

# INSERTION SORT

## 1. Definition

**Insertion Sort** is a simple comparison-based sorting algorithm that builds the final sorted array **one element at a time**, by inserting each new element into its correct position among the previously sorted elements.

## 2. Key Idea

1. Consider first element as sorted.
2. Pick the next element (called *key*).
3. Compare it with elements in the sorted part (left side).
4. Shift larger elements one position to the right.
5. Insert the key in its correct position.
6. Repeat for all elements.

Insertion Sort works similarly to sorting playing cards in your hand.

## 3. Example

Array:

[5, 4, 3, 2, 1]

### Pass 1: key = 4

Shift 5 → place 4

Array → [4, 5, 3, 2, 1]

### Pass 2: key = 3

Shift 5, 4 → place 3

Array → [3, 4, 5, 2, 1]

### Pass 3: key = 2

Shift 5, 4, 3 → place 2

Array → [2, 3, 4, 5, 1]

**Pass 4: key = 1**

Shift 5, 4, 3, 2 → place 1

Array → [1, 2, 3, 4, 5]

Sorted!

## 4. Algorithm (Pseudocode)

```
InsertionSort(arr, n)
  for i = 1 to n-1
    key = arr[i]
    j = i - 1

    while j >= 0 and arr[j] > key
      arr[j+1] = arr[j]      // shift right
      j = j - 1

    arr[j+1] = key           // insert key
```

## 5. Properties

| Property       | Value                     |
|----------------|---------------------------|
| Algorithm Type | Comparison-based          |
| Technique      | Insertion                 |
| In-place       | Yes                       |
| Stable         | ✓ Yes                     |
| Adaptive       | ✓ Yes (best-case $O(n)$ ) |

| <u>Component</u> | <u>Worst Case</u> |
|------------------|-------------------|
| Passes           | $n - 1$           |
| Comparisons      | $n(n - 1)/2$      |
| Shifts           | $n(n - 1)/2$      |
| Time Complexity  | $O(n^2)$          |
| Space Complexity | $O(1)$            |
| Stable?          | Yes               |
| Adaptive?        | Yes               |

## 6. Advantages

- Simple and intuitive
- Efficient for small datasets
- Good for nearly-sorted arrays
- Stable
- Adaptive
- Online algorithm (can sort data as it arrives)

## 7. Disadvantages

- Slow for large datasets
- Worst case  $O(n^2)$
- Many shifts compared to selection sort

## 8. Real-Life Analogy

Sorting cards in your hand:

You pick a card and insert it at the right place among already sorted cards.

## 9. EFFICIENCY ANALYSIS IS THE SAME AS BUBBLE SORT

Insertion sort

9, 4, 12, 13, 15, 2, 1

Step 1 : 9 ] 4, 12, 13, 15, 2, 1

Step 2 : 4 9 ] 12, 13, 15, 2, 1  
 $4 < 9$

Step 3 : 4 9 12 ] 13, 15, 2, 1  
 $12 > 9$

Step 4 : 4 9 12 13 ] 15, 2, 1  
 $13 > 12$

Step 5 : 4 9 12 13 15 ] 2, 1  
 $15 > 13$

Step 6 : 4 9 12 13 2 15 ] 1  
 $2 < 15$   
 4 9 12 2 13 15 ] 1  
 $2 < 13$   
 4 9 2 12 13 15 ] 1  
 $2 < 9$   
 4 2 9 12 13 15 ] 1  
 $2 < 4$   
 2 4 9 12 13 15 ] 1

Step 7 : 2 4 9 12 13 1 15  
 $1 < 13$

2 4 9 12 1 13 15  
 $1 < 12$   
 2 4 9 1 12 13 15  
 $1 < 9$   
 2 4 1 9 12 13 15  
 $1 < 4$   
 2 1 4 9 12 13 15  
 1 2 4 9 12 13 15 ← sorted

total pass = 6.