SAN FRANCISCO STATE UNIVERSITY
Computer Science Department

CSC510 Section 04 – Analysis of Algorithms
Algorithm Challenge 2: Dividing and Conquering

Instructor: Jose Ortiz

Due on October 6 2020 before class

Full Name: ______________________________
Student ID: ______________________________

Assignment Instructions. Must read!

Note: Failure to follow the following instructions in detail will impact your grade negatively.

1. This algorithm challenge is worth 10%, and will be graded using a grading point scale where the maximum possible grade is 100 points. For instance, if your grade in this assignment is 85/100, then this is equivalent to 0.85*10%=8.5% of 10%
2. Part 4 of this algorithm challenge is extra-credit (2%). No partial extra-credit will be given here. In other words, this is either all or nothing
3. Take into account that in this type of assignments, I am more interested in the way you approach the problem rather than your final solution.
4. In this algorithm challenge, you don't need to write working code; only pseudocode

Problem Statement

1. Sort (in ascending order) the items in a file of size 2^x KIB using limited memory. Note that x is a unsigned integer where $x > 0$.

   (a) Rules:

      i. The file is located in disk (not in memory)

      ii. Memory is limited to 2 input buffers and 1 output buffer (4KIB each) Total memory capacity 12 KIB

      iii. Assume that the contents of the file are unsigned integers separated by a comma delimiter. (i.e 3,1,3,100,99...)

      iv. The unsigned integers are not sorted

      v. The file can contain duplicated integers

      vi. When in a file, a digit from an integer is 1 byte (datatype is char). When in a buffer, an integer is 4 bytes (data type is integer).

      vii. Pass number 0 can only use the output buffer. All the remaining passes can use all the available buffers in memory

      viii. All the buffers in memory support ± 4 *bytes* of additional memory allocation.

      ix. The merging process must be done using Merge Sort algorithm.

      x. Temporary files, in disk, can only hold a max size of $((\#pass + 1) * 4)KIB$

   (b) Input and Output

      i. Input: A file containing unsorted unsigned integers in the range of 0 and 100 (both inclusive). For example: 100,67,99,99,1,1,3,24,88,96,37,10,10,88,100,99,99

      ii. Output: A file containing the sorted integers from the input file. For example: 1,1,3,10,10,24,37,67,88,88,96,99,99,99,99,100,100

Your work starts here

1. Describe the algorithm to solve the problem for a given file of size 2^5 and 2^x (any given x). Note that x is a unsigned integer where $x > 0$. You can use tables, diagrams, pics, paragraph description..... to describe the algorithm. Be as clear as possible, and define clearly each step taken during the process.

2. Write the pseudocode that represents your algorithm from part 1. **Note pseudocoding is not the same as coding. In this challenge you are not required to code.**

3. Compute the complexity function, the I/O cost operations, and time complexity of your algorithm using your pseudocode from part 2. For the complexity function and time complexity use the substitution method and then check your result with the Master Theorem. For the I/O cost define it as a function of N. Where N is the number of blocks (4KIB) in pass 0 of this algorithm

4. (Extra-credit 2%) Modify your algorithm from part 1 to support files of any size (not only 2^x). For instance in part 1, we were asumming that the file size is a power of 2 such as 2KIB, 4KIB, 16KB..... However, your new modification should include files of any size such as 3KIB, 37KIB.....

   In addition, do you think that the time complexity of this algorithm has been optimized as a result of the modifications applied? Why?