CoGrammar

Workshop 4 - Computational Complexity

SKILLS FOR LIFE

SKILLS BOOTCAMPS

Department for Education

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**

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 Workshops

Skills Bootcamp Certification Overview

✅ Criteria 4: Original Deadline for Demonstrating Employability without Co-certification

Record a job outcome by 23 September 2024

✅ Criterion 4: Updated Deadline for Imperial College London & University of Manchester Co-certification

Record a job offer by 15 July 2024

CoGrammar

Lecture Objectives

- Understand the importance of measuring algorithmic complexity
- Learn how to effectively compare the performance of different algorithms
- Identify algorithms that are expensive on memory

Which one of the following time complexities is the worst?

A. $O(n^2)$
B. $O(2^n)$
C. $O(n \log n)$
D. $O(\log n)$

What is the fastest growing term in the following equation?
$2n + n^2 + 5$

A. $2n$
B. n^2
C. 5

What is the goal of measuring the space and time complexity of an algorithm?

A. See the exact memory and time that is being used when an algorithm is executed
B. Gain a sense of how an algorithm performs as the input size increases
C. Measure the relationship between the amount of space used and how fast an algorithm performs
D. None of the above

Background

What have we looked at so far

- Data types
- Divide and conquer algorithms

What do they have in common

- No two are made equally
- There is a cost and benefit of choosing a given data structure or algorithm

CoGrammar

Problem Statement

Recap

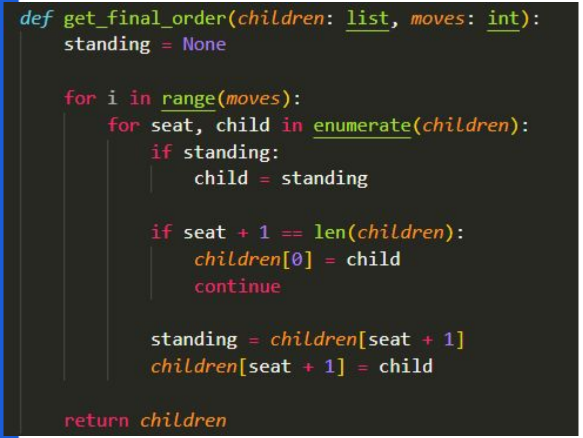
There are `x` children seated in a straight line, each child has a number on their t-shirt to uniquely identify them.

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.

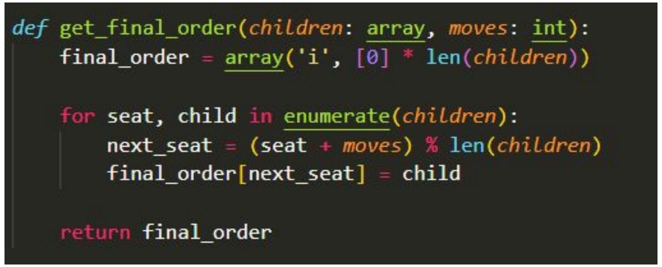
Our goal is to simulate a round of this game and produce the final order that the children will be seated.

Solutions

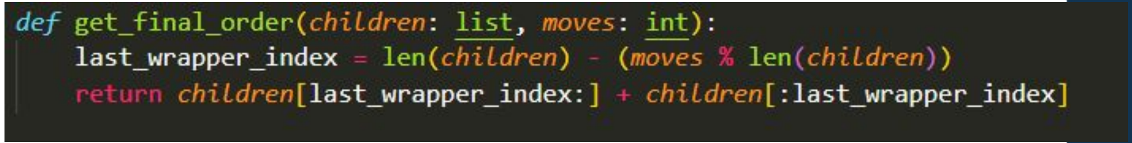
We came up with 3 solutions for this problem take a look here, but how can we determine which one of our solutions is the best one?



```python
def get_final_order(children: list, moves: int):
    standing = None

    for i in range(moves):
        for seat, child in enumerate(children):
            if standing:
                child = standing

            if seat + 1 == len(children):
                children[0] = child
                continue

            standing = children[seat + 1]
            children[seat + 1] = child

    return children
```



```python
def get_final_order(children: array, moves: int):
    final_order = array('i', [0] * len(children))

    for seat, child in enumerate(children):
        next_seat = (seat + moves) % len(children)
        final_order[next_seat] = child

    return final_order
```



```python
def get_final_order(children: list, moves: int):
    last_wrapper_index = len(children) - (moves % len(children))
    return children[last_wrapper_index:] + children[:last_wrapper_index]
```

From the code, which solution would you say is the most efficient and why?

Algorithms

Algorithms

What is it

- A procedure that solves a well-defined computing problem
- Usually takes the form of a single or multiple related functions

Algorithms in a large system

- Typically small parts of a larger system
- Perform a domain specific operation
- Though small, their impact is large
  - A single 5 file function can be the difference between the application responding in 0.005s or 20s

Algorithms

Skills to have as a programmer

- Design
  - One should be able to take a well defined problem or process and translate it into code.
- Measure performance
  - One should be able to identify a good and bad algorithm based on performance

Algorithms

Applying these skills at work

- Software Engineering
  - A good software engineer should be able to translate processes to code
  - Go to person for automating long processes
- Data Science
  - Pandas may not be able to do the advanced data cleaning that you want, so you need to get creative
  - Many data scientists end up getting into Data Engineering which involves building efficient pipelines for large data sets

Algorithms

Applying the skills in the interview

- Translating requirements to code
  - Removes the need to memorize code implementation and just focus on understanding the process.
- Allows you to use coding challenge platforms as a way to practice problem solving skills instead of memorizing solutions
- Measure performance
  - You may be presented with an algorithm and asked what makes it good or bad
  - You can explain what makes your solution good
  - If you are given code to refactor, you can focus on optimising the correct things.

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

CoGrammar

Algorithms

Identify a good algorithm

- Time Complexity - How fast is the algorithm as the input size increases
- Space Complexity - How much additional space is required for the algorithm to perform its operations

CoGrammar

Benchmarking

Process

- Tests the time it takes for a function to perform an operation with varying input sizes
- Can measure that amount of space used to perform the operation

Benefits

- Good for testing latency of deployed applications
- Good for finding bottlenecks in the system and comparing the performance of possible fixes
- Good for estimating potential cloud costs

Cons

- Performance is dependent on available resources

The RAM Model

What is it

- Random Access Machine
- A fictional computer that makes all tests consistent
- Allows us to consistently check the performance of an algorithm
- Makes it easy to compare the performance of different algorithms.

How it works

- Gives each line of code a weight
- Overall complexity is measured by adding up all of the operations in the code.

The RAM Model

Notation
Memory Access - 0 Steps
Arithmetic Operations - 1 Step
Conditional operations - 1 Step
Logical Operations - 1 Step
Loops - 1 Step (per iteration)

Additional Notes
Memory access is based on the data types used, so we will not include this to our equation, but it’s important to choose the right data structure.

The RAM Model

Additional Notes

- When looking and conditional statements, we will only count the logical operations
- We won’t count the logical operations in loops, we will look at the loop as a single step (even though every part does count)

CoGrammar

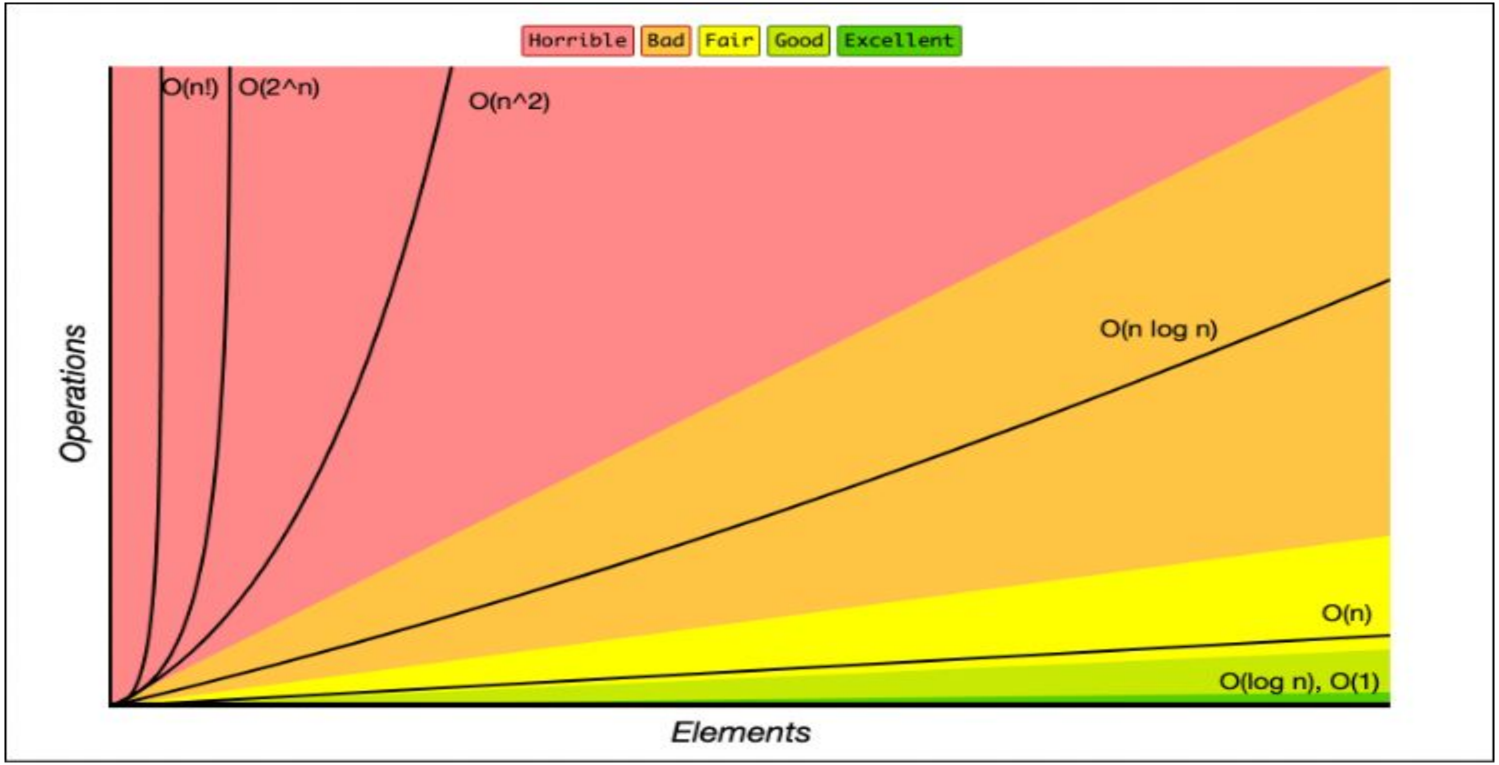
Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

Practical Example

CoGrammar

Big O Notation



Fastest Growing Term

To find the Big O complexity from the RAM model, we need to find the fastest growing term in a given equation.

$f(n) \in O(g(n))$

Explanation: f(n) is contained in g(n), meaning that g(n) grows as fast or faster than g(n).

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

Explanation: $2n + n^2 + 100$ is contained in n^2, meaning that n^2 is the fastest growing term. When plotting $2n + n^2 + 100$ on a graph, the value of n^2 affects the curve more than any other value.

Practical Example

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

Space Complexity

Approach

- We consider operations with value types as constant space (Since the stack memory is preassigned)
- Each object creation counts towards space complexity
- Remember the data type you're working with and take the space complexity of performing operations to account
    - eg, editing strings creates new string objects in memory

Garbage Collector

Background

- When you create an object, it is stored in memory and a variable is used to reference the location in memory
- When you reassign a variable, the object doesn’t get deleted, the object you are pointing to just changes
- Compound data types like strings and array create new objects for doing operations like deletes and resizing (if the language supports those)

Solution

- The garbage collector run periodically to clear any unreachable objects stored in the heap

CoGrammar

Garbage Collector

Additional Notes

- Though efficient, everytime the garbage collector is called, it pauses all operations
  - It’s a good idea to perform operations that minimise calls to the garbage collector
- Variables in functions are cleared by the garbage collector after the function runs
- Global variables are never cleared from memory
  - This is why it’s better to create local variables as their memory usage is short lived

When evaluating the performance of an algorithm, which aspect do we want to focus on

A. The sum of the operations
B. The fastest growing term
C. The total lines of code
D. The constant operations

HyperCo has 20 stores around the world, they need to sync the data from the stores to their data warehouse every day, this process involves getting over 100 000 records per store.

HyperCo's pipeline triggers an Azure Cloud Function which collects the data from the different stores, this service is billed per second.

Which approach should HyperCo take to predict the cost of running the function.

A. Big O Notation
B. Performance Benchmarks
C. RAM Model

Recap

Algorithms

- We looked at what an algorithm is
- We talked about the importance of algorithms for programmers
- We looked at how algorithm design and analysis skills can be used to optimise our code.
- We looked at how algorithms design and analysis skills can be used in the technical interview.

Testing Performance

- We saw that we can use benchmarks for testing the performance on a specific machine
- We looked at the RAM model for getting an idea of the overhead our algorithm will have based on the size of the input
- We looked at how we can translate the RAM model to Big O
- We looked at space complexity and how the garbage collector works.

CoGrammar

Thank you for joining!