STUDENT ID: 10670921

SUMMARY FOR INSERTION SORT

Insertion Sort is an algorithm used to sort a given list of items. It iterates through the list and sorts one element at a time. With each iteration, it selects one element and places it in its correct position with respect to the list by comparing it with the nearest elements. This process is repeated until the last item is looked at and there are no more left to be sorted.

Pseudocode for the insertion sort algorithm

InsertionSort(A[0..*n* − 1]*)*
//Sorts a given array by insertion sort
//Input: An array *A*[0..*n* − 1] of *n* orderable elements
//Output: Array *A*[0..*n* − 1] sorted in nondecreasing order
for *i* ← 1 **to** *n* − 1 **do**
v ← *A*[*i*]
j ← *i* − 1
while *j* ≥ 0 **and** *A*[*j*]> *v* **do**
A[*j* + 1] ← *A*[*j*]
j ← *j* − 1
end while
A[*j* + 1] ← *v*
end for

Advantages of insertion sort

- It's a simple algorithm to implement.
- Performance is very high when operating with small lists.

Disadvantages of insertion sort

Performance suffers when large lists are used, as this could involve carrying out a lot of comparisons and shifting of array items.

Time Complexity

To know how expensive each operation is, we need to calculate the cost of each step. The cost of each step is indicated by the side of the algorithm below

Algorithm	Cost
for $i \leftarrow 1$ **to** $n - 1$ **do**	n
$v \leftarrow A[i]$	n-1
$j \leftarrow i - 1$	n-1
while $j \geq 0$ **and** $A[j] > v$ **do**	$\sum_{j=2}^{n} tj$
$A[j + 1] \leftarrow A[j]$	$\sum_{j=2}^{n} tj-1$
$j \leftarrow j - 1$	$\sum_{j=2}^{n} tj-1$
end while	
$A[j + 1] \leftarrow v$	*n-1*
end for	

Best Case Analysis

This is the situation that occurs when say the input array is already sorted. The steps in the algorithm will still be executed, but the while loop that does the sorting will not be entered; which reduces the complexity greatly. The equation for this will therefore be:

$T(n) = c_1n + c_2(n-1) + c_3(n-1) + c_4(n-1) + c_5(n-1)$
$= (c_1 + c_2 + c_3 + c_4 + c_5)n – (c_2 + c_3 + c_4 + c_5)$
$= an + b$ //simplifying the equation
$=O(n)$ //removing the constants

Worst Case Analysis

And finally, the worst case for our algorithm occurs when the input array is in the complete opposite order from what we want it to be for being sorted. The equation will hence be:

$T(n) = c_1n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^{n} tj + c_5 \sum_{j=2}^{n} tj-1 + c_6 \sum_{j=2}^{n} tj-1 + c_7(n-1)$
$= an^2 + bn + c$
$=O(n^2)$

Space Complexity

Insertion sort is a stable sort space complexity of O(1)