Teaching Guidelines for

Algorithms and Data Structures Using Java

PG-DAC September 2021

Duration: 82 hours (30 class room hours + 28 lab hours + 18 mini project hours + 6 revision/practice hours)

Objective: To reinforce knowledge of problem solving techniques, data structure concepts and analysis of different algorithms using Java.

Prerequisites: Knowledge of programming in C/C++ with object oriented concepts

Evaluation: 100 Marks
Weightage: Theory exam – 40%, Lab exam – 40%, Internals & Mini project – 20%

Text Book:

- Data Abstraction and Problem Solving with Java: Walls and Mirrors by Janet Prichard , Frank M. Carrano / Pearson

References:

- Problem Solving: Best Strategies to Decision Making, Critical Thinking and Positive Thinking by Thomas Richards / Kindle Edition
- Data Abstraction and Problem Solving with Java: Walls and Mirrors by Janet Prichard , Frank M. Carrano / Pearson
- Object-oriented Analysis and Design Using UML - An Introduction to Unified Process and Design Patterns by Mahesh P. Matha / PHI
- Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein

---

(Note: Each Session is of 2 hours)

Session 1:
Problem Solving & Computational Thinking
Lecture:

- Define the problem
- Identify the problem
- Introduction to Problem Solving
- Problem solving basics

Lab:

- Faculty needs to assign different problems, mostly real world problems
- Students (team-wise, two students in a team) need to analyze as per the techniques learned
- Based on the above problems students need to select as per the selection criteria learned
- They need to implement the selected solution and need to do the documentations.
- Introducing the mini project ideas

Sessions 2 & 3:
Algorithms & Data Structures

Objective: At the end of the session students should know, what is the importance of data structure in problem solving. How stacks, queues, circular queues work. Their real world applications. How to solve problems using these data structures.

Lecture:

- Introductory Concepts
- Algorithm Constructs
- Complexity analysis of algorithms (Big O notation)
- O O design: Abstract Data Types (ADTs)
- Basic Data Structures
    - Arrays
    - Stacks
    - Queues
    - Circular Queues

Lab:

- Implement stack through array
- Complexity analysis of loops and recursive algorithms.
- Implement queues with inserting element at different location (First, Last)
- Implement circular queue

Sessions 4 & 5:

Linked List Data Structures

Objective: At the end of the session students should know, what are applications of Linked List, different types of link list. Comparison with arrays as when to use linked list and when to use array.

Lecture:

- Linked lists
    - Singly linked lists
    - Doubly linked lists
    - Circular linked lists
    - Node-based storage with arrays

Lab:

- Implement circular queue using linked list
- Implement stack using linked list

Session 6:

Recursion

Objective: At the end of the session students should know what is recursion, type of recursion, local variable in recursion, stack manipulation during recursion, function complexity

Lecture & Lab:

- What is recursion?
- What is the base condition in recursion.
- Direct and indirect recursion.
- Memory is allocated to different function calls in recursion.
- Pro and cons of recursion

- Function complexity during recursion

Sessions 7 & 8:
Trees & Applications
Objective: At the end of the session students should know what is the use of binary trees, how to create binary search trees. Different tree traversals. What are the applications of binary trees? How to calculate search complexity in binary search trees? What are the limitations of binary search trees? What are different options to overcome the binary search tree limitations.

Lecture:

- Introduction to trees
- Trees and terminology
- Tree traversals
- Binary trees
- Complete binary trees / Almost complete binary tree (ACBT)
- Array implementation of ACBT
- Binary search trees
- AVL tree
- Multi-way tree
- Brief introduction and uses cases of B-Tree /B+Tree.

Lab:

- Write a program to implement a binary search tree and the following operations on it:
    - Create()
    - Tree traversals - Breadth First Search, Depth First Search, Inorder(), Preorder(), Postorder()
    - Delete()

Sessions 9 & 10:
Searching & Sorting Algorithms
Objective: At the end of the session students should know what are the different types of sorting and searching algorithms, why all the sorting algorithms are equally important despite different time/space complexity of the algorithms. How the complexity is calculated for each of them. How to pick a sorting algorithm given the nature of the data to be sorted.
Lecture:

- Objectives of Searching
    - The Sequential Search
    - Analysis of Sequential Search
    - The Binary Search
- Analysis of Binary Search
- Introduction to sorting
    - Selection sort
    - Insertion sort
    - Bubble sort
    - Heapsort

- Mergesort
- Quicksort

- Analysis of sorting algorithms

Lab:

- Writing program to search an item through sequential search technique.
- Implement to find an item in a list through binary search
- Implement sorting algorithm for: insertion sort, Quicksort

Session 11:
Hash Functions and Hash Tables
Objective: At the end of the session students should know what is hashing, what is the importance of hashing, comparative complexity of hashing with other search techniques. Problems (collision) with hashing and what are the different solutions of that.
Lecture:

- Hashing & Introduction to hash tables
- Hash Functions
- Different type of hash functions
- Collision Resolution
- Linear Probing
- Quadratic Probing
- Double Hashing
- Inserting and Deleting an element from a hash table

Lab:

- Implement hashing techniques in different programs solved earlier
- Write a program to implement Hash table
- Fibonacci recursive algorithm improvement using hash table

Sessions 12 & 13:
Graphs & Applications
Objective: At the end of the session students should know what is graph? Why is graph the most generic data structure? Different types of graphs. Different representation of graph? Graph traversals (Breadth First Traversal, Depth First Traversal). Different applications which can be solved with graphs, real world and programming problems with graphs.
Lecture:

- Introduction to graph theory
- Graph Terminology
- Different types of Graphs
- Representation of Graphs
  - Adjacency Matrix
  - Adjacency List
  - Graph Traversal Algorithms (Breadth First Search, Depth First Search)
- Shortest Path

- Level Setting : Dijkstra's algorithm
- Level Correcting: All-pairs shortest path, Floyd-Warshall algorithm

- Spanning Trees
  - Minimum spanning tree algorithms,
  - Prim's algorithm
  - Kruskal's Algorithm

Lab:

- Implement a graph using adjacency Matrix and traverse using Depth First Search.
- Implement a graph and do traversal using stack and queue.

Sessions 14 & 15:
Algorithm Designs
Objective: At the end of the session students should know what are different classes of algorithms. What is the nature of each class of algorithms? How to pick an algorithm for a particular problem. What problems fall under each class of algorithms. What are the worst case, average case and the best case for algorithms?
Lecture:

- What are the different class of algorithms
- How to write efficient Algorithm
- Introduction to algorithm design techniques
- Algorithm Design techniques
- Analysis of an Algorithm
  - Asymptotic Analysis
  - Algorithm Analysis
- Analysis of different type of Algorithms
  - Divide and Conquer Algorithm
  - Greedy algorithm
  - Dynamic Programming algorithm
  - Brute force algorithm
  - Backtracking algorithms
  - Branch-and-bound algorithms
  - Stochastic algorithms
- Complexity
  - Complexity Analysis
  - Space complexity of algorithm
  - Time complexity of algorithm
- Case study on Algorithm Design techniques
- Application of Data structures

Lab + Assignment: (2 hours)

- Study on different Algorithms
- Compare different Algorithms previously programmed and do the analysis

- Mini project implementation guidelines

Mini Project (3 days)

Data Structure Implementation and Applications

Objective: Implementation hands-on of data structures, implementation and design of a mini project covering the non-linear data structure and a shortest algorithm.

- Stack/Queue, Link List
- Merge Sort/Heap Sort
- Binary Search Tree
- Graph
- Dijkstra Algorithm