Lesson Review: Module Summary

Response to and Request for feedback

If you are resubmitting, include a statement outlining the changes you have made to your submission. This section can be short but should be precise. It is a good idea to quote the feedback you are responding to.

If this is your first submission, include a statement about what part of the lesson review you would most like to receive feedback (and why). Your tutor will take this in consideration when reviewing your work, although they may choose to give you feedback on a different thing if they think it's more appropriate.

Module Learning Objectives

I certify that I achieved the following learning objectives for the module (these objectives can be found in the introduction of the module):

Summarising the content:

- Identify the key terms and concepts in the module. For each of these terms and concepts:
  - Define the term and explain the concept **in your own words** (beware plagiarism – this is an assessment task).
  - Summarise the most important results related to these concepts, including theorems and propositions, algorithms and procedures, etc.
  - You can provide examples, figures, diagrams, but only if they help illustrate your point. This is a summary, so it's best to restrict the explanations to the main points.
  - Make sure you include references to the Module Learning Objectives.
- How is this useful? Relate the concepts you learnt in this module to the broader context (other modules, other units within your degree, and more generally)

ADVANCED LEARNING MODULE : COMPLEXITY

COMPLEXITY OF ALGORITHMS AND BIG O NOTATION

The complexity in algorithms refers to how a program performs in reference to time (slow or fast) or the amount of resources such as time or memory required to perform a task or in other words how many steps it will take an algorithm to run. . Suppose we have two algorithms and both of them have same running time but as soon as we change the input, we see a significantly different effect on them.

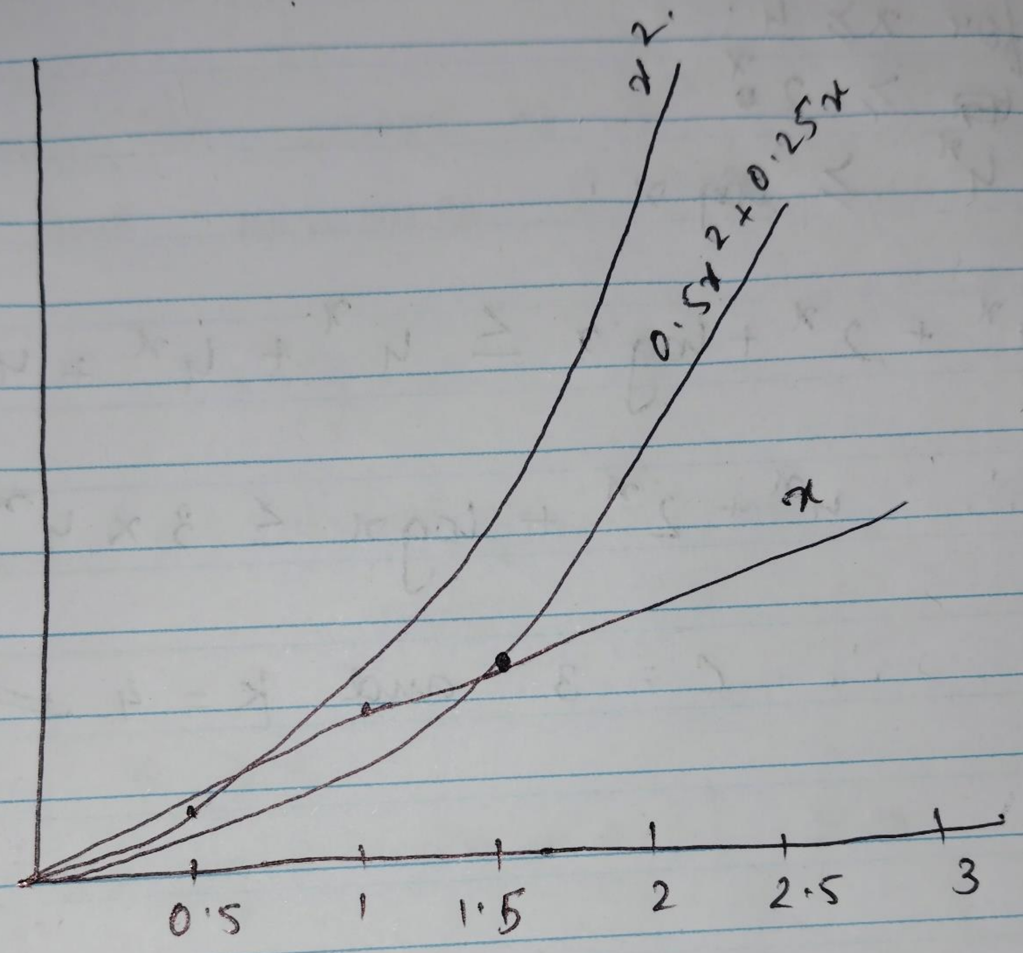
Usually an algorithm with lower complexity can run more efficiently than the one with more steps i.e. more complexity.

BIG O NOTATION INTRODUCTION

The big o notation is the language we use for talking about how long an algorithm takes to run or how much space it uses. In other words, determining how much complexity it holds. As mentioned in the learning module, the big o notation provides the idea of worst possible runtime of a particular algorithm.

COMPARING RUNNING TIME

We can compare the running time of different functions using graphs by comparing the curves. For example, as given in the learning module we have curves x^2 , 1/2x^2+1/4x and x. now if we plot the graph bwe will get something like this


x²
0.5x² + 0.25x
x
0.5
1
1.5
2
2.5
3

We see how the functions look more like straight lines as the input for the function grow. These functions can be grouped under the same category of linear complexity. Which is denoted by **$\mathcal{O}(x)$** as it denotes that the number of operations grows proportionally which we saw here in the graph.

Similarly, for the functions which have their curves as parabolas, show quadratic complexity **$\mathcal{O}(x^2)$.** And do not need to be similar to straight lines like the linear complexity.

FORMALISING THE CONCEPT

Summarising this whole thing we can say that complexity can be classified based on how they behave- linearly, quadratically etc. but there are cases when we might see the graph showing linear complexity when the function is not even linear . so, we need much more than just looking at the curves to be able to compare these algorithms.

CALCULATING WITNESS PAIRS

BIG – O

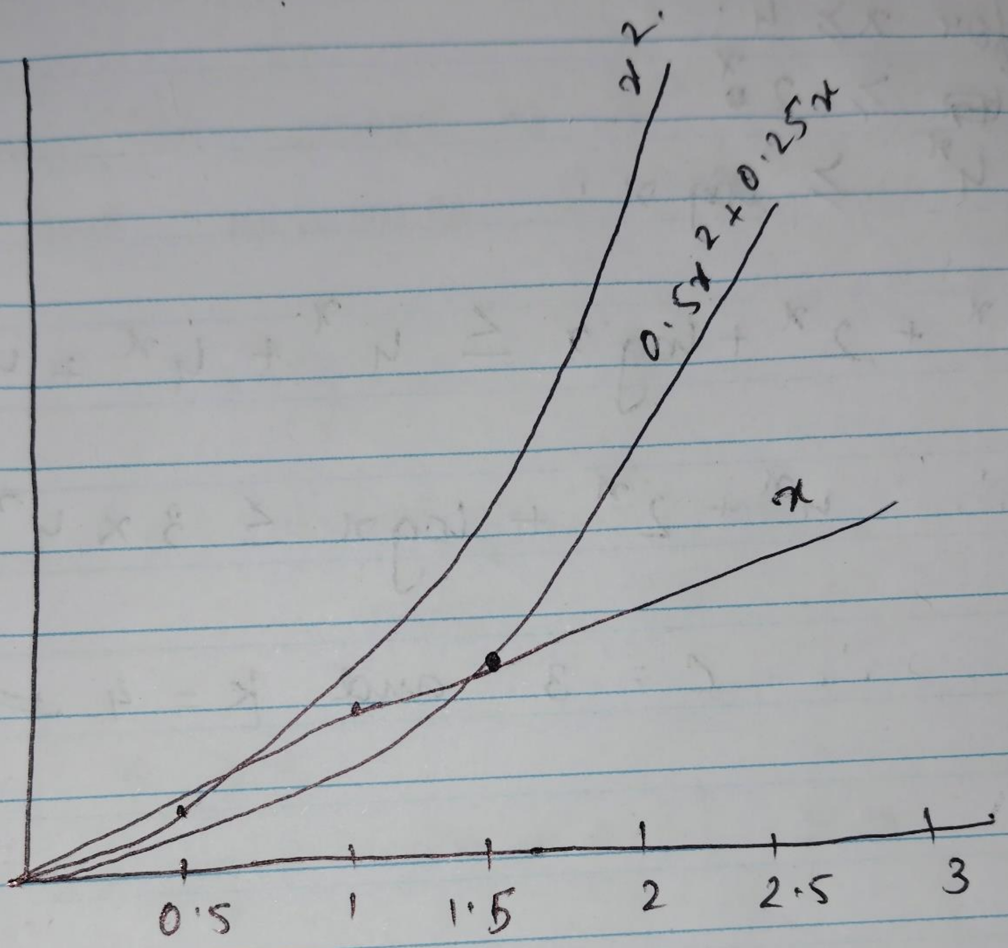
A function f(x) is said to be of order g (x), denoted by $\mathcal{O}(g(x))$ if

$\exists C, k \in \mathbb{R}$ such that $|f(x)| \leq C\ g(x)\ \forall x \geq k$

We say that $f(x) = \mathcal{O}(g(x))$. We refer to C and k as witnesses i.e. the values of C and k are ‘witnessing’ the fact that $f(x)$ is order $g(x)$.

POWERS OF X

Choosing k : although there are no rules to choose k we take k =1 as the first choice.


x^2
$0.5x^2+0.25x$
x
0'5
1
1'5
2
2·5
3

Let us perform an example :

f(x)= 7x^2 = = $\mathcal{O}(x$^3) .

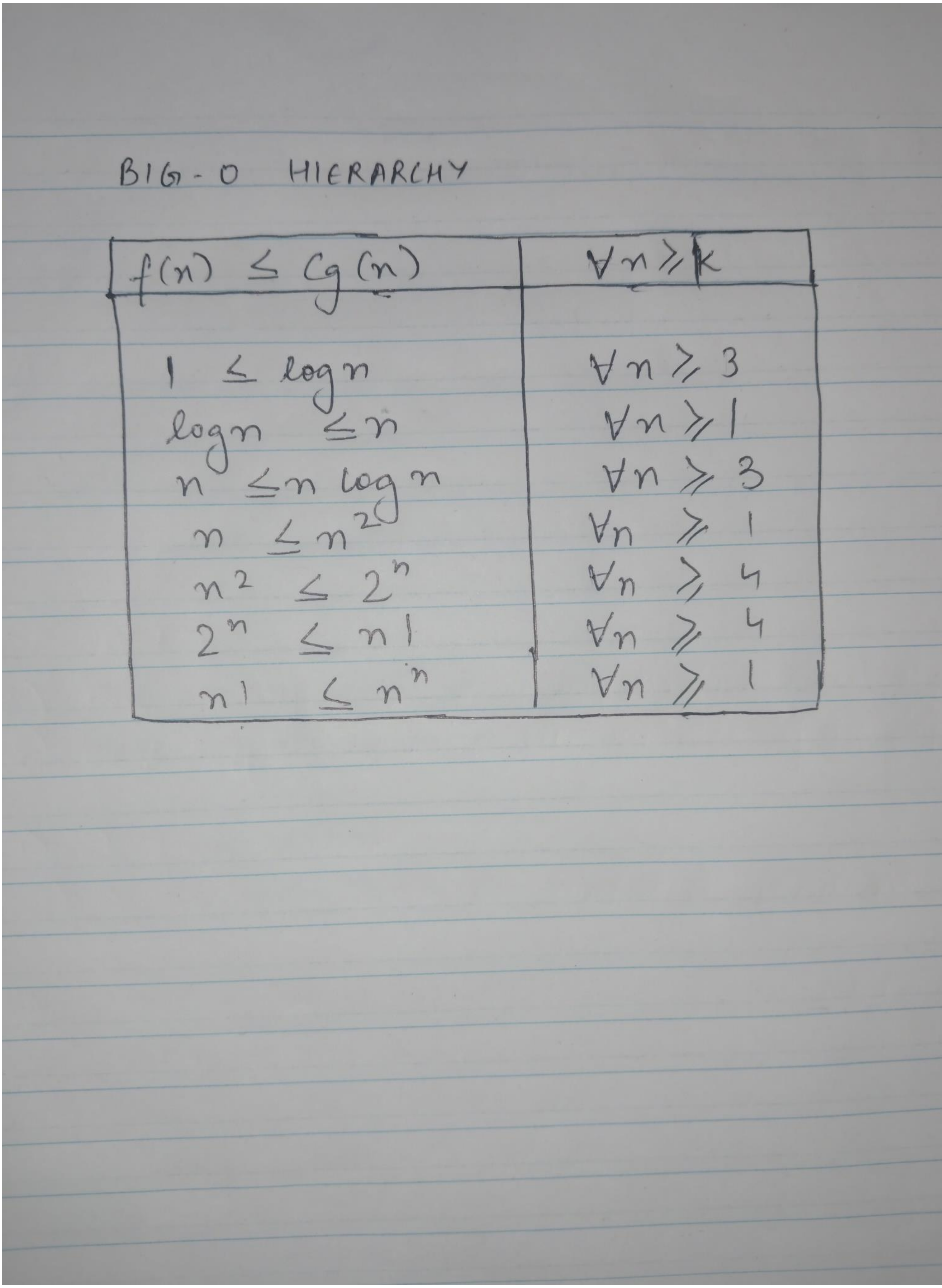
find the witnesses that show that f(x) = $\mathcal{O}$ (x^3)

⇨ Witnesses (k = 1 and C = 7). We have 7x2 ≤ 7x3 for all x ≥ 1. \
⇨ • Witnesses (k = 7 and C = 1). We have 7x2 ≤ 1x3 for any x ≥ 7.

So, 7x2 = $\mathcal{O}(x$3).

OTHER FUNCTIONS
Log x and e^x
Now, if we plot the graph for log x and x we can see that log x **≤ *x* for *x* ≥ 0**.
Also we know that log n = a implies e^a = n

BIG O HERIARCHY.



BIG-O HIERARCHY

$f(n) \le Cg(n)$	$\forall n \geqslant k$
$1 \le \log n$	$\forall n \geqslant 3$
$\log n \le n$	$\forall n \geqslant 1$
$n \le n \log n$	$\forall n \geqslant 3$
$n \le n^2$	$\forall n \geqslant 1$
$n^2 \le 2^n$	$\forall n \geqslant 4$
$2^n \le n!$	$\forall n \geqslant 4$
$n! \le n^n$	$\forall n \geqslant 1$

SORTING ALGORITHMS
There are different types of sorting algorithms and their complexities:
The two types are

- Bubble sort $\mathcal{O}(n^2)$
- Merge sort: $\mathcal{O}(n(\log n))$

I would like to conclude my summary with the believe that this task helped me broaden my horizon about not only mathematics but also programming as that is another unit which uses logic and algorithms. Also the part where we learnt about using graphs to compare the complexity, I could relate to the core module – graphs as even though on a lower level ,this topic was somewhat related.

Reflecting on the content:

- What is the most important thing you learnt in this module?
  the most important thing I learnt from this module was using graphs to determine the complexity of the function or the algorithm.
- How does this relate to what you already know?

  I already know about linear and quadratic graphs and how to generally plot them. Through this module I got to know how to use it to compare the functions.
- Why do you think your course team wants you to learn the content of this module for your degree?

  Algorithms are really necessary to understand as a computer science student and algorithms use logic straightaway, logic and precisely all types of logic is what I need to be familiar with in order to understand my course better.