Data Structures & Algorithms 1

Topic 1 – Introduction

Course Text-Book

- "Data Structures and Algorithms in Java"

  by Robert Lafore
- Core text for both CS210 and CS211
- Course website : moodle CS210
- My email address : pmaguire@cs.nuim.ie

Course Content

- CS210 and CS211 are NOT programming courses
- The idea is to provide you with concepts: tools you need to write efficient programs
- The programming language you use and the way you implement these ideas is up to you
- You will learn efficient structures for storing information and handy algorithms for manipulating that information

CS210

- Java Revision
- Object-oriented programming
- Array Algorithms
- Algorithm Complexity
- Big O Notation

CS210

- Simple Sorting
  - Bubble sort
  - Selection sort
  - Insertion sort
- Stack and Queues
- Linked Lists
- Recursive Mergesort

What is a data structure?

- A data structure is a conceptual structure for organizing information
- There are many different ways of organizing information
- Different structures have different advantages and disadvantages

Data Structure

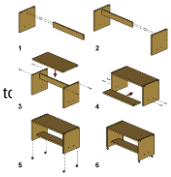
- What are the advantages and disadvantages of this data structure?
- Easy to add to
- Difficult to find stuff
- Wasteful of space

Data Structure

- What are the advantages and disadvantages of this data structure?
- Easy to find stuff
- Laborious to add stuff
- Difficult to expand

What is an algorithm?

- An algorithm is a precise step-by-step plan for solving a problem using a finite sequence of instructions
- But how do we know if a set of instructions is comprehensive?
- It proved extremely difficult for mathematicians to pin down a precise definition for 'algorithm'
- Alan Turing finally managed to provide a comprehensive definition for 'algorithm' in the 1930s: a program that can run on a Turing machine



Hilbert's Entscheidungsproblem

- David Hilbert set up a research program in the 1920s to define the foundations of mathematics
- He wondered whether there was an algorithm that could decide whether any statement was true or false
  - This is called the *Entscheidungsproblem* (German for decision problem)
- Hilbert believed there was no such thing as an unsolvable problem in maths
  - *"We must know, we shall know"*

Alan Turing (1912 – 1954)

- Alan Turing was an English mathematician and logician who is now considered the father of computer science
- In school Turing fell in love with an older male student whose subsequent death shattered his belief in God and turned him into a atheist
- He was elected a fellow in Cambridge based on the strength of his final year dissertation

What is an algorithm?

- Alan Turing wanted to prove that no algorithm could ever resolve the Entscheidungsproblem
- But what exactly is an algorithm?
- Turing observed some computers at work
- At the time women were often employed as 'computers' as they represented a source of cheap labour and had mathematical training

Computers at work

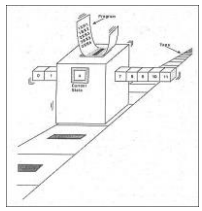
- The ‘computer’ makes marks on a sheet of paper
- She shifts her attention from what she had written earlier to what she is writing now
- She keeps a small amount of information in her working memory
- Turing wanted to strip out all of the detail such as whether the computer is using a pencil or a pen

Turing Machines

- Alan Turing’s greatest contribution was in coming up with a model of computation that provides a complete description of all of the physical steps involved in following an algorithm
- Turing’s model of computation is now called a Turing machine (he called them a-machines)
- Turing accounted for every single step down to the manipulation of the smallest piece of information
- In doing so he had managed to reconcile the field of logic with mechanical processes occurring in the physical world

Turing Machine

- It is important to note that a Turing Machine is just a thought experiment involving an abstract model of computation
- It has an infinitely long tape and a read/write head that can move along the tape changing the values along the way
- This simplistic theoretical machine can be used to simulate the computation of any algorithm



Halting

- Turing realised that for every algorithm that is run there are only two possible outcomes
  - Either the algorithm will terminate at some stage (called halting)
  - Or else it will run forever
  - Obviously it has to be one or the other!

Halts

```
for(int i = 0; i<10; i++){
        System.out.println("Still looping...");
}
```

Loops forever

```
while(true==true){
        System.out.println("Still looping...");
}
```

- But from just looking at an algorithm is it possible to predict in advance which category it will fall into?

How Turing did it

- Turing realised that he could use the halting / looping distinction to expose the limitations of Turing machines
- He proved the Entscheidungsproblem was unsolvable by showing that you can seize up any Turing machine by feeding it a description of itself and giving it the following instructions:

> *Computer,*
>
> *Take this description of an algorithm and check to see if it halts or loops forever. If it halts at some point then go into an infinite loop. However, it you find that it loops forever then halt.*

Feed the computer itself...

“*101010010100011110 101001010001001110 110110010100101010 010110101010...*”.

Turing's influence

- Turing's efforts to solve Hilbert's Entscheidungsproblem led to several breakthroughs
  1. Turing had to specify a model of computation (called the Turing machine) to allow the idea of an algorithm to be comprehensively defined for the first time
  2. To feed the machine itself, Turing had to show that machines, programs and data could all be represented as input data
  3. As a result, Turing realised that it is possible to have one Turing machine which is capable of simulating all other machines (the Universal Turing Machine)
  4. By showing that no algorithm can solve the Entscheidungsproblem, Turing showed that there are some questions which are unsolvable

Machine Program Data

- Before Turing it was assumed that the machine, program and data were all different

Machine: 

Program: 

Data: 

Turing Machines

- Turing's model allowed data, algorithm and machine to be treated as abstract patterns
- Computers represent patterns of information using binary code
  - 11011010110101
- The Church-Turing thesis speculates that the entire universe can be simulated by a Universal Turing Machine

Incomputability

- So, not all problems can be solved by an algorithm
- In fact, there are infinitely more problems than there are algorithms to solve them
- We will study problems that can be solved by algorithms in reasonable amounts of time
- Next year you will study problems that cannot be solved in the module Theory of Computation

Question

- A Turing machine is:

A) The machine built by Turing to solve the Entscheidungsproblem
B) A machine that Turing designed so he could simulate all other computers
C) A device for representing fundamental data structures
D) A concept that formalises the definition of an algorithm
E) An algorithm for computing the answer to the Entscheidungsproblem

Summary

- **Algorithm**: A step-by-step procedure for solving a problem via a computational process
- **Program**: An implementation of an algorithm in some programming language
- **Data Structure**: A conceptual system for organizing the data needed to solve some problem

How is this module relevant?

- This module is about nuts and bolts concepts
- We will learn to write searching and sorting algorithms and data structures from scratch
- Programmers who aren't interested in the nuts and bolts can simply use pre-written objects in Java like LinkedList( ) or Hashtable( ) that do exactly the same thing
- However, if you don't understand the code you're using, your program might end up being far more inefficient than it needs to be

Laying the foundations

- Some software employers are primarily concerned with high-level application development and not too worried about the nuts and bolts
- However, employers like Microsoft, IBM, Google and Intel need "software engineers" – people who can build a system from scratch
- Although your future programming experience may be at a higher level, understanding the core foundations is critical