Problem 1: Bubble Sort & Stable and Adaptive Sorting

a) $\boldsymbol{for}\ i = 1\ \boldsymbol{to}\ length(A) - 1$
$\quad\boldsymbol{for}\ j = length(A)\ \boldsymbol{to}\ i + 1$
$\quad\boldsymbol{if}\ A[j] < A[j-1]$
$\quad swap\ A[j]\ and\ A[j-1]$

b) Worst case:

Number of operations is

$$4(n-1+n-2+\cdots+3+2+1)$$

$$= 2 \times n \times (n-1)$$

$$= cn^2 + cn + 1$$

$$\therefore \mathrm{O}(n^2) \text{ for the worst case}$$

Best case:

Since the elements are already sorted the algorithm does not go into the if statement and therefore the time complexity would be:

$$O(n)$$

as the algorithm only does n comparisons.

Average case:

The average case time complexity is given by:

$$O(n^2)$$

c) Merge sort, insertion sort and bubble sort are stable sorting algorithms as they are comparison based i.e.

$$A[j]\ comes\ before\ A[i]\ if\ and\ only\ if\ A[j] < A[i]$$

Since $i < j$, the relative order is preserved $if\ A[i] \equiv A[j]$, i.e. $A[i]$ comes before $A[j]$.

On the other hand heap sort is unstable. Take an array consisting of elements:

$$21, 20a, 20b, 14, 11, 5, 3$$

(The a and b are used to differentiate the elements.)
Building a minimum heap the 21 is moved to the last position, the 20a is moved as it is the next element. The resulting array after heap sort would be.

$$3, 5, 11, 14, 20b, 20a, 21$$

Heap sort is unstable as any information about the original order of the sequence is lost during heapify.

d) Merge sort is not adaptive as it divides the array all the way down regardless of the order of elements.
On the other hand heap sort, insertion sort and bubble sort are adaptive as they compare elements before swapping them.

Problem 2: Heap Sort

a) Find attached cpp file.