Question 2:

From the code I made, the merge sort has the worst case complexity because there are multiple $\log_2(n)$ levels of recursion occurring throughout the merge function. This results in the overall complexity to be O(n log n) which will result in the worst case complexity because all elements are processed at each level.

Question 3:

Split [8, 42, 25, 3, 3, 2, 27, 3]
Left: [8, 42, 25, 3]
Right: [3, 2, 27, 3]

Left:

Split [8, 42, 25, 3]

Left: [8, 42]

Right: [25, 3]

Split [8, 42,]

Left: [8]

Right: [42]

Merge:

[8, 42]

Split [25, 3]

Left: [25]

Right: [3]

Merge:

[3, 25]

Merge:

[8, 42]

[3, 25]

Compares 8 vs 3 → take 3

Compares 8 vs 25 → take 8

Compares 42 vs 25 → take 25

Take remaining 42

[3, 8, 25, 42]

Right:

Split [3, 2, 27, 3]

Left: [3, 2]

Right: [27, 3]

Split [3, 2,]

Left: [3]

Right: [2]

Merge:

[2, 3]

Split [27, 3]

Left: [27]

Right: [3]

Merge:

[3, 27]

Merge:

Left: [2, 3]

Right: [3, 27]

Compares 2 vs 3 → take 2

Compares 3 vs 3 → take 3 (first 3)

Compares 3 vs 27 → take 3 (second 3)

Take remaining 27

[2, 3, 3, 27]

Combine:

Left: [3, 8, 25, 42]

Right: [2, 3, 3, 27]

Compares 3 vs 2 → take 2

Compares 3 vs 3 → take 3

Compares 8 vs 3 → take 3

Compares 8 vs 3 → take 3

Compares 8 vs 27 → take 8

Compares 25 vs 27 → take 25

Compares 42 vs 27 → take 27

Take remaining 42

[2, 3, 3, 3, 8, 25, 27, 42]

Question 4:

Yes it does, each level processes all 8 elements per level, it matches the worst case behaviour because it has three 3's, and it matches the total amount of operations ($n \cdot \log_2(n) = 8*3 = 24$)