Reem Alsharabi CS3012 Assignment 2

1 Merge Sort

1.1 Code and Output

[4]:
```
import random

def sort(arr):
    if len(arr) <= 1: # if the array has one element
        return arr

    mid = len(arr) // 2
    left = arr[:mid] # start to mid
    right = arr[mid:] # mid to end

    left = sort(left)
    right = sort(right)

    return merge(left, right)

def merge(left, right):
    global comparisons
    merged_arr = []
    i = 0
    j = 0

    while i < len(left) and j < len(right): # until we didn't reach the end of either arrays
        comparisons += 1
        if left[i] < right[j]:
            merged_arr.append(left[i])
            i += 1
        else:
            merged_arr.append(right[j])
            j += 1
    # add the remaining elements of left and right to the merged_arr
    merged_arr += left[i:]
    merged_arr += right[j:]

    return merged_arr

def main():
    arr = [random.randint(1, 1000) for _ in range(100)] # Generate an array of random 100 elements
    print("Sorted array:", sort(arr))
    print("Number of comparisons:", comparisons)

comparisons = 0
main()
```

```
Sorted array: [7, 10, 19, 33, 34, 49, 50, 66, 77, 79, 83, 101, 104, 113, 132,
140, 147, 148, 148, 151, 156, 156, 199, 212, 213, 230, 242, 254, 264, 271, 291,
293, 316, 318, 322, 325, 334, 348, 352, 356, 364, 370, 375, 406, 417, 420, 429,
439, 443, 452, 453, 469, 474, 477, 492, 496, 496, 507, 512, 515, 524, 546, 549,
559, 559, 565, 575, 594, 615, 628, 640, 648, 649, 695, 698, 699, 700, 764, 766,
770, 796, 805, 806, 820, 825, 827, 842, 857, 888, 899, 900, 916, 923, 951, 952,
968, 973, 988, 988, 995]

Number of comparisons: 544
```

1.2 Time Complexity Analysis

The time complexity of Merge Sort is O(n log n), where n is the number of elements to be sorted.

- The O(log n) time complexity comes from divide-and-conquer method that recursively divides the input array into two halves until each subarray contains only one element.
- After dividing the array, the algorithm merges the two subarrays together, which takes O(n) time.