CSC510: Analysis of Algorithms
Homework 2
Recurrences, Divide and Conquer

—

Full Name: Wing Lee
Student ID: 920558688

Assignment Instructions for students. Must read!

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

1. This homework is worth 15%. This homework has two problems. The second problem is divided into several parts.
2. Students must turn this assignment in on Canvas in PDF format, assignments submitted in a file format other than PDF will be considered not submitted and graded as such.
3. Blank assignments, or assignments missing work will be graded as such. No exceptions!. It is the responsibility of students to check if the assignment was submitted in the correct format.
4. Please read the late work policies stated in the syllabus. Homework submissions that do not follow these policies will get an unsatisfactory grade.

1. (3 points) Solve the following recurrence using the back substitution method, and check your results with the Master Theorem. Show ALL your work including at least three substitutions before you move to k steps, and computed summations. No credit will be given if students don't show all their work in detail.

Back Substitution

$$T(n) = 9T(\frac{n}{3}) + n^3 + n^2 + n + 1$$

$$T(\tfrac{n}{3}) = 9\left[9T(\tfrac{n}{3^2}) + (\tfrac{n^3}{3^2}) + (\tfrac{n^2}{3^2}) + \tfrac{n}{3^2} + 1\right] + n^3 + n^2 + n + 1$$

$$= 9^2 T(\tfrac{n}{3^2}) + \tfrac{n^3}{3} + n^2 + 3n + 9 + n^3 + n^2 + n + 1$$

$$= 9^2 T(\tfrac{n}{3^2}) + \left(\tfrac{n^3}{3} + n^3\right) + (2n^2) + (3n+n) + (9+1)$$

$$T(\tfrac{n}{3^2}) = 9^2\left[9T(\tfrac{n}{3^3}) + (\tfrac{n}{3^2})^3 + (\tfrac{n}{3^2})^2 + \tfrac{n}{3^2} + 1\right] + n^3 + n^2 + n + 1$$

$$= 9^3 T(\tfrac{n}{3^3}) + \left(\tfrac{n^3}{3^2}\right) + n^2 + 9n + 18 + \tfrac{n^3}{3} + n^3 + 2n^2 + 3n + n + 9 + 1$$

$$= 9^3 T(\tfrac{n}{3^3}) + \tfrac{n^3}{3^2} + \tfrac{n^3}{3} + n^3 + 3n^2 + 9n + 3n + n + 81 + 9 + 1$$

$$T(\tfrac{n}{3^3}) = 9^3\left[9T(\tfrac{n}{3^4}) + (\tfrac{n}{3^3})^3 + (\tfrac{n}{3^3})^2 + \tfrac{n}{3^3} + 1\right] + n^3 + n^2 + n + 1$$

$$= 9^4 T(\tfrac{n}{3^4}) + \tfrac{n^3}{n^3} + n^2 + 27n + 729 + \tfrac{n^3}{3^2} + \tfrac{n^3}{3} + n^3 + 3n^2 + 9n + 3n + n + 81 + 9 + 1$$

$$= 9^4 T(\tfrac{n}{3^4}) + \tfrac{n^3}{3^3} + \tfrac{n^3}{3^2} + \tfrac{n^3}{3} + n^3 + 4n^2 + 27n + 9n + 3n + n + 729 + 81 + 9 + 1$$

k steps ...

$$T(n) = 9^k T(\tfrac{n}{3^k}) + n^3\left(\tfrac{1}{3^0} + \tfrac{1}{3^1} + \tfrac{1}{3^2} + 1 + \tfrac{1}{3^3} + \tfrac{1}{3^{k-1}}\right) + kn^2 + n\left(1 + 3^1 + 3^2 + 3^3 + 3^{k-1}\right) + \left(9^0 + 9^1 + 9^2 + 9^3 + 9^4 + 9^{k-1}\right)$$

$$T(n) = 9^k T(\tfrac{n}{3^k}) + n^3\left(\sum_{i=0}^{k-1} (\tfrac{1}{3})^i\right) + kn^2 + n\left(\sum_{i=0}^{k-1} (1)^i\right) + \left(\sum_{i=0}^{k-1} (9)^i\right)$$

$n = 9^k$

$T(\tfrac{n}{3^k}) = 1$

$\log_9 n = k$

Master Theorem

$a = 9$
$b = 3$
$k = 3$
$p = 0$

$\log_b a = \log_3 9 = 2$

Case 1 = $a > k \Rightarrow 2 > 3$? false

Case 2 = $a = k \Rightarrow 2 = 3$? false

Case 3 = $a < k \Rightarrow 2 < 3$ ✓ TRUE

$\Theta(n^3 \log^0 n) = \Theta(n^3 * 1)$

$= \Theta(n^3)$

2. (12 points) **Divide and Conquer**

The following description is the problem statement. Please don't do work here

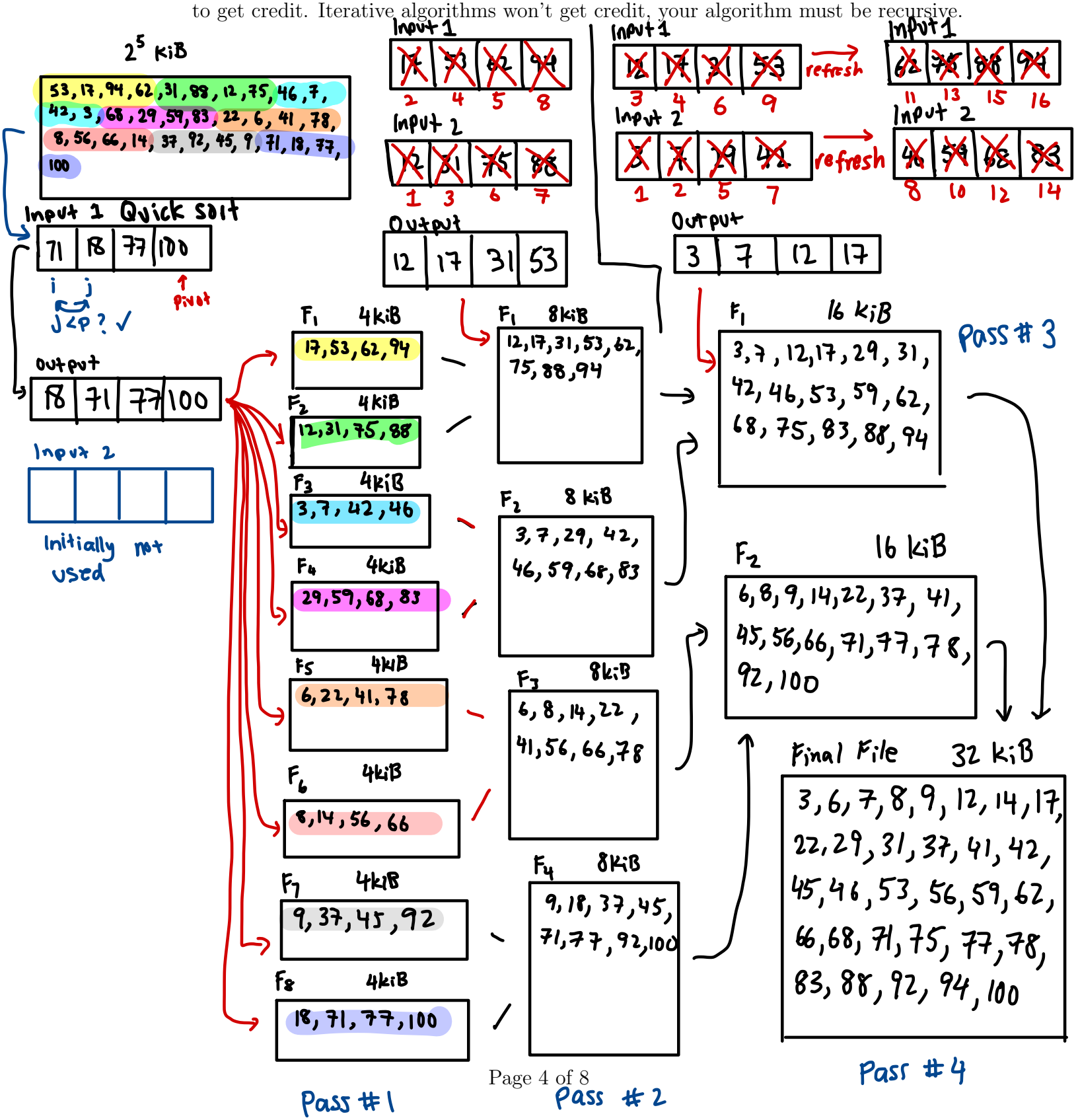
Given a file F that contains integers separated by commas, and a machine with limited memory, sort the contents of this file in ascending order without missing a single bit.

- Assumptions for this problem:
  - The original file F is given as the input for the problem and the size of the file is 2^x KiB, where $x \in \mathbb{N}$
  - You are running this algorithm in a machine with limited memory. Your machine has 12 KiB of memory only, where you have an additional 4 bytes reserved to a pointer that may perform reading or writting operations.
  - In memory, you are only allowed to have 3 buffers (or arrays), one input buffer, and two output buffers. The size of each of the buffers is 4 KiB. Buffers in memory only support integers.
  - You are allowed to store data into temporal files in your hard drive while performing the algorithm. However, these temporal files must be deleted as soon as they are emptied.
  - When placed in a file, every digit of an integer is a char (1 byte). On the other hand, when placed in buffers in memmory, an integer is always 4 bytes.
  - This problem allows duplicated integers
  - Commas in files are always chars (1 kb)
  - The output file must be the same size and contains the same data as the original file, but sorted in ascending order. You can't miss a bit.
  - The algorithm to solve this problem must be recursive

pass = $(\log_2 32) - 1$

Your work starts in this page!!!

(a) (3 points) Design an algorithm for this problem using a Divide and Conquer approach for a given input file of 2^5 KiB. DO NOT write pseudocode in this section, you can only use diagrams as seen in class for other similar problems. Show all the steps in this process to get credit. Iterative algorithms won't get credit, your algorithm must be recursive.



(b) (3 points) Create the recursive pseudocode implementing your Divide and Conquer algorithm for an input file of size 2^x KiB. Note that if you write code in this section or iterative pseudococe, you won't get credit. Your algorithm needs to be implemented using recursive pseudocode to get credit for this problem. In this problem, handwriting is not allowed. Your pseudocode needs to be typed.

Algorithm Divide and Conquer for input file

```
function mergeFile(pass, N)
  if (pass == 0)
    quickSortFile(input1)
    return
  chunk_size = N / (2^pass)
  for i from 1 to 2^pass:
    read input1 from file F starting at (i-1) * chunk_size
    read input2 from file F starting at i * chunk_size
    merge input1 and input2 into output Buffer
    write the output buffer to a temporary file
  mergeFile(pass-1, N)

function fileSort(input_file, output_file, N, total_memory)
  if (N <= total_memory):
    initialize: input1 <- 4 KiB Buffer
    initialize: input2 <- 4 KiB Buffer
    initialize: output <- 4 KiB Buffer
    chunk_size = total_memory / 2
    input1_pointer = 0
    input2_pointer = 0
    while not end of input_file:
      byte = read one byte from input_file
      if input1_pointer < 4 KiB:
        input1[input1_pointer] = byte
        increment input1_pointer
        if byte == ',':
          input1_pointer -= 1
      else if input2_pointer < 4 KiB:
        input2[input2_pointer] = byte
        increment input2_pointer
        if byte == ',':
          input2_pointer -= 1

        if input2_pointer == 4 KiB:
          merge input1 and input2 into the output buffer
          write the output buffer to a temporary file

          input1_pointer = 0
          input2_pointer = 0

    mergeFile(pass, N)
    fileSort(temporary_file, output_file, N, total_memory)
```

(c) (3 points) Compute the Theta time complexity of your recursive pseudocode. First compute complexity and time complexity using Backsubstitution, and then check your results with the Master Theorem.

```
function mergeFile(pass, N)                                   <- T(n)
  if (pass == 0)
    quickSortFile(input1)
    return
  chunk_size = N / (2^pass)
  for i from 1 to 2^pass:                                     -> n
    read input1 from file F starting at (i-1) * chunk_size
    read input2 from file F starting at i * chunk_size
    merge input1 and input2 into output Buffer
    write the output buffer to a temporary file
  mergeFile(pass-1, N)

function fileSort(input_file, output_file, N, total_memory)
  if (N <= total_memory):
    initialize: input1 <- 4 KiB Buffer                        n
    initialize: input2 <- 4 KiB Buffer                        n
    initialize: output <- 4 KiB Buffer                        n
    chunk_size = total_memory / 2                             - T(n/2)
    input1_pointer = 0
    input2_pointer = 0
    while not end of input_file:
      byte = read one byte from input_file
      if input1_pointer < 4 KiB:
        input1[input1_pointer] = byte
        increment input1_pointer
        if byte == ',':
          input1_pointer -= 1
      else if input2_pointer < 4 KiB:
        input2[input2_pointer] = byte
        increment input2_pointer
        if byte == ',':
          input2_pointer -= 1

        if input2_pointer == 4 KiB:
          merge input1 and input2 into the output buffer      - T(n/2)
          write the output buffer to a temporary file

          input1_pointer = 0
          input2_pointer = 0

    mergeFile(pass, N)
    fileSort(temporary_file, output_file, N, total_memory)
```

O(n)

Back substitution

$T(n) = 2T(\frac{n}{2}) + n$

$T(n) = 2^2 T(\frac{n}{2^2}) + \frac{2n}{2} + n$

$T(n) = 2^2 T(\frac{n}{2^2}) + \frac{2n^2}{2} + \frac{2n}{2} + n$

k steps

$T(n) = 2^k T(\frac{n}{2^k}) + n\left(\frac{2}{2} + \frac{2^2}{2^2} + \frac{2^3}{2^3} + \dots + \frac{2^{k-1}}{2^{k-1}}\right)$

$T(\frac{n}{2^k}) = 1$

$2^k = n$

$k = \log_2 n$

$T(n) = 2^{\log_2 n} T(1) + n\left(\sum_{i=0}^{k-1} (\frac{2}{2})^i\right)$

$= n^{\log_2 2} T(1) + 2n\left(1 - \frac{n^{\log_2 2}}{n}\right)$

$T(n) = \theta(n)$

master theorem

$T(n) = 2T(n/2) + n$

$a = 2$

$b = 2$

$k = 1$

$p = 0$

$\log_2 2 = 1 = k$

$\theta(n \log^0 n)$

$= \theta(n)$

(d) (3 points) Optimize your pseudocode from part (b) so now instead of supporting only input files of 2^x KiB, your algorithm will support files of any size (i.e. x KiB). Also, comment on how this optimization will affect your Theta time complexity from part (c).

```
function mergeFile(pass, N, total_memory)
  if pass == 0
    quickSortFile(input1)
    return
  chunk_size = min(N, total_memory / (2^pass))
  for i from 1 to 2^pass:
    read input1 from file F starting at (i-1) * chunk_size
    read input2 from file F starting at i * chunk_size
    if comma is last in input buffer
      move pointer back -1
    merge input1 and input2 into output Buffer
    write the output buffer to a temporary file
  mergeFile(pass-1, N, total_memory)

function fileSort(input_file, output_file, N, total_memory)
  if N <= total_memory:
    initialize: input1 <- buffer of size N
    initialize: input2 <- buffer of size N
    initialize: output <- buffer of size N
    read the entire input_file into input1
    sort input1 in memory
    write input1 to the output_file
  else:
    pass = 0
    while N > total_memory:
      N = N / 2
      pass += 1
    chunk_size = total_memory / 2
    while not end of input_file:
      byte = read one byte from input_file
      if input1_pointer < 4 KiB:
        input1[input1_pointer] = byte
        increment input1_pointer
        if byte == ',':
          input1_pointer -= 1
      else if input2_pointer < 4 KiB:
        input2[input2_pointer] = byte
        increment input2_pointer
        if byte == ',':
          input2_pointer -= 1

        if input2_pointer == 4 KiB:
          merge input1 and input2 into the output buffer
          write the output buffer to a temporary file

          input1_pointer = 0
          input2_pointer = 0

    mergeFile(pass, N)
    fileSort(temporary_file, output_file, N, total_memory)
```

I think this would be slower because you are now supporting all files of any size reason is because of the merging and quicksort and also pointers in consideration

So this would probably be $\Theta(n \log n)$

(e) **(This part is optional and its worth 2 points extracredit)** Use your favorite programming language to implement your pseudocode from part (b) or (d). DO NOT copy and paste your code in this document. Instead, write below the link pointing to the online code editor where you hosted your code. I recommend to use Replit, but you are free to use any other online code editor. Note that your code must have at least 3 unit test with different input files so I can check if the output file is correct. Students completing this section won't get the extracredit if I can't access your code (i.e. broken link or url not loading)

Note: students are allowed to do the extracredit problem ONLY if they completed parts (a), (b), (c) and (d) of this problem