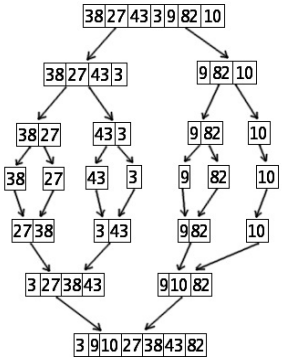
Merge Sort

Merge Sort Algorithm

- The idea behind merge sort is **divide and conquer**
  1. Divide data into 2 equal parts
  2. Recursively sort both halves
  3. Merge the results

Merge Sort Example

1. Divide data into 2 equal parts
2. Recursively sort both halves
3. Merge the results



Merge Sort Example

- 8 45 87 34 28 45 2 32 25 78
  - Divide in half
- 8 45 87 34 28 | 45 2 32 25 78
  - Divide each half in half
- 8 45 87 | 34 28 | 45 2 32 | 25 78
  - Divide each half of half in half
- 8 45 | 87 | 34 | 28 | 45 2 | 32 | 25 | 78
  - Divide remaining elements that have more than one number in half
- 8 | 45 | 87 | 34 | 28 | 45 | 2 | 32 | 25 | 78

Merge Sort Example (continued)

- 8 | 45 | 87 | 34 | 28 | 45 | 2 | 32 | 25 | 78
  - Sort by comparing adjacent elements
  - 8 | 45 → 8 45
  - 87 → 87 (not sorted)
  - 34 | 28 → 28 34
  - 45 | 2 → 2 45
  - 32 → 32 (not sorted)
  - 25 | 78 → 25 78
- 8 45 | 87 | 28 34 | 2 45 | 32 | 25 78

Merge Sort Example (continued)

- 8 45 | 87 | 28 34 | 2 45 | 32 | 25 78
  - Sort by comparing adjacent elements
  - 8 45 | 87 → 8 45 87
  - 28 34 → 28 34 (not sorted)
  - 2 45 | 32 → 2 32 45
  - 25 78 → 25 78 (not sorted)
- 8 45 87 | 28 34 | 2 32 45 | 25 78

Merge Sort Example (continued)

- 8 45 87 | 28 34 | 2 32 45 | 25 78
  - Sort by comparing adjacent elements
  - 8 45 87 | 28 34 → 8 28 34 45 87
  - 2 32 45 | 25 78 → 2 25 32 45 78
- 8 28 34 45 87 | 2 25 32 45 78
  - Sort by comparing adjacent elements
  - 8 28 34 45 87 | 2 25 32 45 78 → 2 8 25 28 32 34 45 45 78 87
- 2 8 25 28 32 34 45 45 78 87

Merge Sort Example (finished)

- 8 45 87 34 28 45 2 32 25 78
- 8 45 87 34 28 | 45 2 32 25 78
- 8 45 87 | 34 28 | 45 2 32 | 25 78
- 8 45 | 87 | 34 | 28 | 45 2 | 32 | 25 | 78
- 8 | 45 | 87 | 34 | 28 | 45 | 2 | 32 | 25 | 78
- 8 45 | 87 | 28 34 | 2 45 | 32 | 25 78
- 8 45 87 | 28 34 | 2 32 45 | 25 78
- 8 28 34 45 87 | 2 25 32 45 78
- 2 8 25 28 32 34 45 45 78 87

How to Sort?

- After splitting the array, how do you sort?
  - Multiple options
    - Selection Sort
    - Insertion Sort
    - Other sorting algorithms

Merge Sort Efficiency

- O(n log n) time complexity
  - Better than selection sort and insertion sort for large arrays
- Why use it?
  - Good option if the data to be sorted can only be efficiently accessed sequentially
    - Some data structures have elements that can only be accessed by going in order (i.e. can't just skip to the middle or end easily)