

Workshop 5 - Constant Time Storage and Access

**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**

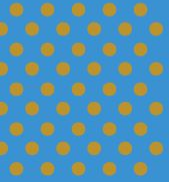
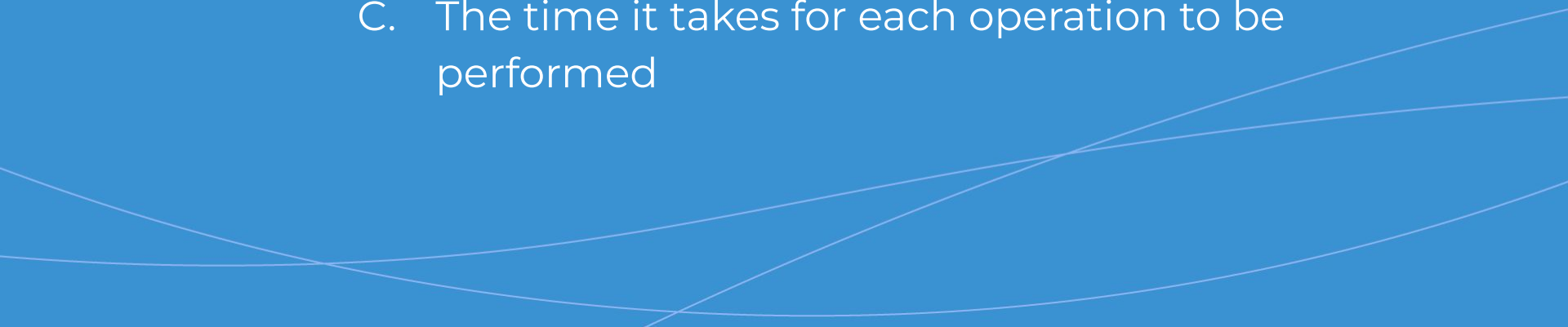


Lecture Objectives

- Understand the importance of memory management when building algorithms
- Understand the different factors that affect constant time access
- Understand the different factors that affect constant storage
- Understand how we can use different data structures for constant time access
- Understand the approaches for achieving constant space





Which factor has the biggest impact on the storage used for an algorithm?

- 
- A. Choice of data structure
 - B. Total number of operations being performed
 - C. The time it takes for each operation to be performed
- 



Which storage option is more efficient?

- 
- A. Heap memory
 - B. Disk storage
 - C. Stack memory
- 



Is the following statement true or false?



Recursion will always result in constant space regardless of the depth of the recursion

- A. True
 - B. False
- 

Background

Memory

- RAM is the fastest way to work with data in our applications
- RAM is limited, so we can't realistically use it for everything
- What can we do to make sure that our applications run fast, but don't use too much memory?

Managing Memory (OS Level)

Memory = RAM

Stack Memory

- The programming language preallocated memory for the stack
 - Usually a few megabytes
- Values in the stack are stored contiguously making it fast to access
- Only value types (primitive data types) are stored in the stack.

Heap Memory

- The OS sets the max allocation for each running application
 - Usually in the gigabytes
- If the application runs over it's allocation, the values are moved to temporary disk storage
- All objects go into the heap

Stack Memory

- Fast to access
- We can't control what goes into the stack
- The memory allocation is very low, but the likelihood of running out of stack space is small
- Accessing values in the stack will always be a constant time

Heap Memory

- If the value is not primitive it will be stored in the heap.
- When an object loses its reference, the garbage collector will be invoked
- Primitive data types will be stored in the heap if they are being stored in compound data types.
- Some objects will make use more more heap memory using certain built in methods.

CoGrammar

Q & A SECTION

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

Constant Space Storage

The amount of memory required to perform an operation does not grow as the input size grows

How to achieve this

- The parameters do not count to the space complexity
- Don't make full or partial copies of the original data structure
- Any variables that does not get affected by the size of the input will be constant.

Demonstrations

Constant Time Access

What is it

The time it takes to access a value should not be affected by the size of the input.

Constant Time Access

Data Structure	Insert	Update	Retrieve	Delete
String			Indexing	
Array		Indexing	Indexing	
List	Append	Indexing	Indexing	Last index
Dictionary	Indexing	Indexing	Indexing	Indexing
Set	Add		Look up	Value
Stack	Push		Pop	
Queue	Enqueue		Dequeue	

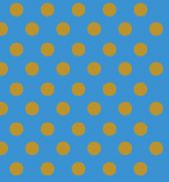
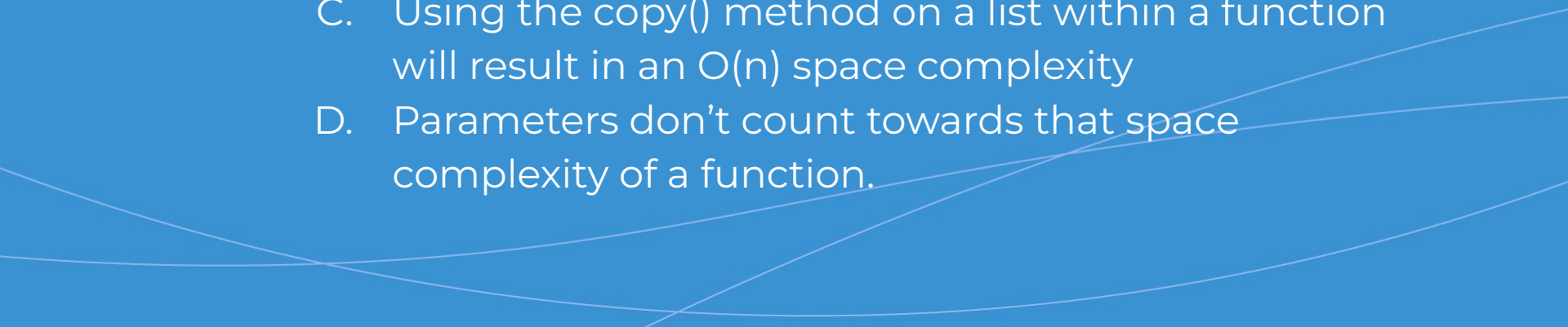
CoGrammar

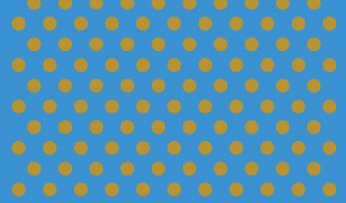
Q & A SECTION

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



Which one of the following statements is false with regards to space and time complexity?

- 
- A. A bad space complexity can lead to the application running slowly
 - B. All constant time operations run for the exact same amount of time
 - C. Using the `copy()` method on a list within a function will result in an $O(n)$ space complexity
 - D. Parameters don't count towards that space complexity of a function.
- 



You need to write code for a microcontroller that has limited resources. Which aspects of your application are you most concerned with optimising?



Recap

- We took a look at how memory works in our application
- We saw that data stored in the stack does not affect our space complexity and provides constant access
- We saw that objects in the heap can be more expensive and can affect the access time and storage used
- We saw how indexing collections can be constant time operations
- We saw how we can use stacks and queue for constant access
- We took a look at how hash sets and hash tables can be used to perform constant time operations.



CoGrammar

Thank you for joining!

