

# BLG 336E - Analysis of Algorithms II

## 2019/2020 Spring

### Final Project

#### Question 1

- You should write all your code in C++ language. Your code should be run with the command line arguments specified for each question.
- Your code should be able to be compiled with default g++ compiler and run under Ubuntu OS. **Even if you are writing your code on a different OS, you should check it via ITU SSH.**
- This is a Final Course Project Assignment, cheating is absolutely unethical and morally unacceptable. It will be punished by a negative grade. Also disciplinary actions will be taken.
- For every part of the homework, programs should be run with different command line arguments. **The codes not using these arguments or giving output in a different layout will not be graded.**

#### 1 - Divide & Conquer (25 pts)

[10 pts] In [1], the authors asserted that execution time of Merge Sort can be decreased with a modification on merging operation. Thus, they created the Enhanced Merge Sort algorithm. The main difference is dividing the array into two sub-lists according to each element's odd-even positions before the recursive sorting operation. An example operation is given in Figure 1. Pseudocode of the algorithm is as given in Figure 1 where the main contribution is the **func** function.

Using the skeleton code **q1\_1.cpp** implement the EMS algorithm. Your code should take the filename as the first argument. An example output for **q1\_test1.txt** is as given below. Here first the unsorted list, then the divisions and lastly the sorted list is displayed.

```
1 g++ q1_1.cpp -o q1_1
  ./q1_1 test1.txt
3 6, 1, 10, 4, 8, 5, 7, 9
  1, 4, 5, 7
```

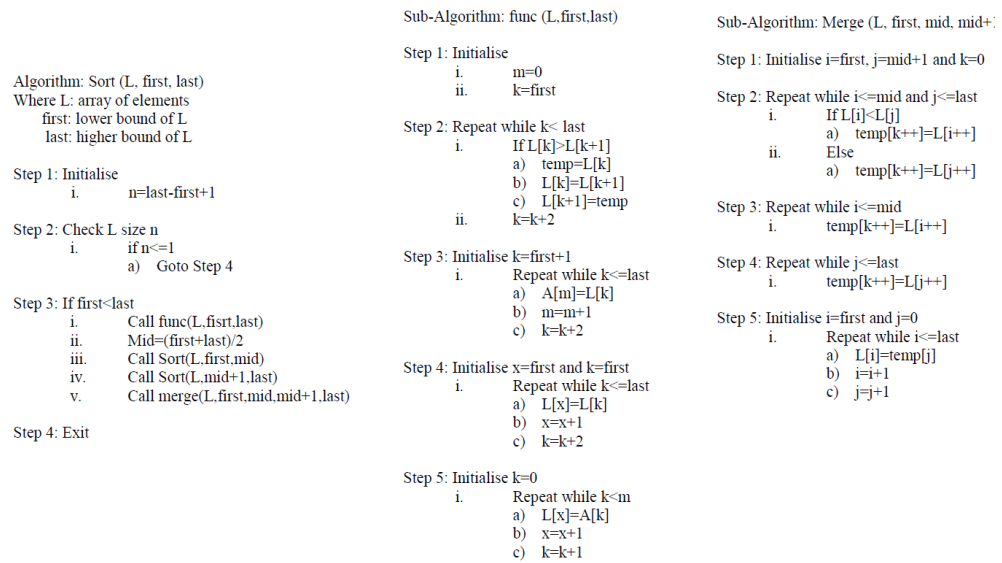


Figure 1: Enhanced Merge Sort algorithm.

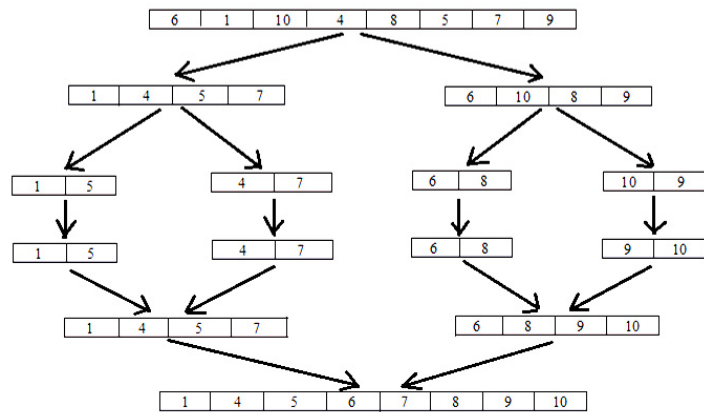


Figure 2: Enhanced Merge Sort on a list

```

5 1, 5
  4, 7
7 6, 10, 8, 9
  6, 8
9 10, 9
  1, 4, 5, 6, 7, 8, 9, 10

```

[10 pts] Min-max sorting is a sorting algorithm which also works in a divide & conquer manner. Basically it can be summarised as follows;

- Find minimum value of the array **in a D&C way**.
- Find maximum value of the array **in a D&C way**.
- Swap these items with the first and last items.
- Recursively follow the same procedure on the remaining part of the array.

Change the code you used in the first part so that for left sub-arrays it uses Min-Max Sorting and right sub-arrays it uses EMS. Output for this part for the same input file is given below.

```

g++ q1_2.cpp -o q1_2
2 ./q1_2 test1.txt
  6, 1, 10, 4, 8, 5, 7, 9
4 Min_max unsorted: 1, 4, 5, 7
  Min: 1 Max: 7
6 Min: 4 Max: 5
  Min_max sorted: 1, 4, 5, 7
  6, 10, 8, 9
8 Min_max unsorted: 6, 8
10 Min: 6 Max: 8
  Min_max sorted: 6, 8
12 10, 9
  Min_max unsorted: 9
14 Min_max sorted: 9
  1, 4, 5, 6, 7, 8, 9, 10

```

[5 pts] Make the complexity analysis of this newly frankenstained algorithm.

## Bibliography

- [1] S. Paira, S. Chandra, and S. S. Alam, “Enhanced merge sort-a new approach to the merging process,” *Procedia Computer Science*, vol. 93, pp. 982–987, 2016. [1](#)