Tanishq Gadkari
011859770

Bubble Sort:

While there are swaps being done
Set swap to false
Iterate through the array
If swaps happens (swaps when arr[n] > arr[n+1])
Set swap to true

Selection Sort:

For each outer loop
Find the smallest element's index within the array
Swap with the respective index of the outer loop

Insertion Sort:

Assume the first element is sorted
Loop through n - 1 times (since first element is sorted)
While there exists a element A[n] < A[n-1]
Swap A[n] and a[n - 1]

Merge Sort:

Divide the array evenly into it's subpart by exploiting the mid of sub each array
Mid = (low + high) / 2;
Call low , mid
Call mid + 1, high
With a base case of if low >= high return
Finally merge

The merge will consist of comparisons between two arrays sliced around the mid value
leftArr = [low, . . ., mid]
rightArr = [mid + 1, . . ., high]

Compare both arrays with each other
If leftArr[i] < rightArr[j]
resultArr[k] = leftArr[i], increment i, k
Else
resultArr[k] = rightArr[j]. Increment j, k

If any leftovers from either left or right array them in the result arr after
Loop through left arr
resultArr[k] = leftArr[i], increment i, k
Loop through right arr
resultArr[k] = rightArr[j], increment j, k

Quick Sort:

The division here will be based on the biases of our pivot and where it ends up, it won't be even like it was for Merge Sort.
I let my pivot be the last element

Partition = partition(arr, low, high)
Call low, partition - 1
Call partition + 1, high

Here the partition function plays a crucial role
Pivot = arr[high]
Partitioning index = low

Loop through the array
    If the element is less than pivot
        Swap it with the partitioning index, increment partitioning index

Once exited the loop means that we travest to the very last element which is our pivot
By this point we should swap our pivot with the partitioning index to put it its rightful place

Return partitioning index to determine how the next division of array will take place

Resources

https://www.programiz.com/dsa/bubble-sort
Only used it for Merge and Quick sort understanding, all the n^2 sorts were original