Insertion Sort is a simple sorting algorithm that builds the final sorted list one item at a time.

It iterates through the input list, removing one element at a time and finding its correct position within the sorted part of the list.

The algorithm repeats this process until the entire list is sorted.

Here’s how the Insertion Sort algorithm works:

1. Start with an unsorted list of elements.
2. Divide the list into two parts: the sorted part and the unsorted part. Initially, the sorted part contains only the first element, and the unsorted part contains the remaining elements.
3. Iterate through the unsorted part of the list, starting from the second element.
4. Select the current element and compare it with the elements in the sorted part of the list, moving from right to left.
5. If the current element is smaller (or larger, depending on the desired sorting order) than the compared element, shift the compared element one position to the right.
6. Repeat step 5 until you find the correct position for the current element within the sorted part of the list.
7. Insert the current element into its correct position within the sorted part of the list.
8. Move the boundary of the sorted part one element to the right.
9. Repeat steps 3-8 until the entire list is sorted.

Example:

Consider the following unsorted list of integers: [5, 3, 8, 2, 1].

Pass 1:

- Select the second element (3) and compare it with the first element (5). Since 3 is smaller, shift 5 to the right. The list becomes

```
[3, 5, 8, 2, 1]
```

Pass 2:

- Select the third element (8) and compare it with the second element (5). Since 8 is greater, no further shifting is needed. The list remains

```
[3, 5, 8, 2, 1]
```

Pass 3:

- Select the fourth element (2) and compare it with the third element (8). Since 2 is smaller, shift 8 to the right.
- Compare 2 with the second element (5). Since 2 is smaller, shift 5 to the right.
- Insert 2 into its correct position within the sorted part. The list becomes

```
[2, 3, 5, 8, 1]
```

Pass 4:

- Select the fifth element (1) and compare it with the fourth element (8). Since 1 is smaller, shift 8 to the right.
- Compare 1 with the third element (5). Since 1 is smaller, shift 5 to the right.
- Compare 1 with the second element (3). Since 1 is smaller, shift 3 to the right.
- Insert 1 into its correct position within the sorted part. The list becomes

```
[1, 2, 3, 5, 8]
```

The list is now sorted in ascending order: [1, 2, 3, 5, 8].

Insertion Sort has a time complexity of $O(n^2)$ in the average and worst cases, where ‘n’ is the number of elements in the list.

It performs well for small lists or partially sorted lists and is often used as an intermediate step within more advanced sorting algorithms.