Name: Ada Onyekaba

CS5008

Spring 2021 March 18

Assignment 8, Primer on proofs

We want to continue to get more comfortable with the mathematical notation used in Algorithms. In problem 1, you are going to write *in plain English* what the expression is and then solve the statement.

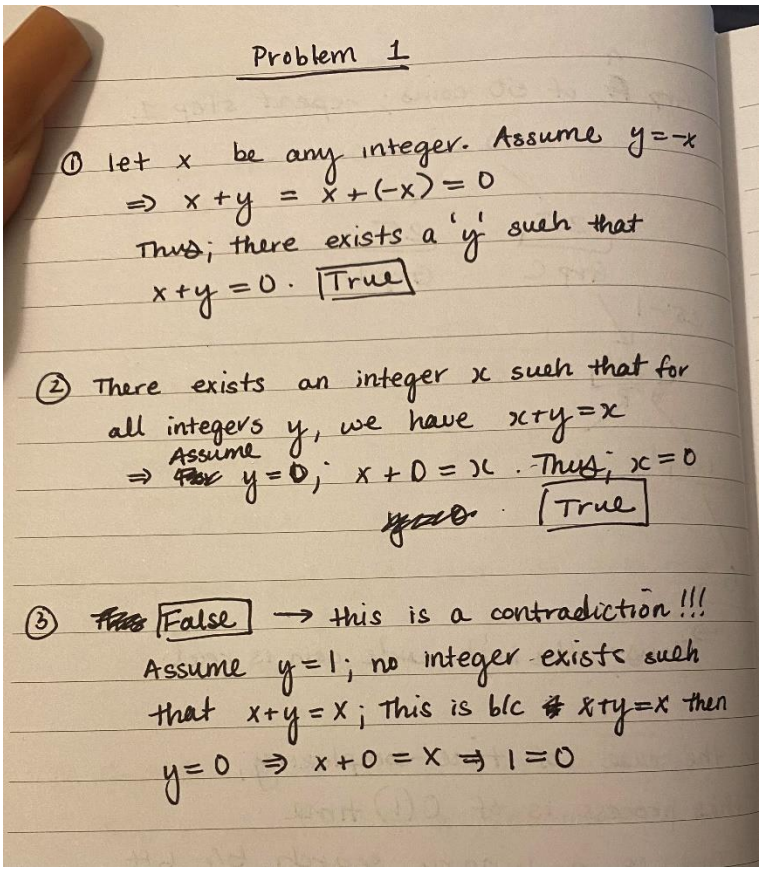
Problem 1 - Quantifiers

Write the following statements as English sentences, then decide whether those statements are true if x and y can be any integers. When deciding if x and y can be any integers, prove your claim with a convincing argument.

1. $\forall x \exists y : x + y = 0$

2. $\exists y \forall x : x + y = x$

3. $\exists x \forall y : x + y = x$



In problem 2 and problem 3, we want to solidify our understanding of Big-O notation. Remember, Big-O notation is about the growth of a function as n grows asymptotically large.

Problem 2 - Growth of Functions

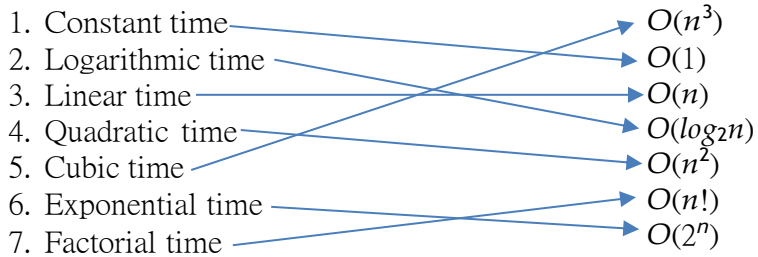
Organize the following functions into six columns. Items in the same column should have the same asymptotic growth rates (they are big-O and big-Θ of each other). If a column is to the left of another column, all its growth rates should be slower than those of the column to its right.

n^2, $n!$, $n \log_2 n$, $3n$, $5n^2 + 3$, 2^n, 10000, $n \log_3 n$, 100, $100n$

10000| log2n, 3log2n | 3n, 100| nlog2n, nlog3n | 5n2+3, n2 | 2n, n!

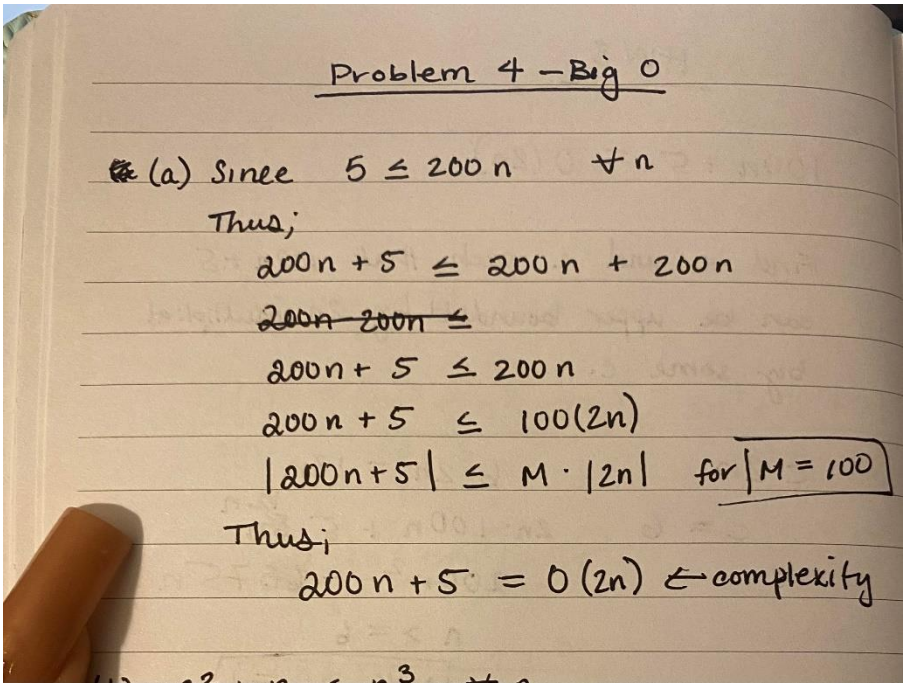
Problem 3 - Function Growth Language

Match the following English explanations to the *best* corresponding Big-O function by drawing a line from the left to the right.

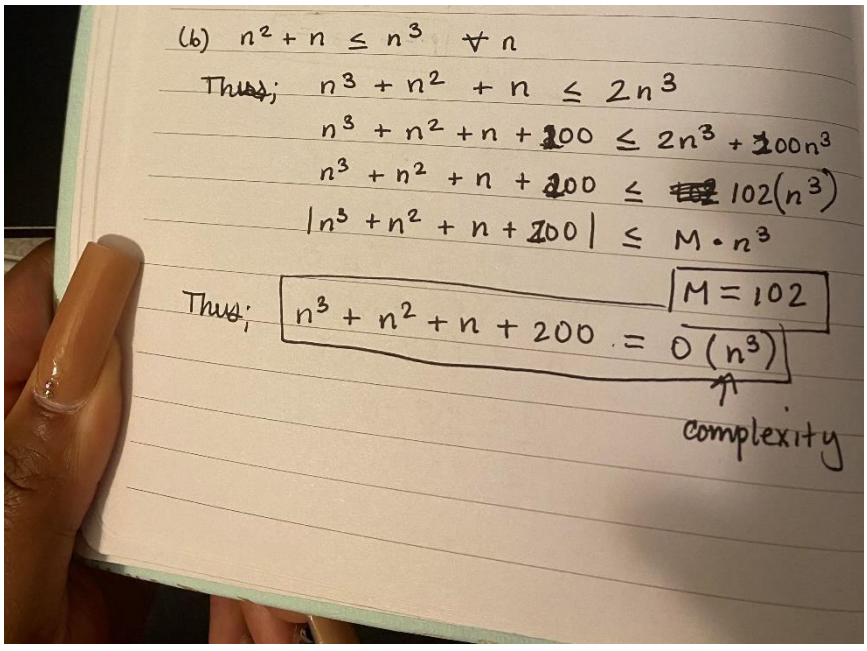


Problem 4 - Big-O

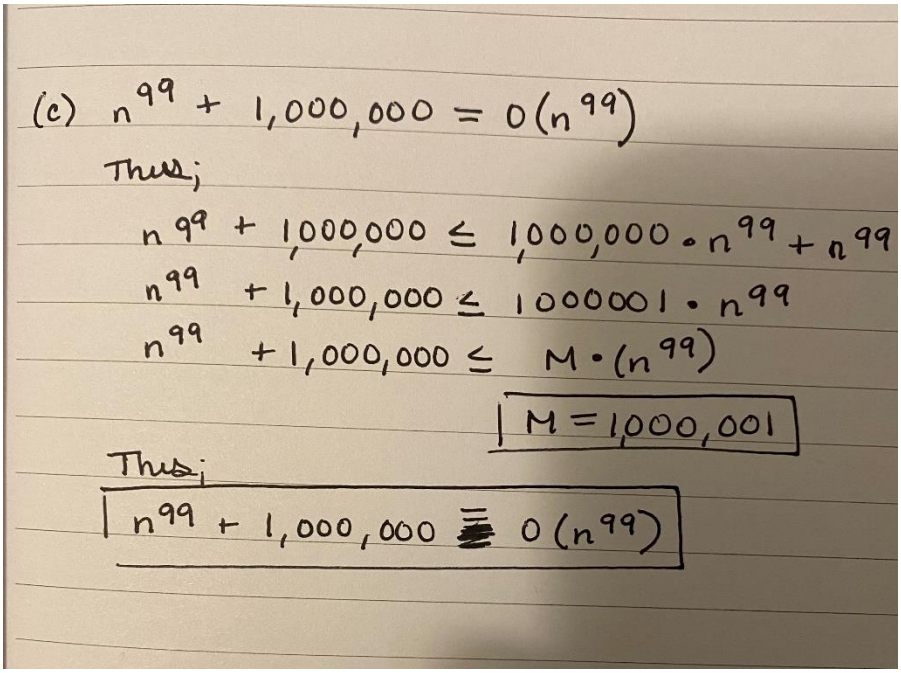
1. Using the definition of big-O, show $100n + 5 = O(2n)$.



2. Using the definition of big-O, show $n^3 + n^2 + n + 100 = O(n^3)$.



3. Using the definition of big-O, show $n^{99} + 10000000 = O(n^{99})$.



(c) $n^{99} + 1,000,000 = O(n^{99})$

Thus;

$n^{99} + 1,000,000 \leq 1,000,000 \cdot n^{99} + n^{99}$

$n^{99} + 1,000,000 \leq 1000001 \cdot n^{99}$

$n^{99} + 1,000,000 \leq M \cdot (n^{99})$

$M = 1,000,001$

Thus;

$n^{99} + 1,000,000 \equiv O(n^{99})$

Problem 4 - Searching

We will consider the problem of search in ordered and unordered arrays.

1. We are given an algorithm called *search* which can tell us *true* or *false* in one step per search query if we have found our desired element in an unordered array of length 2048. How many steps does it take in the worse possible case to search for a given element in the unordered array?

   It takes one step per search for an element in the given array.

2. Describe a *fasterSearch* algorithm to search for an element in an **ordered array**. In your explanation, include the time complexity using Big-O notation and draw or otherwise explain clearly why this algorithm is able to run faster.

   Faster search algorithm deals with the ordered array of data. It is applicable to searching for an element in an ordered array that has a longer length.

   1. find the desired value in an ordered array, the terminal value is compared with the middle value of the ordered array.
   2. If both the terminal value and the middle value of an ordered array does not match with each other then the same iteration is performed for the value next to the middle value of an ordered array.
   3. Big-O is O(1) in an ordered array.
   4. The algorithm is able to run faster if the number of elements in an ordered array is large then algorithm takes longer time in responding.

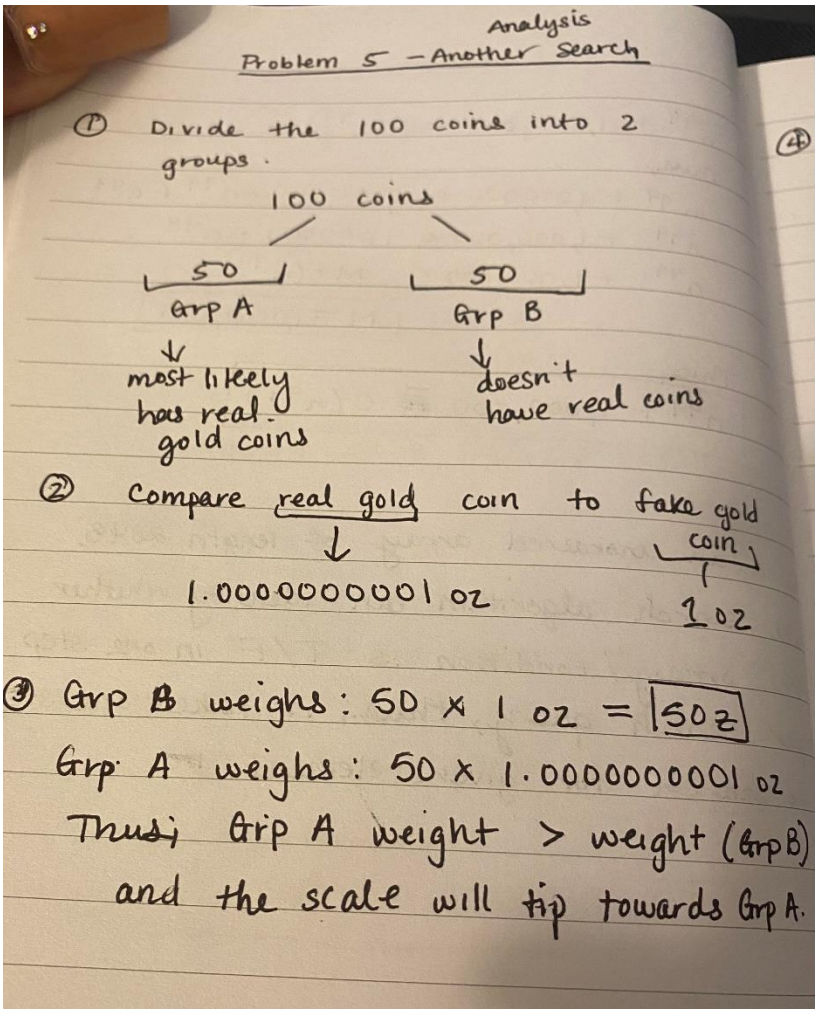
3. How many steps does your fasterSearch algorithm (from the previous part) take to find an element in an ordered array of length 256 in the worse-case? Show the math to support your claim

   For the worst case, the algorithm compares the 256 value with the elements in the ordered array which results in the algorithm taking 256 steps to arrive at the terminal value in Big-O(worst case) scenarios.

Problem 5 - Another Search Analysis

Imagine it is your lucky day, and you are given 100 golden coins. Unfortunately 99 of the gold coins are fake. The fake gold coins all weigh 1 oz. but the 1 real gold weighs 1.0000000001 oz. You are also given one balancing scale that can precisely weigh each of the two sides. If one side is heavier than the other side, you will see the scale tip.

1. Describe an algorithm for finding the real coin. You must also include the algorithm the time complexity. *Hint* Think carefully – or do this experiment with a roommate and think about how many ways you can prune the maximum amount of fake coins using your scale.



2. How many weighing must you do to find the real coin given your algorithm?

it takes 6 rounds of weighing to find the real coin.