INNOPOLIS UNIVERSITY

Data Structures and Algorithms

Adil Khan

Spring 2020

1 Data Structures and Algorithms

- **Course name:** Data Structures and Algorithms
- **Course number:** —

1.1 Course characteristics

1.1.1 Key concepts of the class

- Algorithms
- Algorithm Analysis
- Algorithmic Strategies
- Data Structures

1.1.2 What is the purpose of this course?

This course provides an intensive treatment of a cross-section of the key elements of algorithms and data-structures, with an emphasis on implementing them in modern programming environments, and using them to solve real-world problems. The course will begin with the fundamentals of searching, sorting, lists, stacks, and queues, but will quickly build to cover more advanced topics, including trees, graphs, and algorithmic strategies. It will also cover the analysis of the performance and tractability of algorithms and will build on the concept of Abstract Data Types. A key focus of the course is on effective implementation and good design principles.

1.1.3 Course Objectives Based on Bloom's Taxonomy

- What should a student remember at the end of the course? By the end of the course, the students should be able to recognize and define

- Algorithms
- Abstract Data Types
- Data Structures
- Algorithmic Strategies
- Asymptotic Analysis
- Amortized Analysis

- What should a student be able to understand at the end of the course? By the end of the course, the students should be able to describe and explain (with examples)

- Difference between different abstract data types and data structures
- How to perform asymptotic and amortized analysis
- Difference between various algorithmic strategies
- Different algorithms: such as sorting, searching, etc.
- Different types of tree ADTs, their properties related algorithms
- Graphs, their properties, and related algorithms

- What should a student be able to apply at the end of the course? By the end of the course, the students should be able to apply

- Algorithmic strategies to solve real-life problems
- Asymptotic analysis to Analyze algorithms and software's complexity
- Trees and Graphs (and their theory) to solve complex problems

1.1.4 Course evaluation

Table 1: Course grade breakdown

1**Type**	1**Default points**	**Proposed points**
Labs/seminar classes	20	0
Interim performance assessment	30	30
Exams	50	70

If necessary, please indicate freely your course's features in terms of students' performance assessment: None

1.1.5 Grades range

Table 2: Course grading range

1**Grade**	1**Default range**	**Proposed range**
A. Excellent	90-100	
B. Good	75-89	
C. Satisfactory	60-74	
D. Poor	0-59	

If necessary, please indicate freely your course's grading features: The semester starts with the default range as proposed in the Table **??**, but it may change slightly (usually reduced) depending on how the semester progresses.

1.1.6 Resources and reference material

- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms. The MIT Press 2009.*
- *M. T. Goodrich, R. Tamassia, and M. H. Goldwasser. Data Structures and Algorithms in Java. WILEY 2014.*

1.2 Course Sections

The main sections of the course and approximate hour distribution between them is as follows:

Table 3: Course Sections

Section	Section Title	Teaching Hours
1	Elementary Data Structures, Algorithmic Complexity and Approaches	28
2	Sorting Algorithms and Trees	24
3	Graphs	28

1.2.1 Section 1

Section title: Elementary Data Structures, Algorithmic Complexity and Approaches

Topics covered in this section:
- Algorithms and Their Analysis
- Elementary Data Structures
- Hashing Map and Collision Handling
- Algorithmic Strategies

What forms of evaluation were used to test students' performance in this section?

1**Form**	**Yes/No**
Development of individual parts of software product code	1
Homework and group projects	1
Midterm evaluation	1
Testing (written or computer based)	1
Reports	0
Essays	0
Oral polls	0
Discussions	1

Typical questions for ongoing performance evaluation within this section

1. For a given function give an asymptotic upper bound using "big-Oh" notation
2. Compute the worst case running time of a given algorithm.
3. Insert items into a hashmap given a hash function and a collision handling scheme.
4. Given an algorithm, identify its algorithmic strategy

Typical questions for seminar classes (labs) within this section

1. How to implement various data structures?
2. Implement an algorithm for a given task having a desired worst case time complexity
3. Describe the difference between different types algorithmic strategies
4. Implement a hashmap
5. Solve various practical problems using different algorithmic strategies

Test questions for final assessment in this section

1. For a given function give an asymptotic upper bound using "big-Oh" notation
2. Compute the worst case running time of a given algorithm.
3. Insert items into a hashmap given a hash function and a collision handling scheme.
4. Given an algorithm, identify its algorithmic strategy

1.2.2 Section 2

Section title: Sorting Algorithms and Trees

Topics covered in this section:
- Comparison and Non-comparison Sort
- Binary Search Tree
- Balanced Binary Search Trees
- Tree Traversals
- Priority Queues and Binary Heaps

What forms of evaluation were used to test students' performance in this section?

Typical questions for ongoing performance evaluation within this section

1. Given a BST, answer different questions, such as (a) is the tree an AVL tree? What is the predecessor of a certain node? (b) Will after the removal of a certain node, the resulting tree will be a AVL tree or not?

1Form	Yes/No
Development of individual parts of software product code	1
Homework and group projects	1
Midterm evaluation	1
Testing (written or computer based)	1
Reports	0
Essays	0
Oral polls	0
Discussions	1

2. Similar question as above but for other types of balanced binary search trees, including randomly built binary search trees.
3. Questions related to tree algorithms, such as tree traversals Given a sorting problem defined under some constraints, what sorting algorithm will you use and why?

Typical questions for seminar classes (labs) within this section

1. Implement different types of binary search trees
2. Implement tree traversals
3. Implement different sorting algorithms, such as quicksort, countsort, bucketsort, etc.

Test questions for final assessment in this section

1. Given an unbalanced AVL tree, perform double rotation and show the resulting tree.
2. Given a sequence of elements to be sorted, explain which sorting algorithm you would use to sort the input the fastest and why you chose this sorting algorithm.
3. Implement a sorting algorithm given a problem and specify the big-Oh running time for your algorithm.

1.2.3 Section 3

Section title: Graphs

Topics covered in this section:

- Graph Representations
- Searching in Graphs
- Minimum Spanning Tree
- Shortest Path
- Max-flow Min-cut

What forms of evaluation were used to test students' performance in this section?

1Form	Yes/No
Development of individual parts of software product code	1
Homework and group projects	1
Midterm evaluation	1
Testing (written or computer based)	1
Reports	0
Essays	0
Oral polls	0
Discussions	1

Typical questions for ongoing performance evaluation within this section

1. Given a graph with a certain number of vertices and connected components, compute the largest number of edges that it might have?
2. What is the difference between adjacency list and adjacency matrix representation of a graph?

Typical questions for seminar classes (labs) within this section

1. Implement various graph representations
2. Given a computing problem, devise an algorithm to solve it using Graphs and then implement your algorithm.

Test questions for final assessment in this section

1. Give pseudocode for performing a certain operation in a required time complexity using the adjacency list representation.
2. Give pseudocode for performing a certain operation in a required time complexity using the adjacency list representation.
3. Calculate the maximum flow for a given flow network