**A picture containing text

Description automatically generated**

**School Of Information Technology**

**IT2553 Data Structure and Algo**

|  |  |
| --- | --- |
| **Admin No & Team Members Name:** | 201520M: Eden Will Sng Jin Xuan |
| **PEM Group:** | SF2102 |
| **Module:** | IT2553-02 |
| **Assignment:** | Tutorial 7 |

1. Explain using an example of sorting algorithm, the divide and conquer strategy used in algorithm design.

Example we will be using is merge sort. Merge sort is a comparison type of algorithm that uses the divide and conquer strategy. This is based on recursion , Divide and conquer strategy involves breaking the problem into small sub problem until they become easy enough to solve directly. The solutions to these small sub problems will be combined to become the solution to the original problem

Merge sort has three steps in its divide and conquer strategy

First merge sort divides the partition S into two sub list S1 and S2 which is about n/2 element each. So given 50 elements in a array it will have 2 sub array of 25 elements each named S1 and S2 respectively.

Then after the division S1 and S2 are sorted recursively using the merge sort algorithm

After it is sorted, it is conquered which is to say Merge S1 and S2 into a unique sorted list.

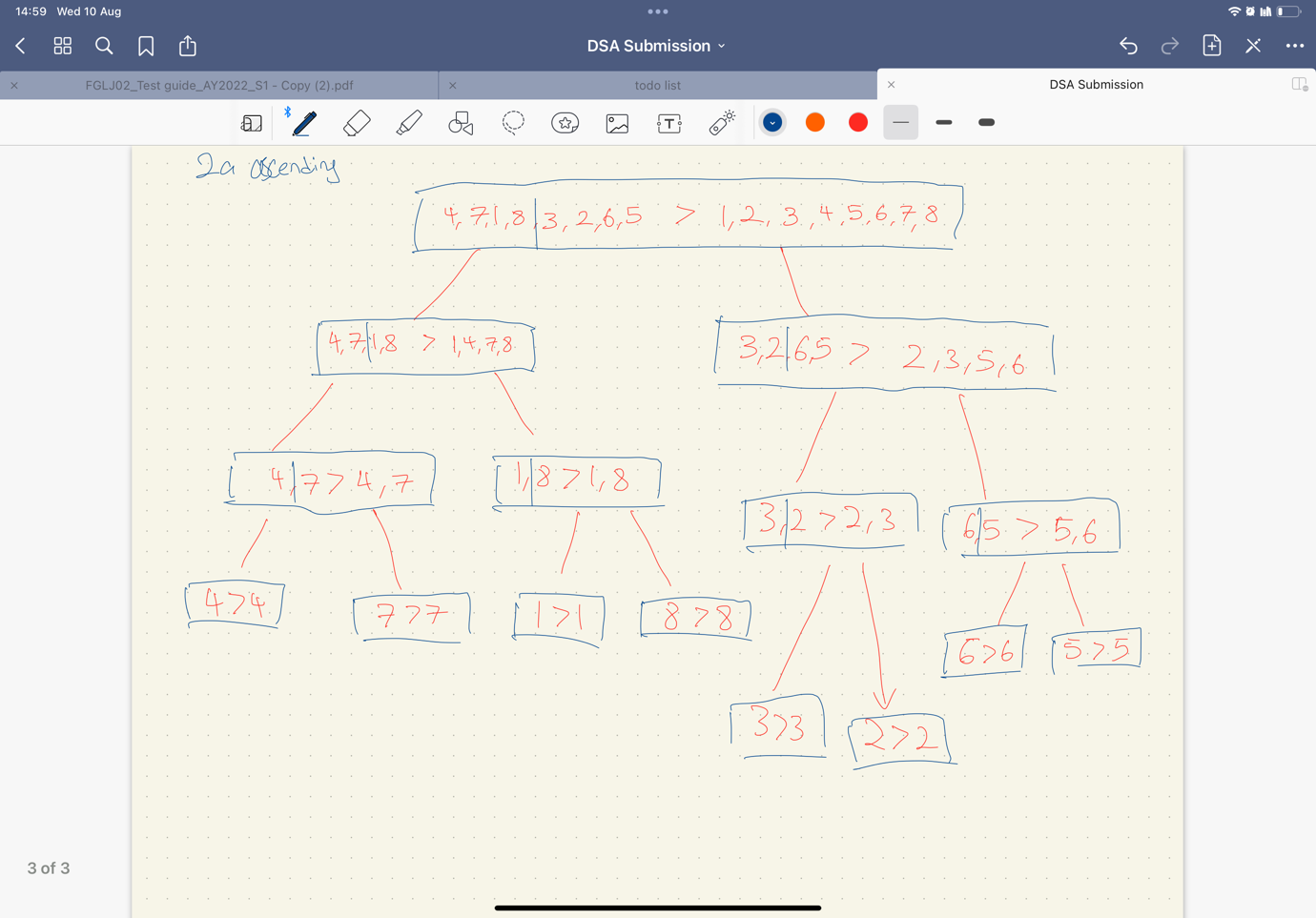
1. Using Merge Sort binary tree, trace the execution of merge sort with the following list of numbers:
   1. 4, 7, 1, 8, 3, 2, 6, 5 (in ascending order)
   2. 5, 2, 7, 8, 1, 4, 6, 3 (in descending order)

2a)

Diagram, text

Description automatically generated

2B)



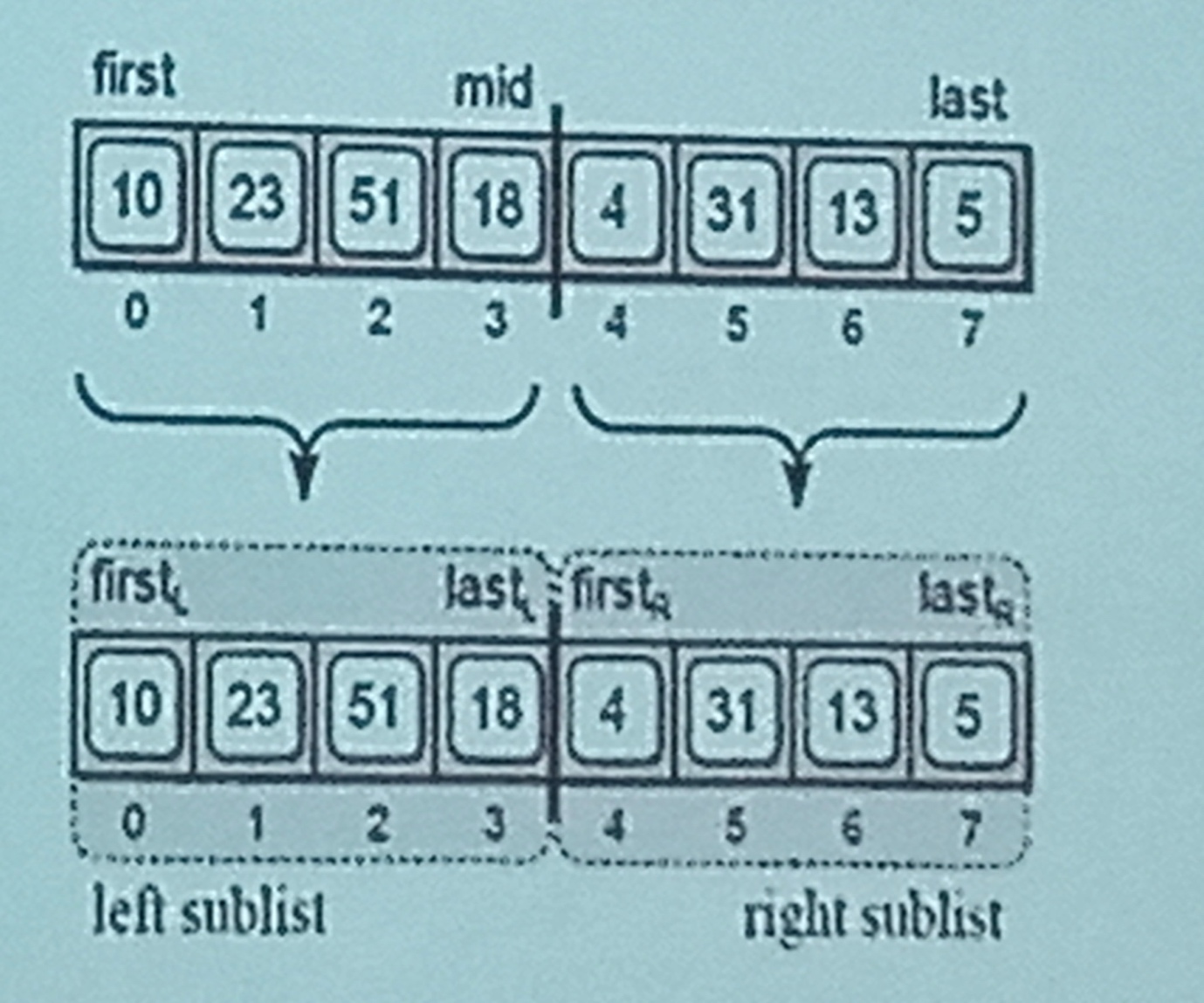
1. In computer science, an in-place algorithm can be described as:

“an algorithm that does not need an extra space and produces an output in the same memory that contains the data by transforming the input ‘in-place’. However, a small constant extra space used for variables is allowed”

Are the sorting algorithms we have discussed so far, i.e. bubble sort, selection sort, insertion sort and merge sort, in-place algorithm?   
  
Bubble sort, selection sort, insertion sort and quick sort are in place.

Quicksort is still in place algorithm because it uses the extra space for recursion function call. It is still in place because it uses the extra space required that is not used to manipulate input but only for recursive calls

Merge sort is not a in place algorithm because it requires an extra o(n) space which is to store the additional array

However, an in-place implementation of Merge Sort is possible e.g. by using index markers to specify a sublist of elements to create virtual sublists within the original physical list (e.g. in Binary Search, we use low & high index markers):  
  
In addition, the mergeSortedLists () method will need to be modified well to support in-place merging of the virtual sub lists.

1. Given a sequence 𝑆 of 𝑛 values, each equal to 0 or 1 e.g. [1, 0, 0, 1, 1, 1, 0]. Describe an in-place method for sorting 𝑆.

[**NOTE**: Do not use the standard sorting algorithms e.g. bubble sort, selection sort, insertion sort etc. to answer this question.]   
Imagine that we colour the 0's blue and the 1's red. Start with a marker at the beginning of the list and one at the end of the list. While the first marker is at a blue element, continue incrementing its index. Likewise, when the second marker is at a red element, continue decrementing its index. When the first marker has reached a red element and the second a blue element, swap the elements. Continue moving the markers and swapping until they meet. At this point, the sequence is ordered.

[NOTE: This approach is like the solution for Practical 06 - Recursion, Question 3, where you were asked to re-arrange a sequence of integer values so that all the even values appear before all the odd values.]

If there are three colours in the sequence, we can order it by doing the above algorithm twice. In the first run, we will move one colour to the front, swapping back elements of the other two colours. Then we can start at the end of the first run and swap the elements of the other two colours in the same way as before. Only this time the first marker will begin where it stopped at the end of the first run.