Discussion Post Unit 4_python example

September 30, 2025

[1]:
```python
def merge_sort(arr):
    """
    Recursively sorts an array using merge sort.
    """

    # Base case: arrays with 0 or 1 element are already sorted
    if len(arr) <= 1:
        return arr

    # Step 1: Split the array into two halves
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    # Step 2: Recursively sort each half
    sorted_left = merge_sort(left_half)
    sorted_right = merge_sort(right_half)

    # Step 3: Merge the two sorted halves
    return merge(sorted_left, sorted_right)


def merge(left, right):
    """
    Merges two sorted arrays into a single sorted array.
    """
    result = []
    i = j = 0

    # Compare elements and take the smaller one
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
```

```
    # Append any remaining elements
    result.extend(left[i:])
    result.extend(right[j:])

    return result


# Example usage
arr = [38, 27, 43, 3, 9, 82, 10]
print("Original array:", arr)
sorted_arr = merge_sort(arr)
print("Sorted array:", sorted_arr)
```

```
Original array: [38, 27, 43, 3, 9, 82, 10]
Sorted array: [3, 9, 10, 27, 38, 43, 82]
```

0.1 How the recursion works step by step

0.1.1 Divide

The array `[38, 27, 43, 3, 9, 82, 10]` is split in half repeatedly:

- First split → `[38, 27, 43]` and `[3, 9, 82, 10]`
- Then `[38, 27, 43]` → `[38]` and `[27, 43]`
- And `[27, 43]` → `[27]` and `[43]`
- This continues until all subarrays are of size 1.

0.1.2 Conquer (sort subarrays)

Arrays of size 1 are already sorted.

0.1.3 Combine (merge)

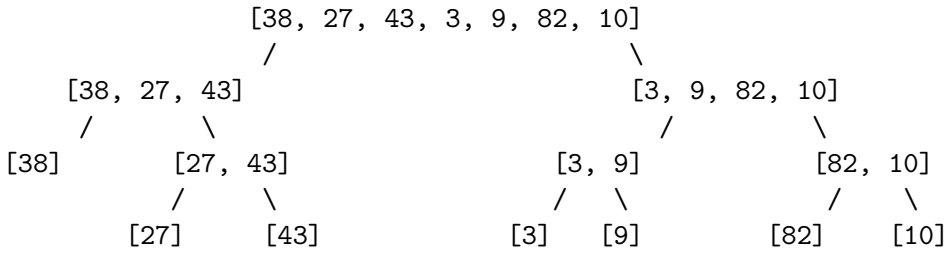
The `merge` function takes two sorted arrays and produces a single sorted array. For example:

- Merge `[27]` and `[43]` → `[27, 43]`
- Merge `[38]` and `[27, 43]` → `[27, 38, 43]`
- Eventually merge `[27, 38, 43]` with `[3, 9, 10, 82]` → `[3, 9, 10, 27, 38, 43, 82]`

This recursive divide-and-conquer approach ensures an **O(n log n)** runtime.

0.2 Merge Sort Recursion Tree (Example)

Original Array: [38, 27, 43, 3, 9, 82, 10]



0.2.1 Merge Steps

1. [27] + [43] → [27, 43]
2. [38] + [27, 43] → [27, 38, 43]
3. [3] + [9] → [3, 9]
4. [82] + [10] → [10, 82]
5. [3, 9] + [10, 82] → [3, 9, 10, 82]
6. [27, 38, 43] + [3, 9, 10, 82] → [3, 9, 10, 27, 38, 43, 82]

Final Sorted Array: **[3, 9, 10, 27, 38, 43, 82]**