Merge Sort

Code and Analysis

Merge Sort Algorithm

Definition:

Merge Sort is a divide-and-conquer sorting algorithm that recursively divides an array into two halves, sorts each half, and then merges the sorted halves back together. This process continues until the entire array is sorted.

Algorithm Steps:

1. Divide the array into two halves.
2. Recursively apply Merge Sort to each half.
3. Merge the two sorted halves by comparing elements and placing them in the correct order.
4. Repeat the merging process until a single sorted array is obtained.

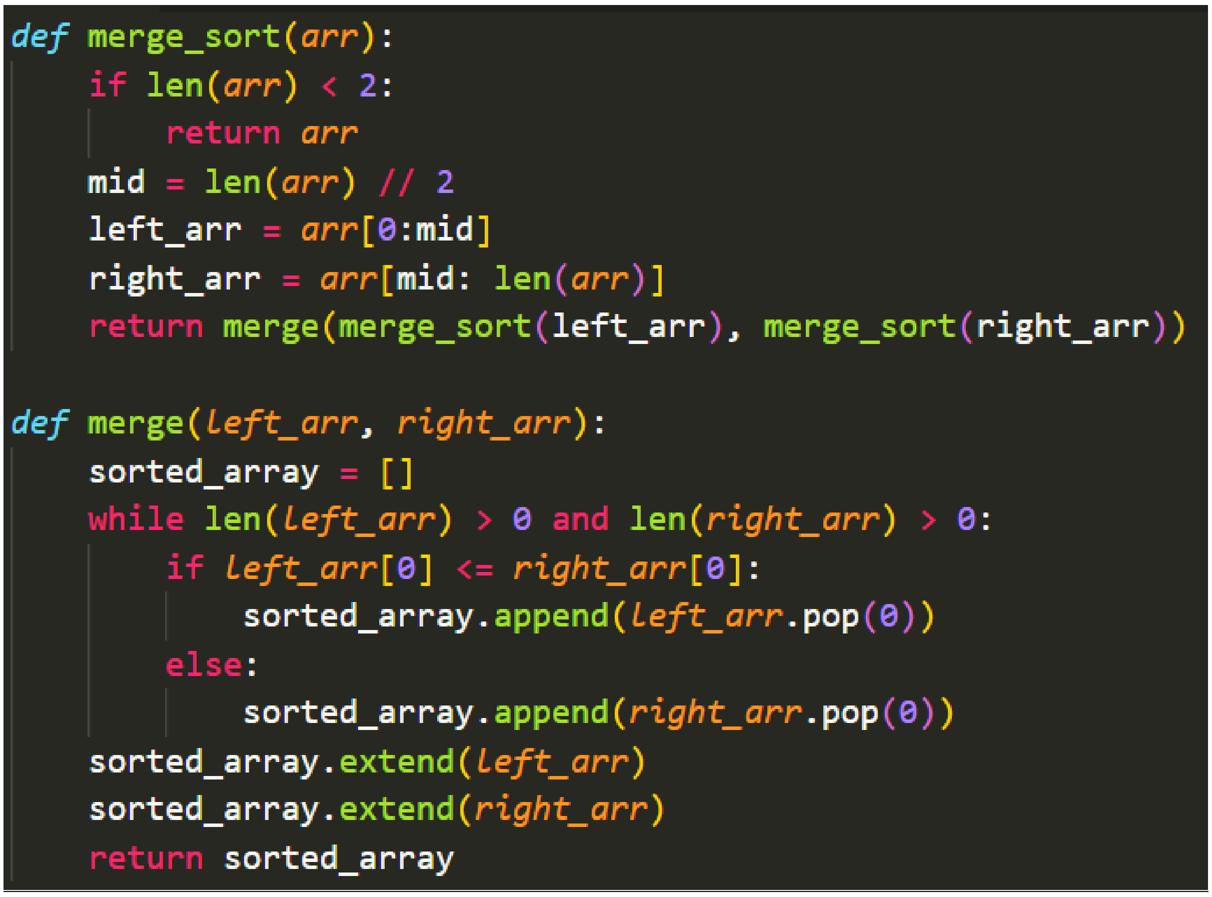
Time Complexity:

- Best Case (Already Sorted): O(n log n)
- Worst Case (Reverse Sorted): O(n log n)
- Average Case: O(n log n)

Space Complexity:

- Auxiliary Space: O(n) (requires additional space for merging subarrays)

Code Implementation (Python):



```python
def merge_sort(arr):
    if len(arr) < 2:
        return arr
    mid = len(arr) // 2
    left_arr = arr[0:mid]
    right_arr = arr[mid: len(arr)]
    return merge(merge_sort(left_arr), merge_sort(right_arr))

def merge(left_arr, right_arr):
    sorted_array = []
    while len(left_arr) > 0 and len(right_arr) > 0:
        if left_arr[0] <= right_arr[0]:
            sorted_array.append(left_arr.pop(0))
        else:
            sorted_array.append(right_arr.pop(0))
    sorted_array.extend(left_arr)
    sorted_array.extend(right_arr)
    return sorted_array
```