Welcome to the CoGrammar CIW: Code Challenge

The session will start shortly...

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

Coding Interview Workshop 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**

Coding Interview Workshop 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

CoGrammar

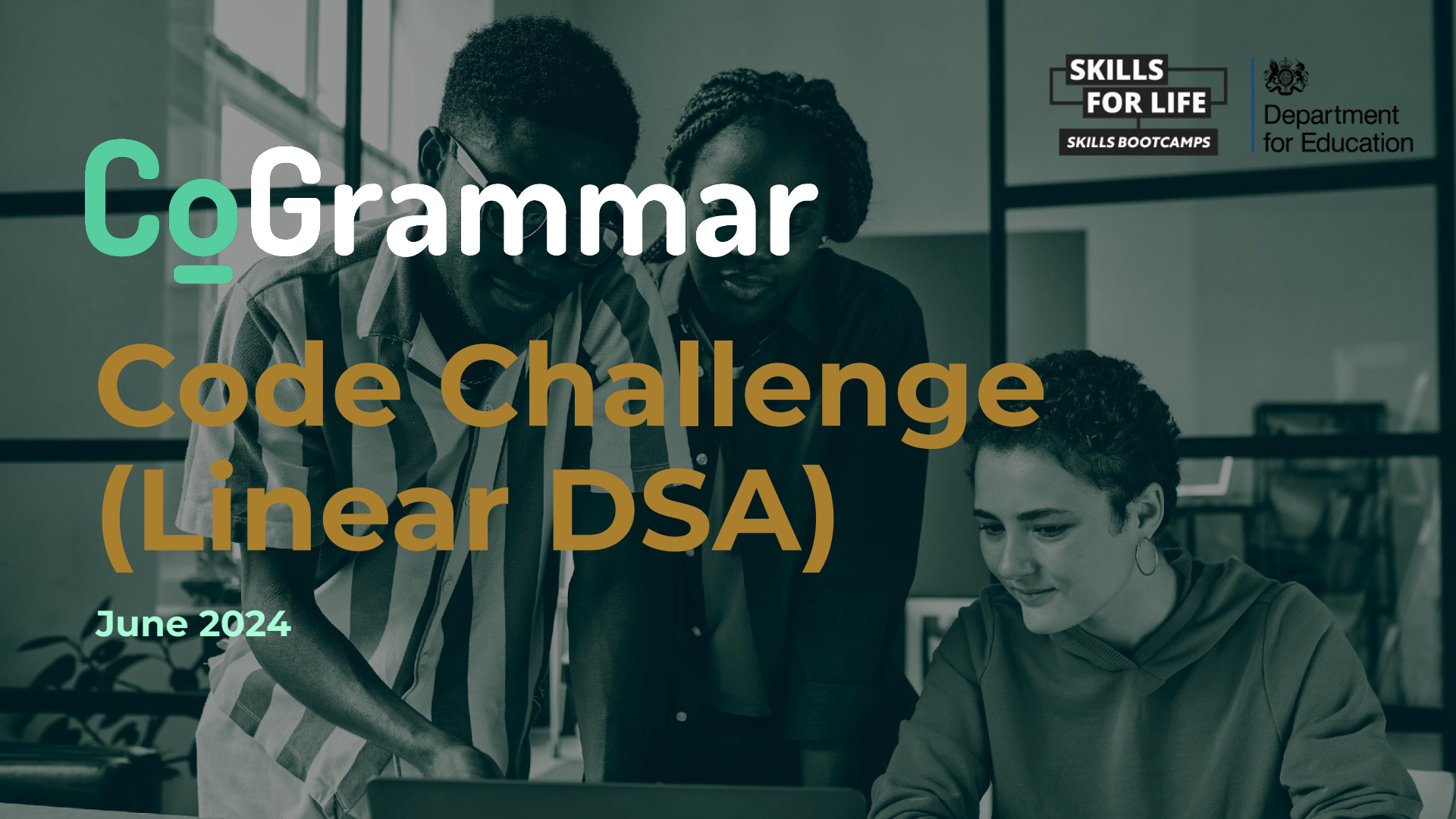
Skills Bootcamp Progression Overview

✅ 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

Code Challenge (Linear DSA)

June 2024

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Agenda

- Understand the concept of Linear Data Structures (Linked Lists | Stacks | Queues)
- Understand how to take a problem statement and create a coding solution from the challenge.
- Analyse the time and space complexity of the given coding challenge
- Optimise the solutions from a brute force approach.

Linear DS

- Used to store Data in a sequential order, each element has a next element and a previous element excluding the first and last elements.

Examples of Linear DS

- Arrays
- Linked Lists
- Stacks
- Queues

Characteristics of Linear DS

- Sequential Storage - Each element is stored one after the other
- Single Level - Each element has only one predecessor and one successor
- Fixed Size/Dynamic size
- Access - Access to elements is typically sequential (To access an item with unknown index you may need to traverse from the beginning)
- Traversal - Involves visiting each element in a sequence
- Insertion/Deletion - This may require shifting or rearranging especially in fixed arrays
- Memory Efficiency - Can be more memory efficient compared to non-linear data structures.

Let’s take a look at a problem!

Leave your questions in the questions section

CoGrammar

Code Challenge.

- You are given two non-empty linked lists representing two non-negative integers. The most significant digit comes first and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list. You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Let’s Breathe!

Let’s take a small break before moving on to the next topic.

CoGrammar

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

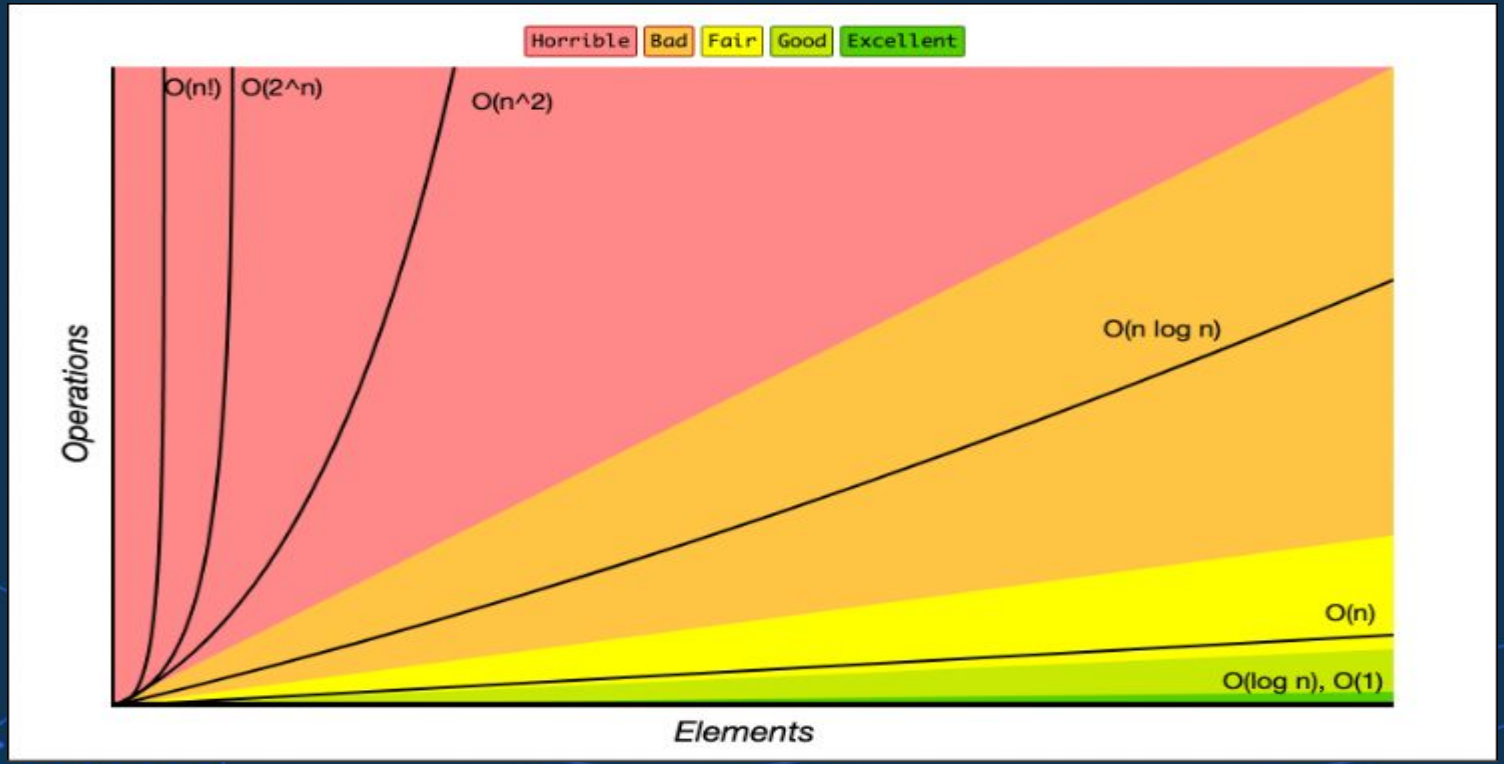
Algorithm Analysis: Time Complexity

CoGrammar

RAM Model

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

Big O


Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Operations
Elements

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

CoGrammar

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

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar