QUESTION

Sorting is one of the issues many computer scientists have been working to optimize algorithm complexity. Merge sort is one of the fastest, therefore, the most popular, sorting algorithms used by many application programs today. Merge sort performs sorts using the following steps:

Step 1: Chop an array of \( N \) elements to \( N \) arrays, each of which has one element

Step 2: Merge two arrays of one element to one array of two elements in such a way that whichever smaller is put in the first (left) element in the array of two elements.

Step 3: Repeat 2 by merging two arrays, each of which consist of two elements to one element that consists of four elements.

Step 4: Recursively repeat Step 3 until all the smaller arrays (called “sub-lists” in the pseudo-code below) are merged to one array of \( N \) elements.

When all the smaller arrays are merged to one array of \( N \) elements, the \( N \) elements must be sorted (the smallest element will be stored on the left hand side of the array, followed by the second smallest, and so on).
Assumptions:

(a) There is an array (“m” in the following pseudo-code) of $N$ elements
(b) Each element in the array is an integer

```c
function merge_sort(list m)
    // Base case. A list of zero or one elements is sorted, by definition.
    if length(m) <= 1
        return m
    // Recursive case. First, *divide* the list into equal-sized sublists.
    var list left, right
    var integer middle = length(m) / 2
    for each x in m before middle
        add x to left
    for each x in m after or equal middle
        add x to right
    // Recursively sort both sublists.
    left = merge_sort(left)
    right = merge_sort(right)
    // *Conquer*: merge the now-sorted sublists.
    return merge(left, right)
```

Questions:

(1) **Which of the following(s) is (are) correct (select all that apply)?**

(a) SIMD merge sort has a better algorithm complexity order than SISD merge sort and its execution time is faster than that of SISD.

(b) SIMD merge sort does not have a better algorithm complexity order than SISD merge sort (the same algorithm order) but its execution time is faster than that of SISD merge sort.

(c) SIMD merge sort does not have a better algorithm complexity order than SISD merge sort (the same algorithm order) and its execution time is same as that of SISD merge sort.

(d) None of the above.

(2) **Justify (prove) your choice(s) above.**
Assume the followings:

- You always have infinite capacity of memory (for both SISD and SIMD architecture computers).
- You always have an infinitely large number of processing units in a SIMD architecture computer.
- All processing units in a SIMD architecture computer can read from a memory address physically at the same time.
- Preparing whatever size of an array requires a constant time (i.e., $O(1)$).