CSCB63 Week 2 Lecture 1

Class	CSCB63
Created	@Oct 18, 2020 3:28 PM
Type	Lecture

Data Structures and Analysis

ADT - Abstract Data Type

This is a set of objects and operations that can be performed on these objects.

Examples:

- Integers, with operations `ADD(x, y)`, `SUBTRACT(x, y)`...
- Stack, with operations `PUSH(s, x)`...

Data Structure

A data structure is an implementation of an `ADT`.

For example, a stack could be implemented by either a singly-linked list, or an array.

ADTs are important for `specification`, and it provides `modularity` where the usage depends only on the definition rather than the implementation. We can change the implmentation of an `ADT` without changing the rest of the program. These abstract data types can also be implemented once and used in lots of different programs.

Summary

An `ADT` is a way to describe `what` the data is and `what` you can do with it.

A `data structure` is a way to describe `how` the data is implemented and `how` the operations are performed.

Complexity

The complexity of an algorithm is the amount of `resources` an algorithm uses. We quantify this by expressing as a function of the `size` of the input.

Types of resources:

- Running time
- Speed (memory)
- number of logic gates

Input Size

The definition of `input size` will depend on which `types` of objects we are talking about:

- Integer: number of bits
- List: number of elements
- Graphs: number of `vertices` and `edges`

The `running time` of an algorithm is the number of `primitive` operations of `steps` executed. This also depends on the problem. The notion of `step` should be machine `independent`.