

Insertion Sort, Part 1

INSERTION SORT is a very useful and much used algorithm, although, as we shall find, it is not one of the most efficient. The basic method is as follows.

At the i^{th} step the first i elements are already sorted and we INSERT the $(i + 1)^{st}$ element in its correct position among the first i elements. We do this by working up the sorted list until we reach an earlier element than the $(i + 1)^{st}$ one.

To learn how the algorithm works, we will discuss its operation on the following List 1.

<u>List 1</u>		<u>List 2</u>		<u>List 3</u>		<u>List 4</u>
NSW		NSW		ACT		ACT
TAS		<u>TAS</u>		NSW		NSW
ACT		ACT		<u>TAS</u>		QLD
QLD	→ 1 comparison	QLD	→ 2 comparisons	QLD	→ 2 comparisons	<u>TAS</u>
VIC		VIC		VIC		VIC
SA		SA		SA		SA
NT		NT		NT		NT
WA		WA		WA		WA

In each list, we have sorted down to the element underlined. We insert the next element into the sorted portion by comparing it with each element (starting at the end) until we reach an earlier one.

List 2 is obtained by looking at the first unsorted element, TAS, and comparing it with NSW. NSW is earlier in the alphabet, so in List 2 we INSERT TAS after NSW. We have sorted the first two elements and used 1 comparison.

List 3 is obtained by first comparing the next unsorted element, ACT, with the element above it, TAS. We don't stop because TAS is later in the alphabet, so compare ACT with NSW. Because ACT is earlier, we INSERT ACT at the top of the list. We have sorted the first 3 elements, using an extra 2 comparisons.

List 4 requires only 2 comparisons because we stop when we reach NSW, which is earlier than QLD.

1. Continue the insertion sort by writing down LIST 5, LIST 6 and LIST 7, recording at each stage the number of comparisons performed.

INSERTION SORT has its worst case (that is the one requiring the most comparisons) when the list is in reverse order. (You should NOT think that every sort has its worst case when the list is in reverse order. Some sorts have their worst case when the given list is already in the correct order.)

2. (a) Let $n, n - 1, \dots, 2, 1$ be the first n numbers in reverse order and let $I(n)$ be the number of comparisons required to put this list in numerical order. Apply INSERTION SORT and write down LIST 1, LIST 2, LIST 3 and LIST 4. Each time indicate how many comparisons are performed.
- (b) Write down LIST n and then LIST $n - 1$ and indicate how many comparisons are needed to move from LIST $n - 1$ to LIST n . **Hint:** Notice that in LIST 2 the first 2 elements are in correct order, in LIST 3 the first 3 elements are in order, and in LIST 4 the first 4 are in order. What about LIST n ?

3. (a) Find $I(4)$. Leave your answer as a sum. (b) What is $I(5)$? Leave your answer as sum.
4. (a) Write down $I(n)$ as a sum.
 (b) Use the following information to find a formula for $I(n)$.

$$\begin{array}{cccccccc}
 1 & + & 2 & + \dots + & (n-2) & + & (n-1) & = I(n) \\
 + & (n-1) & + & (n-2) & + \dots + & 2 & + & 1 & = I(n) \\
 \hline
 n & + & n & + \dots + & n & + & n & = 2I(n)
 \end{array}$$

5. Write down the definition of $I(n) \in \mathcal{O}(n^2)$.
6. Prove that $I(n) \in \mathcal{O}(n^2)$.
7. Let $J(n)$ be the number of comparisons needed to sort the list $1, 2, \dots, n$ using Insertion Sort.
 (a) Write down Lists 1, 2, 3 when $n = 3$ and record the number of comparisons used at each step.
 (b) Find $J(3)$.
 (c) Find $J(4)$.
 (d) Find $J(n)$ and prove that $J(n) \in \mathcal{O}(n)$.
8. The set $\{1, 2\}$ can be used to form 2 different lists and the set $\{1, 2, 3\}$ can be used to form 6 different lists. If T_n is the total number of comparisons required to sort all the lists formed from $\{1, \dots, n\}$ we will now see that $T_3 = 3T_2 + 2(1 + 2 + 2)$.
 (a) Write down each of the six orderings (permutations) of $\{1, 2, 3\}$.
 (b) For the lists 1, 2, 3 and 2, 1, 3 use insertion sort to sort them, setting out your answer as follows:

List 1	List 2	List 3	List 1	List 2	List 3
<u>1</u>	<u>2</u>
2	1
3	3

- (c) Notice that, to the left of the vertical line all we have done is sort the two lists 1, 2 and 2, 1 with 2 elements. Hence up to that stage T_2 comparisons have been required (in total, for both original lists). How many extra comparisons are taken to move from LIST 2 to LIST 3 (in total)?
- (d) Repeat parts (b) and (c) for the lists 1, 3, 2 and 3, 1, 2 which end in 2. (We still use T_2 comparisons to sort the lists 1, 3 and 3, 1.)
- (e) Repeat parts (b) and (c) for the lists 2, 3, 1 and 3, 2, 1 which end in 1.
- (f) Explain why $T_3 = 3T_2 + 2(1 + 2 + 2)$.
 In Sorting Prac 4, we show that $T_4 = 4T_3 + 3!(1 + 2 + 3 + 3)$, which is a special case of $T_n = nT_{n-1} + (n-1)!(1 + 2 + \dots + (n-2) + (n-1) + (n-1))$.
9. Explain why $I(n)$ gives the biggest number of comparisons and $J(n)$ gives the smallest number of comparisons when INSERTION SORT is used to sort a list with n elements.

We have seen that the number of comparisons needed for Insertion Sort is between $n-1$ and $\frac{1}{2}n^2 - \frac{1}{2}n$. In Sorting Prac 4 we will see that the average number is roughly $\frac{1}{4}n^2$.