Merge Sort
Merge Sort

• In merge sort, the idea of the algorithm is to sort smaller arrays and then combine those arrays together (merge them) in sorted order.

• Merge sort leverages something called recursion, which we’ll touch on in more detail in a future video.

In pseudocode:
  • Sort the left half of the array (assuming $n > 1$)
  • Sort the right half of the array (assuming $n > 1$)
  • Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

In pseudocode:
- Sort the left half of the array (assuming $n > 1$)
- Sort the right half of the array (assuming $n > 1$)
- Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming \( n > 1 \))
Sort the right half of the array (assuming \( n > 1 \))
Merge the two halves together
Merge Sort

5 2 1 3 6 4

In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

In pseudocode:
- Sort the left half of the array (assuming $n > 1$)
- Sort the right half of the array (assuming $n > 1$)
- Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
**Merge Sort**

| 5 | 1 | 2 | 3 | 6 | 4 |

**In pseudocode:**
- Sort the left half of the array (assuming \( n > 1 \))
- Sort the right half of the array (assuming \( n > 1 \))
- Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming \( n > 1 \))
Sort the right half of the array (assuming \( n > 1 \))
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming \( n > 1 \))
Sort the right half of the array (assuming \( n > 1 \))
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming \( n > 1 \))
Sort the right half of the array (assuming \( n > 1 \))
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming \( n > 1 \))
Sort the right half of the array (assuming \( n > 1 \))
Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
**Merge Sort**

**In pseudocode:**
- Sort the left half of the array (assuming \( n > 1 \))
- Sort the right half of the array (assuming \( n > 1 \))
- Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
### Merge Sort

In pseudocode:
- Sort the left half of the array (assuming $n > 1$)
- Sort the right half of the array (assuming $n > 1$)
- Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming \( n > 1 \))
Sort the right half of the array (assuming \( n > 1 \))
Merge the two halves together
In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming \( n > 1 \))
Sort the right half of the array (assuming \( n > 1 \))
Merge the two halves together
Merge Sort

In pseudocode:
Sort the left half of the array (assuming $n > 1$)
Sort the right half of the array (assuming $n > 1$)
Merge the two halves together
Merge Sort

• **Worst-case scenario**: We have to split $n$ elements up and then recombine them, effectively doubling the sorted subarrays as we build them up. (combining sorted 1-element arrays into 2-element arrays, combining sorted 2-element arrays into 4-element arrays...)

• **Best-case scenario**: The array is already perfectly sorted. But we still have to split and recombine it back together with this algorithm.
Merge Sort

\( \mathcal{O}(n \log n) \)

\( \Omega(n \log n) \)