REPORT – PROGRAM ASSIGNMENT 3

1. **Description:**

   Merge sort is a divide and conquer algorithm.

   The idea of this algorithm comes from combining 2 sorted lists into 1 new sorted list.

   Merge sort has 2 processes, distributive and merge. The array is sorted by repeating the two processes.

   Find the middle point to divide the array into two halves, recursively the first half and second half, keep repeating until the 2 subarrays have only 1 element, then merge the 2 array sorted halves.

2. **Worst case:** O(nlogn)

   We recursively divide the input array of size n into two parts of sizes $\frac{n}{2}$.

   We have:

   T(n) = $2T(\frac{n}{2}) + cn$, c is a constant

   = $2(2T(\frac{n}{4}) + c\frac{n}{2}) + cn$

   = $4T(\frac{n}{4}) + 2cn$

   = $8T(\frac{n}{8}) + 3cn$

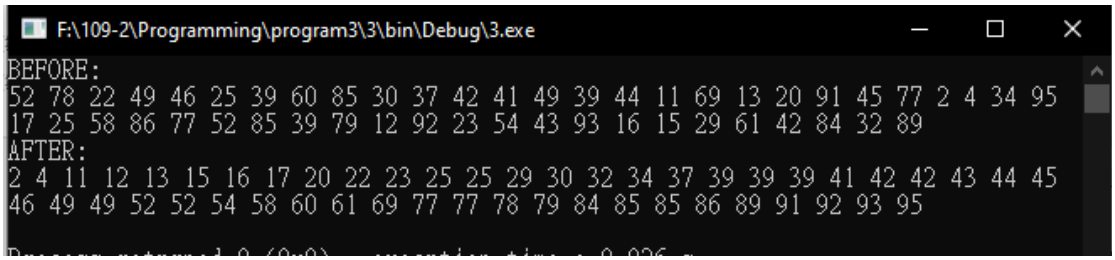
   = $2^kT(\frac{n}{2^k}) + kcn$

   When n = 2^k → k = logn

   → **T(n) = nT(1) + cnlogn**

   T(1) is a constant, so complexity is **nlogn.**

3. Answer:



4. Explanation:

We find the midpoint to divide the array into left and right halves.

Using a recursive algorithm, continue to execute the left (left to mid) and right (mid+1 to right) half, respectively, subdividing the array until there is only 1 element left.

Next step we start merging the 2 sorted arrays. Use 2 temporary arrays to store the left and right halves. Compare elements 2 arrays left and right to merge. Then copy the remaining elements of the left half and right half if any. Make sure the merge array is sorted.

When the recursion returns to the end, we will have a complete sorted array.