**Problem1:**

InsertionSort, BubbleSort are stable because they don’t change the order of equal elements. Even after sorting for equal element at index i and j for i < j before the sorting, then the index of those elements will still be i < j.

**Problem2:**

Text, letter

Description automatically generated

**Problem3: (A)**

**Algorithms** mergeSort(S)

**Input** Sequence S with n integers

**Output** Sorted version of sequence S

if lower S.size() <= 20 then

InsertionSort(S)

Else

(S1, S2) ←Partition(S)

mergeSort(S1)

mergeSort(S2)

returnS

**Problem3: (B)**

**void** mergeSort(**int**[] tempStorage, **int** lower, **int** upper) {

**if** (upper - lower <= 20) {

doInsertionSort(tempStorage, lower, upper);

**return**;

} **else** {

**int** mid = (lower + upper) / 2;

mergeSort(tempStorage, lower, mid); // sort left half

mergeSort(tempStorage, mid + 1, upper); // sort right half

merge(tempStorage, lower, mid + 1, upper); // merge them

}

}

**private** **void** doInsertionSort(**int**[] anArray, **int** lower, **int** upper) {

**int** temp = 0;

**int** j = 0;

**for** (**int** i = lower + 1; i <= upper; ++i) {

temp = theArray[i];

j = i;

**while** (j > 0 && temp < theArray[j - 1]) {

theArray[j] = theArray[j - 1];

j--;

}

theArray[j] = temp;

}

}

**private** **void** merge(**int**[] tempStorage, **int** lower, **int** midPlusOne, **int** upper) {

**int** pos = 0; // tempStorage index

**int** i = lower;

**int** j = midPlusOne;

**int** n = upper - lower + 1;

**while** (i < midPlusOne && j <= upper) {

**if** (theArray[i] <= theArray[j])

tempStorage[pos++] = theArray[i++];

**else**

tempStorage[pos++] = theArray[j++];

}

**while** (i < midPlusOne) {

tempStorage[pos++] = theArray[i++];

}

**while** (j <= upper) {

tempStorage[pos++] = theArray[j++];

}

**for** (j = 0; j < n; ++j) {

theArray[lower + j] = tempStorage[j];

}

}

**Problem3: (C)**

40 ms -> MergeSortPlusInsertion

55 ms -> InsertionSort

151 ms -> SelectionSort

446 ms -> BubbleSort2

464 ms -> BubbleSort1

485 ms -> BubbleSort

The MergeSortPlus runs faster than all the others and this is since, for fewer elements (less than 20 in this case), the insertion sort is much faster and avoids the overhead of the multiple self-calls that would have happened in the merge sort.

**Problem4:**

Letter

Description automatically generated

Letter

Description automatically generated

C) . Generalization: For a given binary tree with height of n, the maximum number of leaves is 2n.