Computing Faculty
Module Descriptors
Spring 2019

BSc in Computing Science

Stage 2

Module	Data Structures and Algorithms
Course code	BSCH-DSA
Credits	10
Important notes	Please note that this module is not intended for first year students.
Allocation of marks	60% Continuous Assessment
	40% Final Examination

Intended Module Learning Outcomes

Upon successful completion of this module, you will be able to:

1. implement and use data structures introduced on the course;
2. apply object-oriented methods when designing data structures;
3. implement both recursive and non-recursive solutions to classical data structure problems;
4. implement both linear and non-linear data structures;
5. analyse simple algorithms using asymptotic analysis;
6. compare the efficiency of algorithms solving similar problems;
7. implement algorithms on data structures and relate these to realistic problems;
8. implement algorithms that process graphs.

Module Objectives

This module builds on the work completed in the Object-oriented Programming module and applies the methods learned there to the design of classes that implement data structures. As in all programming modules, a key objective is the acquisition, on behalf of the learner, of good software engineering skills and the application of these skills to the design and implementation of software components. At the heart of all software design is the implementation of appropriate data structures that provide efficient data models for the problem at hand. Learners develop an in depth knowledge of the standard data structures: stacks, queues, sets, bags and maps; and also learn to implement these using both linear (linked lists, arrays) and non-linear (binary search trees, avl trees, B-trees) data structures.

Module Curriculum

Recursion

- Concept of recursion,
- recursive functions,
- recursion over sequences,
- tail recursion.
- Divide and conquer algorithms.

Analysis of algorithms

- Counting and calculating the time complexity of an algorithm.
- Basic ideas and definitions of asymptotic analysis.
- Big O notation and its application to the evaluation of temporal cost of algorithms.
- Comparing performance using big O notation.
- Basic time and space analysis.

Sorting

- Simple sorting algorithms: insertion, selection, bubble.
- QuickSort. Merge Sort. Heapsort.
- Analysis of sorting algorithms using big O notation.
- Sorting in linear time.

Dynamic Data structures

- Constructing lists using singly linked lists and doubly linked lists.
- Using these linear data structures to implement classes that encapsulate: stacks, queues, priority queues and sets.
- Problem solving with these data structures.
- Concept and definition of Hashtables.
- Implementing classes that encapsulate a hash table and also the use of hash tables in implementing sets.
- Concept and definition of Trees: representing rooted trees, binary search trees, query, insertion, deletion, traversal.
- Optimising the performance of binary trees with avl trees and black-red trees and B-trees.
- Implementing data structures with trees.

Graphs and graph algorithms

- Graphs: basic concepts and representation.
- Breadth first search, depth first search.
- Dijkstra shortest-path algorithm,
- Dijkstra-Prim minimum spanning tree algorithm.