Q1

(a)

Given below is an algorithm:

1. procedure PROC1(array 𝐴, int 𝑛)

2. if 𝑛 > 0 then

3. PROC1(𝐴, 𝑛 − 1)

4. 𝑥 ← 𝐴[𝑛]

5. 𝑖 ← 𝑛 − 1

6. while 𝑖 ≥ 0 and 𝐴[𝑖] > 𝑥 do

7. 𝐴[𝑖 + 1] ← 𝐴[𝑖]

8. 𝑖 ← 𝑖 – 1

9. end while

10. A[i+1] ← x

11. end if

12. end procedure

*Write down the* ***recurrence******relation*** *𝑇(𝑛) for the* ***worst******case******number******of******operations*** *(you may employ the big O notation in your answer) and give its* ***worst******case******time******complexity****.*

***What*** *is the* ***algorithm******computing****?*

Actual code

public int[] proc1(int[] A, int n) {  
 if(n > 0) {  
 proc1(A, n-1);  
 int x = A[n];  
 int i = n - 1;  
 while(i >= 0 && A[i] > x) {  
 A[i + 1] = A[i];  
 i = i - 1;  
 }  
 A[i+1] = x;  
 }  
 return A;  
}

* The algorithm is known as *Insertion sort.*
* Recurrence relation T(n) for the worst case number of operations (may use big O notation):

The recurrence relation of recursive selection sort is T(n) = T(n-1) + n. It can be solved by the method of substitution and is found to be equal to n^2.

* The worst case time complexity of the algorithm is big Theta of (n^2). (Θ(n^2 ))) and it occurs when the input array is sorted in reverse order. (e.g. A = {5, 4, 3, 2, 1}).
* What is the algorithm computing: The algorithm is given as input an array and its size. Then the input array is sorted in ascending order (e.g. inputting (5, 4, 3, 2, 1) will output (1, 2, 3, 4, 5)). The algorithm sorts the array by keeping a subsequence of elements on the left in the correctly sorted order. This subsequence is increased by inserting the next element into its relatively correct position in the sorted subsequence. Adding to the subsequence stops when the last element of the array has been added and sorted.

(b)

(i)

A comparison-based sorting algorithm is a sorting algorithm that can only gain information about items in the input sequence (a1, a2, …, an) by performing pairwise-comparisons. The only requirement for applying a comparison-based sorting algorithm is that the operator forms a total preorder over the data. Meaning:

* If ai <= aj and aj <= az then ai <= az (transitivity)
* For all ai and aj, ai <= aj or aj <= ai (connexity)

In other words, the algorithm can be applied only when the input data can be ordered.

(ii) The translation of the pseudocode:

public int[] proc2(int[] array) {

for(int j = 0; j < array.length - 1; j++) {

int min = j;

for(int i = j + 1; i < array.length; i++) {

if(array[i] < array[min]) {

min = i;

}

}

int temp = array[j];

array[j] = array[min];

array[min] = temp;

}

return array;

}

This is a selection sort.

Selection sort is in place and NOT stable. Its worst case time complexity is the same as for insertion sort – big Theta of n^2 (Θ(n^2 )).