Assignment: Analysis of Sort Algorithms

- Rishant Dutt
801104289

* Theoretical Analysis :

1) Insertion Sort -

→ Runtime depends on size of input, N.

→ The basic operation is, the comparison

'val < items[j]'

→ Best-case: Already sorted list,
∴ while loop breaks on first comparison everytime.
∴ O(n)

→ Worst-case: The input is sorted in reverse order, because maximum shifts will be required
∴ $O(n^2)$

2) Selection Sort -

→ Runtime depends on size of input, N.

→ The basic operation is, the comparison

'items[pos] > items[j]'

→ Best case and Worst - case :

Since the algorithm does not depend on the specific numbers in the list, selection sort had the same time complexity for both worst and best cases, i.e.,

$O(n^2)$, ∵ it scans the list of remaining n-1 elements for each element

3) Merge Sort -

→ Runtime depends on size of input.

→ The basic operation is, the comparison

'len(items) > 1'

→ Worst-case and Best-case:

Merge sort's complexity is same for both cases because the algorithm does not change it's behaviour based on the ~~order~~ inherit order of the list.

here,

$$T(n) = T(n/2) + O(n)$$

using case II of Master method,

$$O(T(n)) = O(n \log n)$$

4) Bubble sort:

→ Runtime depends on the size of the input, N.

→ The basic operation is, comparison

'items[j+1] < items[j]'

→ Best-case: The list is already sorted, no swaps required,

complexity is, $O(n)$.

→ Worst-case: The list is sorted in reverse order, all remaining elements are swapped, for each index,

complexity is, $O(n^2)$.