# CUHK (SZ)

**Course Outline**

1. **Course Identity**
   1. **Course as listed in CUHK (SZ)**

The information in this block should be exactly as approved by CUHK Senate. In case there are any differences, please explain in the table below.

|  |  |
| --- | --- |
| Course code | CSC3100 |
| Course title (English) | Data Structures |
| Course title (Chinese) | 数据结构 |
| Units | 3 |
| Description (English) | The concept of abstract data types and the advantages of data abstraction are introduced. Various commonly used abstraction data types including array, stack, queue, list, tree, and graph, and their implementations will be discussed. In addition, advanced data structures such as heap (Min-max heap, Deaps, Binomial heaps, Fibonacci heaps) and search structures (Optimal binary search treesAVL trees 2-3-4 trees, Read-black trees, B-Trees) will be covered. Some applications such as searching and sorting are also described to illustrate the use of data structures introduced in the textbook. |
| Description (Chinese) |  |

# Corresponding course in CUHK

Please give details of the *closest* corresponding course in CUHK (as approved by CUHK Senate and listed in course list). If the course in SZ maps to more than one course in CUHK, please make multiple copies of the block below.

|  |  |
| --- | --- |
| Course code | CSCI2100/ESTR2102 |
| Course title (English) | Data Structures |
| Course title (Chinese) |  |
| Units | 3 |
| Description (English) | The concept of abstract data types and the advantages of data abstraction are introduced. Various commonly used abstract data types including vector, list, stack, queue, tree, and set and their implementations using different data structures (array, pointer based structures, linked list, 2-3 tree, B-tree, etc.) will be discussed. Sample applications such as searching, sorting, etc., will also be used to illustrate the use of data abstraction in computer programming. Analysis of the performance of searching and sorting algorithms. Application of data structure principles. |
| Description (Chinese) |  |

|  |  |
| --- | --- |
| Course code | CSCI2520 |
| Course title (English) | Data Structures and Applications |
| Course title (Chinese) |  |
| Units | 3 |
| Description (English) | This course formally examines the relationship between abstract data types and data structures. The implementation of abstract data types using various data structures will be discussed. Abstract data types including list, stack, queue, symbol table, tree and graph will be introduced. Introductory complexity analysis and big-O notation will be illustrated with simple algorithms such as searching and sorting. |
| Description (Chinese) |  |

1. **Prerequisites / Co-requisites**

Please state prerequisites and co-requisites, in terms of courses in CUHK (SZ)\* or any other requirements (e.g., having taken certain subjects in high school).

(\* Because course codes may not yet be stable, please provide both course code and course tile.)

# Prerequisites

CSC1001 Introduction to Computer Science: Programming Methodology

# Co-requisites

N/A

1. **Learning Outcomes**

Upon completing this course, students will:

* Understand the important role of data structure in computer science
* Learn the concepts and implementations of various data structures and the associated operations on them
* Understand the advantages and disadvantages of each data structure covered in the course
* Learn some fundamental algorithms
* Be able to analyze a given algorithm in terms of correctness and complexity
* Be willing to learn more in computer science
* Be prepared for further study in data structures and algorithms, and further in computer science

# Course syllabus

The goal of this course is to provide an introduction to the concept of Data Structures. Some programming homework assignments will also be given to let students understand and practice those data structures introduced in the classes. Topics covered will include the following:

        Basic Concept

       Arrays

  Array as an abstract data type

   Polynomial abstract data type

   Sparse matrix abstract data type

   String abstract data type

        Stacks and Queues

   Stack abstract data type

   Queue abstract data type

   Evaluation of expressions

   Multiple stacks and queues

       Lists

   Singly linked list

   Dynamically linked stacks and queues

   Polynomials

   Additional list operations

   Equivalence relations

   Sparse matrices

   Double linked lists

       Trees

   Binary tree and traversals

   Additional binary tree operations

   Threaded binary trees

   Heap

   Binary search trees

   Selection trees

   Set representations

   Counting binary trees

       Graphs

   Graph abstract data type

   Elementary graph operations

   Minimum cost spanning trees

   Shortest paths and transitive closure

   Activity networks

       Internal Sorting

   Searching and list verification

   Insertion sort

   Quick sort

   Merge sort

   Heap sort

   Radix sort

   List and table sorts

   External sorting

       Hashing

   The symbol table abstract data type

   Static hashing

   Dynamic hashing

       Heap Structures

   Min-max heap

   Deaps

   Binomial heaps

   Fibonacci heaps

       Search Structures (Selected topics)

   Optimal binary search trees

   AVL trees

   2-3-4 trees

   Read-black trees

   B-Trees

# Assessment Scheme

The assessment will be based on the students’ performance with respect to the course outcomes.

The assessment will be mainly from the following aspects with associated weights.

|  |  |
| --- | --- |
| **Component/ method** | **% weight** |
| Homework Assignments | 40% |
| Midterm Exam | 20% |
| Final Exam | 40% |

1. **Grade Descriptors**

The final grades will be given based on the following criterions.

|  |  |
| --- | --- |
| **Grade** | **Overall course** |
| **A** | Outstanding performance on learning all the data structures, algorithms, and analysis methods covered in this course. Have exceptional ability to choose proper data structures for designing tasks. Be able to analyze the correctness and complexity of an arbitrarily given algorithm and further design an algorithm with desired complexity. Have exceptional ability to use the algorithms learned to solve real world problems. |
| **A-** | Generally outstanding performance on learning most of the data structures, algorithms, and analysis methods covered in this course. Have good ability to choose proper data structures for designing tasks. Be able to analyze the correctness and complexity of an arbitrarily given algorithm. Have good ability to use the algorithms learned to solve real world problems. |
| **B** | Good performance on learning most of the data structures, algorithms, and analysis methods covered in this course. Have good ability to choose proper data structures for designing tasks. Be able to analyze the correctness and complexity of some given algorithms. Have good ability to use most of the algorithms learned to solve real world problems. |
| **C** | Good performance on learning many of the data structures, algorithms, and analysis methods covered in this course. Have the ability to distinguish the various data structures and may choose proper data structures for some designing tasks. Be able to analyze the correctness and complexity of some given algorithms. Have good ability to use some of the algorithms learned to solve real world problems. |
| **D** | Good performance on learning many of the simple data structures, algorithms, and analysis methods covered in this course. Have the ability to distinguish the commonly used elementary data structures and may choose proper data structures for some simple designing tasks. Be able to analyze the correctness and complexity of some simple algorithms. Have good ability to use some of the simple algorithms learned to solve real world problems. |
| **F** | Unsatisfactory performance on learning many of the data structures, algorithms, and analysis methods covered in this course. Fail to distinguish the commonly used elementary data structures. Not be able to analyze the correctness and complexity of some simple algorithms. Have poor ability to use the algorithms learned to solve real world problems. |

1. **Feedback for evaluation**

* Formal Course and Teaching Evaluation by students after finishing all the lectures;
* Informal feedback to the instructor and/or teaching assistant(s) through face-to- face talking, phone calls, emails, the course website, etc.;
* Advisor-advisee program and programme review

# Reading

* 1. **Required**

T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, “Introduction to Algorithms (3rd Edition)”, The MIT Press, 2009.

# Recommended

# Course components

|  |  |
| --- | --- |
| **Activity** | **Hours/week** |
| Lecture | 3 |
| Tutorial | 8 |

1. **Indicative teaching plan (tentative)**

|  |  |
| --- | --- |
| **Week** | **Content/ topic/ activity** |
| 1 | Introduction |
| 2 | Growth of functions |
| 3 | Divide and conquer |
| 4 | Elementary data structures |
| 5 | Heap |
| 6 | Sorting algorithms |
| 7 | Hashing |
| 8 | Trees |
| 9 | Red-black tree and AVL tree  tre |
| 10 | Graph algorithms I |
| 11 | Graph algorithms II |
| 12 | Advanced topics I |
| 13 | Advanced topics II |
| 14 | Review |