Assignment 2

Student: Hermann Mjelde Hamnnes

Exchange students from: Østfold University College

II.2415 Advanced Algorithmic & Programming

Professor in the course: Dr. NOUBEG

Due: March 12th, 2025

Table of content

Tutorial Course 2

Part 1

Algorithm walkthrough
Input -> list of numbers

1. Check that the list is not empty.

6	5	3	1	8	7	2	4

2. If the list is bigger than one. Then split the list in half.

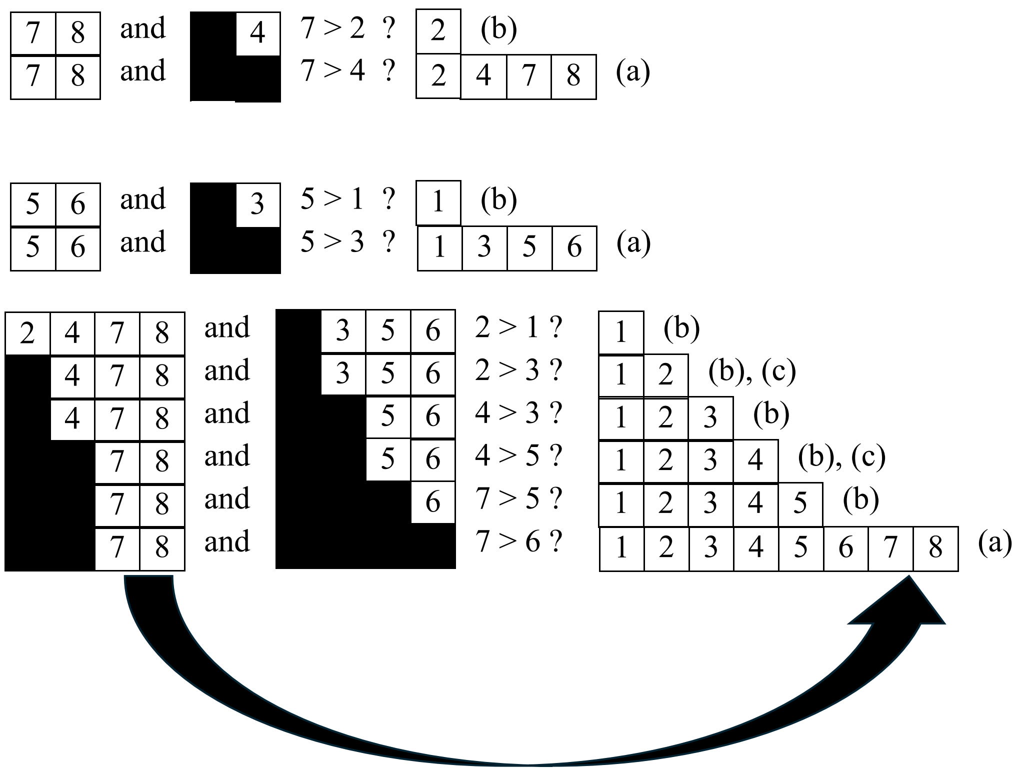
6	5	3	1
8	7	2	4

8	7
2	4
6	5
3	1

8
7
2
4
6
5
3
1

3. Start with comparing the first value in the list with the first value in another list. Choose the lesser value and add it to a new list. And then we need to consider the following aspects in a loop fashion util the end of one of the lists is reached:
    a. If the current value from the first list has been evaluated as bigger than the last value of the second list, then append the current value from the second list to the new list and append the remaining values of the first list to the new list and return the new list.
    b. Else If the value of the first list is grater than the value of the second list then; append the value of the second list to the new table.
    c. Else move to the next value in the first list and start from step a.

8 > 7 ?	7	8
2 > 4 ?	2	4
6 > 5 ?	5	6
3 > 1 ?	1	3


7 8 and 4 7 > 2 ? 2 (b)
7 8 and 7 > 4 ? 2 4 7 8 (a)
5 6 and 3 5 > 1 ? 1 (b)
5 6 and 5 > 3 ? 1 3 5 6 (a)
2 4 7 8 and 3 5 6 2 > 1 ? 1 (b)
4 7 8 and 3 5 6 2 > 3 ? 1 2 (b), (c)
4 7 8 and 5 6 4 > 3 ? 1 2 3 (b)
7 8 and 5 6 4 > 5 ? 1 2 3 4 (b), (c)
7 8 and 6 7 > 5 ? 1 2 3 4 5 (b)
7 8 and 7 > 6 ? 1 2 3 4 5 6 7 8 (a)

Q1

The programmatical representation will therefore look something like this:

```
#We input the list [6, 5, 3, 1, 8, 7, 2, 4]
mergeSort(int[] list)

    int[] listA
    int[] listB

    if list.length <= 1 then #This will be the base case
        do return list;
    end if

    listA = list[0 : list.length / 2]
    listB = list[list.length / 2 : list.length]

    #This part will be the recursive splitting of the list.
    #It will continue until the base case is reached.
    listA = mergeSort(listA) #[6, 5, 3, 1] -> [6, 5] [3, 1] -> [6] [5] [3] [1]
    listB = mergeSort(listB) #[8, 7, 2, 4] -> [8, 7] [2, 4] -> [8] [7] [2] [4]

    #This part will be the recursive merging of the lists.
    #At this point we will wait for each sub-list to be compared with the its other
    #half. We need to imagine that within the the both input values for the merge method
    #there are now being returned a merged list until we are left with the two original
    #list A and B with their original contents just in a sorted order.
    return merge(listA, listB)

    #The merging will therefore work something like this:
    #listA <- merge([5, 6], [1, 3]) <- [merge([6], [5])], [merge([3], [1])]
    #listB <- merge([7, 8], [2, 4]) <- [merge([8], [7])], [merge([2], [4])]
    #return merge([1, 3, 5, 6], [2, 4, 7, 8])
    #Which ultimately will return the list [1, 2, 3, 4, 5, 6, 7, 8]
```

```




#I will here provide a walkthrough of the input [7, 8] and [2, 4].
merge(int[] listA, int[] listB)
    int[] mergedList = new Integer[listA.length + listB.length];
        int a = 0, b = 0, m = 0;

        #m is here used to keep track of what position in
        #the new list we are at.

        #This code will be the same logic as explained in step 3 in
        #the word document.
        #In the walkthrough below, the prefix number will refer to what
        #iteration we are at in the corresponding loop.
        while (a < listA.length && b < listB.length) {
            #1: 0 < listA.length -> True AND 0 < listA.length -> True
            #2: 0 < listA.length -> True AND 1 < listA.length -> True
            #3: 0 < listA.length -> True AND 2 < listA.length -> False


            if (listA[a] <= listB[b]) {
                #1: 7 <= 2 -> False
                #2: 7 <= 4 -> False
                mergedList[m++] = listA[a++];
            } else {
                #1: mergedList = [2, , , ]
                #2: mergedList = [2, 4, , ]
                mergedList[m++] = listB[b++];
            }
        }

        while (a < listA.length) {
            #1: 0 < listA.length -> True
            #2: 1 < listA.length -> True
            #3: 1 < listA.length -> False
            mergedList[m++] = listA[a++];
            #1: mergedList = [2, 4, 7, ]
            #1: mergedList = [2, 4, 7, 8]
        }

        while (b < listB.length) {
            mergedList[m++] = listB[b++];
        }

        return mergedList;

```

Q2

Here is an implementation of the algorithm written in Java.

```java
package eleve.hhamnnes.tutorial2.first_part;

import java.util.Arrays;
import eleve.hhamnnes.tutorial2.interfaces.MergeSortAlgorithm;

public class MergeSortRecursiveAlgorithm implements MergeSortAlgorithm {

    @Override
    public Integer[] execute(Integer[] list) {
        if (list.length <= 1) {
            return list;
        }

        int middleIndex = list.length / 2;

        Integer[] listA = Arrays.copyOfRange(list, 0, middleIndex);
        Integer[] listB = Arrays.copyOfRange(list, middleIndex, list.length);

        listA = execute(listA);
        listB = execute(listB);

        return merge(listA, listB);
    }

    private Integer[] merge(Integer[] listA, Integer[] listB) {
        Integer[] mergedList = new Integer[listA.length + listB.length];
        int a = 0, b = 0, m = 0;

        while (a < listA.length && b < listB.length) {
            if (listA[a] <= listB[b]) {
                mergedList[m++] = listA[a++];
            } else {
                mergedList[m++] = listB[b++];
            }
        }

        while (a < listA.length) {
            mergedList[m++] = listA[a++];
        }

        while (b < listB.length) {
            mergedList[m++] = listB[b++];
        }

        return mergedList;
    }
}

```

Table of resources

This list of resources is based on the APA 7th style. The mentioned styled is described on the following website: https://www.kildekompasset.no/en/ (downloaded 01.02.2025)

Okeke, C. (2023, July 17). *Mastering Big O Notation: Understanding Time and Space Complexity in Algorithms*. Medium. https://medium.com/@DevChy/introduction-to-big-o-notation-time-and-space-complexity-f747ea5bca58

Tutorialpoint. (downloaded 2nd of March 2025). *Data Structures - Asymptotic Analysis*. Tutorialpoint. https://www.tutorialspoint.com/data_structures_algorithms/asymptotic_analysis.htm

Last comments

Please review the code in my GitHub repository: https://github.com/hhamnnes/Assignment1-Advanced-Algorithm-and-programming