Algorithms and their Applications
CS2004 (2020-2021)

Dr Mahir Arzoky

1. Introduction to Module

Where Did it all Start? – Part 1

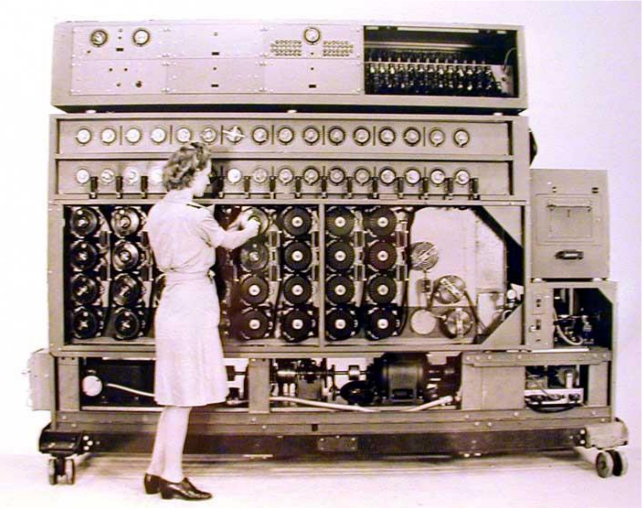
- In ~1900, a very famous German Mathematician called **David Hilbert** (1862-1943) listed 23 unsolved mathematical problems
- He later refined problem number ten to ask "is there a way of determining if an arbitrary first-order symbolic logic statement is true or false?"
  - Don't worry about what this means!

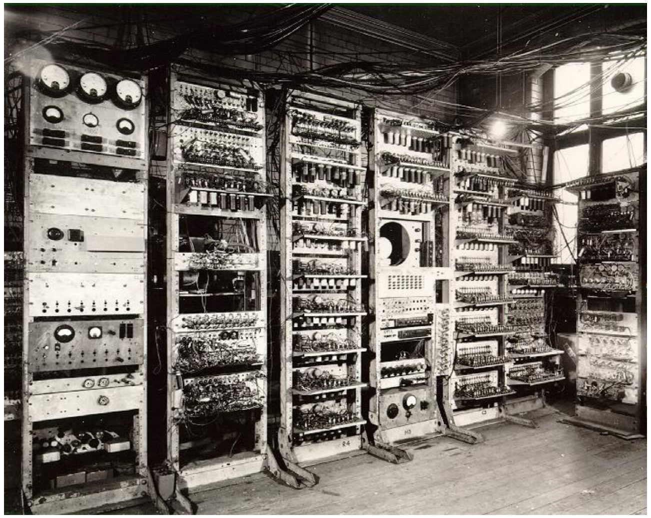
Where Did it all Start? – Part 2

- This became known as the "*Entscheidungsproblem*" or "**Decision Problem**"
- The "**halting problem**"
  - Can we find a program that can predict whether any other program and its input will halt or run forever…
- **Alan Turing** showed it is impossible to solve halting problem
- This was used to show that the "Decision Problem" was impossible to solve
- In 1936 Turing (1912-1954) created two concepts: "Turing Machine" and "Universal Turing Machine"
  - Covered at level one – CS1005 Logic and Computation

Where Did it all Start? – Part 3

- In 1939 Alan Turing used his ideas to create a machine called "The Bombe" which helped crack the German enigma code
- In 1948, the first computer (called the Baby) and associated computer program were developed at Manchester University, UK, based on Turing's ideas





Alan Turing

- Alan Mathison Turing (OBE, FRS)
- 23rd June 1912 to 7th June 1954
- Mathematician
- Computer scientist – the first?
- The theoretical basis for the stored-program computer
- Bletchley Park - WWII
- Code Breaker
- Pardoned by the UK government in 2013
- He will be the face of the new £50 bank note…

Computer Science and Computation

- Computer Science is the study of computation
  - A discipline that studies computable problems and computational structures
  - A discipline that involves the understanding and design of computers and computational processes
  - Interdisciplinary
- Computation is the procedure of calculating i.e. determining something by mathematical or logical methods
  - "In natural science, Nature has given us a world and we're just to discover its laws. In computers, we can stuff laws into it and create a world", - Professor Alan Kay
  - "The Physical Origin of Universal Computing" By Michael Nielson, https://www.scientificamerican.com/article/the-physical-origin-of-universal-computing/
- Computability is about what computers can do and cannot do
- Characterises problems that can be solved algorithmically

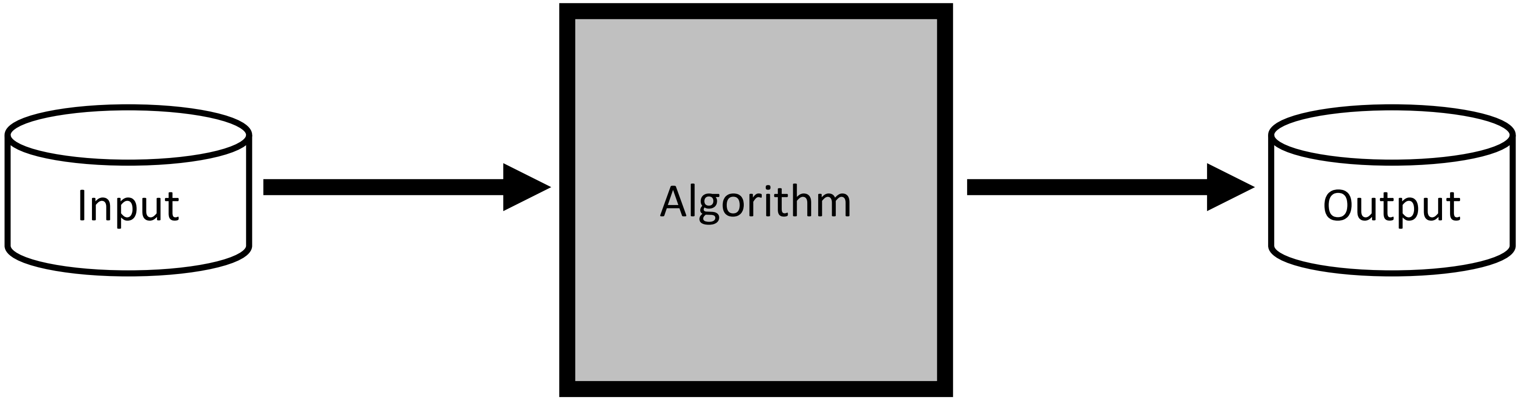
So What is a Computer and Computer Program?

- A **computer** is a device for carrying out certain types of algorithms
- An **algorithm** (in our context) is a set of steps for solving a problem
- A **computer programming language** is a "language" for describing algorithms to computers
- A **computer program** is (usually) a description of a single algorithm and/or problem
- **Computer software** is a collection of programs that carry out a similar or related task

What is an Algorithm?

- The term **algorithm** comes from Persian scholar Muhammad ibn Musa al-Khwarizmi
  - c. 780 to c. 850
  - Mathematician, astronomer, geographer, and scholar in the House of Wisdom in Baghdad
  - He was one of the fathers of algebra
- Algorithms are everywhere
- Typically algorithms are associated with a computer but we (humans) have algorithms too!
- The crafting of efficient algorithms led to a whole science focused around computation
  - This evolved into the discipline of computer science

What is an Algorithm? - Continued



- An algorithm is any well-defined computational procedure that takes some value(s) as input and produces some value(s) as output
- An algorithm can be seen as a tool for solving a well-specified **computational problem**

Why Study Algorithms?

- Some algorithms that solve the same problem can differ massively in their efficiency
- Computers may be getting faster but they are not infinitely fast!
  - For example, sorting a list of billions of numbers could take a computer a very long time
  - If we do not make the algorithms efficient then they may take too long to run!

What is the Analysis of an Algorithm?

- This is an indication of the time a computer will take to solve a problem given the size of the problem
- If we do not analyse our algorithms we do not know how long they will take
- Time is abstract, and is usually measured in operations
  - This allows us to compare two algorithms independent of the speed of a computer
- Also known as the computational complexity of an algorithm

Algorithm Example

Fake Coin Problem

- ❑ You have 64 one pound coins (***N*=64**)
- ❑ All the coins are supposed to be the **same weight and size**
- ❑ You find out that **one of the coins is fake** and weighs **less** than the rest
- ❑ We want to find the fake coin…
- ❑ Luckily, you have a balance scale!

Fake Coin Problem

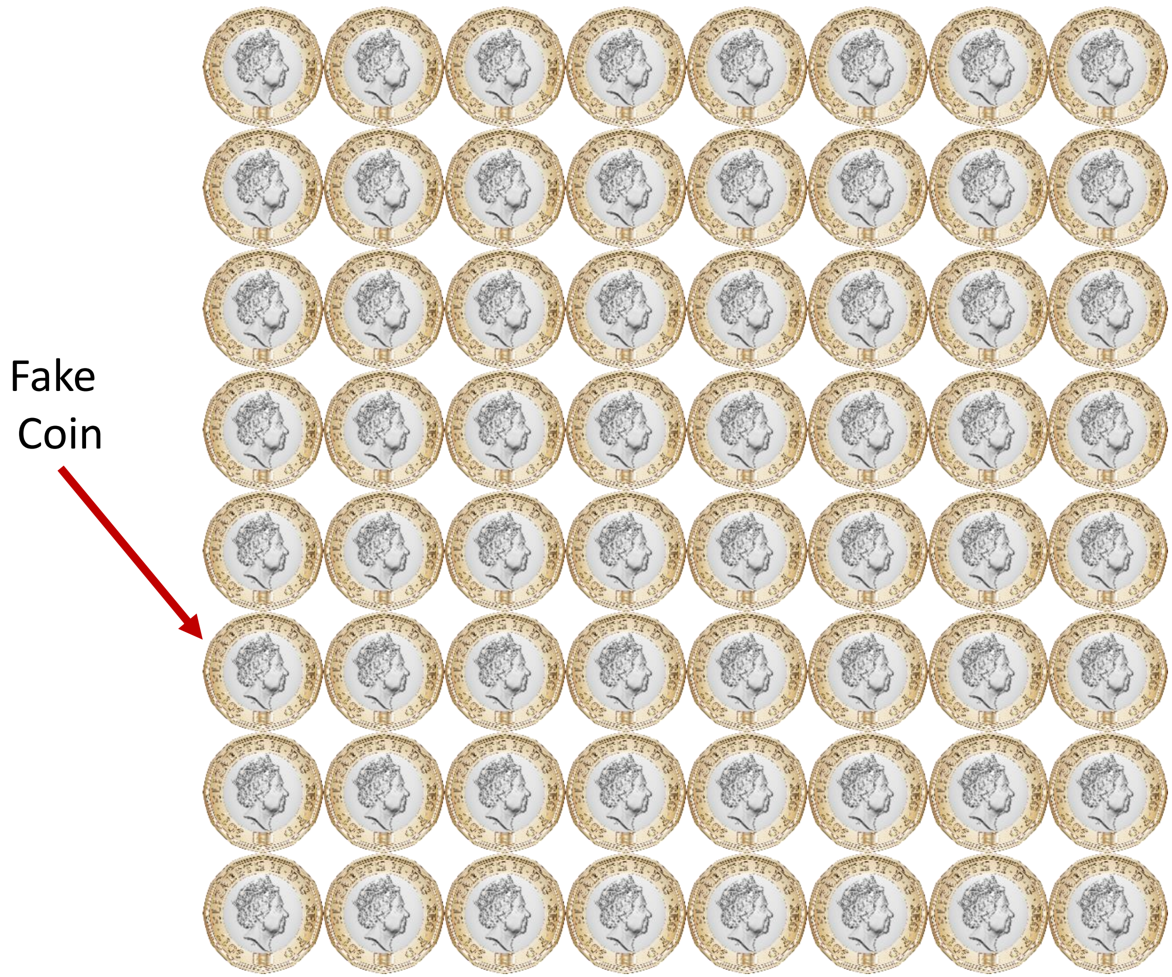
- Outline an algorithm for finding the fake coin
- Think about the followings:
  - How many times do you have to weigh?
  - Easiest approach is not necessarily the best approach!
  - Ideally, we want to design an algorithm that will take the shortest period of time
  - We want an algorithm that is efficient, for example, what if we are then given 10,000 (*N*=10,000) coins and told that one is fake?

Example Solution One

Example Solution One – Pseudocode

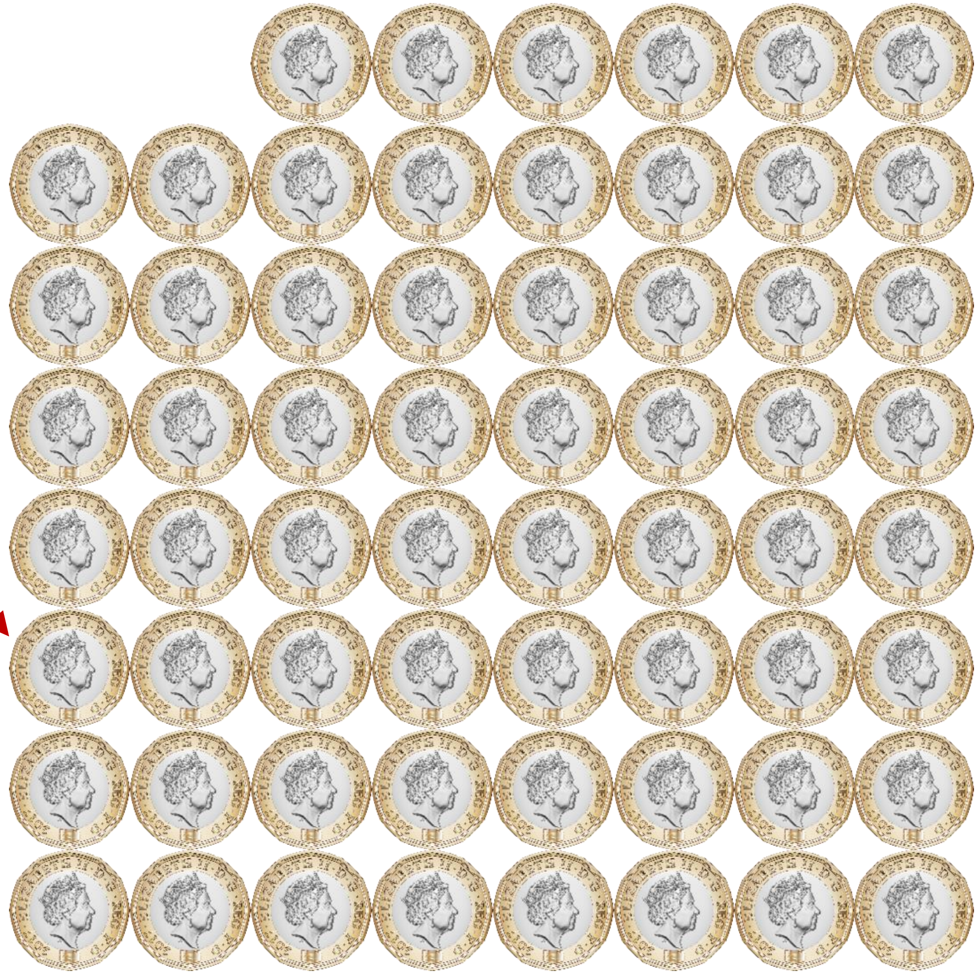
- Take two coins at a time, one on each side the scales:

```
If weight(Left) = weight(Right) Then
   Take the next two coins
Else
   Coin at the lighter side is fake
End If
Repeat the above until either the fake coin is
found or you have checked all the coins
```


Fake
Coin

STEP 1

Fake Coin

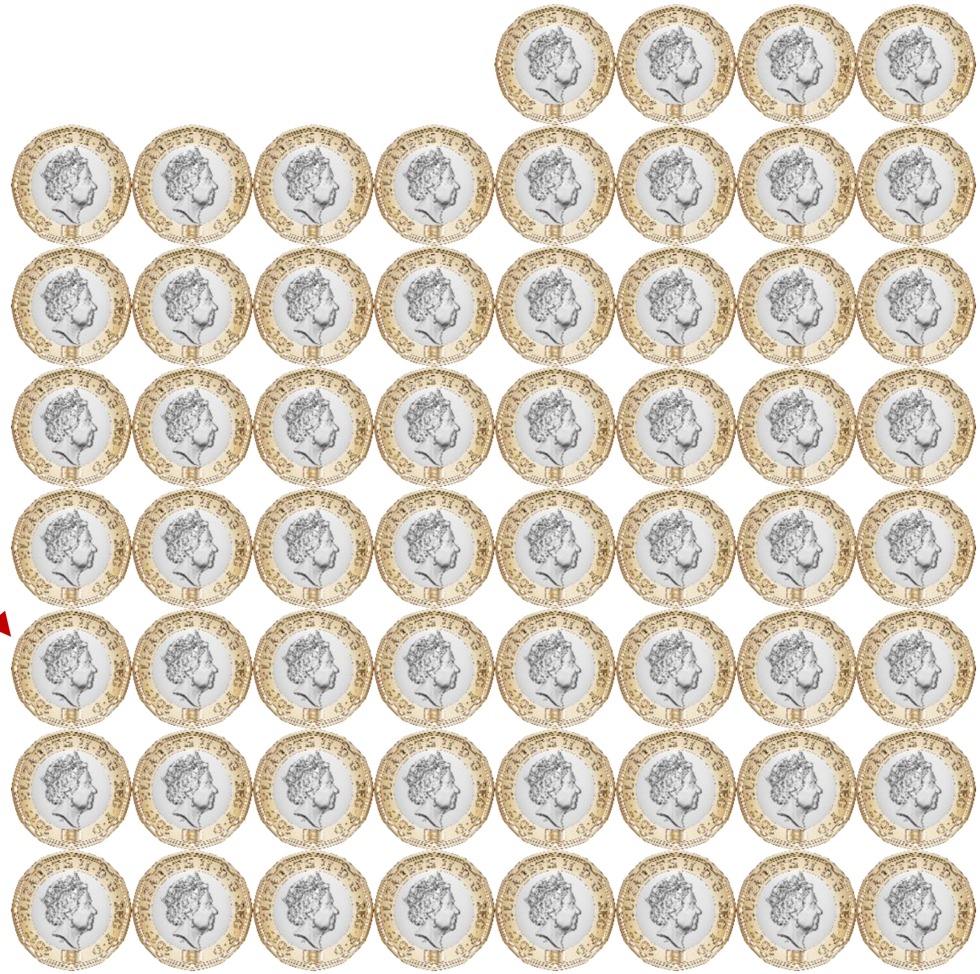


STEP 1

Balanced!

STEP 2

Fake
Coin



STEP 21

Example Solution One – Continued

- In the worst case, i.e. the fake coin is the last or the one before the last
  - The IF Then-Else statement will have to run *N*/2 → 64/2 → 32 times
  - O(*N*) - linear algorithm
- In the best case (and if we are very lucky) , e.g. the fake coin is the first or second, the same statement will only have to run once

- <u>But, can we do better?</u>

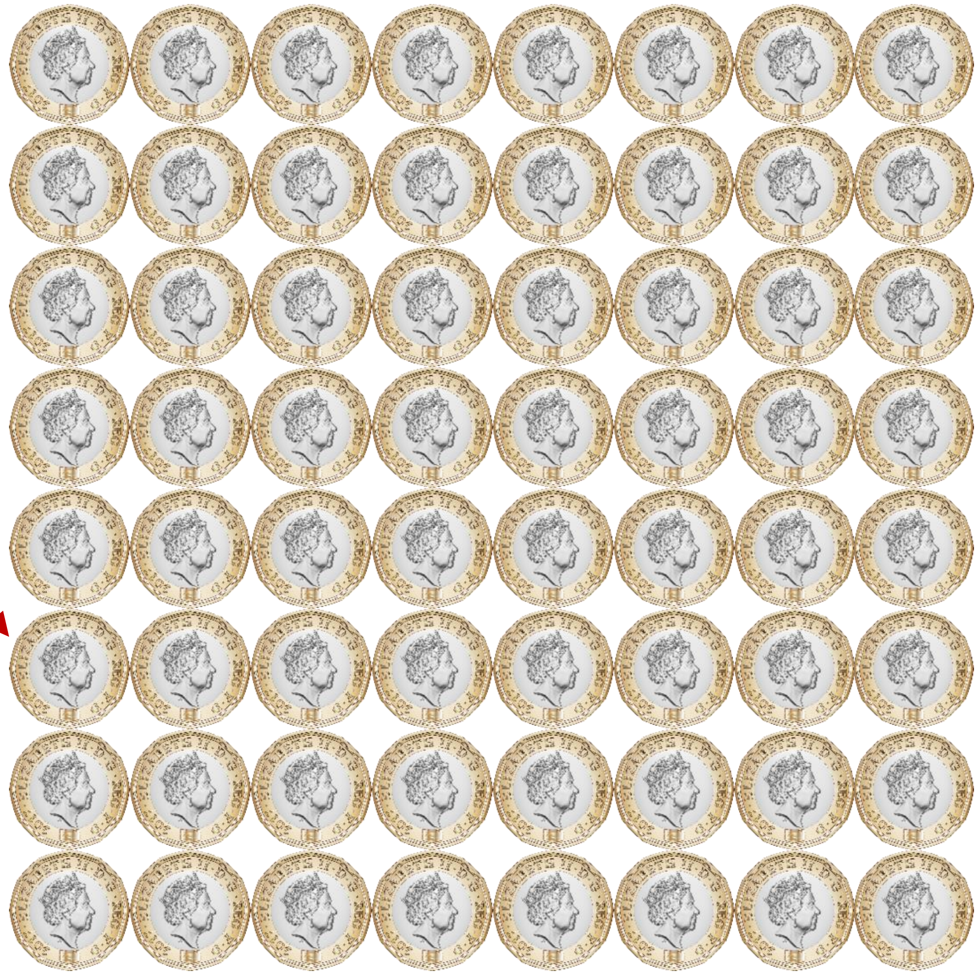
Example Solution Two – Divide and Conquer Algorithm

Example Solution Two – Pseudocode

- Split the set of coins into two subsets of equal size; N/2 go to the left side of the scale, N/2 go to the right side

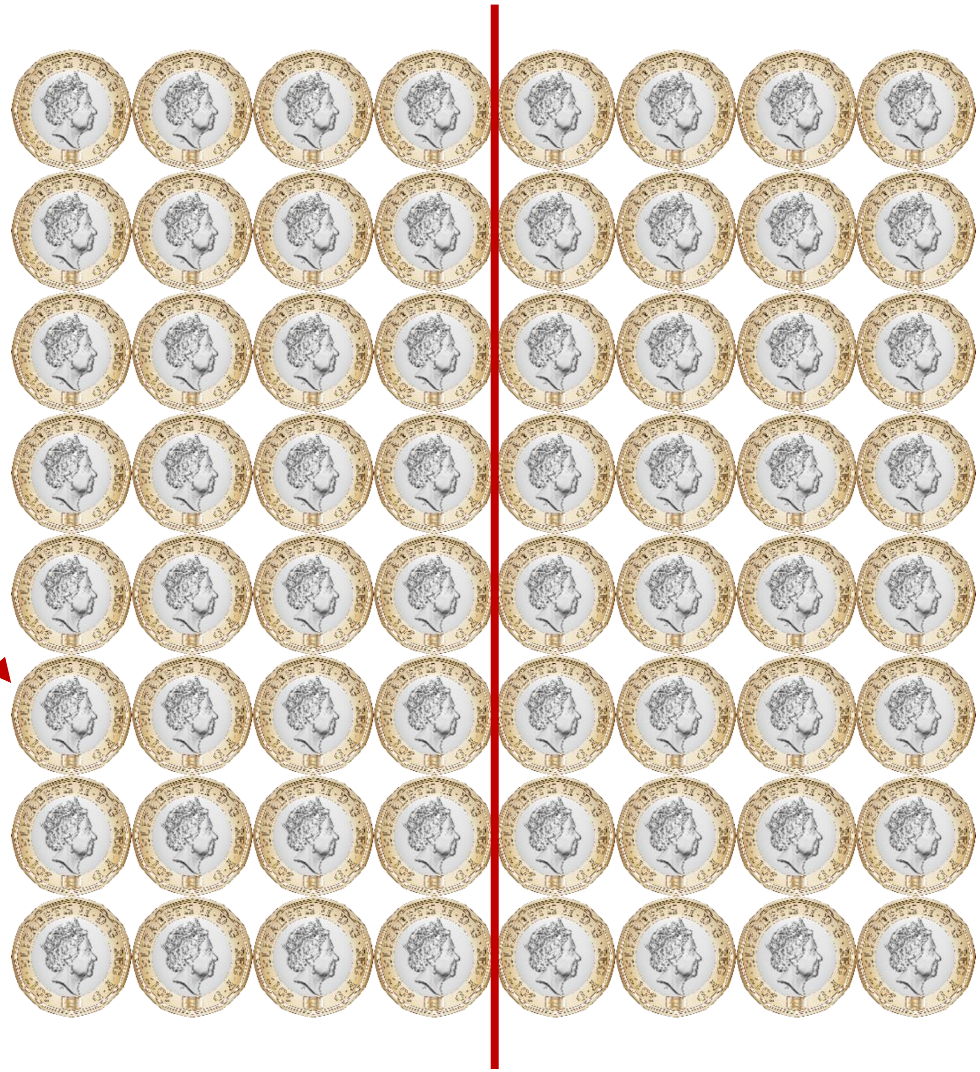
```
If weight(left) ≠ weight(right) Then
    The lighter side contains the fake coin
    Split the lighter N/2 coins …
Else
    Neither side contains the fake coin
End If
```

Fake Coin



STEP 1

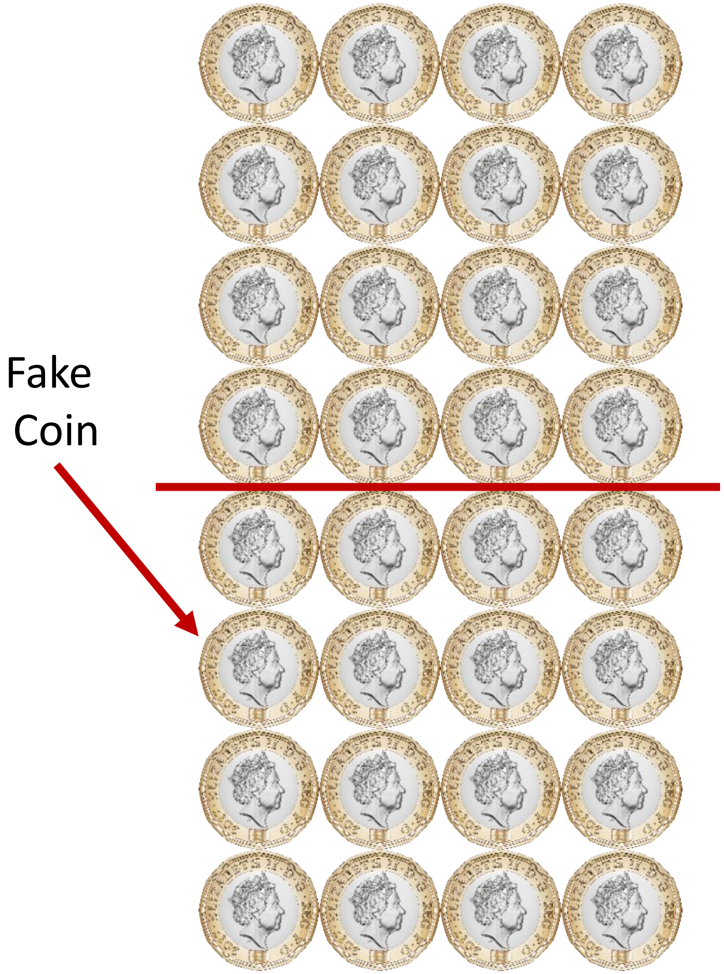
Fake
Coin



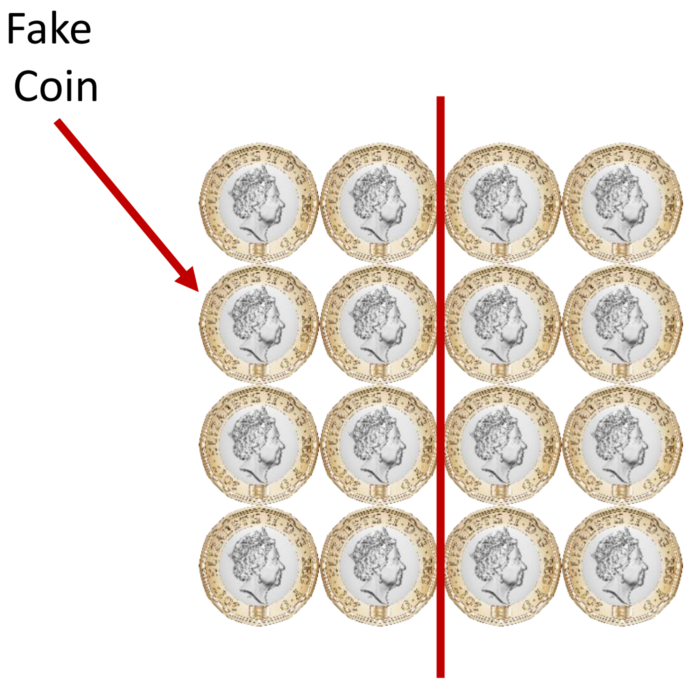
STEP 1

Fake Coin

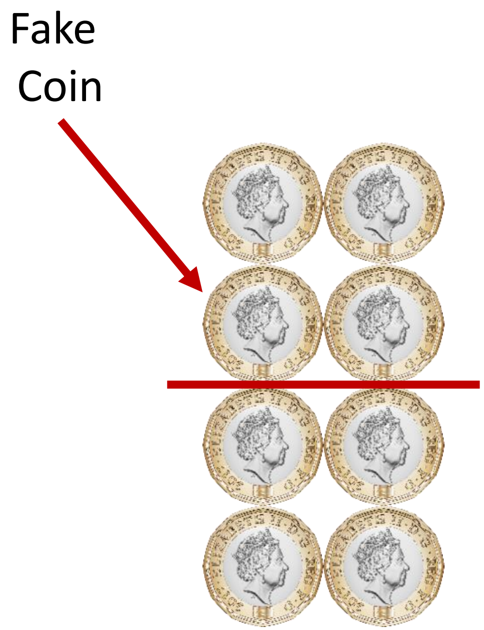
STEP 2



STEP 3

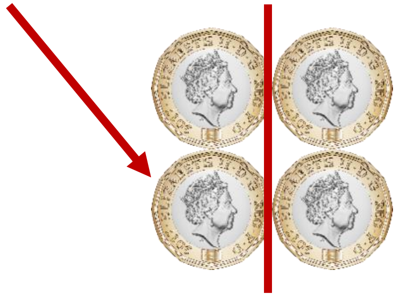


STEP 4



STEP 5

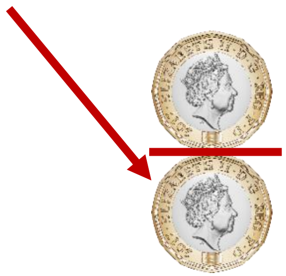
Fake Coin

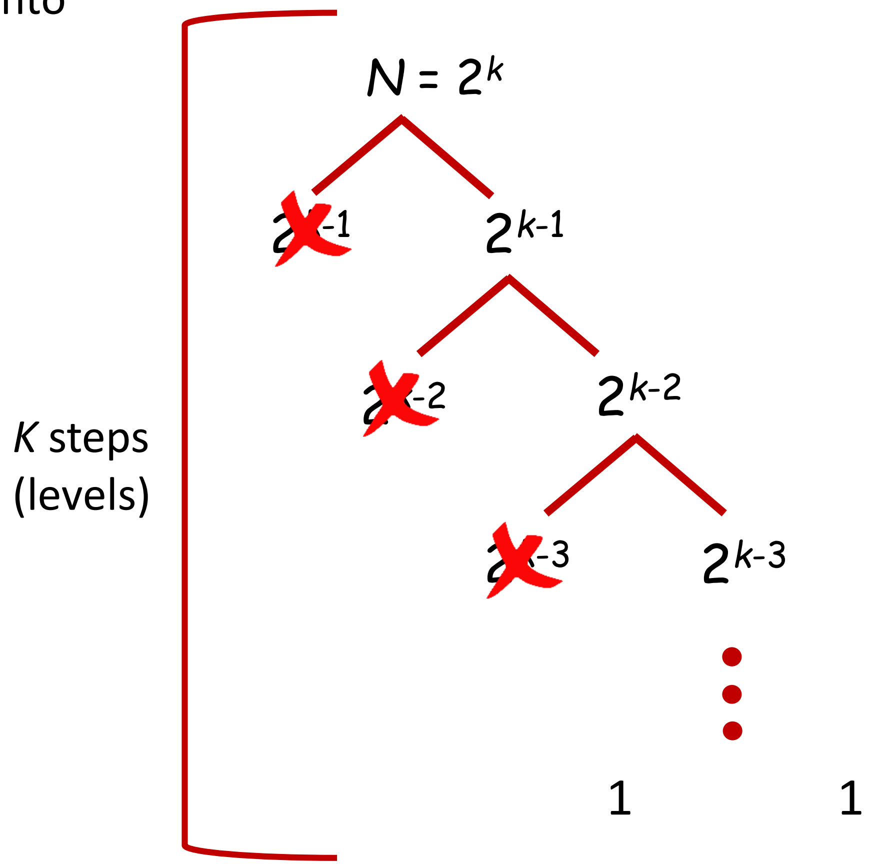


STEP 6

- Remember we started with N=64
- This algorithm divides the coins into 2 equal parts
- We are left with the fake coin after six comparisons
- $N = 64 = 2^6$

Fake Coin





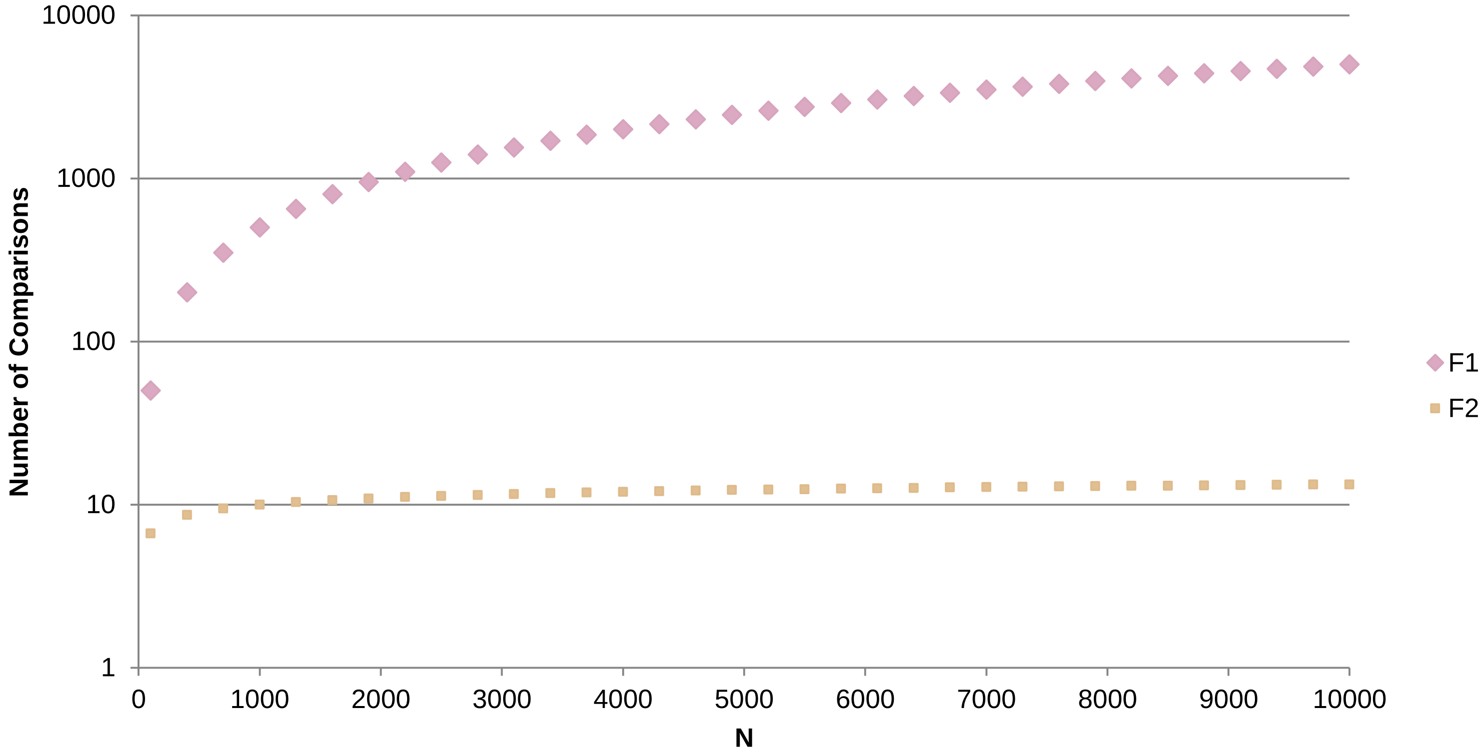
- If $N = 2^k$, then $k = \log_2 N$
- So the order of this algorithm is O(log N)

Example Solution Two – Continued

- Essentially this is a form of **Binary Search**
- The number of splits is equivalent to the number of levels (k) in the corresponding complete binary tree (we will cover this later)
- So the complexity will be $O(k) \rightarrow O(\log N)$
- Since there is only one comparison on each level, the worst case is to conduct log2(N+1) comparisons
- Note: $2^K = N$, $\log_2(N) = k$, e.g. $\log_2(8) = 3$

Example Solution Comparison

- Solution 1: $F1(N) = N/2$, $F1 \in O(N)$
- Solution 2: $F2(N) = \log_2(N+1)$, $F2 \in O(\log_2(N))$



The Analysis of Algorithms

- So there can be many competing algorithms for solving a given problem
- This module will enable you to gain experience in analysing and comparing different algorithms
- Later on in the module we will discuss that there are some problems that are impossible to solve computationally

Introduction to the Module

What are the Aims of this Module?

"This module provides an understanding of a set of useful data abstractions and algorithms. It aims to stimulate students' critical thinking and develop their ability to choose appropriate algorithms in solving practical problems and implementing them in software."

What is to be Taught?

- The concepts of algorithms
  - What is an algorithm?
  - What is a program?
  - What is software?
  - Input and output
  - What is the analysis of algorithms?
- The concepts of data structures
  - Notation, Lists, Queues, Trees, Graphs, Hash tables, etc
- A number of different algorithms
  - Sorting, Searching, Graph traversal, Heuristic search, Evolutionary Computation, Data Clustering, Bin Packing, Optimisation
- A number of different applications
  - Examples of searching, Graph based problems, Bin packing problems, Data clustering gene expression data, Parameter optimisation and The Travelling Salesperson Problem

What is NOT going to be Taught?

- Java
  - This module assumes and requires that you can program in Java
- We are going to teach you how to implement algorithms in Java
- There will be plenty of opportunity to practice
- Most laboratory sessions will require you to program in Java
  - If you are rusty in Java then you should start revising now!
  - Java resources will be made available

Module Team

Dr Mahir Arzoky
Module Leader

Dr Stephen Swift

Dr Alina Miron

Dr Lorraine Ayad

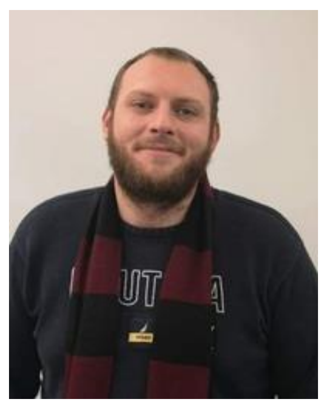
GTA Team


Afees Adebode


Anas Alkasasbeh


Ashley Mann


Faisal Maramazi


Nikolaos Markatos


Zulkarnain Choudhry

Module Structure

- This module spans two terms
- There may not be a lecture or laboratory session every week
  - There is most weeks however...
  - Check on Blackboard (CS2004 Algorithms and their Applications Schedule) – Provisional!

CS2004 Algorithms and their Applications (2020-2021)
Lectures, Laboratories and Assessment

Week Number	LECTURE Week 1: Monday, 16:00-17:00, Online Term 1 (rest) & Term 2: Mondays, 09:00-10:00, Online	LABORATORY Term 1: Online & TOWA 045 (advance booking only) Term 2: Online & TOWA 407/408 (advance booking only)	Laboratory that might be examined in Task #1 and/or #2
1	1. Introduction to Module	No Laboratory	N/A
2	2. Foundations of Algorithm Analysis	1. Familiarisation with the Java Programming Environment 2. Design and Implementation of Simple Algorithms	No
3	3. Mathematical Foundation	3. Mathematical Foundation	Yes
4	4. Time Complexity and Asymptotic Notation	4. Analysis of Algorithms	Yes
5	5. Data Structures and their Applications	5. Data Structures	Yes
6	6. Classic Algorithms – Sorting	6. Sorting Algorithms	Yes
7	[Reading Week – No Lecture]	None – Private Study on Worksheets	N/A
8	7. Classic Algorithms – Graph Traversal	7. Graph Traversal – MST	Yes
9	8. Graph Traversal – Dijkstra's Algorithm	Catch Up Laboratory	N/A
10	9. Search and Fitness	8. Fitness Functions – the Scales Problem	Yes
11	10. Hill Climbing and Simulated Annealing	9. Hill Climbing – the Scales Problem	Yes
12	No Lecture	No Laboratory	N/A
Christmas	(Weeks 13-15) No Lectures	No Laboratories	N/A
16	11. Applications, Introduction and Parameter Optimisation	10. Parameter Estimation – Projectile Modelling	Yes
17	12. Tabu Search and Iterated Local Search	Catch Up Laboratory	N/A
18	13. An Introduction to Genetic Algorithms	11. A Simple Genetic Algorithm Applied to the Scales Problem	Yes
19	14. Further Evolutionary Computation	12. Further Genetic Algorithms	Yes
20	15. Ant Colony Optimisation and Particle Swarm Optimisation	13. Ant Colony Optimisation and Particle Swarm Optimisation	No
21	16. The Travelling Salesperson Problem	14. The Travelling Salesperson Problem	Yes
22	[Reading Week – No Lecture]	None – Private Study on Worksheets	N/A
23	17. Bin Packing and Data Clustering	15. Data Clustering	Yes
24	No Lecture	Catch Up Laboratory	N/A
25	No Lecture	**CODERUNNER EXAMINATION**	N/A
26	No Lecture	No Laboratory	N/A
Easter	(Weeks 27-29) No Lectures	No Laboratories	N/A
30	18. Revision Lecture	No Laboratory	N/A

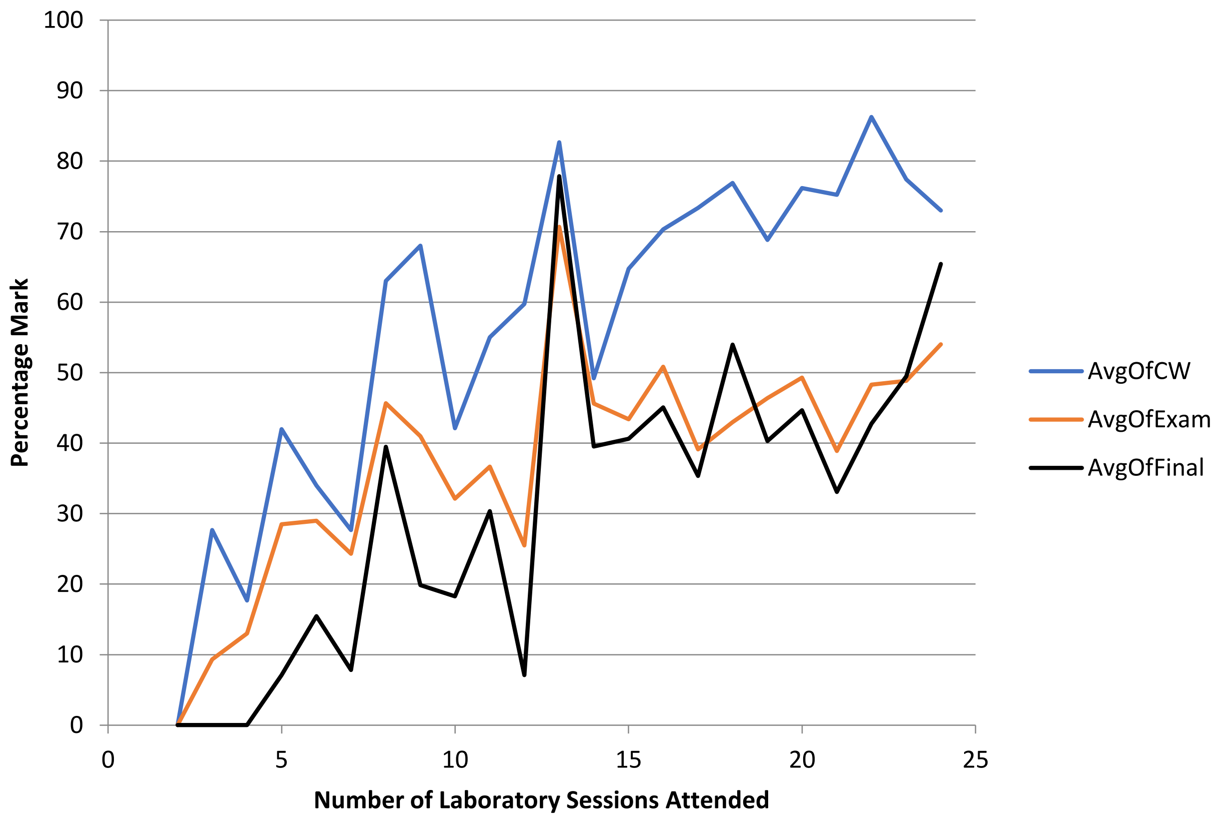
Lectures

- Online only
- Schedule on Blackboard:
  - Week 1: Monday 4-5pm
  - Weeks 2-6, 8-11, 16-21, 23, 30: Mondays, 9-10am
- Cover the theory aspects of the module
- Make sure you attend every lecture
- Lectures will be recorded, and the recording will be made available

Laboratory Sessions

- These will consist of worksheets on the subject of the week's lecture
- Starts week 2 (next week)
  - Laboratory 1
    - Mondays 13:00 to 15:00
  - Laboratory 2
    - Mondays 15:00 to 17:00
- Only attend your session
- Online and TOWA 045 (Term 1) / TOWA 407/408 (Term 2)
- Booking system will be made available for in-person attendance
- 2 hours supervised – there will be Staff/GTAs/ATAs present
- There will be a demo at the start of the session
- Laboratory sessions are NOT recorded – make sure you attend!

Laboratory Sessions - Continued



Correlation

	No. Labs
CW	0.85
Exam	0.73
Final	0.77

Assessment

- **Laboratory worksheets / Coursework (60% weight)**
  - A number of the laboratory worksheets will be assessed
  - Similar (but not the same) to the way Java was assessed - CodeRunner
- **Exam (40% weight)**
  - Theory based
  - This will test the concepts of algorithms and data structures
  - There will be no programming needed in the exam

Laboratory Sheets / Coursework

- Two tasks - making 60% of the module…
- Task #1: 30% of the coursework
  - Sample "topics" from assessed worksheets
  - All class tests (CodeRunner) needs to be completed by Monday Week 21
  - There will be four class tests in total and they will be released every few weeks
  - All four 'class' tests must be passed to pass Task #1
- Task #2: 70% of the coursework
  - CodeRunner Examination
  - During the laboratory sessions of week 25
  - If you have successfully passed Task #1 then Task #2 builds on the marks attained from Task #1
  - If you have not passed Task #1, you can still attempt Task #2 but you will be capped at D- grade

CodeRunner

- ❑ CodeRunner is an open-source question-type plug-in for Moodle
- ❑ It runs program code submitted by you in answer to a programming questions
- ❑ You can correct your code and resubmit (for a small penalty)
- ❑ Some of your solution program code will be automatically run against test datasets for accuracy
- ❑ CodeRunner will be used for both Task #1 and Task #2
- ❑ **If you haven't done the worksheets then you will NOT have time during the class tests…**

Exam

- A three hour **WiseFlow exam**
  - The exact time and date is not available yet
- This will consist of Essay-type questions.
- Past papers will be made available soon…

Resources

- Lecture slides, lecture recordings, laboratory worksheets, past paper etc...
- This module does not follow any set text
- The text books listed on Blackboard (in the study guide) are a very useful reference
- Eclipse and Java have a very well documented help system

Acknowledgment

- Materials has been “borrowed” from:
  - Dr Stephen Swift who wrote and ran the module from 2011-2019
  - FoIT
  - Jorum (JISC funded lecture notes repository – now closed)
    - Rong Yang, University of West England
    - Dr Jim Smith, University of the West of England

Next Topic

- Lecture
  - We will next look at algorithm analysis in more detail
- Laboratory
  - There are no formal laboratories this week