Computational Thinking with Algorithms - Introduction

Dominic Carr

Galway-Mayo Institute of Technology

dominic.carr@gmit.ie

What We Will Cover

1. Goals of this Session

2. Module Introduction
   - Module Descriptor
   - Learning Outcomes
   - Indicative Content
   - Library Resources

Goals of this Session

Goals

- To understand....
  - What the module is about
  - The learning outcomes of the module
  - What the content will be like

Module Introduction

Module Descriptor

The aim of this module is to provide a comprehensive grounding in solving **computational problems** and *designing algorithms*.

Learning Outcomes

- LO1 Apply structured methodologies to problem solving in computing.
- LO2 Design algorithms to solve computational problems.
- LO3 Critically evaluate and assess the performance of algorithms.
- LO4 Translate real-world problems into computational problems.

Indicative Content I

- Computational Thinking
  - Understanding and describing problems
  - Modelling real-world problems
  - Abstraction and experimentation
  - Undecidable problems
- Algorithm design
  - Searching
  - Sorting
  - Data structures
  - Flow diagrams

Indicative Content II

- Analysis of Algorithms
  - Sizing a problem
  - Rates of growth
  - Best, average and worst cases
  - Benchmarks
- Using the literature
  - Accessing the literature
  - Reading the work of others
  - Writing literature

Books]

- **Algorithmics - The Spirit of Computing** - 3rd Edition David Harel & Yishai Feldman Springer
- **Algorithms in a Nutshell** George T. Heineman, Gary Pollice, and Stanley Selkow O' Reilly
- **Data Structures and Algorithms in Java - (4th edition)** Michael T. Goodrich and Roberto Tamassia John Wiley & Sons Inc

What is Computer Science? I

"We need to *do away* with the myth that computer science is about **computers**. *Computer science is no more about computers than astronomy is about telescopes* ... Science is not about tools, it is about how we use them and what we find out when we do." - Michael R. Fellows, Ian Parberry

What is Computer Science? II

Computer science is the study of *processes that interact with data* and that can be represented as data in the form of programs. It enables the use of *algorithms to manipulate, store, and communicate digital information*.

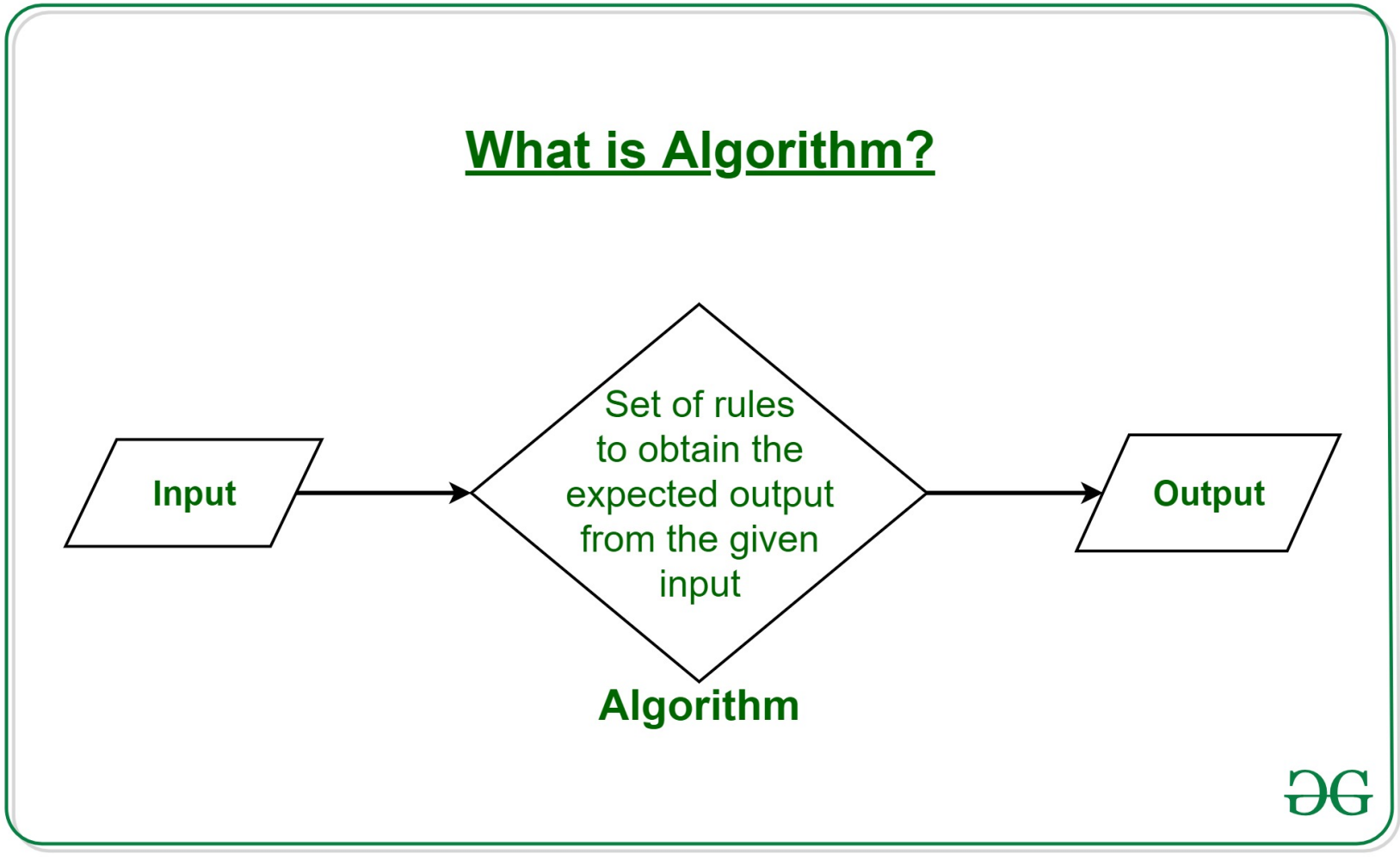
What is an Algorithm? I

Dictionary says...

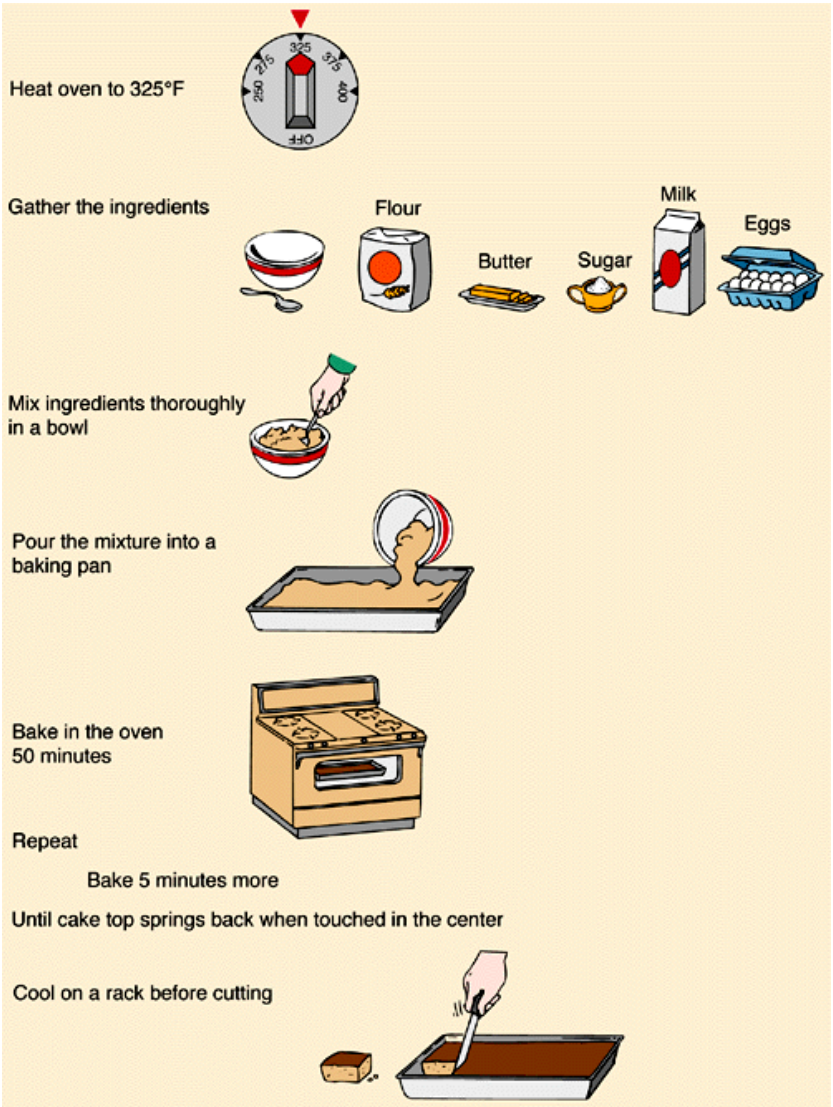
A set of rules that must be followed when solving a particular problem

- Set of instructions
- A "recipe"
- a process to be followed
- Can be very simple....or very complicated
- Can be more than one way of solving a problem
  - i.e. more than one algorithm

What is an Algorithm? II



What is an Algorithm? III



What is an Algorithm? IV

- Make tea
  - Fill kettle
  - Boil kettle
  - Put tea bag in a cup
  - Pour the water
  - Add milk
- Alternatively
  - Put tea bag in a cup
  - Take to water boiler
  - Open the spout for hot water
  - Add sugar
  - Stir

What is an Algorithm? V

- Properties of an algorithm
  - Speed
  - Efficiency
  - Readability
  - Complexity
  - Computability?

What is an Algorithm? VI

A programming task can be decomposed as follows:

- Problem Solving
  - Produce an order sequence of steps (operations) which solve the problem
  - this is the algorithm
- Implementation
  - Take that series of steps and convert it into a real programming language
  - So that the machine can run the algorithm

Why does it matter? I

You follow algorithms in daily life and without them you would not be able to function!

- crossing the road
- Brushing teeth
- Cooking
- Clean
- ....

Why does it matter? II

There are many computer-driven algorithms which you interact with every day:

- GPS directions which utilize a "shortest-path" algorithm
- Encryption algorithms to protect online banking and purchasing
- Delivery services using algorithms to assign drivers
- Spam detection on your email services
- Content filtering on your Facebook feed
- ....

Why does it matter? III

While humans can use imprecise algorithms to accomplish tasks, the same cannot be said for computers

- E.g. when driving to your workplace, your algorithm may contain a rule like: "If traffic is heavy, take an alternate route"
- We know intuitively what is meant by "heavy traffic", but a computer must be explicitly programmed to deal with this scenario
- Therefore, algorithms for computational devices must be described precisely enough that a computer can run them successfully, achieving the desired outcome

Why does it matter? IV

Computer do precisely what we tell them to do

Design criteria for algorithms I

A well-designed algorithm should:

- Produce a "correct" solution for a given input
- Use computational resources efficiently

Design criteria for algorithms II

- Correctness
  - Easy to define in (simple) real world problems, e.g.
    - Is a book by author X present on this bookshelf (boolean true/false answer)
    - What is the shortest route between two points (one unique solution)
    - Sort pupils in a class by increasing height (one correct answer)
  - More difficult to define correctness in other real world problems, e.g.
    - Which route will take the least time to traverse (answer depends on factors other than geometry, i.e. traffic conditions)
    - Handwriting recognition (is that character a number 5 or a letter S)?

Design criteria for algorithms III

- In this module we will focus mainly on problems which have knowable solutions
- Incorrect solutions are acceptable occasionally, so long as the expected probability of returning an incorrect solution can be quantified/controlled
- For many practical applications, algorithms have been developed which have a guaranteed margin of error, rather than a guarantee of complete correctness

Design criteria for algorithms IV

- Efficiency
  - If a routing algorithm could produce a route that avoided all congestion, but took an hour to do so, would it be useful?
  - An algorithm that produces a correct solution but takes a long time to run may be useless in practice. This is the reason that more efficient algorithms that compute approximate solutions with guaranteed margins of error are often adopted in practice

Design criteria for algorithms V

- Factors which may be taken into consideration when evaluating the efficiency of an algorithm:
  - Time (real world/ CPU time)
  - Memory (RAM) usage
  - Storage (Hard disk) usage
  - Number of read/write operations (especially on disk) • Network usage
  - Random bit requirements (some algorithms require a source of random numbers) • We will focus mostly on time efficiency in this module

Design criteria for algorithms VI

How does the time taken for an algorithm to execute vary with the size of the input?

The End