

**MASTER OF COMPUTER APPLICATIONS (MCA)**  
**2-YEAR FULL TIME PROGRAMME**  
**Post Graduate Curriculum Framework**  
**under NEP 2020**  
**(w.e.f 2025)**

**DEPARTMENT OF COMPUTER  
SCIENCE FACULTY OF  
MATHEMATICAL SCIENCES  
UNIVERSITY OF DELHI  
DELHI  
110007**

## **Master of Computer Application (MCA) Program Details**

### **Affiliation**

The proposed programme shall be governed by the Department of Computer Science, Faculty of Mathematical Sciences, University of Delhi, Delhi-110007.

### **Programme Structure and Objectives**

The MCA is a four-semester program spanning two years. It has two structures - **MCA with Coursework, and MCA with Coursework and Internship.**

First three semesters comprise of course - work. In the fourth semester, students are required to do an industry project with at least one supervisor from the department.

#### **The Programme objectives of MCA with Coursework are to**

- Equip the students with the **foundations of** computer science.
- **Enable the students to follow the career path of their choice by choosing** courses from a wide list of specialised courses with progression.
- **Prepare the students to take up a career in the highly competitive IT industry with Applications of Computer Science.**

#### **The Programme objectives of MCA with Coursework and Internship are to**

- Equip the students with the **foundations of** computer science.
- **Enable the students to follow the career path of their choice by choosing** courses from a wide list of specialised courses with progression.
- Provide students **with hands-on experience on live projects**, fostering practical skills and enabling them to prepare technical reports.
- **Prepare the students to take up a career in the highly competitive IT industry with applications of Computer Science.**

**Project:** Each student shall carry out a project in the fourth semester. The Project will be carried out under the supervision of the teacher(s) of the department. When the project is carried out in an external organization (academic institution/ industry), a supervisor may also be appointed from the external organization. The project work will be evaluated jointly by the internal supervisor and an examiner to be appointed by the department in consultation with the internal supervisor.

The project evaluation shall be as follows:

- Mid-semester evaluation: 30% weight
- End-semester evaluation
  - Dissertation: 30%weight
  - Viva-voce: 40%weight



## Semester I

Semester I					
	Number of core courses	3			
Course Code	Course Title	Credits in each core course			
		Theory	Tutorial	Practical	Total
DSC101	<a href="#">Object Oriented Programming</a>	3	0	1	4
DSC102	<a href="#">Data Structures</a>	3	0	1	4
DSC103	<a href="#">Database Systems</a>	3	0	1	4
SEC101	<a href="#">Software Tools and Techniques</a>	0	0	2	2
	Total credits in core course	14			
	Number of DSE/GE courses	2*			
	DSE 1	3	0	1	4
	DSE2/ GE1	3	0	1	4
	Total credits in DSE/GE	8			
	Total credits in Semester I	22			

\*Select two DSEs or one DSE and one GE

List of DSEs for Semester I		
Course Code	Course Title	L-T-P
DSE101	<a href="#">Network Science</a>	3-0-1
DSE102	<a href="#">Graph Theory</a>	3-0-1
DSE103	<a href="#">Computer Organization and Architecture</a>	3-0-1
DSE104	<a href="#">Java Programming</a>	3-0-1
DSE105	<a href="#">Statistical Methods</a>	3-0-1
DSE106	<a href="#">Web Technologies</a>	3-0-1

*Sup 6*

Semester II					
	Number of core courses	3			
Course Code	Course Title	Credits in each core course			
		Theory	Tutorial	Practical	Total
DSC201	<a href="#">Design and Analysis of Algorithms</a>	3	0	1	4
DSC202	<a href="#">Operating Systems</a>	3	0	1	4
DSC203	<a href="#">Artificial Intelligence and Machine Learning</a>	3	0	1	4
SEC201	<a href="#">Scientific Writing and Computational Analysis Tools</a>	0	0	2	2
	Total credits in core course	14			
	Number of DSE/GE courses	2*			
	DSE3	3	0	1	4
	DSE4 / GE2	3	0	1	4
	Total credits in DSE/GE	8			
	Total credits in Semester II	22			

\*Select two DSEs or one DSE and one GE

List of DSEs for Semester II		
Course Code	Course Title	L-T-P
DSE201	<a href="#">Social Networks Analysis</a>	3-0-1
DSE202	<a href="#">Combinatorial Optimization</a>	3-0-1
DSE203	<a href="#">Cyber Security</a>	3-0-1
DSE204	<a href="#">Information Retrieval</a>	3-0-1
DSE205	<a href="#">Digital Image Processing</a>	3-0-1
DSE206	<a href="#">Data Mining</a>	3-0-1

*Nguyen*

Semester III					
	Number of core courses	2			
Course Code	Course Title	Credits in each core course			
		Theory	Tutorial	Practical	Total
DSC301	<a href="#">Data Communication and Computer Networks</a>	3	0	1	4
DSC302	<a href="#">Information Security</a>	3	0	1	4
SEC	TBD	0	0	2	2
	Total credits in core course	10			
	Number of DSE/GE courses	3*			
	DSE 5	3	0	1	4
	DSE 6	3	0	1	4
	DSE 7 / GE 3	3	0	1	4
	Total credits in GE/DSE	12			
	Total credits in Semester III	22			

List of DSE for Semester III		
Course Code	Course Title	L-T-P
TBD		

### Semester IV (Structure 1 Coursework)

Semester IV (Structure 1 Coursework)					
	Number of core courses	2			
Course Code	Course Title	Credits in each core course			
		Theory	Tutorial	Practical	Total
DSC401	<a href="#">Software Engineering</a>	3	0	1	4
DSC402	<a href="#">Deep Learning</a>	3	0	1	4
SEC	TBD	0	0	2	2
	Total credits in core course	10			
	Number of DSE/GE courses	3*			
	DSE 8	3	0	1	4
	DSE 9	3	0	1	4
	DSE 10 / GE 4	3	0	1	4
	Total credits in GE/DSE	12			
	Total credits in Semester IV	22			

*Handwritten signature*

List of DSE for Semester IV (Structure 1 Coursework)		
Course Code	Course Title	L-T-P
TBD		

Semester IV (Structure 2 Coursework+Internship)					
	Number of core courses	2			
Course Code	Course Title	Credits in each core course			
		Theory	Tutorial	Practical	Total
DSC401	Internship	0	0	16	16
	Techniques of Report Writing	0	0	2	2
	Total credits in core course	18			
	Number of DSE/GE courses	1*			
	DSE 11 / GE 5	3	0	1	4
	Total credits in GE/DSE	4			
	Total credits in Semester IV	22			

List of DSE for Semester IV (Structure 2 Coursework+Internship)		
Course Code	Course Title	L-T-P
TBD		

The following outcomes must be achieved by the end of the fourth semester in addition to the outcomes mentioned for Structure I for **Structure 2 (Coursework+Internship)**.

- i) Completion of experimentation/ fieldwork or similar tasks.
- ii) Submission of the project report.

*Ngubé*

## **SEMESTER - I**

### **DSC101: OBJECT-ORIENTED PROGRAMMING [3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSC101	4	3	0	1	NIL

#### **Course Objectives:**

The course aims to develop the student's problem-solving skills. The course also focuses on debugging skills. The student learns to develop modular, well-documented code.

#### **Course Learning Outcomes:**

On completing this course, the student will be able to:

1. select a suitable programming construct and in-built data structures for the given problem.
2. design, develop, document, and debug modular programs.
3. use recursion as a programming paradigm for problem-solving.
4. apply object-oriented paradigm for problem-solving.

#### **Syllabus:**

##### **Unit-I (12 hours)**

Notion of class, object, identifier, keyword, and literal; basic data types: int, float, string, Boolean; basic operators (arithmetic, relational, logical, assignment), standard libraries.

##### **Unit-II (12 hours)**

Modular program development, input and output statements, control statements: branching, looping, exit function, break, continue, and switch-break; arrays and pointers, testing and debugging a program.

##### **Unit-III (6 hours)**

Use of recursion as a programming paradigm for problem-solving.

##### **Unit-IV (15 hours)**

Object Oriented Programming Concepts: Use of classes, inheritance, and Polymorphism. Exception Handling and File Handling: Reading and writing text and structured files, errors and exceptions.

#### **Readings:**

1. R. G. Dromey, *How to Solve it by Computer*, Pearson, 2006.

2. Stanley B. Lippman, and Josée Lajoie, *C++ PRIMER*, Addison-Wesley, 2019.
3. Bjarne Stroustrup, *The C++ Programming Language (4th Edition)* Addison-Wesley, 2013

### **List of Practicals:**

1. Write a program for **BMI Calculator** which accepts height and weight from the user and computes BMI value and category (underweight, normal, overweight, obese), where BMI is defined as the ratio of weight in kilograms by height in metres squared. **(2 hours)**
2. Write a program for **Electricity Bill Calculation** which accepts the number of units consumed and calculates the total bill amount using slab-based rates (e.g., ₹3/unit for first 100 units, ₹5/unit for next 100, ₹8/unit for above 200). It also prints the final bill with slab-wise charges. **(2 hours)**
3. Write a program that takes a paragraph from the user and calculates the number of words, sentences, and characters (excluding spaces), and identifies the longest word. **(2 hours)**
4. Write a program that accepts the marks of five subjects for a student, calculates the average, and assigns a grade based on a standard grading scale (A, if average>85, B if >70, C if >55, D if >40, F otherwise). **(2 hours)**
5. Write a menu-driven modular program for a **Contact Manager** with options to add, search, delete, and view contacts using loops and control statements. Choose appropriate data type to save details. **(2 hours)**
6. Write a program that accepts an array of integers from the user and a target value, then searches for the value using both linear and binary search functions and returns the index if found. **(2 hours)**
7. Write a recursive program for the following: **(4 hours)**
  - To compute the factorial of a given non-negative integer.
  - To find  $n^{\text{th}}$  number in the Fibonacci series
  - To solve the **Tower of Hanoi** problem for n disks, showing the steps to move disks from source to destination.
  - To add multiplies two number without using multiplication operator.
8. Write a program to implement a Student Record System using object-oriented programming concepts. Create a class called Student that includes attributes such as student ID, name, age, department, and marks. Define methods to input student details, store them in an appropriate data structure and display the details of a student based on their student ID. The program should allow the user to add multiple students and retrieve information of any specific student by searching with the ID. Use constructors for initializing student objects and demonstrate encapsulation by keeping attributes private and accessing them through getter methods. **(4 hours)**
9. Write a program to implement a Bank Account Management System using object-oriented programming principles. Begin by creating a base class called BankAccount





that includes attributes such as account number, account holder name, and balance. Define member functions to perform common operations like deposit(amount), withdraw(amount), and check\_balance() for displaying the current balance.

Then, create two subclasses: SavingsAccount and CurrentAccount. In the SavingsAccount subclass, include additional rules such as maintaining a minimum balance (e.g., ₹1000), and deduct a penalty if the balance falls below the minimum. In the CurrentAccount subclass, implement features such as an overdraft limit (e.g., up to ₹5000), allowing withdrawals beyond the current balance up to the limit. Override appropriate methods to reflect the different rules in each account type.

Ensure that the program uses constructors for initialization and proper encapsulation of data members. The program should allow the user to create accounts, perform transactions (deposit/withdraw), and view account details through a menu-driven interface. **(4 hours)**

- 10.** Write a program to model a Library Management System using object-oriented programming concepts. Define a class Book with attributes such as book\_id, title, author, and availability status. Include methods for displaying book details, updating availability, and checking whether a book is currently issued or not.

Create a base class User that includes common attributes such as user\_id, name, and borrowed\_books. Then, define two subclasses Student and Faculty, which inherit from User. Implement method overriding to define different borrowing rules: for example, Student users can borrow up to 3 books for a maximum of 15 days, whereas Faculty users can borrow up to 5 books for a maximum of 30 days. Include a method borrow\_book(book) and return\_book(book) with logic to check borrowing limits and due date rules based on the user type.

Also, define a class BorrowedBook to store borrowing details like book\_id, user\_id, borrow\_date, and due\_date. Ensure the program maintains a list of all books and users and supports operations like: adding new books to the system, registering users, borrowing and returning books, and displaying user or book details.

The program should use constructors for initializing objects, apply encapsulation to restrict direct access to sensitive attributes, and demonstrate inheritance and method overriding effectively. You may also extend it with simple menus or command-line interaction for a smoother user experience. **(4 hours)**

- 11.** Develop a file-based student database system where user input is stored in a text file and options are provided to add, view, and search records, with exception handling for input errors and missing data. **(4 hours)**



## **DSC102: DATA STRUCTURES [3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSC102	4	3	0	1	NIL

### **Course Objectives:**

The course is designed to develop the student's ability to develop algorithms for creating and manipulating basic data structures. The students also learn to compare and contrast different data structures for the problem.

### **Course Learning Outcomes:**

On completing this course, the student will be able to:

1. develop programs using fundamental data structures such as sets, lists, stacks, queues, trees, graphs, as well as advanced structures like balanced trees and heaps.
2. identify and apply the most appropriate data structure for solving specific computational problems efficiently.
3. analyze the time and space complexity of algorithms and data structures to make informed choices in program design.
4. optimize the use of data structures through suitable programming constructs to enhance performance across various scenarios.
5. implement and evaluate real-world applications using a combination of data structures, demonstrating problem-solving and critical thinking skills.

### **Syllabus:**

#### **Unit-I**

**(8 hours)**

Growth of functions, Recurrence relations: Asymptotic notations, solving recurrences using recursion trees.

#### **Unit-II**

**(14 hours)**

Basic data Structures: Primitive Data Types, Abstract Data Types, Linear vs Non-Linear Data Structures, Arrays - Static and Dynamic, 2D Arrays, Linked Lists - Single, Doubly linked, Circular; Stacks and Queues using arrays and linked lists; operations, their analysis and applications.

#### **Unit-III**

**(12 hours)**

Trees: Binary Tree, Tree Traversals, Binary Search Tree, Height Balanced Trees: AVL, Red-Black Trees, B and B+ Trees, Heaps, Priority Queues, operations, their analysis and applications.

#### **Unit-IV**

**(11 hours)**

Sets: Sets, Multisets, Maps, Hash Tables, Dictionaries. Graphs: Representation of Graphs, Searching in Graphs – breadth first search and its applications, depth first search and its applications, Shortest paths, Spanning Trees.



### Readings:

1. Goodrich, M., Tamassia, R. and Mount D, *Data Structures and Algorithms in C++/Java*, 2nd Edition, Wiley, 2016.
2. Elliot B. Koffman, Paul A.T. Wolfgang, *Objects, Abstraction, Data Structures and Design Using C++/Java*, 1st Edition, Wiley Global Education. 2005.
3. T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, 3rd Edition, Prentice-Hall of India Learning Pvt. Ltd. 2010.

### List of Practical's:

1. Write a recursive function to solve a recurrence relation and manually build the recursion tree. Use code and time measurement to validate asymptotic complexity. **(4 hours)**
2. Create static and dynamic arrays, perform insertion, deletion, and resizing operations. Measure time complexity of each operation and explain memory overhead in dynamic arrays. **(6 hours)**
3. Implement all types of linked lists. Perform insertion and deletion at head, tail, and a given position. Demonstrate forward and backward traversal in doubly linked lists. **(8 hours)**
4. Build stack and queue data structures using arrays and linked lists. Perform push, pop, enqueue, and dequeue operations, and demonstrate applications like expression evaluation and palindrome checking. **(6 hours)**
5. Construct a binary tree from given input and perform preorder, inorder, and postorder traversals (both recursively and iteratively). **(6 hours)**
6. Create a BST and perform insert, delete, and search operations. Extend the implementation to AVL tree and demonstrate automatic balancing after operations. **(4 hours) (Optional)**
7. Create a hash table using open addressing and resolve collisions with linear and quadratic probing. Demonstrate performance with a load factor and compare efficiency. **(2 hours) (Optional)**
8. Implement graph using adjacency list/matrix. Perform BFS and DFS with applications like connected components and cycle detection. Use Dijkstra's algorithm for shortest paths and Kruskal's/Prim's for minimum spanning tree. **(4 hours) (Optional)**
9. Implement graph using adjacency list/matrix. Perform BFS and DFS with applications like connected components and cycle detection. Use Dijkstra's algorithm for shortest paths and Kruskal's/Prim's for minimum spanning tree. **(4 hours) (Optional)**

### DSC103: DATABASE SYSTEMS [3-0-1]



Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSC103	4	3	0	1	NIL

### Course Objectives:

This course introduces the fundamentals of database systems, including data modeling, abstraction, and the DBMS architecture. It covers conceptual and relational database design using ER/EER models and integrity constraints. Students will learn to query and manipulate data using relational algebra, calculus, and SQL. Advanced topics like normalization, transaction management, and an introduction to NoSQL and XML databases are also explored.

## **Course Learning Outcomes:**

On completion of this course, the student will be able to:

1. apply basic database concepts, including the structure and operation of the relational data model.
2. apply logical database design principles, including data normalization and E-R/EE-R diagrams to an application,
3. apply the integrity constraints to the application at hand.
4. answer database queries using Structured Query Language (SQL).
5. enumerate the concurrency control issues and identify their resolution strategies.
6. design and implement database projects.

## **Syllabus:**

### **Unit-I**

**(6 Hours)**

Introduction to Database Systems: Data modelling for a database, abstraction and data integration, three-level DBMS architecture

### **Unit-II**

**(10 Hours)**

Database Design and Modelling: Entity-Relationship (ER) Model, Extended Entity-Relationship (EER) Model, Relational Model: Relations, Conversion of ER Diagrams to Relations, Integrity Constraints

### **Unit-III**

**(14 Hours)**

Querying and Manipulating Data: Relational Algebra, Relational Domain & Tuple Calculus, Structured Query Language (SQL): DDL, DML, Views, Embedded SQL

### **Unit-IV**

**(15 Hours)**

**Advanced Database Concepts:** Functional Dependencies, Determining Keys, Normalization, Lossless Join and Dependency-Preserving Decomposition, Transaction Management: ACID Properties, Concurrency Control, Transaction Recovery, Introduction to NoSQL and XML databases

## **Readings:**

1. Silberschatz, H., Korth, S. and Sudarshan, S. *Database System Concepts*. 6th Edition, McGraw Hill, 2014.
2. Elmasri, R. and Navathe, S. B. *Fundamentals of Database Systems*. 7th Edition, Pearson, 2016.
3. Ramakrishnan, R. and Gehrke, J. *Database Management Systems*. 3rd Edition, McGraw Hill, 2014.
4. Lewis, P., Bernstein, A. and Kifer, M. *Databases and Transaction Processing – An Application-Oriented Approach*. Prentice Hall, 2003.

### **List of Practical's:**

1. Define a schema for a real-world application with keys and constraints (e.g., Library/College System). Perform data manipulation using INSERT, UPDATE, DELETE, and SELECT statements in SQL. Use COUNT, SUM, AVG, MIN, MAX, GROUP BY, and HAVING. **(4 Hours)**
2. Model entities and relationships; map EER to tables. Perform multi-table queries using various types of JOIN and subqueries (correlated and nested). Apply and test NOT NULL, CHECK, DEFAULT, PRIMARY KEY, FOREIGN KEY, UNIQUE. **(4 Hours)**
3. Define simple and complex views; demonstrate access abstraction. Identify anomalies, functional dependencies, decompose tables, and implement a normalized schema in SQL. **(2 Hours)**
4. Interfacing SQL with Python programming language. Implement GUI-based environment using TkInter. **(4 Hours)**
5. Use START TRANSACTION, COMMIT, ROLLBACK, and simulate dirty read/lost update problems. Demonstrate locking behavior (e.g., using InnoDB), isolation levels, and test serializability. **(2 Hours)**
6. Choose a domain (e.g., Event Management, Hostel Management, E-commerce), build a full schema, write complex queries, use views, apply constraints, and simulate transactions. **(14 Hours)**

### **SEC101: SOFTWARE TOOLS AND TECHNIQUES [0-0-2]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
SEC101	2	0	0	2	

### **Course Objective:**

The objective of this course is to develop proficiency in the use of software tools required for project development.

### **Course Learning Outcomes:**

Upon successful completion of this course, students will

1. have knowledge of the fundamental shell tools and scripting techniques used in Linux or Unix-like environments.
2. understand the process of booting systems, using live USBs, and understanding their role in system recovery and troubleshooting.
3. apply Markdown for documentation, including its syntax and its applications in writing clear, readable documents.
4. Use Git for version control, including branching, merging, and resolving conflicts in



## **Syllabus:**

### **Unit I**

Foundations of Shell and Command-line Environment: Shell Tools and Scripting, Editors (Vim), Command-line Environment, Debugging and Profiling, Common command-line flags/patterns, Booting + Live USBs, Metaprogramming: Working with Daemons, FUSE, Markdown

### **Unit II**

Advanced Tools and Applications: Version Control (Git), Markdown language, Backup strategies: full, incremental, and differential, Vagrant and VMs, Docker and Containerization, Cloud Providers Overview, manage public and private clouds using OpenStack

## **Readings:**

1. Newham C., *Learning the bash shell: Unix shell programming*. O'Reilly Media, Inc., 2005
2. Shotts W. *The Linux command line: a complete introduction*. No Starch Press. 2019.
3. <https://git-scm.com/book/en/v2>
4. <https://www.techtarget.com/searchdatabackup/feature/Full-incremental-or-differential-How-to-choose-the-correct-backup-type>
5. <https://developer.hashicorp.com/vagrant/docs>
6. <https://docs.docker.com/get-started/>
7. <https://docs.openstack.org/2025.1/>

## **List of Practical's**

1. Shell Tools and Scripting: Write a script to display system information (uptime, disk usage, memory), use of variables, conditionals, and loops, Text Processing with Shell: Use grep, awk, sed, cut, sort, uniq, wc to process log files, count failed SSH login attempts from `/var/log/auth.log`. **(4 hours)**
2. Vim Editor Mastery: Open a file, insert text, delete lines, use undo/redo, use search/replace, split windows, macros, and customise. `vimrc`. Command-line Environment: Bash Configuration and Aliases, Create and modify `.bashrc` or `.Zshrc` with aliases and functions, set environment variables, use history, jobs, fg, bg, kill, &amp;amp;, nohup. **(4 hours)**
3. Shell Script Debugging: Use `bash -x` and `set -x` to trace script execution, insert logging and echo for runtime info, Measure script runtime with time, perf, strace, and top, Common Command-line Flags/Patterns, explore tools like find, xargs, tee, tr, yes, watch, cut, paste, and pipeline, Batch rename files or extract columns from CSV using shell commands. **(4 hours)**
4. Creating and Using a Live USB: Create a bootable USB using dd, balenaEtcher, or Rufus, Boot from USB and explore the Linux live environment, Mount and access partitions, Exploring Boot Process, use dmesg, journalctl, systemctl, and lsblk to inspect boot logs and services. **(4 hours)**
5. Metaprogramming: Working with Daemons, FUSE, convert a script into a background

daemon process, manage it using systemctl or cron, Install FUSE, mount a simple FUSE filesystem (e.g., using hello.py or passthrough\_fuse), Create a basic virtual FS using Python or C with fusepy. **(2 hours)**

6. Writing and Rendering Markdown: create a .md file with headings, lists, code blocks, links, and images. Convert Markdown to HTML/PDF using pandoc or preview with VSCode/Typora. Optional: Use grip or markdown-cli for local web preview. **(2 hours)**
7. Git Basics: Initialize a Git repository, Track files using git add, git commit, and git status, Branching and Merging, Create, switch, and merge branches, Solve a basic merge conflict. **(4 hours)**
8. Remote Repository and Collaboration, push to GitHub or GitLab, Clone, pull, and contribute using forks and pull requests. **(2 hours)**
9. Understand and implement Full and Incremental Backup strategies. Use tar or cp to create a full backup of the directory. **(2 hours)**
10. Schedule incremental backups using rsync or borg. Demonstrate how changes from the last full backup are saved using tools like rdiff-backup. **(2 hours)**
11. Vagrant and Virtual Machines (VMs): write a Vagrantfile to launch a Linux VM using VirtualBox and automate software installation using shell scripts or Ansible within the Vagrant VM. **(2 hours)**
12. Docker and Containerization: Install Docker on your system and verify the installation using the command *docker info*. Use *docker pull* to download official images of nginx and Ubuntu. Start a container and explore the nginx container by accessing it via the browser. Stop the container **(2 hours)**.
13. In continuation of Lab exercise 11, write a Dockerfile to containerise a simple web app (e.g., Python Flask or Node.js) specifying dependencies and configurations. Build the Docker image and run the container. **(2 hours)**
14. Install *Docker Compose*. Write a simple docker-compose.yml file that defines two services: A web app (use Web App created in Lab exercise 12) and A database (e.g., MySQL or PostgreSQL). Demonstrate the Start and stop of both the web and database services. Further, show how Docker can be used to run multiple instances of the web app. **(4 hours)**
15. Explore the three leading cloud platforms: Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). Further, make a document containing a comparison table that contrasts the main features and services provided by these three cloud service providers. Gather details on the following categories: Compute Services, Storage Services, Networking Services, Machine Learning and Analytics, and Security Services. **(4 hours)**
16. Deploying a Virtual Machine on AWS by creating an EC2 Instance on AWS. Select the right AMI (Amazon Machine Image), instance type, and configure security groups. Further, launch and connect to the EC2 instance created. **(2 hours)**
17. Deploying a Virtual Machine on Azure. Further, deploy a VM on Google Cloud Platform (GCP) using Compute Engine. Further explore the Google Cloud Console and basic VM settings (region, machine type, firewall rules). **(4 hours) (Optional)**
18. Networking Setup: Create Virtual Private Networks (VPCs) in AWS, Virtual Network in



Azure, and VPC in GCP. Configure subnets and firewall rules for VMs to communicate securely. **(2 hours) (Optional)**

- 19. Monitoring and Management:** Introduce cloud-native monitoring tools for VM performance (e.g., CPU usage, network traffic). Set up basic alerts and logs for VM health and performance monitoring. **(2 hours)**
- 20. Cloud providers offer a wide variety of instance types optimised for different workloads** (e.g., compute-optimised, memory-optimised, storage-optimised). Discuss AWS EC2 instance families such as T-series and M-series. Similarly, explore Azure's virtual machine types (e.g., A-series, D-series, E-Series. Review GCP machine types (e.g., N1-standard, C2, and M-series). Compare the trade-offs between choosing a general-purpose instance and specialised instances for specific workloads. **(6 hours)**
- 21. Build and Manage Public and Private Clouds using OpenStack:** Install OpenStack using DevStack or Packstack on a VM or cloud host, launch instances, create networks, and assign floating IPs using Horizon (web UI) or CLI, write a script to automate instance creation using openstack CLI or heat templates. **(2 hours)**
- 22. Backup Containers and VM Snapshots:** Demonstrate backing up Docker volumes and VM snapshots. Use Git + Docker + Vagrant + Cloud deployment for a CI/CD mini project. **(4 hours)**





## List of DSEs for Semester I

### DSE101: NETWORK SCIENCE [3-0-1]

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSE101	4	3	0	1	Basic knowledge of probability theory

#### Course Objectives:

The course aims to acquaint the students with the graph theory concepts relevant for network science. The students learn dynamics of and on networks in the context of applications from disciplines like biology, sociology, and economics.

#### Course Learning Outcomes:

At the end of the course, the student will be

1. able to appreciate the ubiquity of the graph data model
2. able to understand the importance of graph-theoretic concepts in social network analysis
3. able to understand the structural features of a network
4. familiar with the theoretical graph generation models
5. identify community structures in networks
6. able to write programs to solve complex network problems

#### Syllabus:

**Unit-I** **(5 hours)**  
Introduction to complex systems and networks, modelling of complex systems, review of graph theory.

**Unit-II** **(10 hours)**  
Network properties: Local and global properties, clustering coefficient, All-pair-shortest path-based properties, centrality measures for directed and undirected networks, degree distribution, clustering coefficient.

**Unit-III** **(15 hours)**  
Graph models: Random graph model, degree distribution, small world network model, power laws and scale-free networks, Barabasi-Albert (preferential attachment) network model, measuring preferential attachment

**Unit IV** **(15 hours)**  
**Community structure in networks:** Communities and community detection in networks, Hierarchical algorithms for community detection, Modularity-based community detection algorithms, label propagation algorithm, multi-level graph.



### **Readings:**

1. Pósfai, Márton, and Albert-László Barabási, Network Science, Cambridge University Press, 2016.
2. Newman, MEJ. Networks: An Introduction, Oxford University Press, 2010.
3. Easley David, and Jon Kleinberg, Networks, crowds, and markets: Reasoning about a highly connected world, Cambridge university press, 2010.
4. Meira Jr, W. A. G. N. E. R., and M. J. Zaki, Data mining and analysis, Fundamental Concepts and Algorithms, 2014.

### **List of Practicals:**

1. Create and visualise a simple network and understand nodes, edges, and basic network structure. Use NetworkX (Python), Gephi. **(4 hours)**
2. Visualise Networks Using User Attributes. Use node color, size, or layout to reflect attributes like age or interest. **(2 hours)**
3. Implement Local and Global Network Properties. Also, measure average path length, diameter, and network density. **(2 hours)**
4. Calculate Node Centrality Measures and compare degree, closeness, betweenness, and eigenvector centrality. Use social graph e.g., Zachary's Karate Club and dolphin datasets. **(8 hours)**
5. Plot and analyze degree distributions in real-world and synthetic networks. **(2 hours)**
6. Implement Erdős–Rényi random graphs and assess their structural properties. **(2hours)**
7. Generate Small World Networks using Watts-Strogatz model and measure clustering and path length. **(2 hours)**
8. Implement Barabási–Albert Model and Generate preferential attachment networks and observe scale-free properties. **(2 hours)**
9. Implement and apply Hierarchical Community Detection algorithms to discover community structures. **(4 hours)**
10. Use the Louvain algorithm to find communities and optimize communities using modularity. **(2 hours)**

### **DSE102: Graph Theory [3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSE102	4	3	0	1	

### **Course Objectives:**

This course will thoroughly introduce the basic concepts of graphs theory, graph properties and formulations of typical graph problems. The student will learn to model which diverse applications in many areas of computing, social and natural sciences.

## **Course Learning Outcomes:**

Upon successful completion of this course, the student will be able to

1. model problems using different types of basic graphs like trees, spanning tree, bipartite and planar graphs.
2. understand and identify special graphs like Euler graphs and Hamiltonian graphs.
3. have increased ability to understand various forms of connectedness in a graph.
4. appreciate different graph-coloring problems, matching problems and their solutions.

## **Syllabus:**

### **Unit-I**

**(9 Hours)**

Fundamental Concepts: Definitions, examples of problems in graph theory, adjacency and incidence matrices, isomorphisms, paths, walks, cycles, components, cut-edges, cut-vertices, bipartite graphs, Eulerian graphs, Hamiltonian graphs, vertex degrees, reconstruction conjecture, extremal problems, degree sequences, directed graphs, de Bruijn cycles, orientations and tournaments, The Chinese postman problem.

### **Unit-II**

**(9 Hours)**

Trees: Trees and forests, characterizations of trees, spanning trees, radius and diameter, enumeration of trees, Cayley's formula, Prüfer code, counting spanning trees, deletion-contraction, the matrix tree theorem, graceful labeling, minimum spanning trees (Kruskal's algorithm), shortest paths (Dijkstra's algorithm).

### **Unit-III**

**(16 Hours)**

Matching and Covers: Matchings, maximal and maximum matchings, M-augmenting paths, Hall's theorem and consequences, Min-max theorems, maximum matchings and vertex covers, independent sets and edge covers, Connectivity, vertex cuts, Edge-connectivity.

Connectivity and Paths: Blocks, k-connected graphs, Menger's theorem, line graphs, dual graphs, network flow problems, flows and source/sink cuts, Ford-Fulkerson algorithm, Max-flow min-cut theorem.

### **Unit-IV**

**(11 Hours)**

Graph Coloring: Vertex colorings, bounds on chromatic numbers, Chromatic numbers of graphs constructed from smaller graphs, chromatic polynomials, properties of the chromatic polynomial, the deletion-contraction recurrence.

Planar Graphs: Planar graphs, Euler's formula, Kuratowski's theorem, five- and four-color theorems.

## **Readings:**

1. West, Douglas B, *Introduction to Graph Theory*, 2<sup>nd</sup> Edition, Pearson, 2017.
2. Chartrand, Gary and Zhang, Ping, *Introduction to Graph Theory*, Tata McGraw Hill, 2017.
3. Gross, Jonathan L., Yellen, Jay and Anderson, Mark, *Graph Theory and Its Applications*, 3<sup>rd</sup> Edition, Taylor & Francis, 2024.



### **References:**

1. Deo, Narsingh, *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall India Learning Private Limited, New edition, 1979.

### **List of practical:**

*You may choose a suitable programming language and all necessary/relevant inputs/problems.*

1. Write a program to check whether the graph is: Bipartite, Eulerian, Hamiltonian or not. **(4 Hours)**
2. Write a program to find the minimum spanning trees of the graph. **(2 Hours)**
3. Write a program to find the shortest paths of the graph. **(2 Hours)**
4. Write a program to implement the Prüfer code of the graph. **(2 Hours)**
5. Write a program to implement the maximum and maximal matching of the graph. **(4 Hours)**
6. Write a program to implement the Ford-Fulkerson algorithm, Max-flow, and Min-cut theorem of the graph. **(6 Hours)**
7. Write a program to implement the Vertex colouring of the graph. **(2 Hours)**
8. Write a program to find the chromatic numbers of the graph. **(2 Hours)**
9. Write a program to check whether the graph is planar or not. **(2 Hours)**
10. Write a program to solve the Chinese postman problem. **(4 Hours)**

## **DSE103: COMPUTER ORGANIZATION AND ARCHITECTURE** **[3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSC103	4	3	0	1	

### **Course Objectives:**

The course is designed to introduce basic building blocks of digital electronics, design and architecture of computer systems. This course aims to develop the skill to architect a digital computer system using a simulator.

### **Course Learning Outcomes:**

On completing this course, the student will be able to:

1. describe the basic organization of computer hardware.



2. represent and manipulate data – number systems, conversion between different number systems, perform binary arithmetic.
3. design simple combinational and sequential logic circuits - flip-flops, counters, shift registers, adders, subtractor, multiplexer, demultiplexer, and Arithmetic/Logic unit.
4. design a CPU simple computer / microprocessor: instruction format, instruction set, addressing modes, bus structure, input/output architecture, memory unit, Arithmetic/Logic and control unit, data, instruction and address flow.

## **Syllabus:**

### **Unit-I (10 hours)**

Basic Building Blocks: Boolean logic and Boolean algebra, tri-state logic; flip-flops, counters, shift registers, adders, subtractors, encoders, decoders, multiplexers, demultiplexers.

### **Unit-II (20 hours)**

Processor Design: CPU organisation, register organisation, stack organisation, microprogrammed control unit, RISC architecture; microprocessor architecture, modern computing architectures. Bus and memory transfers, arithmetic, logic shift micro-operations; basic computer organization: common bus system, instruction formats, instruction cycle, interrupt cycle, input/output configuration.

### **Unit-III (6 hours)**

Memory Unit: Primary memory, secondary memory, associative memory, sequential access, direct access storage devices.

### **Unit-IV (9 hours)**

Input-Output Architecture: Input/output devices; data transfer schemes - programmed I/O and DMA transfer; data transfer schemes for microprocessors.

## **Readings:**

1. M. Morris Mano, *Computer System Architecture*, Revised 3rd Edition, Pearson, 2018.
2. W. Stallings, *Computer Organization and Architecture: Designing for Performance*, 9th Edition, Pearson Education, 2012.
3. A.S. Tanenbaum, *Structured Computer Organization*, 6th Edition, Prentice-Hall of India, 2012.
2. J.P. Hayes, *Computer System Architecture & Organization*, 3rd Edition, McGraw-Hill Education, 2017.

## **List of Practical's:**

1. Design the circuits for Half Adder, Half Subtractor, Full Adder, and Full Subtractor using basic logic gates. **(4 hours)**
2. Design and simulate a multiplexer and demultiplexer (16:1 MUX, 1:16 DEMUX). **(2 hours)**
3. Implement encoders and decoders (e.g., 3-to-8 decoder, 8-to-3 encoder) using logic gates. **(2 hours)**
4. Design asynchronous and synchronous counters using flip-flops. **(2 hours)**

5. Design a common Bus system for four registers of size 4-bits using multiplexer and tri-state buffers. **(4 hours)**
6. Design and simulate circuits for the following Arithmetic Microoperations: **(6 hours)**
  - Binary Adder
  - Binary Adder-Subtractor
  - Binary Incrementer
  - Binary decrementer
  - 4-bit Arithmetic Circuit for all the Arithmetic Microoperations
7. Design a logic circuit that performs basic logical operations like AND, OR, NOT, and XOR. **(2 hours)**
8. Design the circuits for arithmetic and logical shift microoperations. **(4 hours)**
9. Design and simulate a circuit for complete general register set architecture with common ALU. **(4 hours)**

### **DSE104: JAVA PROGRAMMING [3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSE104	4	3	0	1	

#### **Course Objectives:**

The objective of this course is to develop skills in object-oriented concepts, writing programs using exception handling techniques, multithreading and user interface design of Java programming.

#### **Course Learning Outcomes:**

Upon successful completion of this course, the student will be able to

1. understand the object-oriented concepts, like Classes, Objects, Inheritance, Polymorphism and exceptions.
2. design, implement, document, test, and debug a Java application consisting of multiple classes and multithreading with input/output through files.
3. create Java applications with a graphical user interface (GUI).

#### **Syllabus:**

##### **Unit-I**

**(7 Hours)**

Introductory Concepts: program, identifiers, variables, constants, primitive data types, expressions, control statements, structured data types, arrays, functions.

**Unit-II****(10 Hours)**

Object Oriented Concepts: Abstraction, encapsulation, objects, classes, methods, constructors, inheritance, polymorphism, static and dynamic binding, overloading, Abstract classes, Interfaces and Packages.

**Unit-III****(9 Hours)**

Exception handling: Throw and Exception, Throw, try and catch Blocks, Multiple Catch Blocks, Finally Clause, Throwable Class, Types of Exceptions, java.lang Exceptions, Built-In Exceptions.

**Unit-IV****(19 Hours)**

File Handling: Byte Stream, Character Stream, File I/O Basics, File Operations, Serialization.

GUI Design: GUI based I/O, Input and Message Dialog boxes, Swing components, Displaying text and images in window.

**Readings:**

1. James Gosling, Bill Joy, Guy L. Steele Jr, Gilad Bracha, Alex Buckley, *The Java Language Specification*, Java SE 7 Edition, Addison-Wesley, 2013.
2. Cay S. Horstmann, *Core Java - Vol. I – Fundamentals*, 10<sup>th</sup> Edition, Pearson, 2017.
3. Deitel & Deitel, *Java-How to Program*, 9<sup>th</sup> Edition, Pearson Education, 2012.

**References:**

1. Richard Johnson, *An Introduction to Java Programming and Object-Oriented Application Development*, Thomson Learning, 2006.
2. Herbert Schildt, *Java: The Complete Reference*, 10<sup>th</sup> Edition, McGraw-Hill Education, 2018.

**List of practical:**

*You may choose all necessary/relevant inputs/ problems.*

1. Write a java program to design a class account using inheritance and static that shows all the functions (withdrawal and deposit) of the bank. **(2 Hours)**
2. Write a java program for method overloading and constructor overloading. **(2 Hours)**
3. Write a java program for dynamic method dispatch and final as class, variables, and methods. **(4 Hours)**
4. Write a java program to represent abstract class with example. **(2 Hours)**
5. Write a java program to implement inner classes and interface using extends keyword. **(2 Hours)**
6. Write a java program to create a user-defined package. **(2 Hours)**
7. Write a java program to implement exception handling techniques. **(4 Hours)**
8. Write a java program for the producer and consumer problem using threads. **(2 Hours)**



9. Write a java program to display the file class properties and load phone numbers with names from a text file. **(2 Hours)**
10. Write a java program to compute the factorial value using an Applet. **(2 Hours)**
11. Write a java program for handling Mouse events and Key events. **(4 Hours)**
12. Write a java program that works as a simple calculator. Use a Grid Layout to arrange Buttons for digits and for the +, -, \*, % operations. Add a text field to display the result. **(2 Hours)**

### **DSE105: Statistical Methods [3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSE105	4	3	0	1	Nil

#### **Course Objectives:**

This course aims to equip students with the skills necessary to apply statistical methods for various applications.

#### **Course Learning Outcomes:**

On completing this course, the student will be able to:

1. apply descriptive statistical techniques to summarize and interpret data
2. apply inferential statistical methods, including hypothesis testing and confidence interval estimation.
3. perform and interpret simple and multiple linear regression analysis
4. apply principles of experimental design in the context of a problem

#### **Syllabus:**

##### **Unit-I (10 hours)**

Introduction: Descriptive statistics: measures of central tendency and variability, representation of data: stem and leaf diagram, histogram, boxplot, and ogive; bar diagram and its variations, Pie charts.

##### **Unit-II (10 hours)**

Probability distributions: discrete and continuous, joint and conditional probability; theory of attributes: coefficient of association and coefficient of colligation.

##### **Unit-III (15 hours)**

Statistical Inference: Parameter and statistic; sampling distributions, confidence intervals and margin of error, hypothesis testing; non-parametric inference: non-parametric tests: Mann-Whitney U test, Kruskal-Wallis test, Spearman's rank correlation coefficient.

##### **Unit-IV (10 hours)**

Regression and Classification: Correlation: measure and significance, simple linear regression,



multiple linear regression, one-way classification, analysis of variance, two-way classification, analysis of covariance, curvilinear regression, factorial experiments.

### **Readings:**

1. Robert S. Witte and John S. Witte, *Statistics*, John Wiley & Sons Inc; 11th edition, 2021
2. Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, *An Introduction to Statistical Learning*, Springer, 2023.
3. G. W. Snedecor, W. G. Cochran, *Statistical Methods*, Iowa State University Press, 1973
4. John A. Rice, *Mathematical Statistics and Data Analysis*, Cengage, 2013

### **List of Practical's:**

Write the Python program for the following-

Refer research paper: Kumar, M. S., & Sahu, P. P. (2013). Employment Growth, Education and Skills in India: Emerging Perspectives. *Indian Journal of Labour Economics*, 56(1), 95-122.

1. Select any data table and a variable from the selected data table. Compute central tendencies: arithmetic mean, geometric mean, median and standard deviation for the selected variable. **(4 hours)**
2. Refer table 9 for data of states for the year 1983 of primary sector for the persons who are educated up to middle level. Plot a box plot. Do same exercise for the year 1993-94 and comment on the shape of both box plots **(4 hours)**
3. Refer total population of various years in table 2: Plot an Ogive and Pie-chart **(4 hours)**
4. Refer table 4: is there any significant impact of gender on the education level? Test it on 95% confidence using Chi-square test **(4 hours)**
5. Refer table 5: From year 1983 to 2009-10, check the impact of time on non-literacy at 5% level of significance using Mann-Whitney U test **(4 hours)**
6. Refer table 5: From year 1983 to 2009-10, check the impact of time on non-literacy at 5% level of significance using Kruskal-Wallis test **(4 hours)**
7. Refer the data of 1994 and 2000 of table 2. Using Spearman's rank correlation, check whether data of 1994 and 2000 is strongly correlated? **(2 hours)**
8. Select a table and variables of your choice. Test the impact of treatment factor at 5% level of significance using one-way analysis of variance **(4 hours)**



## **DSE106: WEB TECHNOLOGIES [3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE106	4	3	0	1	

### **Course Objectives:**

This course provides a comprehensive understanding of web technologies and their applications, covering networking concepts, front-end development with modern frameworks, client-side programming with JavaScript, and server-side programming using Django. Additionally, Students will learn how to develop, test, and deploy dynamic web applications efficiently.

### **Course Learning Outcomes:**

On completion of this course, students will be able to

1. understand web technologies concepts and networking protocols significance
2. build responsive web interfaces using HTML5, CSS, and frameworks like Bootstrap and React.js with a mobile-first approach.
3. apply various client-side web technologies for front-end development.
4. create server-side applications with Django, handling routing, authentication, databases, and REST APIs.
5. apply CI/CD practices to automate the development, testing, and deployment of web applications.

### **Syllabus:**

#### **Unit-I**

**(11 hours)**

Introduction to Networking and Web Technologies: Internet and its Evolution, World Wide Web, Web 2.0, Web 3.0, Network Communication Protocols: HTTP/HTTPS, SMTP, IMAP, POP, FTP, Client-Server Architecture, Web Applications Architecture, Application and Web Servers, Web Clients, Cloud Technologies: AWS, Azure, Google Cloud, API-First Architecture, RESTful APIs, WebSockets, Real-time Communication, Microservices Architecture, DevOps Concepts: Continuous Integration/Continuous Deployment (CI/CD) Pipelines

#### **Unit-II**

**(10 hours)**

**Front-end Development:** Introduction to HTML5, HTML elements, HTML tags, lists, tables, frames, forms, Basics of XHTML, CSS Style Sheets, Responsive Design and Mobile-First Development, CSS Frameworks: Bootstrap, Materialize, Modern JavaScript Frameworks: Introduction to React.js, Angular, or Vue.js, Component-Based Architecture, CSS Preprocessors: SASS, LESS,, Front-end Testing: Jest, Mocha, Web Performance Optimization: Lazy Loading, Code Splitting



### **Unit-III (12 hours)**

**Client-Side Programming:** Modern JavaScript (ES6+): Arrow Functions, Async/Await, Destructuring, Template Literals, Variables and Data Types, Literals, Functions, Objects, Arrays, Built-in Objects and Event Handling, Modifying Element Style, Document Trees, Advanced JavaScript Features: Closures, Promises, Async/Await, TypeScript Introduction, Single-Page Application (SPA) Development with React.js, State Management with Redux (for React) or Vuex (for Vue.js), Integrating APIs in React or Vue for dynamic content.

### **Unit-IV (12 hours)**

**Server-Side Programming:** Introduction to Web Frameworks: Overview of Django, Setting up a Django Project, Routing and Views in Django, Templates in Django, Working with Forms in Django, Handling User Authentication and Authorization in Django, Authentication and Authorization: JWT (JSON Web Tokens), OAuth2.0, Introduction to Databases: Using SQLite, PostgreSQL, or MySQL with Django, CRUD Operations with Django ORM, REST APIs with Django Rest Framework, Session and Cookie Management in Django, Deployment of Django Applications

#### **Resources:**

1. Resources for developers by developers (Mozilla Firefox) <https://developer.mozilla.org/en-US/>
2. The OdinProject, <https://www.theodinproject.com/>
3. React: The library for web and native user interfaces, <https://react.dev/>
4. Django: the web framework for perfectionists with deadlines, <https://www.djangoproject.com/>
5. Scrimba: Helping developers learn faster by merging video and coding into one, <https://scrimba.com/>

#### **List of Practical's:**

The student will choose a project (website to be developed) of their choice and will do the following.

1. Initialise the project repository in GitHub and define the project scope in the README. **(2 hours)**
2. Create wireframes using Figma or some other tool to design a basic UI structure. **(4 hours)**
3. Develop 2-3 static web pages using HTML and CSS. **(2 hours)**
4. Implement basic client-side functionality such as whether the email entered format is correct or not, strength of the password etc **(4 hours)**
5. Set up a React.js project and create simple components. **(4 hours)**
6. Develop a few reusable components and handle state management depending on the project requirement **(4 hours)**
7. Set up a backend framework using Django **(4 hours)**
8. Create user login and registration features using Django **(4 hours)**



9. Set up and configure a relational database. Further, connect the React.js front-end with backend APIs. Integrate external APIs into the application, if any. **(4 hours)**
10. Deploy the project to a cloud platform of your choice. (additional)



## **SEMESTER – II**

### **DSC201: DESIGN AND ANALYSIS OF ALGORITHMS [3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSC201	4	3	0	1	

#### **Course Objectives:**

The course introduces the techniques for algorithm design and analysis of the asymptotic performance of these algorithms. This course aims to achieve competence in designing efficient algorithms using different data structures for real-world problems.

#### **Course Learning Outcomes:**

On completing this course, the student will be able to:

1. describe various algorithm design techniques, including iteration, divide and conquer, dynamic programming, and greedy approach algorithms.
2. analyze the strengths and weaknesses of each technique.
3. identify and apply technique(s) suitable for simple applications.
4. demonstrate the correctness of algorithms and analyze their time complexity theoretically and practically.
5. model simple problems as graphs and solve them using Graph Algorithms.

#### **Syllabus:**

##### **Unit-I (4 hours)**

Computational Complexity: Review of Growth of Functions, asymptotic notation, Master's Theorem.

##### **Unit-II (20 hours)**

Searching and Sorting Techniques - Linear search, Binary search, insertion sort – time complexity. Binary Search, Merge sort and Quick sort – time complexity. Lower bounding techniques: Decision Trees. Linear Sorting: Count Sort, Radix Sort, Bucket Sort. Randomized algorithms: Introduction to random numbers, randomized Qsort, randomly built BST

##### **Unit-III (12 hours)**

Optimization Techniques: Greedy Algorithms: Interval Scheduling, Minimum Spanning Trees – Prim's algorithm, Kruskal Algorithm, Shortest Path Problem – Dijkstra's algorithm. Dynamic Programming: Weighted Interval Scheduling, Matrix chain multiplication, Knapsack problem, Longest common subsequence.



## Unit-IV

(9 hours)

String Processing: Brute-force method, KMP algorithm. Introduction to class NP, NP-Hard and NP-Complete.

### Readings:

1. Kleinberg, J. and Tardos, E. *Algorithm Design*. 1<sup>st</sup> Edition, Pearson Education India, 2013.
2. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, 4<sup>th</sup> Edition, The MIT Press, 2022.
3. Sara Baase, Allen Van Gelder, *Computer Algorithm – Introduction to Design and Analysis*, 3rd edition, 2009, Pearson Education.

### Practicals

#### General Instructions:

1. All programs to be developed/uploaded immediately on GitHub.
2. Results obtained in a practical may be required in subsequent practical's. So, save them and keep them handy.
3. A program must be completely automated with no manual intervention - from taking the input values like n, m to generating the records of the results. Generating data sets can be internal to the program. It will be better to save it in a file also, in case it is required to be reused.
4. Time should not include the time for reading the input and writing the output.
5. If GenAI tools are used to create a part of the code, prior permission must be sought in writing, and it must be recorded as documentation at the top of the program.
6. Libraries for basic data structures like Stacks, Queues, Binary Search trees, Hashmaps can be used. For any other specialised part, prior permission must be sought.

### List of Practicals

1. Implement Insert Sort and run it on a synthetic data of (Name : string type, age : float type) for n = 10, 20, ...100. For each n, randomly generate at least 10 data sets. Report the average number of comparisons for each n. Plot a graph and obtain a best-fit curve. **(6 hours)**
  - a. write a function to sort on age
  - b. write a function to sort on names
  - c. sort on age and then on names so that the ordering on age is maintained. i.e. For (Reeta, 18.3), (Reeta 17.8), (Geet, 18.3), the output should be (Geet, 18.3), (Reeta 17.8), (Reeta, 18.3). I.e. If there are two people with the name Reeta their records should appear so that they are sorted on their ages.
2. Program based on real life Application of Sorting: for example weather data or sort the edges of a graph on its weights - used in connectivity planning. **(2 hours)**
3. Repeat 1 and 2 with Merge Sort and Compare with the results of Insertion Sort. **(4 hours)**
4. Repeat 1 and 2 with Quick Sort and Compare with the results of Insertion Sort and Merge Sort. **(4 hours)**
5. Repeat 1 and 2 with Randomized Quick Sort and Compare with the results of Quick Sort. **(2 hours)**



6. Implement radix Sort and run it on a synthetic data of (Name : string consisting of 10 characters, age in years: integers in the range 10 to 20 ) for n = 10, 20, ...100. For each n, randomly generate at least 10 data sets. Report the average number of comparisons for each n. Plot a graph and obtain a best-fit curve. **(4 hours)**
  - a. write a function to sort on age
  - b. write a function to sort on names
  - c. sort on age and then on names so that the ordering on age is maintained.
  - d. Compare with the results of Insertion Sort, Merge Sort, Quicksort and Randomized Quick Sort.
7. Programs on Graphs, as required. **(2 hours)**
8. Problem Solving based on Sorting, Searching, Graph problems, Greedy and Dynamic Programming. **(6 hours)**

### **DSC202: OPERATING SYSTEMS [3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSC202	4	3	0	1	

#### **Course Objectives:**

This course aims to provide a functional perspective of the operating systems and develop skills to experiment with different components, tasks and services, including job scheduling, memory management, device management, process management and file management.

#### **Course Learning Outcomes:**

On completing this course, the student will be able to:

1. describe the basic functions of an Operating System.
2. distinguish between different types of operating systems to use each of them most efficiently in the respective application areas.
3. describe different techniques for managing computer resources like CPU, memory, file and devices.
4. implement algorithms for managing computer resources.
5. experiment with different components, tasks and services of the operating system.

#### **Syllabus:**

##### **Unit-I (5 hours)**

Introduction: Defining an Operating System, Design Goals, Evolutionary history of operating systems; Concept of User, job and Resources; Batch processing, Multi-programming, Time sharing; Structure and Functions of Operating System.



## Unit-II

(18 hours)

Process Management and Synchronization: Process states, State Transitions, Process Control Structure, Context Switching, Process Scheduling, Threads. Process Interaction, Shared Data and Critical Section, Mutual Exclusion, Busy form of waiting, Lock and unlock primitives, Synchronization, Classical Problems of Synchronization, Semaphores, Monitors, Conditional Critical Regions, System Deadlock, Wait for Graph, Deadlock Handling Techniques: Prevention, Avoidance, Detection and Recovery.

## Unit III

(15 hours)

Memory Management: Address Binding, Dynamic Loading and Linking Concepts, Logical and Physical Addresses, Contiguous Allocation, Fragmentation, Paging, Segmentation, Combined Systems, Virtual Memory, Demand Paging, Page fault, Page replacement algorithms, Global Vs Local Allocation, Thrashing, Working Set Model, Pre-Paging.

## Unit IV

(7 hours)

File and Secondary Storage Management: File Attributes, File Types, File Access Methods, Directory Structure, File System Organization and Mounting, Allocation Methods, Free Space management; Disk Structure, Logical and Physical View, Disk Head Scheduling, Formatting, Swap Management.

### Readings:

1. Silberschatz, Galvin, and Gagne, *Operating Systems concepts*, Wiley, 2009.
2. Gary Nutt, Nabendu Chaki, Sarmistha Neogy, *Operating Systems: A Modern Approach* (3rd ed.), Addison Wesley, 2009.
3. Tanenbaum, Andrew S., and Herbert Bos. *Modern operating systems*. Pearson Education, 2015.
4. Stallings, William. *Operating systems: internals and design principles*. Prentice Hall Press, 2011.
2. D.M. Dhamdhare, *Operating Systems: A Concept Based Approach* (2nd ed.), Tata McGrawHill, 2007.

### List of Practical's:

1. Write a C program that demonstrates process creation using fork (), executes a new program using exec(), and waits for child process termination using wait() system call. **(6 hours)**
2. Implement multiple threading using the thread library. Ensure graceful thread termination using join() to prevent abnormal exit. **(4 hours)**
3. Design a multithreaded program using pthreads and semaphores to simulate and demonstrate both mutual exclusion and deadlock scenarios. **(4 hours)**
4. Implement a race condition and then resolves it using: **(8 hours)**
  - pthread\_mutex\_lock() and pthread\_mutex\_trylock()
  - POSIX semaphores
  - Condition variables
  - Reader-writer locks (pthread\_rwlock).
5. Develop a solution to the classic producer-consumer problem using a bounded buffer. Use mutexes for mutual exclusion and semaphores to synchronise producer and consumer threads.





(4 hours)

6. Write a C program that demonstrates interprocess communication between a parent and child process using unnamed pipes (pipe () system call). (2 hours)

7. Write a program that demonstrates file reading and writing operations. (2 hours)

## **DSC203: ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**[3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSC203	4	3	0	1	Basics of probability and statistics

### **Course Objectives:**

Beginning with a comprehensive overview of the AI techniques, the course introduces the supervised and unsupervised ML techniques, along with their applications in solving real-world problems. The course also covers evaluation and validation methods for ML models.

### **Course Learning Outcomes:**

On completion of this course, the student will be able to:

1. Understand foundational AI concepts, intelligent agents, logic systems, and classical search techniques.
2. Explore key machine learning models including regression, classification, and clustering algorithms.
3. Analyze model performance through bias-variance tradeoff, feature selection, and dimensionality reduction.
4. Learn artificial neural network architectures, training procedures, and optimization techniques.
5. Gain insights into advanced neural models including RNNs, SOMs, and Generative AI techniques.

### **Syllabus:**

#### **Unit-I**

**(15 hours)**

Introduction, Evolution, Application of AI, AI Problems, Problem Formulation, Intelligent Agents, Reasoning and Logic, Propositional Logic, First-order Logic, Inference in First-order Logic, Forward and Backward Chaining, Expert System, Solving Problems by Searching, Search - Uninformed Search Techniques, Heuristic Search Techniques, Advanced Search Techniques.

#### **Unit-II**

**(10 hours)**

Machine Learning: Introduction, Supervised vs. Unsupervised learning, Regression: Linear regression with one variable, Linear regression with multiple variables, Logistic Regression,



Polynomial regression, Classification: Decision trees, Naive Bayes classifier, Support Vector Machine, Kernel functions, Clustering: K-mean, hierarchical.

### **Unit-III**

**(10 hours)**

Bias-variance tradeoff, Feature scaling, feature selection methods - Lasso, Ridge, Curse of dimensionality, dimensionality reduction methods - Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA).

### **Unit IV**

**(10 hours)**

Introduction, Activation Artificial Neural Networks: Function, Optimization algorithm, Gradient descent, Backpropagation Algorithms, Training Procedures Networks- Perceptron's, Multilayer Perceptron's, Hopfield neural network, recurrent neural network, Self-organising maps, Generative AI.

### **Readings:**

1. Alpaydin, Ethem. *Introduction to machine learning*. MIT Press, 2020.
2. Russell, Stuart J., and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.
3. Bishop, Christopher M., and Nasser M. Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. Springer, 2006.
4. Shalev-Shwartz, Shai, and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.
5. Michalski, Ryszard Stanislaw, Jaime Guillermo Carbonell, and Tom M. Mitchell, eds. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

### **List of Practical's':**

1. Write a program to implement the working of Breadth-First Search (BFS) and Depth-First Search (DFS). **(2 hours)**
2. Build a simple Linear Regression model with one and multiple variables using scikit-learn. **(2 hours)**
3. Implement Logistic Regression for binary classification (e.g., spam detection). **(2 hours)**
4. Train and visualise Decision Tree and Naive Bayes classifiers on a sample dataset. **(4 hours)**
5. Build and evaluate a Support Vector Machine (SVM) with the kernel trick on a classification problem. **(4 hours)**
6. Implement K-Means and Hierarchical clustering and visualise clustering results. **(4 hours)**
7. Apply feature scaling and evaluate its impact on model performance. Further, use Lasso and Ridge regression to perform feature selection and regularisation, and compare model performance with and without scaling and regularisation. **(4 hours)**
8. Perform Dimensionality Reduction using PCA and LDA and visualise transformed features. **(4 hours)**
9. Build a Multilayer Perceptron from scratch or using TensorFlow/Keras; train it on a classification task (e.g., MNIST). **(4 hours)**



## **SEC201: SCIENTIFIC WRITING AND COMPUTATIONAL ANALYSIS TOOLS [0-0-2]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
SEC201	2	0	0	2	NIL

### **Course Objective:**

The course aims to equip students with the skills to create professional documents using LaTeX and master MATLAB for data analysis, signal processing, image processing, and system modelling. Students will also learn to work with MATLAB functions, scripts, and Simulink for simulation-based tasks.

### **Course Learning Outcomes:**

Upon successful completion of this course, students will be able to

1. Create a basic LaTeX document, format text with different styles, and apply fundamental document structures such as sections, paragraphs, and formatting options.
2. create and manipulate matrices and arrays in MATLAB, understanding their importance in numerical computation and data analysis.
3. apply optimisation techniques and perform curve fitting tasks in MATLAB, including fitting data to a model and minimising error.

### **Syllabus:**

#### **Unit-I**

LaTeX : Basic Document Setup, and Formatting Text, Lists and Tables, Multi-Column Layouts, Mathematical Equations, Figures and Graphics, Customizing Fonts and Colors, References, Citations, and Headers/Footers/Page Numbering, Creating a Resume or CV.

#### **Unit-II**

MATLAB: Basic MATLAB Operations, Matrices and Arrays, Plotting and Visualization, Loops and Conditional Statements, Functions and Scripts, File Handling, Signal Processing, Image Processing, Optimization and Curve Fitting, Simulink Basics

### **Readings:**

1. <https://guides.nyu.edu/LaTeX/sample-document>
2. Brian R. Hunt, Ronald L. Lipsman, Jonathan M. Rosenberg. *A Guide to MATLAB: For Beginners and Experienced Users*, Cambridge University Press, 2017.

### **List of Practical's :**

1. Basic Document Setup and Formatting Text: Create a new document with a specific page size, orientation, and margins, Format a paragraph using different font styles (bold, italic, underline) and sizes, Apply alignment options (left, right, center, justified) to different text blocks, Use styles to format headings and body text consistently. (4 hours)



2. Lists and Tables, Multi-Column Layouts: Create a bulleted and numbered list for an agenda or shopping list, design a table to represent student grades or inventory details, merge and split cells in the table, and apply borders and shading. Create a newsletter-style document using two or three-column layouts. **(4 hours)**
3. Mathematical Equations: Insert and format basic math expressions, such as fractions, superscripts, and subscripts. Write complex equations, like the quadratic formula, matrices, and integrals, using equation tools or LaTeX syntax (if applicable), and align multiple equations properly using equation editors or tab alignment. **(4 hours)**
4. Figures and Graphics, Customising Fonts and Colours: Insert an image or figure into the document with a caption, Resize and position the image using wrap text and alignment options, Change font family and colour scheme for different document elements, Create a cover page with custom fonts, colours, and images. **(4 hours)**
5. References, Citations, and Headers/Footers/Page Numbering: Insert a bibliography and add citations using a reference manager or built-in citation tools, Apply consistent header and footer designs across the document, Insert and format page numbers (e.g., Roman numerals for intro, Arabic for content), Create a Table of Contents and update it automatically. **(4 hours)**
6. Creating a Resume or CV: Use a template or design a CV layout from scratch with appropriate sections (Education, Experience, Skills), Insert a profile photo, format contact details, and add social media links, Apply proper use of whitespace, bullets, and alignment to ensure clarity and readability, Export the CV as a PDF and check formatting consistency. **(4 hours)**
7. Introduction to MATLAB Interface and Basic Commands: Using command window, editor, and workspace, Arithmetic operations, using help, clc, clear, who, whos. Variable Assignment and Data Types: Creating scalars, vectors, complex numbers, Type conversion, and precision handling, Matrix Creation and Manipulation: Defining matrices, transposition, reshaping, Indexing, slicing, concatenation, Matrix multiplication, inversion, determinant, eigenvalues. **(4 hours)**
8. Plotting and Visualization 2D and 3D Plotting: plot(), subplot(), title(), xlabel(), ylabel(), legend(), 3D plots: mesh(), surf(), contour(), Data Visualization with Customization: Bar graphs, pie charts, histograms, Line styles, markers, color changes. **(4 hours)**
9. Loops and Conditional Statements: Using for, while, and if-else, writing loops to sum a series, the Fibonacci series, and Conditionals to check prime numbers or grading systems, Functions and Scripts, Creating User-Defined Functions, writing a function to compute factorial, and standard deviation. **(4 hours)**
10. Using Scripts for Automation: Writing scripts to read input, process, and display output, File Handling: Reading and Writing Files, reading data from .txt, .csv using fopen, fscanf, textscan, readmatrix(), Writing output to files. **(4 hours)**
11. Signal Processing: Basic Signal Generation and Analysis, generating sine, square, and triangular waves, plotting signals, and performing FFT, Filtering Signals: Applying FIR and IIR filters, using filter (), butter (), and freqz(). **(4 hours)**



12. Image Processing: Image Reading and Display, Read and display an image using imread(), imshow(), and rgb2gray(), Image Enhancement and Filtering: Histogram equalization, edge detection, smoothing. **(8 hours)**
13. Optimisation and Curve Fitting: Curve Fitting using polyfit() and fit(), Fit a polynomial or custom function to data, Evaluate goodness of fit, optimisation using fminsearch, fmincon: Minimise a nonlinear function with constraints. **(8 hours)**

## List of DSEs for Semester II

### **DSE201: Social Networks Analysis [3-0-1]**

Course title & Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE 201	4	3	0	1	Basic knowledge of graphs

#### **Course Objectives:**

The course aims to equip students with various SNA approaches to data collection, cleaning, and pre-processing of network data.

#### **Course Learning Outcomes:**

On completing this course, the student will be able to:

1. explain the basic concepts and principles of social network.
2. identify different types of social networks and their characteristics.
3. implement and apply various social network analysis techniques, such as influence maximisation, community detection, link prediction, and information diffusion.
4. apply network models to understand phenomena such as social influence, diffusion of innovations, and community formation.

#### **Syllabus:**

##### **Unit-I**

**(9 hours)**

Introduction to Social Network Analysis, Types of Networks, Nodes, Edges, Node Centrality, betweenness, closeness, eigenvector centrality, network centralisation, Assortativity, Transitivity, Reciprocity, Similarity, Degeneracy and Network Measure, Networks Structures, Network Visualisation, Tie Strength, Trust, Understanding Structure Through User Attributes



and Behaviour.

## **Unit-II (12 hours)**

Link Analysis and Link Prediction: Applications of Link Analysis, Signed Networks, Strong and Weak Ties, Link Analysis and Algorithms, Page Rank, Personalized PageRank, DivRank, SimRank, PathSim. Temporal Changes in a Network, Evaluation Link Prediction Algorithms, Heuristic Models, Probabilistic Models, Applications of Link Prediction.

## **Unit-III (12 hours)**

Community Detection: Applications of Community Detection, Types of Communities, Community Detection Algorithms, Disjoint Community Detection, Overlapping Community Detection, Local Community Detection, Evaluation of Community Detection Algorithms.

## **Unit IV: (12 hours)**

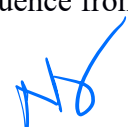
Influence Maximisation: Applications of Influence Maximisation, Diffusion Models, Independent Cascade Model, Linear Threshold Model, Triggering Model, Time-Aware Diffusion Model, Non-Progressive Diffusion Model. Influence Maximization Algorithms, Simulation-Based Algorithms, Proxy-Based Algorithms, Sketch-Based Algorithms, Community-Based Influence Maximization, and Context-Aware Influence Maximization.

### **Readings:**

1. Chakraborty, T. *Social Network Analysis*, Wiley India, 2021.
2. Knoke, D. and Yang, S. *Social network analysis*. SAGE Publications, 2019.
3. Golbeck, J. *Analyzing the social web*, Morgan Kaufmann, 2013.
4. Wasserman, S and Faust, K. *Social network analysis: Methods and applications*, Cambridge University Press, 2012.
5. Newman, M.E.J. *Networks: An introduction*. Oxford University Press, 2010.
6. Chen, W. Castillo, C and Lakshmanan, L.V.S. *Information and influence propagation in social networks*, Springer Nature, 2014
7. Srinivas, V and Mitra, P. *Link prediction in social networks: role of power law distribution*, New York: Springer International Publishing, 2016

### **List of Practical's:**

1. Implement heuristic link prediction methods such as Common Neighbors, Adamic-Adar, and Jaccard Coefficient to identify potential future links in a network. **(4 hours)**
2. Apply SimRank and PathSim to compute similarity scores for link prediction in graph-based data. **(4 hours)**
4. Compare heuristic and probabilistic link prediction models using evaluation metrics like precision and recall. **(4 hours)**
5. Use the Girvan–Newman algorithm to detect disjoint communities by iteratively removing high-betweenness edges. **(4 hours)**
6. Apply the Louvain or Leiden algorithm to uncover modular communities in large-scale networks. **(4 hours)**
7. Evaluate community detection results using modularity and normalised mutual information (NMI). **(4 hours)**
8. Simulate the Independent Cascade Model (ICM) to observe the spread of influence from



- a given set of seed nodes. **(4 hours)**
9. Simulate the Linear Threshold Model (LTM) to model influence propagation based on node activation thresholds. **(4 hours)**
  10. Implement Greedy and CELF algorithms to efficiently select the top-k influential nodes for maximizing spread. **(4 hours)**
  11. Benchmark different influence maximisation techniques, including simulation-based, sketch-based, and community-based methods. **(4 hours)**

### **DSE202: Combinatorial Optimization [3-0-1]**

Course title & Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE202	4	3	0	1	knowledge of Linear Algebra

#### **Course Objectives:**

The course aims to equip students with the technique of linear and integer programs to solve optimization problems via LP-based solutions to shortest path problems, minimum spanning tree problem, max-flow problem and maximum matching problem.

#### **Course Learning Outcomes (CO):**

On completion of this course, the student will be able to:

1. model problems using linear and integer programs.
2. differentiate between the computational complexities of LP and IP.
3. understand polyhedral analysis and apply it to develop algorithms.
4. understand the concept of duality and use it to design exact and approximate algorithms.
5. understand and explain the mathematical theory forming the basis of many algorithms for combinatorial optimization (particularly graph theoretic).

#### **Syllabus:**

##### **Unit-I (15 hours)**

Introduction: Optimization problems, neighborhoods, local and global optima, convex sets and functions, simplex method, degeneracy; duality and dual simplex algorithm, computational considerations for the simplex and dual simplex algorithms-Dantzig-Wolfe algorithms.

##### **Unit-II (10 hours)**

Integer Linear Programming: Cutting plane algorithms, branch and bound technique and



approximation algorithms for traveling salesman problem.

### **Unit-III (15 hours)**

Graph Algorithms: Primal-Dual algorithm and its application to shortest path (Dijkstra's algorithm, Floyd-Warshall algorithms), max-flow problem (Ford and Fulkerson labeling algorithms), matching problems (bipartite matching algorithm, non-bipartite matching algorithms, bipartite weighted matching-hungarian method for the assignment problem, non-bipartite weighted matching problem), efficient spanning tree algorithms.

### **Unit-IV (5hours)**

Matroids: Independence Systems and Matroids, Duality, Matroid Intersection.

### **Readings:**

1. Korte, Bernhard H., Jens Vygen, B. Korte, and J. Vygen. *Combinatorial optimization*. Springer, 2018.
2. Matoušek, Jiří, and Bernd Gärtner. *Understanding and using linear programming*. Springer, 2007.
3. Papadimitriou, Christos H., and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Publication, 1998.
4. Bazaraa, Mokhtar S., John J. Jarvis, and Hanif D. Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.
5. Taha, Hamdy A. *Operations research: an introduction*. Pearson Education India, 2014.

### **List of Practical's:**

1. WAP to implement Simplex Method (4 hours)
2. WAP to implement Dual Simplex Method (6 hours)
3. WAP to implement Primal-Dual algorithm for shortest path problem (6 hours)
4. WAP to implement Floyd-Warshall algorithm for shortest path problem (6 hours)
5. Implement algorithms for matching problems. (8hours)

## **DSE203: Cyber Security [3-0-1]**

Course title & Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE203	4	3	0	1	

### **Course Objectives:**

This course will be responsible for laying the foundation for creating a comprehensive understanding and expertise in the field of cybersecurity. This paper will set the level field for all the students to be able to come on par and move together as they must go deeper into hardcore cybersecurity topics during the course duration.





### **Course Learning Outcomes:**

On completion of this course, the student will be able to:

1. state the need and scope for cyber laws.
2. enumerate various network attacks, describe their sources, and mechanisms of prevention.
3. describe the genesis of SCADA policies and their implementation framework.
4. carry out malware analysis.

### **Syllabus:**

#### **Unit-I**

**(7 hours)**

Introduction: Cyberspace, Internet, Internet of things, Cyber Crimes, cyber criminals, Cyber security, Cyber Security Threats, Cyber laws and legislation, Law Enforcement Roles and Responses.

#### **Unit-II**

**(15 hours)**

Cyberspace Attacks: Network Threat Vectors, MITM, OWASP, ARP Spoofing, IP & MAC Spoofing, DNS Attacks, SYN Flooding attacks, UDP ping-pong and fraggle attacks, TCP port scanning and reflection attacks, DoS, DDOS. Network Penetration Testing Threat assessment, Penetration testing tools, Penetration testing, Vulnerability Analysis, Threat matrices, Firewall and IDS/IPS, Wireless networks, Wireless Fidelity (Wi-Fi), Wireless network security protocols, Nmap, Network fingerprinting, BackTrack, Metasploit.

#### **Unit-III:**

**(15 hours)**

Introduction to SCADA (supervisory control and data acquisition): Understanding SCADA security policies, SCADA Physical and Logical Security, understanding differences between physical and logical security, define perimeter controls and terms, define various security zones, understand communication cyber threats, Understand firewall, architectures.

#### **Unit-IV**

**(8 hours)**

Introduction Malware Analysis: Static Analysis, Code Review, Dynamic Analysis, Behavioral analysis of malicious executable, Sandbox Technologies, Reverse-engineering malware, Defeat anti-reverse engineering technique, automated analysis, intercepting network connections, Network flow analysis, Malicious Code Analysis, Network analysis.

### **Readings:**

1. Peter W. Singer and Allan Friedman, *Cybersecurity and Cyberwar*, Oxford University Press, 2014.
2. Michael Sikorski, Andrew Honig, *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software* 2012, No Starch Press, San Francisco.
3. Dejay, Murugan, *Cyber Forensics* Oxford university press India Edition, 2018.
4. R. Rajkumar, D. de. Niz and M. Klein, *Cyber Physical Systems*, Addison-Wesely, 2017
5. Jonathan Clough, *Principles of Cybercrime*, Cambridge University Press, 24-Sep-2015.

### **References:**

1. CEH official Certified Ethical Hacking Review Guide, Wiley India Edition, 2015.



### **List of Practical's:**

**Disclaimer:** Please note that these lab exercises are for educational purposes only. All activities should be performed within the boundaries of the law.

1. Identify your machine's IP configuration using *ipconfig* (Windows) or *ifconfig* (Linux). Ping various local and remote hosts to verify connectivity, trace the route to a website using *tracert* or *traceroute*, perform DNS lookups using *nslookup*, and gather domain registration details using the *whois* tool. **(2 hours)**
  2. Use tools like *John the Ripper* and *Hydra* to crack passwords. The task will involve using wordlist-based and brute-force techniques to break into password-protected accounts or services. **(4 hours)**
  3. SET tool is one of the tools available in Kali Linux. Explore the tool and list down all types of Social Engineering attacks available in the tool. Further, make a document mentioning the Name of the attack, and how the attack can be launched. How can I make my system safe from these attacks? **(6 hours)**
  4. Use the Social Engineering Toolkit (SET) available in Kali Linux to clone a vulnerable website (e.g., <http://www.itsecgames.com/>). Compare the cloned website with the original, explore different social engineering attacks such as phishing or credential harvesting. **(4 hours)**
  5. Use *theHarvester* tool to collect at least 25 email addresses from various publicly available sources on the internet. These addresses can later be used for phishing simulations or further social engineering attacks. **(2 hours)**
  6. Utilize tools from the *iHunt intelligent framework* available on web to create a sock puppet account. Document the tools used, their functionality, and how this can be applied in a simulated information-gathering exercise. **(2 hours)**
  7. Using the email addresses collected in Lab 6, perform further information gathering. Use search engines and social media platforms to find publicly available information about the email owners. Document the results using the tools given on the *iHunt intelligent framework* for email investigation. **(2 hours)**
- Note: Don't actually contact the person of interest. Just gather information which available publicly. Don't break any law.*
8. Conduct a vulnerability scan on target systems using *NMap*. Analyze the results to identify potential vulnerabilities such as open ports, weak configurations, and security loopholes. **(4hours)**
  9. Use the *Metasploit framework* to exploit vulnerabilities and gain access to a remote Linux machine. The exercise involves using Kali Linux with Metasploit to exploit a vulnerable system. **(4hours)**
  10. A lab exercise related to SCADA & Industrial Control Systems (ICS) Security **(4 hours)**



## **DSE204: Information Retrieval [3-0-1]**

Course Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE204	4	3	0	1	Basic understanding of Statistics and Probability is required.

### **Course Objectives:**

This course introduces the basics of Information Retrieval (IR), focusing on how information is organized, searched, and ranked. Students will learn about different search models, document processing techniques, and ways to measure search accuracy. The course also covers web search methods, including link analysis and web crawling.

### **Course Learning Outcomes:**

Upon successful completion of this course, students will be able to

1. exhibit the understanding of the fundamental concepts of Information Retrieval (IR), including information need, relevance, and early developments in IR systems.
2. apply different retrieval models, such as Boolean retrieval and ranked retrieval, to effectively search and organize information.
3. evaluate search performance using precision-recall, ranking measures, and other standard evaluation metrics.
4. process and represent documents using techniques like vector space modelling, feature selection, stemming, and similarity measures.
5. explore web search techniques, including link analysis methods like PageRank and HITS, as well as web crawling strategies.

### **Syllabus:**

#### **Unit-I**

**(15 hours)**

Introduction to Information Retrieval (IR), information, information need, and relevance, the IR system and its components, early developments in IR, user interfaces in IR, retrieval and IR models, Boolean retrieval, term vocabulary and postings list, ranked retrieval, inverted index, index construction, index compression.

#### **Unit-II**

**(10 hours)**

Document processing, document representation techniques, vector space model, feature selection for IR, stop words, stemming, concept of document similarity, evaluation of information retrieval systems, notion of precision and recall, precision-recall curve.

#### **Unit-III**

**(10 hours)**

Standard performance measures, MAP, reciprocal ranks, F-measure, NDCG, rank correlation,



standard datasets for IR evaluation, web search and link analysis, web crawling techniques, link analysis methods, PageRank algorithm, HITS algorithm.

#### **Unit-IV**

**(10 hours)**

Classification and Clustering: Notion of supervised and unsupervised algorithms, Naive Bayes, nearest neighbour and Rochio's algorithms for text classification, text clustering methods such as K-Means.

#### **Readings:**

1. Baeza-Yates, Ricardo, and Berthier Ribeiro-Neto. *Modern information retrieval*. Vol. 463, no. 1999. New York: ACM Press, 1999.
2. Schütze, Hinrich, Christopher D. Manning, and Prabhakar Raghavan. *Introduction to information retrieval*. Vol. 39. Cambridge: Cambridge University Press, 2008.
3. Grossman, David A., and Ophir Frieder. *Information retrieval: Algorithms and heuristics*. Vol. 15. Springer Science & Business Media, 2004.
4. Butcher, Stefan, Charles LA Clarke, and Gordon V. Cormack. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2016.
5. Croft, W. Bruce, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*. Vol. 520. Reading: Addison-Wesley, 2010.

#### **List of Practical's**

1. Design and implement a Boolean retrieval system that can process queries using AND, OR, and NOT operators on a given set of text documents. **(2 hours)**
2. Develop a text preprocessing pipeline that performs tokenization, case folding, stop-word removal, and stemming on a document corpus. **(2 hours)**
3. Write a program that constructs an inverted index for a collection of documents and supports retrieval of documents based on single-word queries, then implement index compression using variable-byte encoding and analyze the reduction in index size and its effect on retrieval performance. **(4 hours)**
4. Compute the Term Frequency-Inverse Document Frequency (TF-IDF) values for each term in a document collection and store them in a structured format. **(2 hours)**
6. Implement a ranked retrieval system using the vector space model and cosine similarity to score and rank documents based on a user query. **(2 hours)**
7. Use a publicly available IR dataset to implement and compare different retrieval models (e.g., Boolean, Vector Space), then calculate evaluation metrics including precision, recall, F1-score, MAP, and NDCG using provided relevance judgments, and finally, generate precision-recall curves for multiple queries to interpret the performance of each model. **(4 hours)**
8. Implement both the PageRank and HITS algorithms on a manually created web graph, simulate multiple iterations to observe rank convergence, and compute hub and authority scores for each node. **(4 hours)**
9. Train and evaluate a Naive Bayes classifier to categorize documents into predefined classes such as spam vs. ham or news categories. **(2 hours)**
10. Apply the K-Means clustering algorithm to a set of TF-IDF document vectors and visualize the resulting clusters using a 2D projection. **(4 hours)**
11. Develop a basic web crawler that extracts and stores titles and content from publicly accessible websites while respecting robots.txt. **(4 hours)**



## **DSE205: Digital Image Processing [3-0-1]**

Course title & Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice	
DSE205	4	3	0	1	

### **Course Objectives:**

The course will thoroughly introduce image enhancement in the spatial and frequency domains, followed by image morphological operations such as dilation, erosion, hit-or-miss transformations, image segmentation and image compression.

### **Course Learning Outcomes:**

Upon successful completion of this course, the student will

1. be able to enhance the quality of an image using various transformations.
2. analyze the transform of an image in spatial domain to frequency domain.
3. apply required morphological operations to an image.
4. adequate to segment an image using various approaches.

### **Syllabus:**

#### **Unit-I (7 Hours)**

**Introduction:** Applications of digital image processing, steps in digital image processing: image acquisition, image sampling and quantization, basic relationships between pixels.

#### **Unit-II (14 Hours)**

Image enhancement: grey level transformations, histogram processing, local enhancement, image subtraction, image averaging, spatial filtering: smoothing and sharpening filters, Discrete Fourier transformation, filtering in the frequency domain: smoothing and sharpening filters, image restoration in spatial and frequency domains.

#### **Unit-III (12 Hours)**

Morphological image processing and Image Compression: erosion and dilation, opening and closing, hit-or-miss transformation, and some basic morphological algorithms, Image compression models, error-free compression techniques, lossy compression techniques, JPEG, MPEG.

#### **Unit-IV (12 Hours)**

Image segmentation Point: line and edge detection, gradient operator, edge linking and boundary detection, thresholding, region-based segmentation, representation schemes like chain codes, polygonal approximations, boundary segments, skeleton of a region, boundary descriptor, regional descriptor.

### **Readings:**

1. Gonzalez, Rafael C. and Woods, Richard E., *Digital Image Processing*, 4<sup>th</sup> edition, Pearson Education, 2018.
2. Annadurai, S. and Shanmugalakshmi, R., *Fundamentals of Digital Image Processing*, 1<sup>st</sup> edition, Pearson, 2006.
3. Joshi, M. A., *Digital Image Processing: An Algorithmic Approach*, 2<sup>nd</sup> edition, PHI Learning, 2020.

### **References:**

1. Jahne, Bernd, *Digital Image Processing*, 6<sup>th</sup> edition, Springer, 2005.
2. Chandra, B. and Majumder, D.D., *Digital Image Processing and Analysis*, 2<sup>nd</sup> edition, Prentice Hall India Learning Private Limited, 2011.

### **List of practical:**

*You may choose a suitable programming language and all necessary/relevant inputs/problems.*

1. Implement a program for displaying grey-scale and RGB colour images, with those for accessing pixel locations, and investigate adding and subtracting a scalar value from an individual location. **(2 Hours)**
2. Implement a program to perform the image sampling and quantization technique and display the resulting images. **(4 Hours)**
3. Implement a program to apply histogram equalization to a colour image, and apply contrast stretching to the colour example image. Experiment with different parameter values to find an optimum for the visualization of this image. **(2 Hours)**
4. Implement a program to add different levels of salt and pepper and Gaussian noise to images, both in colour and grey-scale. Investigate the usefulness of all filtering for removing different levels of image noise. **(2 Hours)**
5. Write a program for all image restoration techniques. **(4 Hours)**
6. Write a program for the Fourier transform of an image. **(2 Hours)**
7. Write a program for image spatial and sharpening filtering. **(4 Hours)**
8. Implement a program to find the result of erosion, dilation, opening and closing with the below structuring element and choose any image. **(2 Hours)**

a) 1 0 0      b) 1 1 1  
     0 1 0      1 1 1  
     0 0 1      1 1 1

9. Write a program for all the compression techniques of an image. **(4 Hours)**



10. Write a program for all image segmentation techniques. (4 Hours)

### **DSE206: DATA MINING [3-0-1]**

Course title & Code	Credits	Credit distribution of the course			Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/Practice	
DSE 206	4	3	0	1	-

#### **Course Objectives:**

In this course, the objective is to introduce the KDD process. The course should enable students to translate real-world problems into predictive and descriptive tasks. The course also covers data cleaning and visualization, supervised and unsupervised mining techniques.

#### **Course Learning Outcomes:**

At the end of the course, the student will be able to:

1. distinguish between the process of knowledge discovery and Data Mining.
2. play with basic data exploration methods to develop understanding of given data.
3. identify suitable pre-processing method for give problem.
4. describe different data mining tasks and algorithms.
5. use programming tools (e.g. Weka/Python/R etc) for solving data mining tasks.
6. follow formal notations and understand the mathematical concepts underlying data mining algorithms

#### **Syllabus:**

##### **Unit I**

**(9 hours)**

Overview: The process of knowledge discovery in databases, predictive and descriptive data mining techniques, and unsupervised learning techniques. Data pre-processing: Data cleaning, Data transformation, Data reduction, Discretization

##### **Unit-II**

**(12 hours)**

Classification: Supervised learning/mining tasks, Decision trees, Decision rules, Statistical (Bayesian) classification, Instance-based methods (nearest neighbour), Evaluation and Validation methods.

##### **Unit-III**

**(12 hours)**

Clustering: Basic issues in clustering, Partitioning methods ( k-means, expectation maximisation), Hierarchical methods for clustering, Density-based methods, Cluster Validation methods and metrics.

##### **Unit-IV**

**(12 hours)**

Association Rule Mining: Frequent item set, Maximal and Closed item sets, Apriori property, Apriori algorithm.

#### **Readings:**





1. Han, Jiawei, Jian Pei, and Hanghang Tong. *Data Mining: Concepts and Techniques* 3<sup>rd</sup> edition. Morgan Kaufmann, 2022.
2. Zaki, Mohammed J., WM Jr, *Data Mining and Analysis: Fundamental Concepts and Algorithms*, Cambridge University Press, 2014.
3. Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*, Addison Wesley, 2006.
4. Charu, C. *Data Mining: The Textbook*, Springer, 2015

### **List of Practical's:**

#### **1. Data Preprocessing & Visualization (4 Hours)**

- Load and explore real-world datasets using Python/Weka/R
- Perform data cleaning: handle missing values, detect/treat outliers
- Apply data transformation: normalization, encoding categorical variables
- Use data reduction techniques: PCA, sampling
- Apply discretization/binning on continuous features
- Visualize data using histograms, boxplots, scatter plots, and heatmaps

#### **2. Supervised Learning – Classification (4 Hours)**

- Implement a Decision Tree classifier and visualise the tree
- Apply Naive Bayes classifier and evaluate accuracy
- Implement k-Nearest Neighbor (k-NN) classifier with different values of  $k$
- Evaluate classifiers using confusion matrix, precision, recall, and F1-score
- Perform k-fold cross-validation and plot ROC-AUC curves

#### **3. Unsupervised Learning – Clustering (8 Hours)**

- Apply k-Means clustering and determine optimal  $k$  using Elbow/Silhouette method
- Perform Agglomerative Hierarchical clustering and plot dendrogram
- Apply DBSCAN for density-based clustering and tune parameters (eps, minPts)
- Visualize and interpret clustering results

#### **4. Association Rule Mining (4 Hours)**

- Use Apriori algorithm to find frequent itemsets from transaction data
- Generate association rules using confidence and lift thresholds
- Analyze discovered rules for business decision insights

#### **5. Mini-Project (10 Hours)**

- Select a real-world dataset (e.g., healthcare, e-commerce, finance)
- Apply full KDD process: preprocessing, modelling, (classification/clustering/association), and evaluation
- Summarize findings with visualizations and model insights

