, 22, 30, 49, 50, 77, 77, 98]** )