Hugh Baldwin
up2157117

Data Structures and Algorithms

M21270
TB1

University of Portsmouth
BSc Computer Science
2nd Year

Contents

- Mid-Unit assessment (30%) on 8th November
- Exam (70%) in main assessment period

Lecture - Workshop 1 and 2 Async

16:00 27/09/23 Dalin Zhou

Data Structures

- A data structure is a way to store and organise data
  - A collection of elements
  - A set of relations between the elements
- Classification of data structures
  - Linear
    - Unique predecessor and successor
    - e.g. Stack, Queue, etc
  - Hierarchical
    - Unique predecessor, many successors
    - e.g. Family tree, management structre, etc
  - Graph
    - Many predecessors, many successors
    - e.g. Railway or road map, social network
  - Set
    - No predecessor or successor
    - e.g. A class of students
- Static vs Dynamic
  - A static data structure has a fixed size, which cannot be exceeded and must be allocated in it's entirety when created
  - A dynamic data structure has a dynamic size, which can change at runtime and only uses as much memory as is needed for it's contents
- Most data structures follow CRUD(S) for their basic operations;
  - Create - Add a new element
  - Read - Read an existing element
  - Update - Modify an existing element
  - Destroy (or Delete) - Remove an existing element
  - (Search) - Search for an element matching certain search conditions
- Each program would need a different data structre - there is no one-size-fits-all

Abstract Data Types

- An ADT is a collection of data and associated methods
- The data in the ADT cannot be directly accessed, only by using the methods defined in the ADT
- Stacks
  - A stack is a collection of objects in which only the top-most element can be modified at any time
  - This means it is a LIFO (Last-in first-out) structure
  - A stack must implement the following methods:
    * Push - Add an item to the top of the stack
    * Pop - Remove the item from the top of the stack
    * Peek - Examine the item at the top of the stack
    * Empty - Determine if the stack is empty
    * Full - Determine if the stack is full
    * A stack is typically implemented using an array, which is hidden behind the methods of the ADT
- Queues
  - A queue is a collection of objects in which the object which has been in the queue for the longest time is removed first
  - This means it is a FIFO (First-in first-out) structure
  - A queue must implement the following methods:
    * Enqueue - Add a new item to the tail of the queue
    * Dequeue - Remove the item at the head of the queue
  - A queue is typically implemented using an array, which is hidden behind the methods of the ADT
  - There are two methods of implementing a queue
    * Fixed head
      · The head of the queue is always at the 0th position in the array, and elements are shifted as they are dequeued
    * Dynamic head
      · The head of the queue changes to the index of the current head of the queue, and elements are not shifted
      · Since this leaves empty spaces before the head, space may be wasted. To solve this, a circular array could be used

Algorithms

- An algorithm is a procedure that takes a value or set there of as input and produces a value or set there of as an output
- A sequence of computational steps that transforms the input into the output
- Ideally, we would always use the most efficient algorithm that is available
- Classification of algorithms
  - Brute-force
  - Divide and conquer
  - Backtracking
  - Greedy
  - etc

Big-O Notation

- To determine the Big-O of an algorithm
  - Count how many basic operations (assignment, addition, multiplication and division) there are for the worst-case scenario (e.g. 10 items in a stack of length 10)
  - Ignore the less dominant terms of this equation
  - Ignore any constant coefficients
  - The remaining terms become the Big-O complexity of the algorithm
- Example 1:
  - There are $2n + 1$ operations for an algorithm
  - We can ignore the $+1$ as this has less impact, leaving $2n$
  - We can ignore the constant coefficient 2, leaving n
  - So the Big-O notation would be $O(n)$
- Example 2:
  - There are $2n^2 + n + 1$ operations for an algorithm
  - We can ignore the $+1$, leaving $2n^2 + n$
  - We can ignore the $+n$, leaving $2n^2$
  - We can ignore the 2, leaving n^2
  - So the Big-O notation would be $O(n^2)$
- Order of dominance:
  - Constant < Linear < Logarithmic < Quadratic < Cubic < ...
  - $O(1) < O(n) < O(n \log_2 n) < O(n^2) < O(n^3) <$ ...