

# **DATA STRUCTURES & ALGORITHMS**

## **Course Syllabus**

### **1. General Information**

**Course name:** Data Structures & Algorithms

**Course code:** INT1306\_CLC

**Number of credits:** 3

### **2. Objectives**

#### **Knowledge:**

The aim of this course is to provide learners the important knowledge of algorithms and data structures, including:

- Knowledge of algorithm analysis, design and evaluation.
- Knowledge of traditional algorithmic models.
- Knowledge of abstract data structures.

#### **Skills:**

The aim of this course is to equip learners in solving specific problems of computer science with skills in:

- Proficiency in selecting and implementing appropriate data structures for object representation.
- Apply traditional algorithmic models on the selected data structures to solve specific problems of computer science.
- Enhance analytical thinking, algorithm design, and evaluation skills based on theory of complexity analysis of algorithms.

#### **Attitude:**

Learners are required to attend the classes and complete assignments/projects.

### **3. Abstracts**

This course introduces students to the basic knowledge of algorithms and data structures. For each algorithm, the course provides learners with knowledge of representation, complexity evaluation, implementation, evaluation and applications. For each data structure, the course provides learners with concepts, definitions, representations, operations, implementation and applications. Learners can use the knowledge learned in solving specific problems of computer science.

### **4. Teaching and learning methods**

Lectures: 30h

Exercises: 08h

Projects: 0h

Lab: 06h  
Individual reading: 01h

## 5. Prerequisites:

## 6. Learning outcomes

After studying this course, the learner could:

[LO1]: Understand the general concepts of data structures and algorithms. Enhance analytical thinking and evaluation skills based on theory of complexity analysis of algorithms.

[LO2]: Understand and implement traditional algorithmic models (Sorting and searching techniques, algorithm design techniques) to solve specific problems of computer science.

[LO3]: Proficiency in selecting and implementing appropriate data structures for object representation.

## 7. Assignment criteria

Learning outcomes	Assignment criteria
[LO1]: Understand the general concepts of data structures and algorithms. Enhance analytical thinking and evaluation skills based on theory of complexity analysis of algorithms.	<ul style="list-style-type: none"><li>- General concepts of data structures and algorithms</li><li>- Complexity analysis of algorithms</li></ul>
[LO2]: Understand and implement traditional algorithmic models (Sorting and searching techniques, algorithm design techniques) to solve specific problems of computer science.	<ul style="list-style-type: none"><li>- Sorting and searching techniques</li><li>- Algorithm design techniques</li></ul>
[LO3]: Proficiency in selecting and implementing appropriate data structures for object representation.	<ul style="list-style-type: none"><li>- Abstract data structures</li></ul>

## 8. Outlines

### Chapter 1. Introduction to data structures and algorithms

- 1.1. Introduction to data structures
  - 1.1.1. Data structure definition
  - 1.1.2. Classification of data structures
  - 1.1.3. Application of data structures
- 1.2. Introduction to algorithms
  - 1.2.1. Algorithm definition
  - 1.2.2. Algorithm specification
  - 1.2.3. Algorithm representation
    - 1.2.3.1. Using a natural language

- 1.2.3.2. Using formal languages
- 1.2.3.3. Using a programming language code
- 1.2.4. Most important type of algorithms
- 1.2.5. Complexity analysis of algorithms
  - 1.2.5.1. Asymptotic notations
  - 1.2.5.2. Worst, average, best case analysis of algorithms
  - 1.2.5.3. Lower and upper bound theory
  - 1.2.5.4. Analysis of loops
  - 1.2.5.5. Solving recurrences
  - 1.2.5.6. NP, NP-Completeness introduction
- 1.2.6. Applications of algorithms
- 1.3. Summarization of chapter 1

## **Chapter 2. Sorting and searching techniques**

- 2.1. Sorting techniques
  - 2.1.1. Sorting problem
  - 2.1.2. Selection sort algorithm
    - 2.1.2.1. Idea of Selection sort algorithm
    - 2.1.2.2. Example illustrating operations for the Selection sort algorithm
    - 2.1.2.3. Selection sort representation
    - 2.1.2.4. Evaluating algorithmic complexity of Selection sort algorithm
  - 2.1.3. Insertion sort algorithm
    - 2.1.3.1. Idea of Insertion sort algorithm
    - 2.1.3.2. Example illustrating operations for the Insertion sort algorithm
    - 2.1.3.3. Insertion sort representation
    - 2.1.3.4. Evaluating algorithmic complexity of Insertion sort algorithm
  - 2.1.4. Bubble sort algorithm
    - 2.1.4.1. Idea of Bubble sort algorithm
    - 2.1.4.2. Example illustrating operations for the Bubble sort algorithm
    - 2.1.4.3. Bubble sort representation
    - 2.1.4.4. Evaluating algorithmic complexity of Bubble sort algorithm
  - 2.1.5. Quick sort algorithm
    - 2.1.5.1. Idea of Quick sort algorithm
    - 2.1.5.2. Example illustrating operations for the Quick sort algorithm
    - 2.1.5.3. Quick sort representation
    - 2.1.5.4. Evaluating algorithmic complexity of Quick sort algorithm
  - 2.1.6. Merge sort algorithm
    - 2.1.6.1. Idea of Merge sort algorithm
    - 2.1.6.2. Example illustrating operations for the Merge sort algorithm
    - 2.1.6.3. Merge sort representation
    - 2.1.6.4. Evaluating algorithmic complexity of Merge sort algorithm
  - 2.1.7. Radix sort algorithm
    - 2.1.7.1. Idea of Radix sort algorithm
    - 2.1.7.2. Example illustrating operations for the Radix sort algorithm
    - 2.1.7.3. Radix sort representation
    - 2.1.7.4. Evaluating algorithmic complexity of Radix sort algorithm
  - 2.1.8. Applications of sorting algorithms
- 2.2. Searching techniques
  - 2.2.1. Searching problem
  - 2.2.2. Linear search algorithm

- 2.2.2.1. Idea of Linear search algorithm
- 2.2.2.2. Example illustrating operations for the Linear search algorithm
- 2.2.2.3. Linear search representation
- 2.2.2.4. Evaluating algorithmic complexity of Linear search algorithm
- 2.2.3. Binary search algorithm
  - 2.2.3.1. Idea of Binary search algorithm
  - 2.2.3.2. Example illustrating operations for the Binary search algorithm
  - 2.2.3.3. Binary search representation
  - 2.2.3.4. Evaluating algorithmic complexity of Binary search algorithm
- 2.2.4. Interpolation search algorithm
  - 2.2.4.1. Idea of Interpolation search algorithm
  - 2.2.4.2. Example illustrating operations for the Interpolation search algorithm
  - 2.2.4.3. Interpolation search representation
  - 2.2.4.4. Evaluating algorithmic complexity of Interpolation search algorithm
- 2.2.5. Hash table
- 2.2.6. Applications of searching algorithms
- 2.3. Summarization of chapter 2

### **Chapter 3. Algorithm design techniques**

- 3.1. Next generation algorithm
  - 3.1.1. Algorithmic idea
  - 3.1.2. Example illustrating operations for the next generation algorithm
  - 3.1.3. Algorithmic representation
  - 3.1.4. Evaluate algorithmic complexity
  - 3.1.5. Applications of next generation algorithm
- 3.2. Recursion algorithm
  - 3.2.1. Algorithmic idea
  - 3.2.2. Example illustrating operations for the recursion algorithm
  - 3.2.3. Algorithmic representation
  - 3.2.4. Evaluate algorithmic complexity
  - 3.2.5. Applications of recursion algorithm
- 3.3. Back-track algorithm
  - 3.3.1. Algorithmic idea
  - 3.3.2. Example illustrating operations for the back-track algorithm
  - 3.3.3. Algorithmic representation
  - 3.3.4. Evaluate algorithmic complexity
  - 3.3.5. Applications of back-track algorithm
- 3.4. Branch and bound algorithm
  - 3.4.1. Algorithmic idea
  - 3.4.2. Example illustrating operations for the branch and bound algorithm
  - 3.4.3. Algorithmic representation
  - 3.4.4. Evaluate algorithmic complexity
  - 3.4.5. Applications of branch and bound algorithm
- 3.5. Greedy algorithm
  - 3.5.1. Algorithmic idea
  - 3.5.2. Example illustrating operations for the greedy algorithm
  - 3.5.3. Algorithmic representation
  - 3.5.4. Evaluate algorithmic complexity
  - 3.5.5. Applications of greedy algorithm
- 3.6. Divide and conquer algorithm

- 3.6.1. Algorithmic idea
- 3.6.2. Example illustrating operations for the divide and conquer algorithm
- 3.6.3. Algorithmic representation
- 3.6.4. Evaluate algorithmic complexity
- 3.6.5. Applications of divide and conquer algorithm
- 3.7. Dynamic programming algorithm
  - 3.7.1. Algorithmic idea
  - 3.7.2. Example illustrating operations for the dynamic programming algorithm
  - 3.7.3. Algorithmic representation
  - 3.7.4. Evaluate algorithmic complexity
  - 3.7.5. Applications of dynamic programming algorithm
- 3.8. Summarization of chapter 3

## **Chapter 4. Abstract data structures**

- 4.1. Linked list data structure
  - 4.1.1. Simply linked list
    - 4.1.1.1. Simply linked list definition
    - 4.1.1.2. Simply linked list representation
    - 4.1.1.3. Simply linked list operations
  - 4.1.2. Doubly linked list
    - 4.1.2.1. Doubly linked list definition
    - 4.1.2.2. Doubly linked list representation
    - 4.1.2.3. Doubly linked list operations
  - 4.1.3. Circular linked list
    - 4.1.3.1. Circular linked list definition
    - 4.1.3.2. Circular linked list representation
    - 4.1.3.3. Circular linked list operations
  - 4.1.4. Sorted linked list
    - 4.1.4.1. Sorted linked list definition
    - 4.1.4.2. Sorted linked list representation
    - 4.1.4.3. Sorted linked list operations
  - 4.1.5. Linked list applications
- 4.2. Stack data structure
  - 4.2.1. Stack definition
  - 4.2.2. Stack based on array
    - 4.2.2.1. Stack representation
    - 4.2.2.2. Stack operations
  - 4.2.3. Stack based on linked list
    - 4.2.3.1. Stack representation
    - 4.2.3.2. Stack operations
  - 4.2.4. Stack applications
- 4.3. Queue data structure
  - 4.3.1. Queue
    - 4.3.1.1. Introduction
    - 4.3.1.2. Queue representation
    - 4.3.1.3. Queue operations
    - 4.3.1.4. Queue applications
  - 4.3.2. Circular queue
    - 4.3.2.1. Introduction
    - 4.3.2.2. Circular queue representation

- 4.3.2.3. Circular queue operations
    - 4.3.2.4. Circular queue applications
  - 4.3.3. Priority Queue
    - 4.3.3.1. Introduction
    - 4.3.3.2. Priority queue representation
    - 4.3.3.3. Priority queue operations
    - 4.3.3.4. Priority queue applications
  - 4.3.4. Double ended queue (Deque)
  - 4.3.4.1. Introduction
  - 4.3.4.2. Dequeue representation
  - 4.3.4.3. Dequeue operations
  - 4.3.4.4. Dequeue applications
- 4.4. Binary tree data structure
  - 4.4.1. Introduction
  - 4.4.2. Simple binary tree
    - 4.4.2.1. Simple binary tree representation
    - 4.4.2.2. Simple binary tree operations
    - 4.4.2.3. Simple binary tree applications
  - 4.4.3. Binary search tree (BST)
    - 4.4.3.1. BST representation
    - 4.4.3.2. BST operations
    - 4.4.3.3. BST applications
  - 4.4.4. AVL tree
    - 4.4.4.1. AVL representation
    - 4.4.4.2. AVL operations
    - 4.4.4.3. AVL applications
  - 4.4.5. Heap data structure
    - 4.4.5.1. Heap representation
    - 4.4.5.2. Min heap operations
    - 4.4.5.3. Max heap operations
    - 4.4.5.4. Heap applications
- 4.5. Graph data structure
  - 4.5.1. Introduction
  - 4.5.2. Graph representation
    - 4.5.2.1. Adjacency matrix
    - 4.5.2.2. Edge list
    - 4.5.2.3. Adjacency list
  - 4.5.3. Depth first search algorithm (DFS)
    - 4.5.3.1. DFS representation algorithm
    - 4.5.3.2. DFS complexity
    - 4.5.3.3. DFS applications
  - 4.5.4. Breadth first search algorithm (BFS)
    - 4.5.4.1. BFS representation algorithm
    - 4.5.4.2. BFS complexity
    - 4.5.4.3. BFS applications
  - 4.5.5. Euler cycles algorithm
    - 4.5.5.1. Representation algorithm
    - 4.5.5.2. Complexity algorithm
    - 4.5.5.3. Euler applications
  - 4.5.6. Shortest path algorithms

- 4.5.6.1. Dijkstra algorithm
- 4.5.6.2. Bellman-Ford algorithm
- 4.5.6.3. Floyd-Warshall algorithm
- 4.5.7. Minimum spanning tree
  - 4.5.7.1. Kruskal algorithm
  - 4.5.7.2. Prim algorithm
  - 4.5.7.3. Boruvka algorithm
- 4.5.8. Graph applications
- 4.6. Summarization of chapter 4

## 9. Required Textbooks

- [1]. M. Main, W. Savitch, Data Structures and Other Objects Using C++, 4th Edition, 2010.
- [2]. Donald E. Knuth, The Art of Programming, Vol 1, 2, 3, 4, the 4th Edition, Addison-Wesley Publishing Company, 2005.

## 10. Suggested Textbooks

- [1]. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, 3rd Edition, the MIT Press, 2009.
- [2]. Bradley N. Miller, Problem Solving with Algorithms and Data Structures Using Python, 2nd Edition, 2011

## 11. Schedule

Main contents	Duration	Specific contents
<b>Chapter 1. Introduction to data structures and algorithms</b>	4h lecture	1.1. Introduction to data structures 1.2. Introduction to algorithms 1.3. Summarization of chapter 1
<b>Chapter 2. Sorting and searching techniques</b>	4h lecture 2h exercise 2h lab	2.1. Sorting techniques 2.2. Searching techniques 2.3. Summarization of chapter 2
<b>Chapter 3. Algorithm design techniques</b>	12h lecture 3h exercise 2h lab	3.1. Next generation algorithm 3.2. Recursion algorithm 3.3. Back-track algorithm 3.4. Branch and bound algorithm 3.5. Greedy algorithm 3.6. Divide and conquer algorithm 3.7. Dynamic programming algorithm 3.8. Summarization of chapter 3
<b>Chapter 4. Abstract data structures</b>	10h lecture 3h exercise 2h lab	4.1. Linked list data structure 4.2. Stack data structure 4.3. Queue data structure

		4.4. Binary tree data structure 4.5. Graph data structure 4.6. Summarization of chapter 4
--	--	---

**12. Grading Policy**

Attendance:	10%
Exercises:	20%
Mid-term projects/exams:	20%
Final examination:	50%