Data Structures and Algorithms, Spring '23 section 1 Dr. Hristescu

Lab 10

Assigned: Thursday, April 6
Due: 10pm Monday, April 10

Solutions to problems different than specified in the description or not following the instructions provided in this lab will not receive consideration and thus any credit. Follow the instructions carefully.

Problem0: **Analyze** a variant of the MergeSort algorithm discussed in class that sorts an array into ascending order. Consider that instead of splitting the problem into **2** sub-problems, it is split into **3** sub-problems. Explain all the steps in this **3-WaySplit MergeSort** variant and analyze the time complexity of this method for solving a problem of input **n** (assume for simplicity that the size of the input **n** is a **power of 3)** by:

- *analyzing* the **number of item comparisons** in every step of the algorithm. Give the **EXACT** number of comparisons in the **worst case** and the **best case**.
- *finding* the **recurrence relation** for the **number of item comparisons** performed.
- *using* backwards substitution (with the values from the **worst case** analysis) to find the **EXACT closed form** solution for the number of item comparisons performed.
- *expressing* the time **complexity** of this **3-WaySplit MergeSort** variant using the **Big-O notation**.

As a conclusion of your analysis **in addition** to all the steps above, answer the following questions:

- What is the EXACT worst case number of comparisons for the 2-way merge (original method) and the 3-way merge (this method)? Which one is more efficient?
- What is the complexity of the 2 methods? Express it using the Big-O notation. Which method is more efficient?

Place the analysis in a text file called Problem0.txt and include it in allfiles and then the pdf.

Problem1:

Implement the **iterative MergeSort** method as discussed in class to sort a collection of integer data stored in an array in **increasing/ascending** order by completing this frame while carefully following the instructions given in class. Solutions not using this frame and not following the given instructions will not receive consideration and therefore credit. All the 4 marked methods need to be completed as specified in order to receive credit using the required signatures and without adding any additional methods. Only simple local variables can be used in addition to the two 1-dimensional array of size n (specified by the user).
Test your method by running it on this script (name the file testMergeSort make it executable by setting the correct permissions with **chmod 700 testMergeSort**). Your execution should look similar to sorting_out.txt (not the same because the randomly generated numbers will be different).

Problem2:

Lab10Conclusions: Summarize in a few sentences what you have learned from working on this lab.

EXTRA CREDIT:(will NOT be considered for CREDIT without a correct Problem 1)

Extra credit I

Implement the **recursive QuickSort** using the Divide & Conquer approach **as discussed in class** to work on integer data stored in an **array** data structure. The method should sort the data in ascending (increasing) order and should also **count** and **display** the **number of item comparisons** and **swaps** that are performed on the considered input sequences. Make sure to use as base cases subproblems of size 1 and 2.
Test your method on **randomly selected/generated** input. Compare the theoretical analysis done in class with the results obtained from running the code on various input sequences. For the pivot consider either random choice - but not first index, median of 3 or any other **good** choice). Clearly document your choice.

Extra credit II

Time and **compare the execution** of the 2 methods on the **SAME** (sizable) input. Which method performs better for increasing input size?

Submit: Follow the instructions to turn in the lab electronically using **Lab10** for the name of your directory.
Develop your own extensive input suite in addition to the one provided and test your solutions thoroughly on it. Submit all the code developed for this lab and all sample runs on the input suite you developed.