Algorithms and Data Structures

Assignment 5

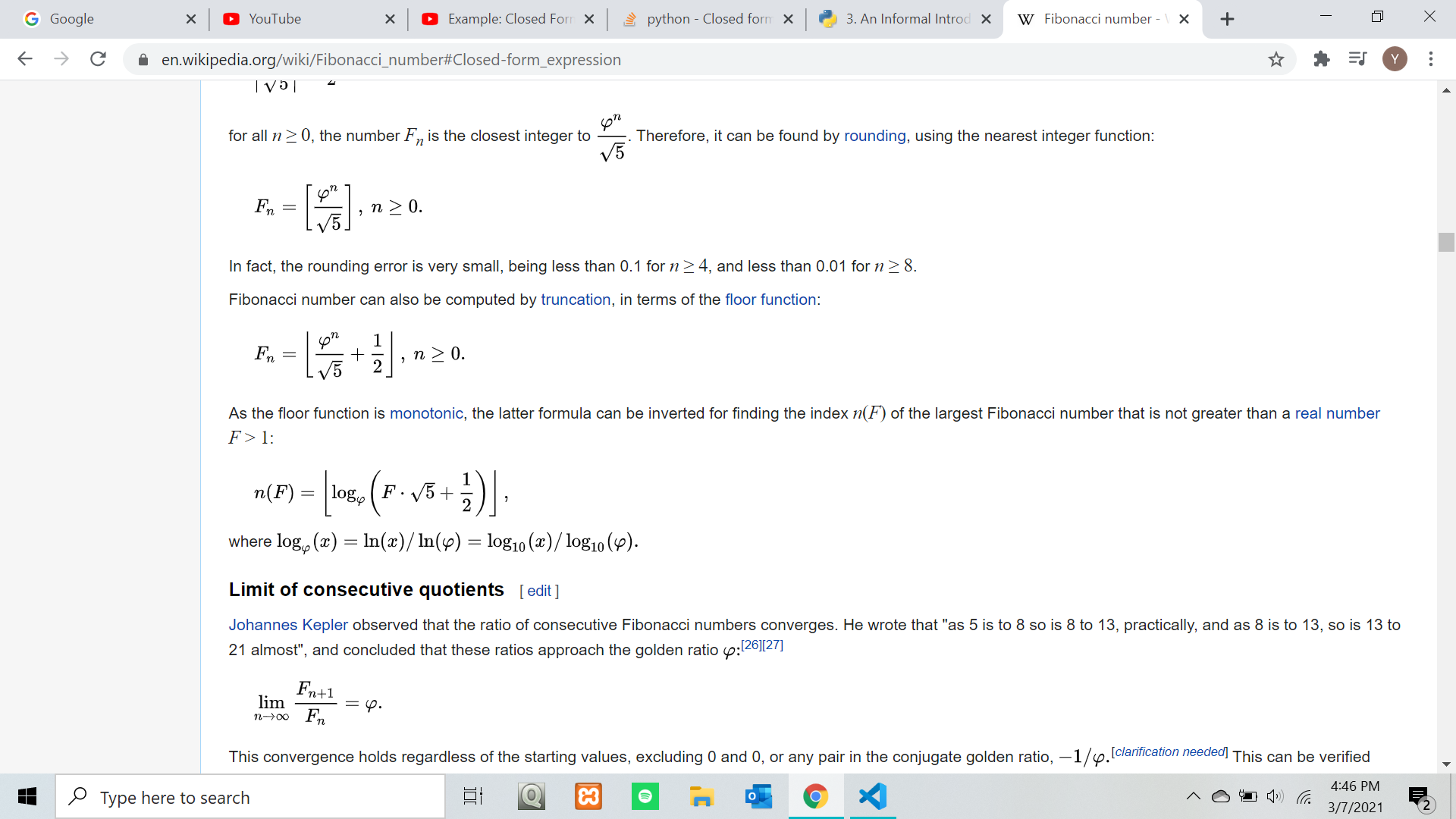
Yen Ling Wong

[y.wong@jacobs-university.de](mailto:y.wong@jacobs-university.de)

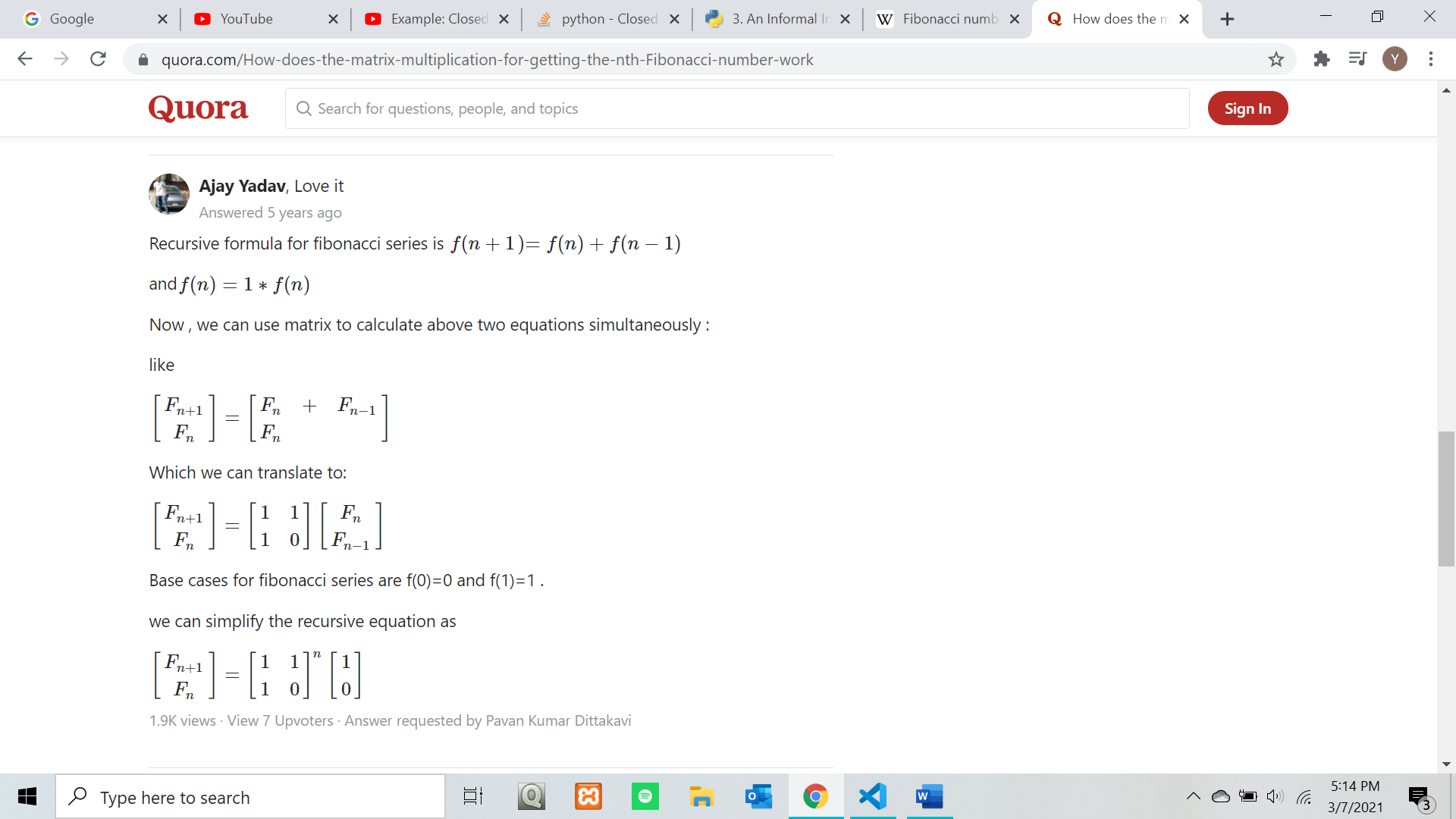
Problem 5.1 (Fibonacci Numbers)

1. Implement all four methods to compute Fibonacci numbers that were discussed in the lecture: (1) naive recursive, (2) bottom up, (3) closed form, and (4) using the matrix representation.

Enclosed in the zip file is a Python file a5\_p1.py with the implementation of all four methods. For the closed form method, the formula was derived from the picture below. (Source: https://en.wikipedia.org/wiki/Fibonacci\_number#Closed-form\_expression)



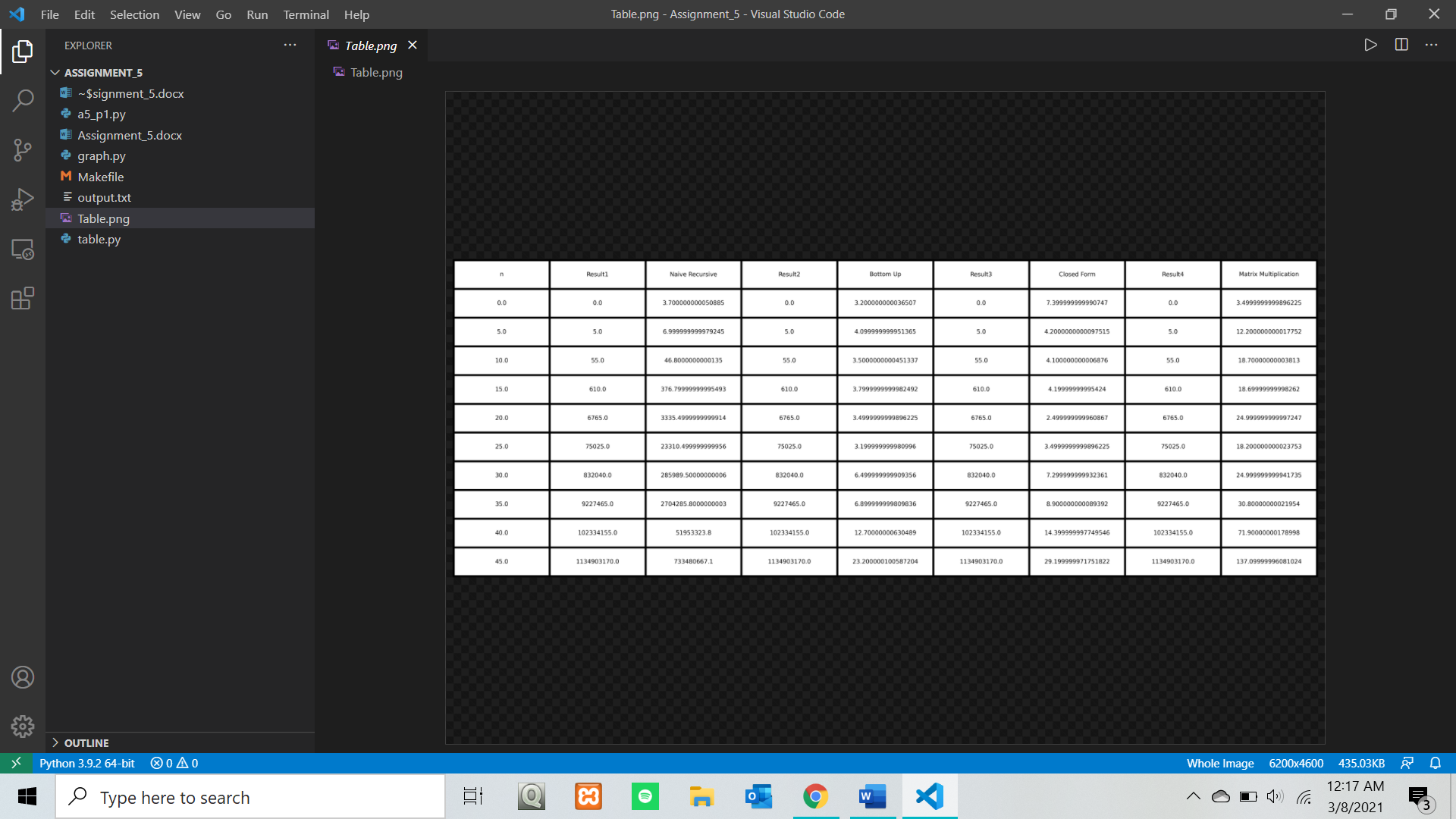
For matrix multiplication, the formula is as follows:



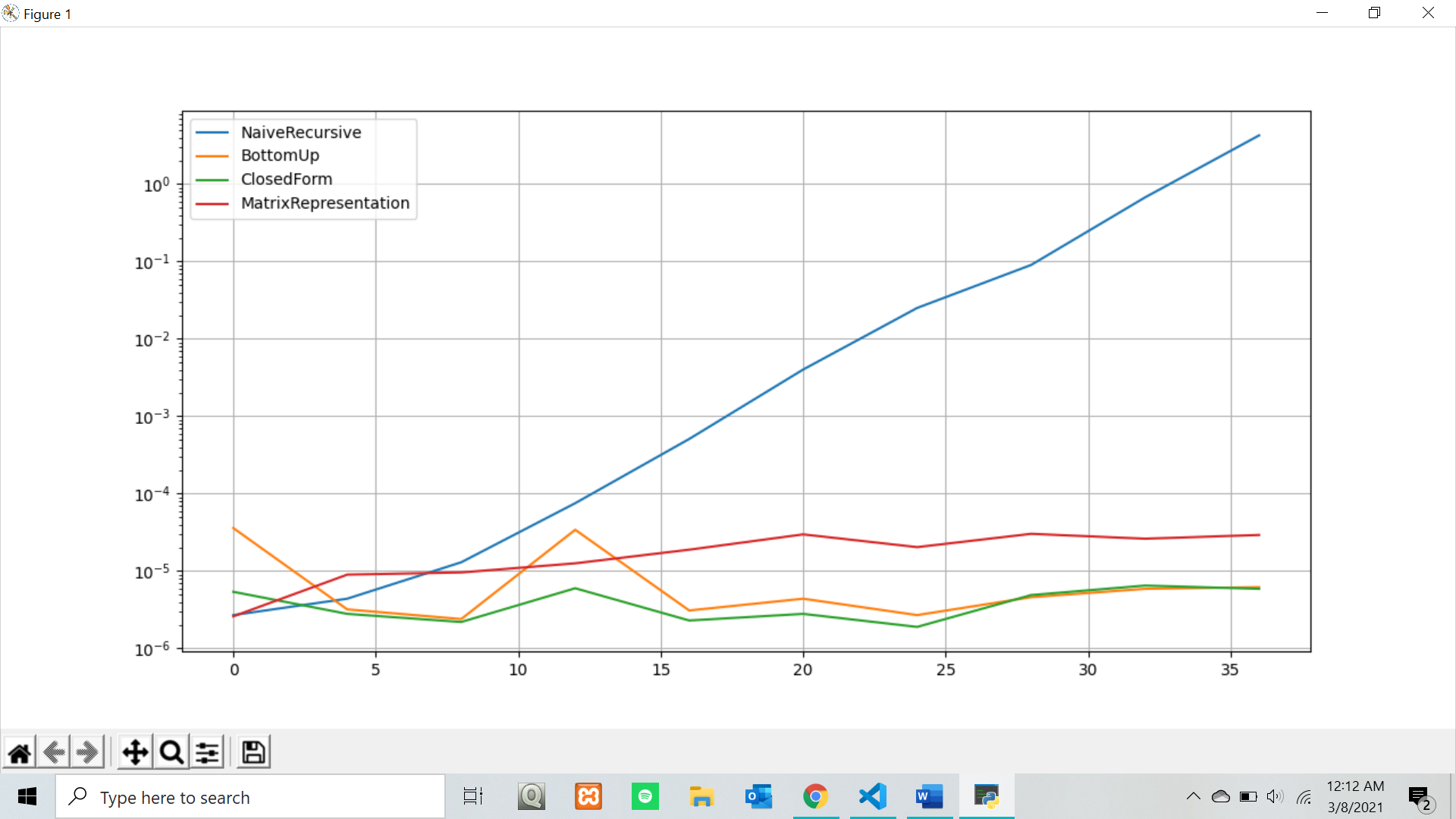
Source: <https://www.quora.com/How-does-the-matrix-multiplication-for-getting-the-nth-Fibonacci-number-work>

The matrix [1 1] [ 1 0] is multiplied n times to add F[n+1] and F[n].

1. The code in a5\_p1.py will write the results into a text file named output.txt. The code in table.py will read from said file and produce a graph called Table.png.



1. From my results, all methods do return the same value of n. However, in practice, the closed form method might produce some rounding errors because of the square root. I have solved this by typecasting the return value of the function as an integer.
2. The line plot is produced from graph.py. Attached is an image of the result, which is also included in the zip file.

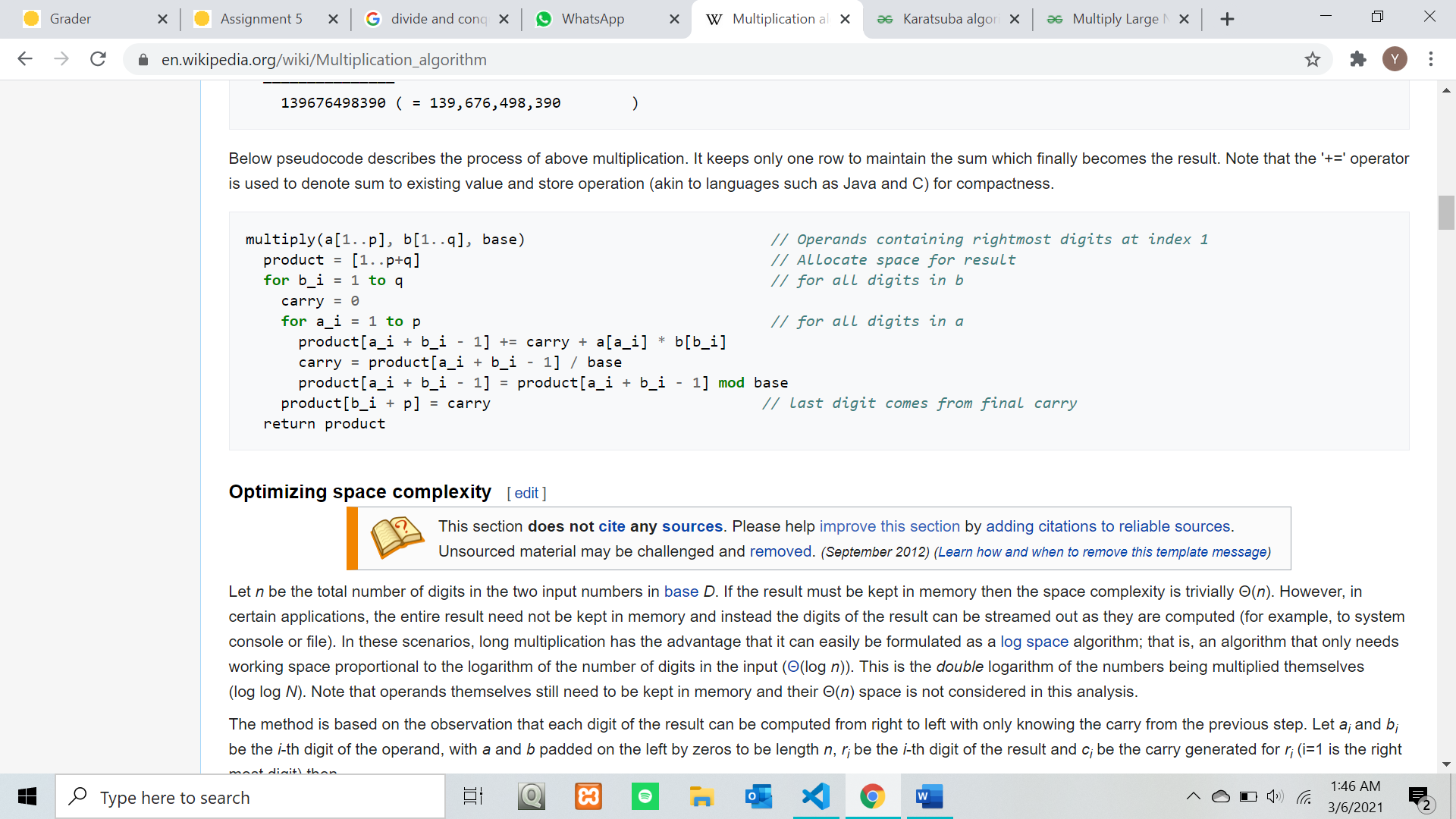


Problem 5.2 (Divide & Conquer and Solving Recurrences)

Consider the problem of multiplying two large integers a and b with n bits each (they are so large in terms of digits that you cannot store them in any basic data type like long long int or similar). You can assume that addition, subtraction, and bit shifting can be done in linear time, i.e., in Θ(n).

1. (2 points) Derive the asymptotic time complexity depending on the number of bits n for a brute-force implementation of the multiplication.

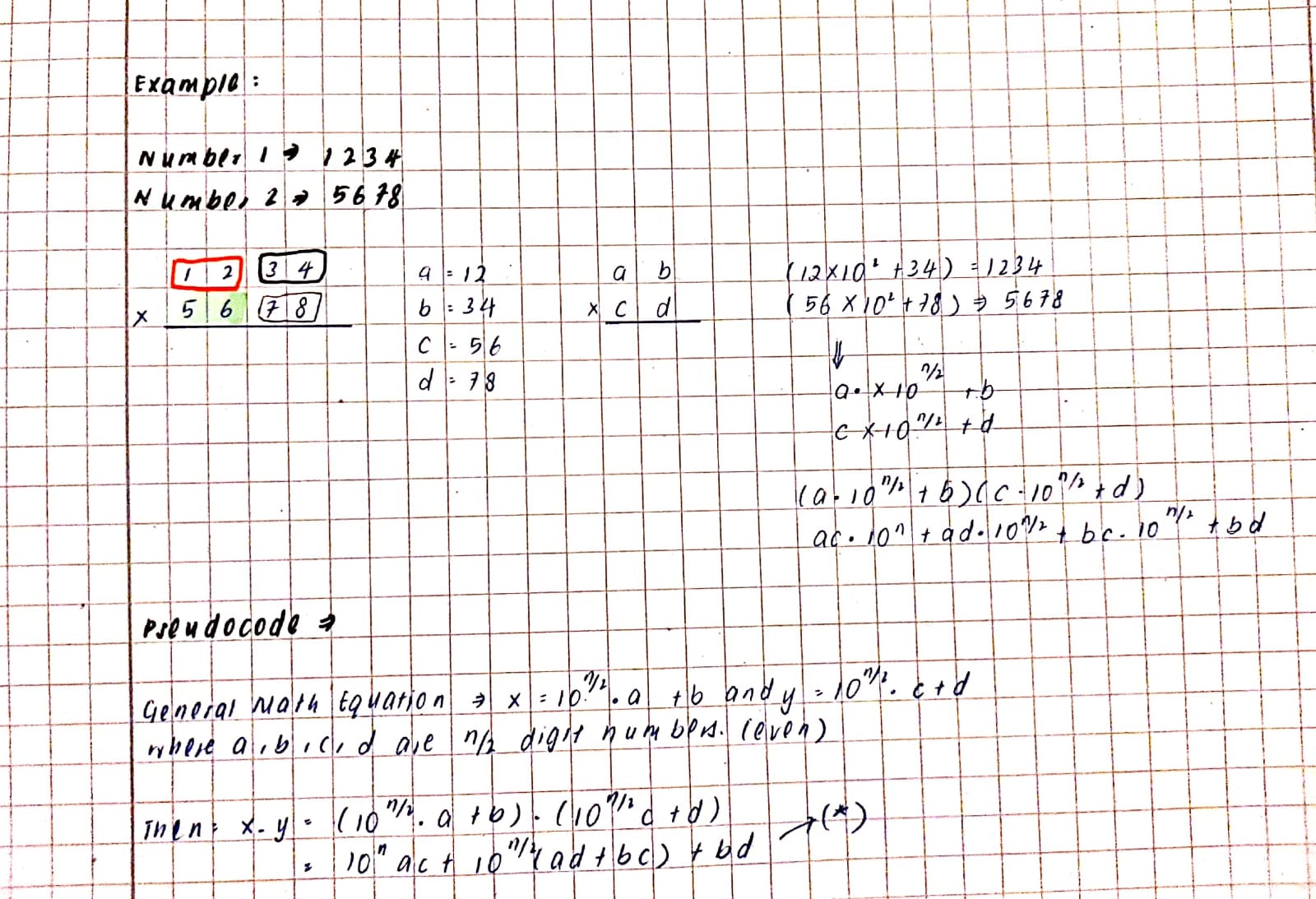
Pseudo-code of Brute Force Implementation for Decimal Representation of Long Integers

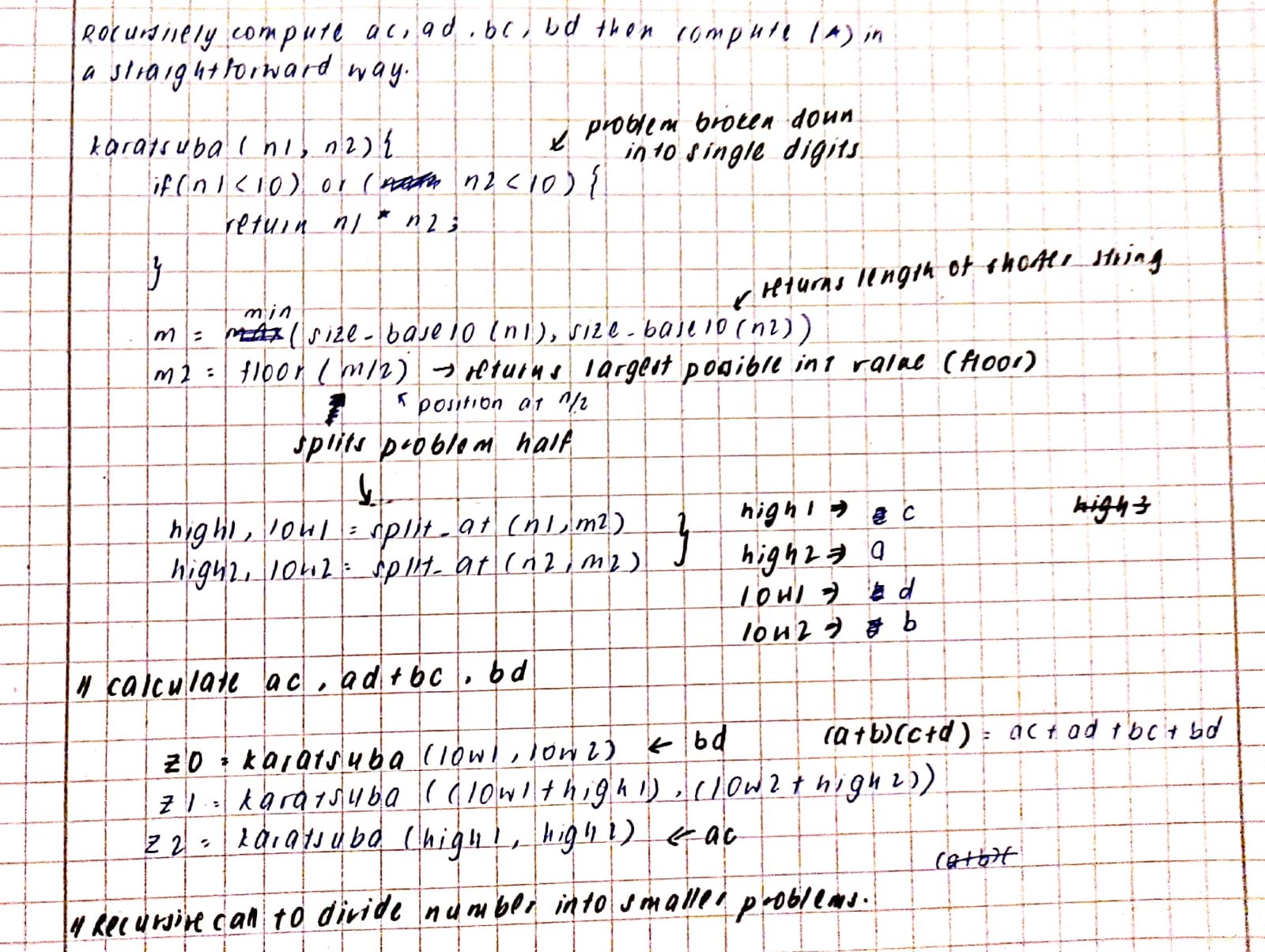


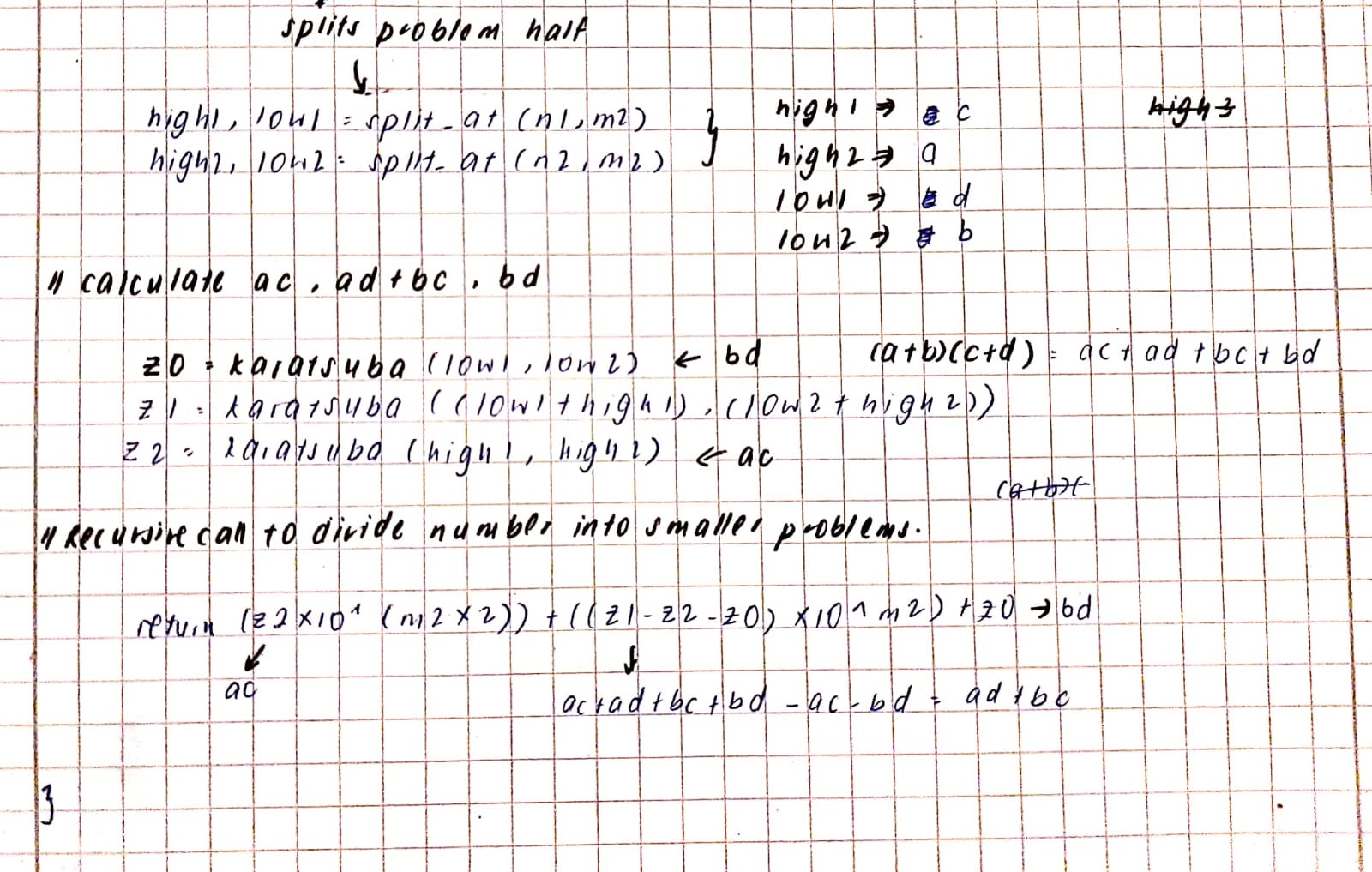
Source: <https://en.wikipedia.org/wiki/Multiplication_algorithm>

Explanation: Lines 6 to 8 represent simple arithmetic operations such as addition, division and mod. In line 6, the current digit of both numbers is multiplied then added to the stored result from the previous operation and the value of ‘carry’, which is the bit overflow from the previous calculation. Then, the new ‘carry’ is calculated in line 7 and brought to the next iteration and in line 8, the result is calculated again using mod (how much it overflows from the base). These three simple arithmetic operation costs constant time each, which is 3Θ(1) in total. Given that a number has n bits, the total cost would be n \* 3Θ(1) = 3Θ(n) (linear time). Since there is a nested for loop within an outer for loop, the total operations done would be p \* q times, and the asymptotic time complexity is Θ(n^2).

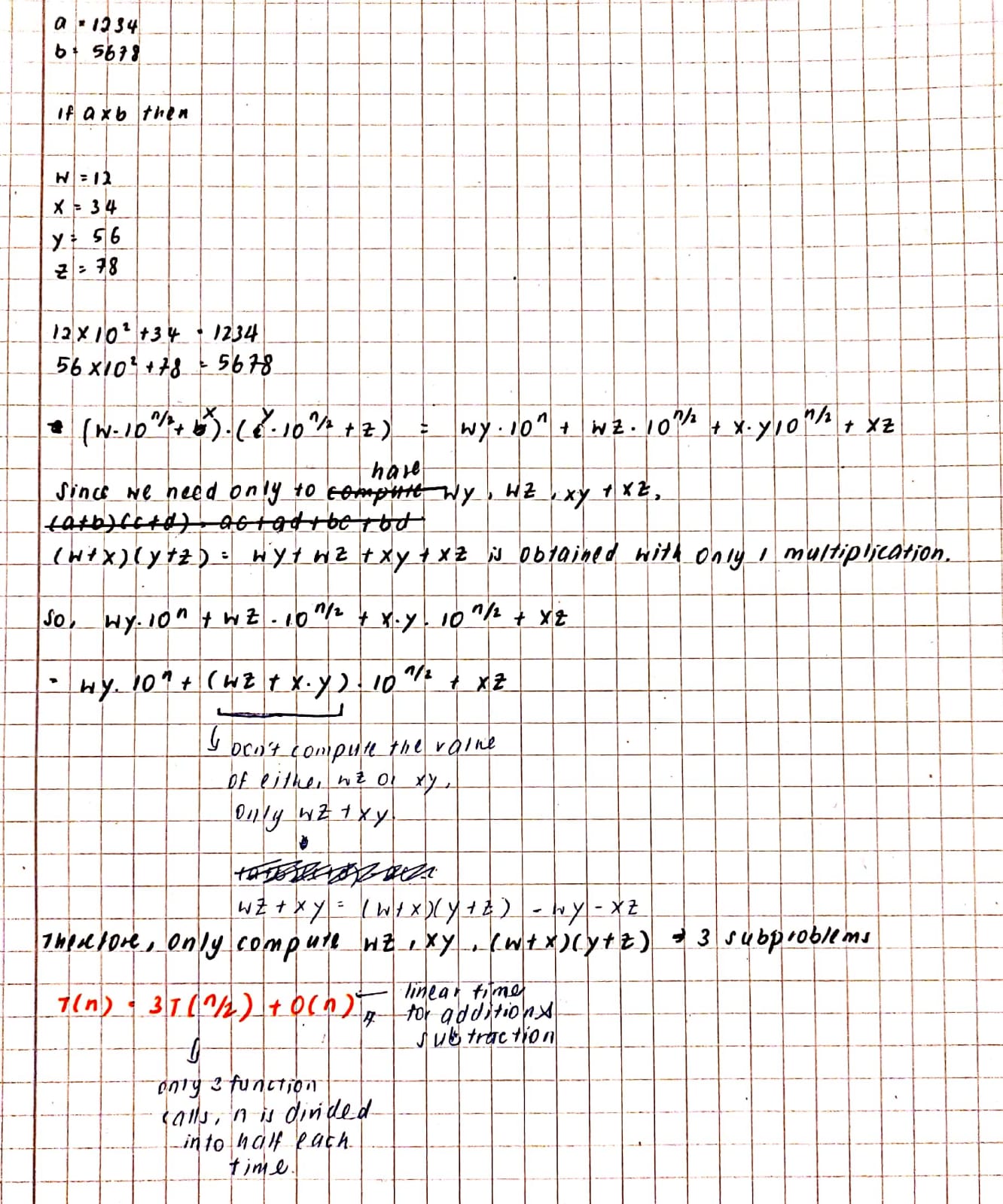
1. Source: https://en.wikipedia.org/wiki/Karatsuba\_algorithm#Pseudocode



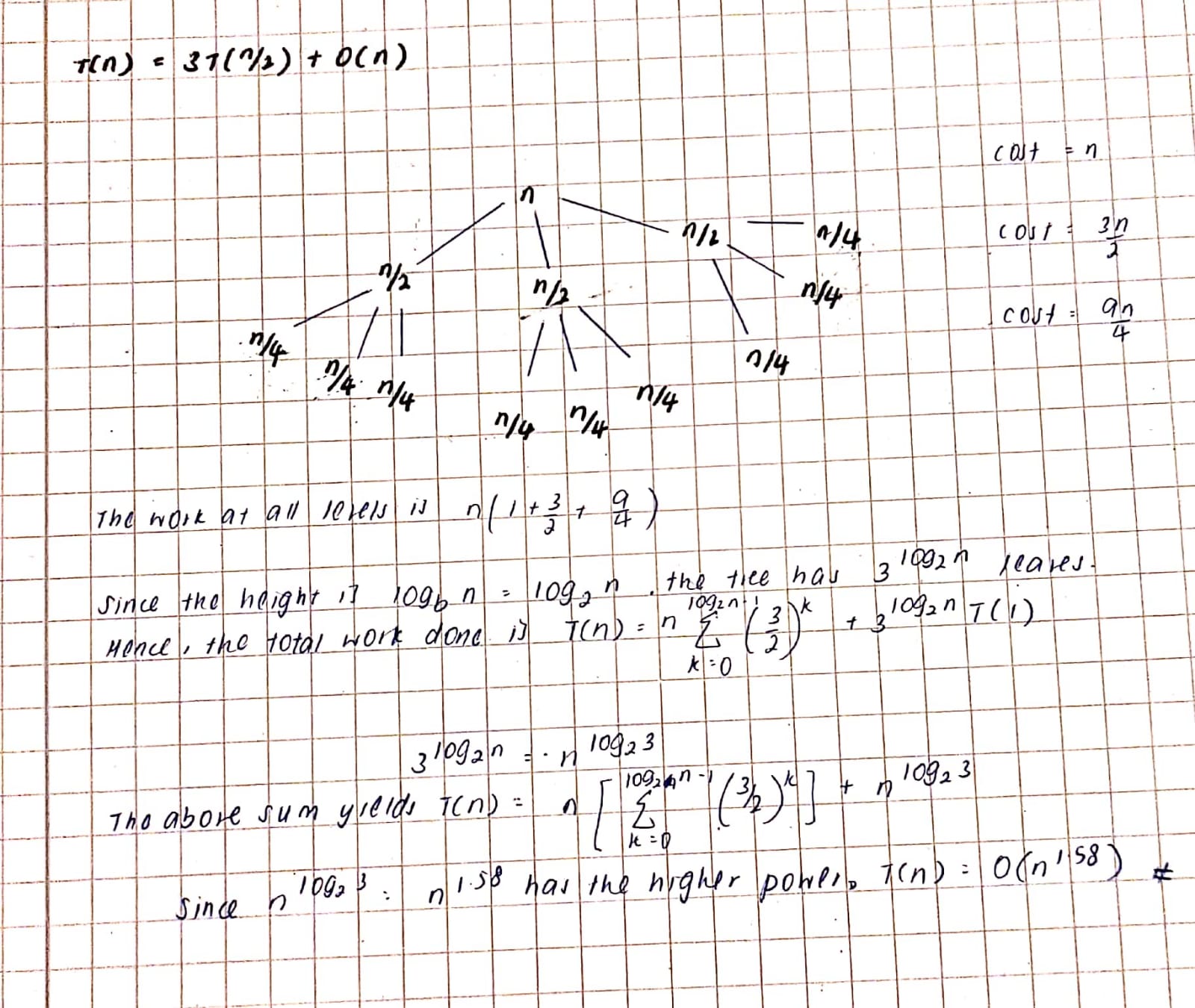




1. Derive a recurrence for the time complexity of the Divide & Conquer algorithm you developed for subpoint (b).



1. Solve the recurrence in subpoint (c) using the recursion tree method



1. Validate the result you got in subpoint (d) by using the master theorem to solve the recurrence again.

