

# Comprehensive Analysis of Fundamental Algorithms in Rust

Thor

November 3, 2025

# Contents

<b>1</b>	<b>Insertion Sort</b>	<b>2</b>
1.1	Idea . . . . .	2
1.2	Pseudocode (CLRS) . . . . .	2
1.3	Complexity . . . . .	2
<b>2</b>	<b>Example: Insertion Sort (Inline)</b>	<b>3</b>
2.1	Implementation Listing . . . . .	3
	<b>References</b>	<b>4</b>

# Chapter 1

## Insertion Sort

### 1.1 Idea

Insertion Sort builds the final sorted array one element at a time by shifting larger elements to the right.

### 1.2 Pseudocode (CLRS)

---

**Algorithm 1** INSERTION-SORT( $A$ )

---

```
1: Input: Array  $A[1..n]$ 
2: for  $j \leftarrow 2$  to  $n$  do
3:    $key \leftarrow A[j]$ 
4:    $i \leftarrow j - 1$ 
5:   while  $i > 0$  and  $A[i] > key$  do
6:      $A[i + 1] \leftarrow A[i]$ 
7:      $i \leftarrow i - 1$ 
8:   end while
9:    $A[i + 1] \leftarrow key$ 
10: end for
```

---

### 1.3 Complexity

Worst-case time  $\Theta(n^2)$ ; space  $\Theta(1)$ .

## Chapter 2

# Example: Insertion Sort (Inline)

### 2.1 Implementation Listing

```
pub fn insertion_sort<T>(arr: &mut [T])
where
    T: PartialOrd + Copy,
{
    if arr.len() < 2 { return; }
    for i in 1..arr.len() {
        let key = arr[i];
        let mut j = i;
        while j > 0 && arr[j - 1] > key {
            arr[j] = arr[j - 1];
            j -= 1;
        }
        arr[j] = key;
    }
}
```

Listing 2.1: Insertion Sort in Rust (algorithms crate)

## References