Computer Science

Algorithms (COMP0005)

Description

Aims:
To develop programming and problem solving skills, to encourage a thoughtful approach to analysis and design problems, to familiarise students with logical and mathematical inference and argumentation.

Learning outcomes:
On successful completion of the module, a student will be able to: apply this knowledge to specification of algorithms and logic programming.

Content:
Searching and sorting algorithms:
-Elementary searching (Linear and binary search);
-Binary search trees;
-Hash Tables, Hashing, Hash functions;

Graph algorithms:
-Directed acyclic graphs;
-PERT chart;
-Dijkstra’s algorithm;
-Bellman-Ford algorithm;
-Floyd-Warshall algorithm;

Text algorithms:
-String matching;
-Longest Common Sequence;
-Sequence/subsequence;

Analysis of algorithms:
-Algorithmic complexity;
-Upper and lower bounds;
-O notation;
-Best, worst and average cases;
-Classifications of algorithms;

Linked Lists:
-Linked structures;
-Singly-linked list, Doubly-linked list;
-Circular linked list;

Key information

Year: 2019/20
Credit value: 15 (150 study hours)
Delivery: UG L4, Campus-based
Reading List: View on UCL website
Tutor: Dr Ifat Yasin
Term: Term 2
Timetable: View on UCL website

Assessment

- Coursework: 30%
- Coursework: 30%
- Written examination (main exam period): 40%

Find out more

For more information about the department, programmes, relevant open days and to browse other modules, visit ucl.ac.uk

Disclaimer: All information correct as of June 2019. Please note that aspects of the module may be subject to change. UCL will make best efforts to inform applicants of major changes.
Abstract Data Types:
- ADTs;
- Defining the ADT;
- Implementing the ADT;
- Bag ADT;
- List-based Implementation;
- Queue ADT;

Dynamic Programming:
- Elements of dynamic programming;
- Rod cutting;
- Matrix-chain multiplication;
- Optimal binary search trees;

Recursion:
- Properties;
- Run time stack, tail recursion;
- Recursive applications, recursive binary search, exponential operation;

Analysis of searching and sorting algorithms:
- Binary search;
- Selection sort, Insertion sort, mergesort, quicksort, radix sort;
- Bubble sort, lower bounds for sorting;
- Working with sorted lists;
- Heapsort, building a heap, heapsort algorithm, priority queues;

Greedy Algorithms:
- Elements of greedy strategy;
- Huffman codes;
- Matroids;

Requisites:
In order to be eligible to select this module, a student must be registered on a programme for which it is a formally-approved option or elective choice AND must have taken in Term 1 Theory of Computation (COMP0003).