CSC510 Analysis of Algorithms
Section 04

Homework 3–
Divide and Conquer

Instructor: Jose Ortiz

Full Name: John King
Student ID: 920628771

Homework Instructions. All Students Must Read This!

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

1. This homework is worth 10% of your final grade.
2. Handwriting work **IS NOT** allowed in this homework.
3. 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.
4. 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.
5. Any indication of work done by AI tools such as ChatGPT won't get credit. Please read the policies outlined in the course's syllabus regarding this matter.
6. Please read the late work policies stated in the syllabus. Homework submissions that do not follow these policies will get an unsatisfactory grade.

Problem Statement

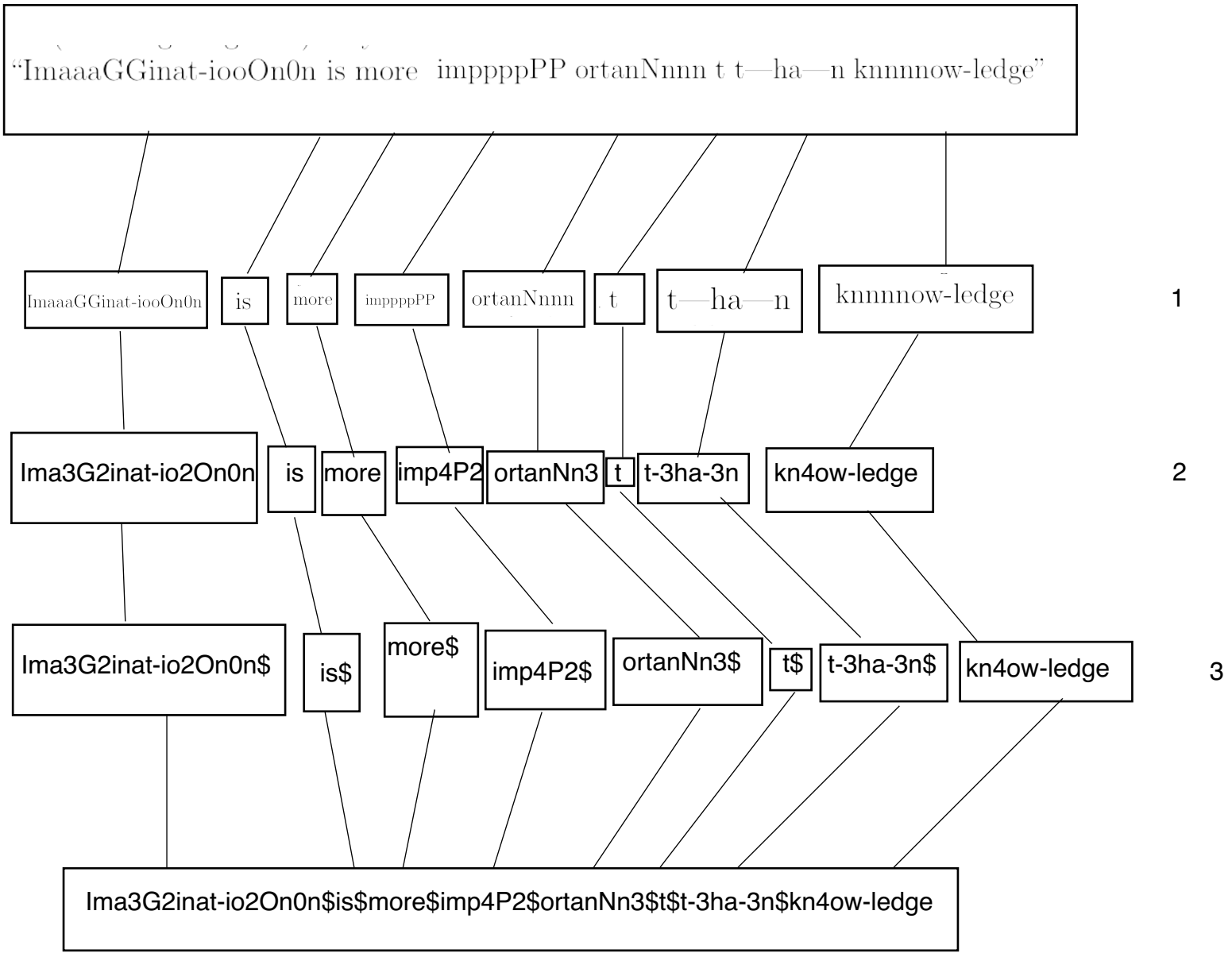
You are given string S. Your goal is to design an algorithm that creates segments from the content of the string encoded as follows: if a segment contains a single character repeated n times, it should be encoded as (character)(n) (e.g., "aaa" becomes "a3", and "AA" becomes A2). If a segment contains a mix of unique characters (each appearing exactly once), it should be encoded by simply concatenating those characters (e.g., "abc" becomes "abc", or "p" becomes "p"). The algorithm should aim to compress the length of the original string.

- **Problem Constraints:**
  - **Students MUST solve this problem using a Divide and Conquer approach, only.** No credit will be given for other approaches used.
  - String S contains alphanumeric characters (including symbols) from the English alphabet. Letters may be uppercase and lowercase, and may be organized into words.
  - There may be word spacing between words in string S. Words spacing must also be considered in this problem and must be marked with the symbol $. For example, " ", will become "$4" because it has 4 word spacing
  - The order of the string S must be preserved. For example, "aabaa" IS NOT "a4b" or "ba4". Instead, it will become "a2ba2"
  - The length of string S is between 1 and 10^4.
  - The implementation of the Divide and Conquer approach for this problem MUST BE recursive.
  - The algorithm **MUST** have a constant space complexity of $\Theta(1)$
- **Input Format:**
  - A string S of size n
- **Output Format:**
  - A string S of size m where $m <= n$
- **Example:**
  - **Input:** a string $S =$ "PPPphhhysssssssiccccccs issss thhhEEE uuuunIveRRRRssssse's oPPpppeeeeeratttttting sys-tem."
  - **Output:** $S =$ "P3ph3ys8ic6s$is4$th3E3$6u4nIveR4s5e's$oP2p3e5rat5ing$sys-tem."

Your work starts here. Good Luck!

1. (4 points) Design an algorithm for this problem using a Divide and Conquer approach.

   (a) (2 points) Create a **step-by-step** graphical visualization (i.e using diagrams) of your solution for this problem using the following string S = "ImaaaGGinat-iooOn0n is more imppppPP ortanNnnn t t—ha—n knnnnow-ledge". Students are allowed to hand-drawn their solutions for this problem.



(b) (2 points) Create an step-by-step algorithm for this problem (in plain english). Your algorithm must work for all the given strings S, including empty strings. In addition, your algorithm must consider all the corner cases that you may encounter in this problem. Neither code nor pseudocode is allowed in this problem

1) find the first space encountered in S
2) if the string S is empty, return the input
3) recursively run the function for the characters between spaces
4) on each run of the function,
    a) count the characters that are the same and next to each other and add them up until a space is reached
    b) add the character and the number of characters to the final print string for each different character in this section
    c) delete the string from the begin to the first space encountered, find the next space and run again recursively
    d) if there are no more spaces, repeat steps a-b
5) print the final string

2. (2 points) Pseudocode your algorithm from problem (1) part (b). Note that your pseudocode must be understood also by non-technical readers to get credit for this problem, and it must be pseudocoded using the Divide and Conquer approach defined by your algorithm

```
FUNCTION HW(S, substringStartNum, finalS)
    INITIALIZE: currchar
    INITIALIZE: newcurr := First character of S
    INITIALIZE: count =: 0
    INITIALIZE: spaceNum
    INITIALIZE: spaceCount
    IF there is a space in S
        spaceNum =: index of first space
    ELSE
        spaceNum =: 0
    END IF
    INITIALIZE: i =: spaceNum
    FOR i < length of S
        INITIALIZE: char =: spaceNum
        IF char is a space
            INCREMENT: spaceCount
        ELSE
            substringnum =: i
            exit loop
        END IF
    END FOR
    INITIALIZE: j =: 0
    FOR j < substringnum
        currchar =: S at index j
        IF currchar = newchar
            INCREMENT: count
        ELSE
            newcurr =: curr
            finalstr =: add character if S at j-1, as well as count to finalstr
            IF curr = a space
                finalstr =: add a $ and space count to finalstr
            END IF
            count =: 0
            DECREMENT: j
        END IF
    END FOR
    S =: cut off beginning of string up until substringNum
    IF substringNum = 0
        newcurr =: character at index 1 of S
        FOR j < length of S
            curr =: S at index j
            IF curr = newcurr
                INCREMENT count
            END IF
            IF j = length of S - 1 or curr not = newcurr
                newcurr =: curr
                finalstr = S at index j and count added to finalstr
                count =: 1
            END IF
        END FOR
        finalstr =: replace all 0 in finalstr to blank
        finalstr =: replace all 1 in finalstr to blank
        RETURN finalstr
    END IF
    RETURN HW(S, substringNum, finalstr)
END FUNCTION
```

3. (3 points) Compute the Theta time complexity of your algorithm:

   (a) (1 point) Show step-by-step the process to create your $T(n)$ function from your pseudocode in problem (2).

   See steps above

   T(n/2) + n + n + n + 14

   T(n) = T(n/2) + 3n + 14

   (b) (1 point) Use the Master Theorem to compute the Theta time complexity of your algorithm from your $T(n)$ function. Show the process step-by-step to get credit

   f(n) = Θ(n^k log^p n)

   T(n/2) + 3n + 14

   case 3
   p = 0 so

   a = 1
   p = 0

   b = 2
   k = 1

   logb(a) = 0

   logb(a) < k = 0 < 1

   Θ(n1 log^0 n) = Θ(n)

   (c) (1 point) Prove that the space complexity of your algorithm meets the constraint outlined in the problem statement (constant space complexity $\Theta(1)$)

   The space complexity is Θ(1) because the space is not being increased dependent on S. By the way the code is written, S could only ever get smaller or remain the same if there are no repeat characters. If the string got longer on each run, it could increase the space complexity, but since it is impossible to get bigger, it could only ever be Θ(1). Since S is only initialized in the first run, and nothing is getting appended each run, the space complexity is proven to be Θ(1) because no memory is getting increased dependent on the size of n.