



Welcome to the **Co**Grammar Presentation Name

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.



Software Engineering Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
(Fundamental British Values: Mutual Respect and Tolerance)
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions](#)

Software Engineering Session Housekeeping cont.

- For all **non-academic questions**, please submit a query:
www.hyperiondev.com/support
- Report a **safeguarding** incident:
www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

Skills Bootcamp

8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✓ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks

Guided Learning Hours (GLH):

Minimum of 15 hours

Task Completion: First four tasks

Due Date: 24 March 2024

✓ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**

Software Engineering - **13 tasks**

Web Development - **13 tasks**

Due Date: 28 April 2024

Skills Bootcamp Progression Overview

✓ Criterion 3: Course Progress

Completion: All mandatory tasks,
including Build Your Brand and
resubmissions by study period end
Interview Invitation: Within 4 weeks
post-course
Guided Learning Hours: Minimum of
112 hours by support end date
(10.5 hours average, each week)

✓ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship
Outcome: Document within 12
weeks post-graduation
Relevance: Progression to
employment or related
opportunity

CoGrammar

Algorithm Design and Analysis

June 2024

Agenda

❖ Topics

Technical Interview






Interview Structure

Who will be There

- You
- 1 or more (usually more) senior members of the team
- The hiring manager (sometimes)

What are they Looking for

- Problem solving skills
 - Knowledge of the field
 - Growth potential
- 



Interview Structure

What Might They Ask

- Coding
 - You are given a sentence, you need to sort the words from longest to shortest, how would you do this?
- Design
 - Given this image of a webpage, draw a circle around the parts that can be turned into components.
- Machine Learning
 - How are feature selection and feature engineering approaches different?

How to Succeed

- Don't skip questions
- Practice problem solving approach, don't cram coding questions
- Build projects to gain more technical knowledge

Question and Answer

Drop your questions in the question section

CoGrammar



Algorithm Design





Algorithms

What are they

- A procedure that solves a specific well defined problem
- Used to solve computational problems

What Should I Do With Them

- Design Algorithms
- Analyse Algorithms

Algorithm Design

Process

1. Read and understand the problem until you can rephrase it
 - a. If you're in an interview, ask follow up questions to get a better understanding.
2. Discover edge cases and constraints from your understanding
 - a. If you're in an interview, ask about what should happen at each edge case you discover
3. Visualise the problem and the process required to solve it
4. Understand the steps from the visualisation
 - a. State the steps in words to formalise your understanding
 - b. Identify values that need to be tracked,
 - i. How the input is being stored
 - ii. How the output will be stored and any temporary values.
5. Choose your data structures
6. Write your first draft code
7. Refine your code, repeat the process



Benefits of Algorithm Design

Benefits

- Removes the need to memorise specific coding challenges, giving you confidence when you go into your interview
- Allows you to vocalise the steps required to solve the problem, this is very important in an interview

Additional Notes

- Use Leetcode and Hackerrank to practice and refine your process,
 - The more you practice the process, the easier it is to do in your head without additional visual aids
- Having a process in the interview is the most important thing
 - If you're not able to write the code, showing that you have a process should still be enough to help you pass

Let's take a look at a problem!

Leave your questions in the
questions section





Story Time

There are x children seated in a straight line, each child has a number on their t-shirt to uniquely identify them.

The children are playing a game where the teacher will say a number y and each child will need to move y seats to the right.

If there are no open seats to the right of the child, they will need to wrap around to the first open seat on the left.

Our goal is to simulate a round of this game and produce the final order that the children will be seated.



Algorithm Analysis: Time Complexity





Analysing Algorithms

Why

- We need to have something measurable to let us know if our code is good
- In an interview, you might be asked whether there is anything else you can do in your code.
 - This might be a trick question to see if you know the efficiency of your code
 - This might be a hint that your code isn't 100% there
- If we need to change our code, we need to know that the new code is better

RAM Model

What is it

- Random Access Machine
- A fictional computer that measures the effect of each line of code.
- From the RAM, we get an equation, this will help us determine how good or bad our algorithm is.
- From the RAM equation, we can see the fastest growing term and get the Big O of our algorithm.

RAM Model

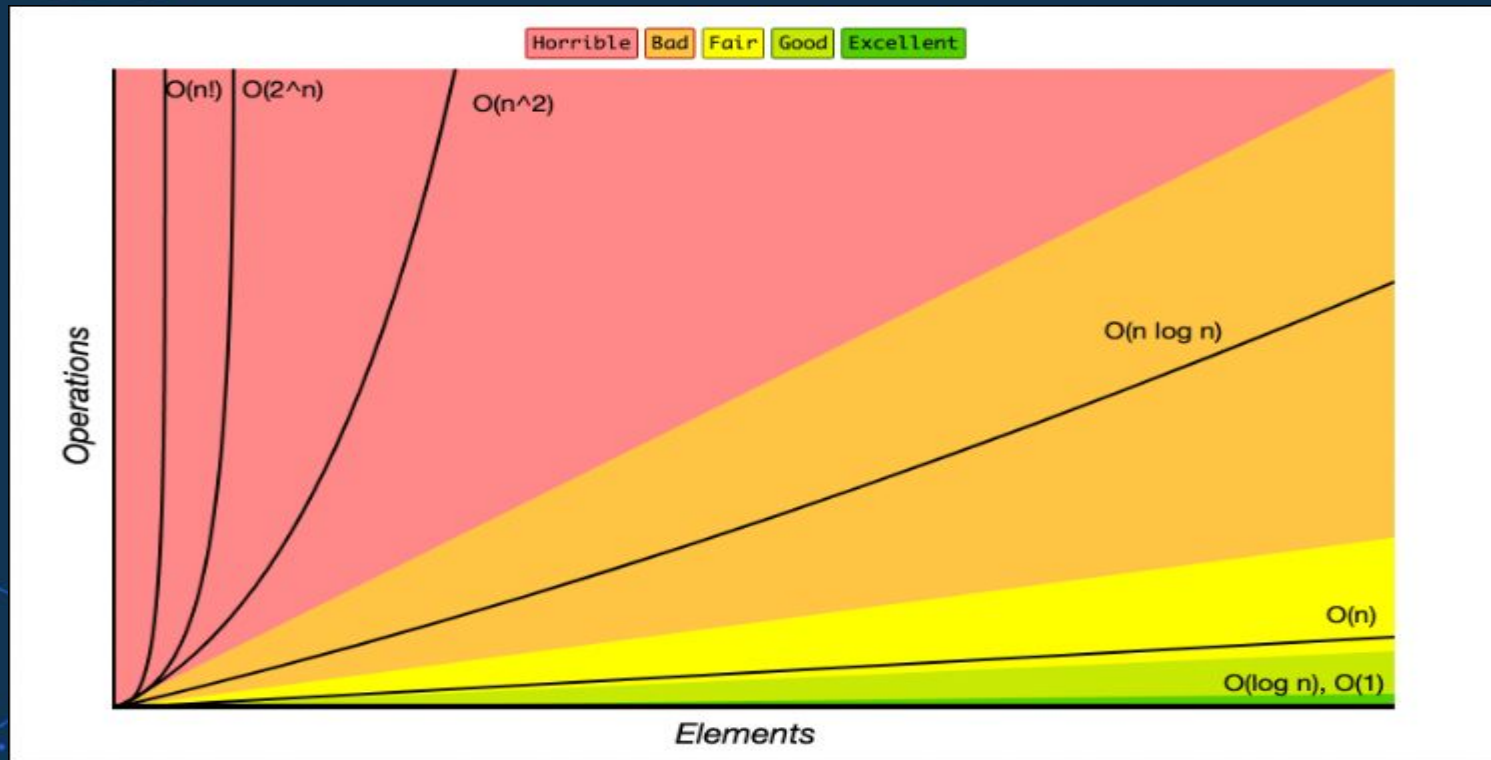
Operation	Cost
Memory Access	0
Arithmetic Operations (+, -, /, %, etc)	1
Logical Operation (&&, etc)	1
Comparison Operations (<, >, ==)	1
Loops (while, for, foreach, etc)	1

RAM Model

Additional Notes

- We count each iteration of the loop as a step
- Operations in a loop will be counted every time they are invoked (might require knowledge of input for code inside a conditional statement)
- We don't count the individual operations in a for loop
- We count each individual comparison and logical operation in a while loop per iteration

Big O



Big O

Fastest Growing Term

- To find the Big O Complexity from the RAM model, we need to find the fastest growing term
- This is the term that will eventually outgrow every other value in our equation as n gets bigger
 - $f(n) \in O(g(n))$

Given the equation: **$2n + n^2 + 100$** , we can see that as the value of n grows, n^2 will outgrow every other term.

Result: **$2n + n^2 + 100 \in O(n^2)$** , our Big O will be $O(n^2)$

If we had the following: $2n + mn + 100$,

$2n + mn + 100 \in O(mn)$

Our Big O would be $O(mn)$, but to fit with computer science, it would be $O(n)$ since that's the closest way to represent it.

Algorithm Analysis: Space Complexity





Space Complexity

What is it

- The measure of how much additional space our algorithms take as our input size grows
- Important to know for operations that require a constant space usage.



Space Complexity Approach

Equation

- We can't use the RAM model because memory works different to computation
- We use the **$S(P) = c + Sp$** equation to get the memory allocation required for the algorithm.
- We look for two things, the fixed part operations and the variable part operations
- **c** - The sum of all fixed part operations
- **Sp** - The sum of all variable part operations.



Space Complexity Approach

Fixed Part Operations

- The memory allocation remains constant regardless of the input that is passed
- We only count them as a single allocation when they are defined, any further references to them won't count to our equation.

Variable Part Operations

- The amount of memory that's allocated changes based on the input that is passed
- We count the initial creation of the object as 1 step
- Each addition to the data structure will be 1 allocations (or how many ever allocations match the operation)
- If the size grows and shrinks or is dependent on conditional statements, we will need to test worst case inputs to get a good reading.

Thank you for attending



Department
for Education

CoGrammar

