MergeSort’s Merge Algorithm

merge(data[], leftPosition, midPosition, rightPosition){

// Calculate size of each half of the section

// Add one to include middle element (remember these are position values)

leftLength = (midPosition – leftPosition) + 1)

// Don’t add one for right length as we don’t want to include the midPosition element here as well

rightLength = (rightPosition – midPosition)

// Create temp arrays, one for each side of the section

leftArray = new array[leftLength]

rightArray = new Array[rightLength]

// Fill temp arrays with the data from their side of the section

for(i = 0, i < leftArray length, i++)

leftArray[i] = data[leftPosition + i]

for(i = 0, i < rightArray length, i++)

rightArray[i] = data[(midPosition + i)+1]

// Merge the temp arrays in order

// Track where we are in temp data

leftPos = 0

rightPos = 0

// Track where we are inserting into in main array

mergedArrayPos = leftPosition

// While still within bounds of BOTH temp arrays

while leftPos < leftLength && rightPos < rightLength

// if current left array value is <= right array value

if leftArray[leftPos] <= rightArray[rightPos]

// copy left array value to main array (data) and move on by 1

data[mergedArrayPos] = leftArray[leftPos]

leftPos++

else // Otherwise, do same for right array

// copy right array value to main array & move on by 1

data[mergedArrayPos] = rightArray[rightPos]

rightPos++

// move on to next slot in main array

mergedArrayPos++

// At this stage, one of the arrays is empty but we don’t know which

// If left still has data, add that to main array

while(leftPos < leftLength){

data[mergedArrayPos] = leftArray[leftPos]

leftPos++

mergedArrayPos++

// If right still has data, add that to main array

while(rightPos < rightLength)

data[mergedArrayPos] = rightArray[rightPos]

rightPos++

mergedArrayPos++