www.teachics.org

Data Structures Using C

Module 5 - PART 1

Table of Contents

01

Introduction to Graph

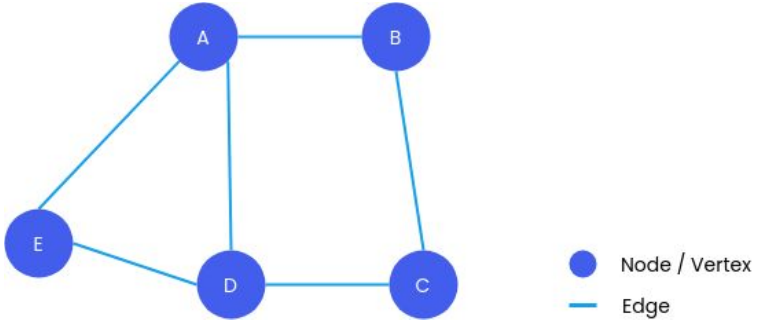
Introduction, Definition, Basic Terminology & Types.

Definition

A **Graph** is a non-linear data structure consisting of nodes and edges.

Definition : A graph **G=<V,E>** consist of two sets: V and E, where

- V is the set of **nodes** (vertices or points)
- E is the set of **edges**, identified with a unique pair of nodes in V, denoted by e=[u, v]**.**



Terminology

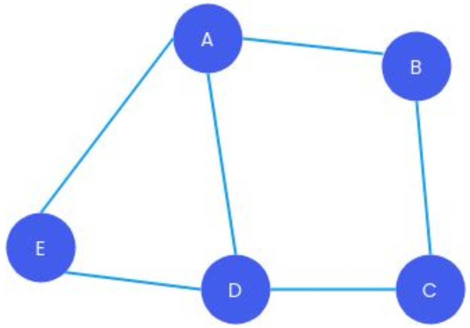
- Suppose e = [u, v]. Then nodes u and v are called **endpoints** of e, and u and v are called **adjacent nodes** or neighbours.
- The **degree** of node u, is the number of edges containing u.
- If u does not belong to any edge(degree 0)-then u is called **isolated edge**.
- A **path** in a graph is a finite sequence of edges which joins a sequence of vertices.
- **Path length** is equal to the number of edges present in the path.
- A path is said to be **closed** if it starts and ends at the same node.
- A path is said to be **simple** if all the nodes are distinct with exception that first node may equal last node .
- A **cycle** is a closed simple path with length 3 or more.
- A graph is said to be **connected** if and only if there is a path between any two of its nodes.
- A graph G is said to be **complete** if every node u in G is adjacent to every other node v in G.
- An edge e is called a **loop** if it has identical endpoints, that is, if e = [u,u].

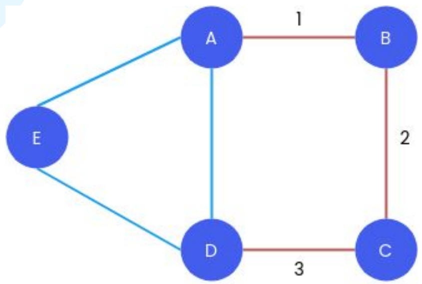
Terminology

- If there are multiple edges between the same pair of nodes in a graph, then the edges are called **parallel edges**.
- A graph containing self loop or parallel edges or both is called a **multigraph**.

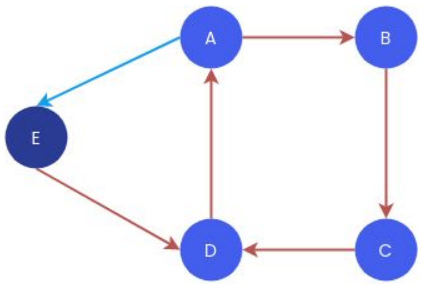
Example:

- A, B, C, D, E - **Nodes**.
- A and B are **adjacent nodes.**
- A-B, B-C, C-D, D-C, D-E, E-A - **Edges**.
- **Degree of A** = 3.
- A-B-C-D-E - **Path from A to E** with **length 4.**
- A-B-C - **Path from A to C** with **length 2.**
- B-C-D-A-B - **Closed path from B to B**.
- E-A-B-C-D-E is a **cycle**.

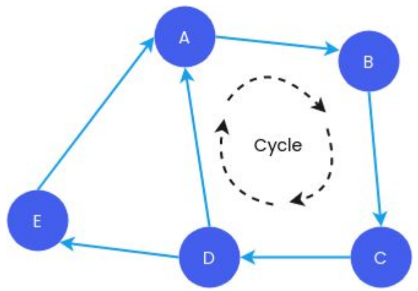


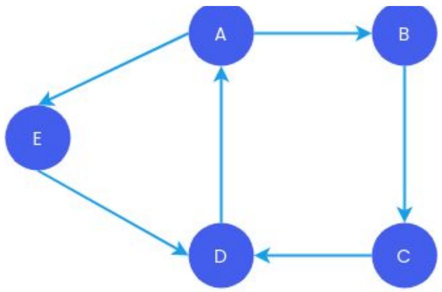


Path from A to D with length 3

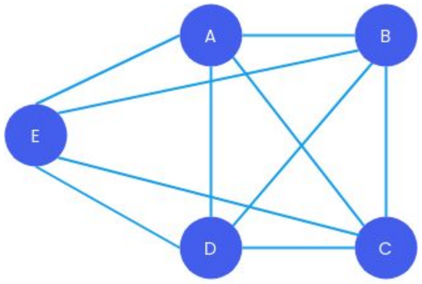


Not a simple path (E-D-A-B-C-D-A)

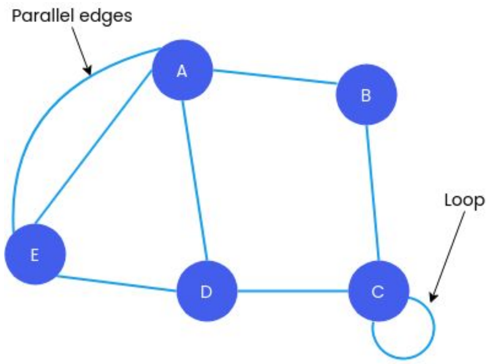




Connected Graph



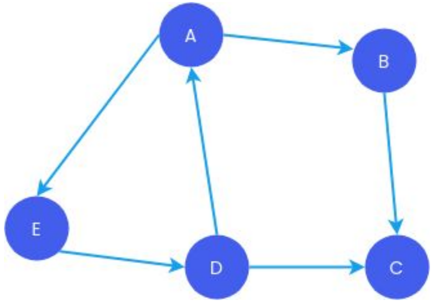
Complete Graph



Multigraph

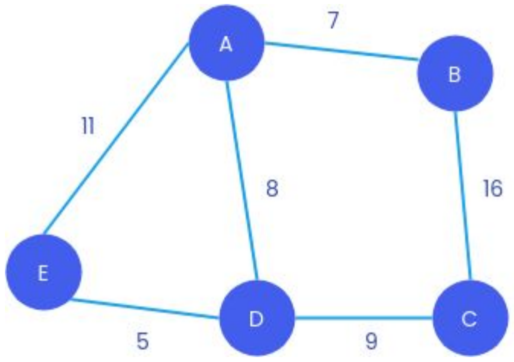
Directed Graph

- A **directed graph** is a graph with the property that its **edges** have **directions**.
- Each edge e is identified with an **ordered pair** [u, v].
- They are also called a digraph.
- Suppose a digraph has a directed edge e = [u, v]
  - e begins at u, and ends at v.
  - u is the origin or initial point of e, and v is the destination or terminal point of e.
- The **outdegree** of a node u is the number of edges beginning at u.
- The **indegree** of a node u is the number of edges ending at u.
- A node u is called a **source** if it has a positive outdegree but zero indegree.
- A node u is called a **sink** if it has a zero outdegree but a positive indegree.
- A node v is said to be reachable from a node u if there is a directed path from v to u.
- A directed graph G is said to be connected(strongly connected) if for each pair u, v of nodes in G there is a path from u to v and there is also a path from v to u.



Weighted Graph

- A **weighted graph** is a graph with the property that its edges have a number associated with it.
- The number is called **weight** of the edge.
- A weighted graph can be undirected or directed.
- The **weight of a path** P is the sum of the weights of the edges along the path P.



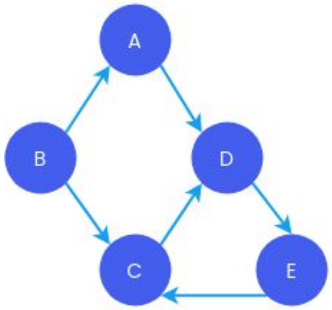
02

Graph Representation

Sequential & Linked list representation of graphs.

Sequential Representation : Adjacency Matrix

- A graph having N nodes can be represented using a NxN square matrix where rows and column numbers represent the vertices.
- A cell at the cross-section of the vertices can have only values 0 or 1.
- An entry $(v_i, v_j) = 1$ only when there exists an edge from v_i to v_j.
- In case of weighted graph, the edge weight is stored instead of 1.
- Adjacency matrix will be a sparse, hence a great deal of space will be wasted.
- It may be difficult to add or delete new nodes.



	A	B	C	D	E
A	0	0	0	1	0
B	1	0	1	0	0
C	0	0	0	1	0
D	0	0	0	0	1
E	0	0	1	0	0

Path Matrix

- For a directed graph G with adjacency matrix A, the element A_{ij} of matrix $\mathbf{A^k}$ gives the **number of paths of length k** from vertex v_i to v_j.
- For example, an element A_{ij} of A^2 will denote the number of paths of length 2 from vertex v_i to v_j.
- Now we define a matrix B_r ,

  $\boldsymbol{B_r = A+A^2+A^3+ \ldots +A^r}$
- The ij entry of the matrix B, gives the number of paths of lengths r or less from node v_i to v_j.
- If G has m nodes, a simple path from any v_i to v_j must have length m-1 or less.
- Similarly a cycle must have length m or less.
- That is if there is a non-zero ij entry in the matrix B_m , there is a simple path from v_i to v_j.
- A **path matrix** is obtained by replacing all non-zero elements in B_m by 1.
- Then P_{ij} = 1 if and only if there is a path from v_i to v_j.

Path Matrix

	A	B	C	D	E
A	0	0	0	1	0
B	1	0	1	0	0
C	0	0	0	1	0
D	0	0	0	0	1
E	0	0	1	0	0

A

	A	B	C	D	E
A	0	0	0	0	1
B	0	0	0	2	0
C	0	0	0	0	1
D	0	0	1	0	1
E	0	0	0	1	0

A^2

	A	B	C	D	E
A	0	0	1	0	0
B	0	0	0	0	1
C	0	0	1	0	0
D	0	0	0	1	0
E	0	0	0	0	1

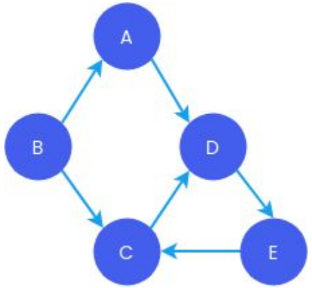
A^3

	A	B	C	D	E
A	0	0	0	1	0
B	0	0	2	0	0
C	0	0	0	1	0
D	0	0	0	0	1
E	0	0	1	0	0

A^4

	A	B	C	D	E
A	0	0	0	1	0
B	0	0	0	2	0
C	0	0	0	0	1
D	0	0	1	0	0
E	0	0	0	1	0

A^5



	A	B	C	D	E
A	0	0	1	3	1
B	1	0	3	4	1
C	0	0	1	2	2
D	0	0	2	1	3
E	0	0	2	2	0

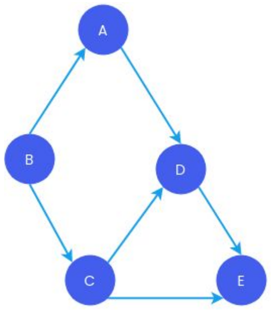
$B=A+A^2+A^3+A^4+A^5$

	A	B	C	D	E
A	0	0	1	1	1
B	1	0	1	1	1
C	0	0	1	1	1
D	0	0	1	1	1
E	0	0	1	1	0

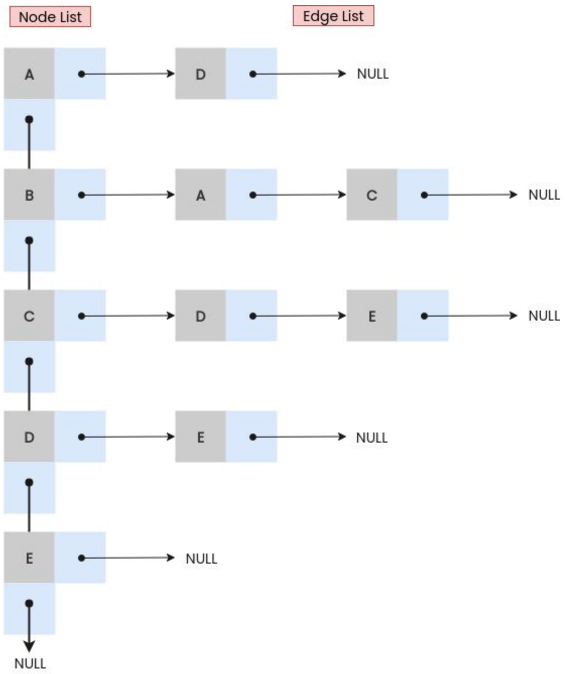
P

Linked Representation

- The linked representation contains two lists, a **NODE** list and an **EDGE**(Adjacency) list.
- Each element in the NODE list will correspond to a node in the graph.
  - **Name** of the Node.
  - Pointer to **adjacent nodes**.
  - Pointer to **next node**.
- Each element in the edge list will represents an edge of the graph.
  - **Destination node** of the edge.
  - **Link** that link together the edges with the same initial node.



Node	Adjacency List
A	D
B	A, C
C	D, E
D	E
E	-



03

Traversal of Graph

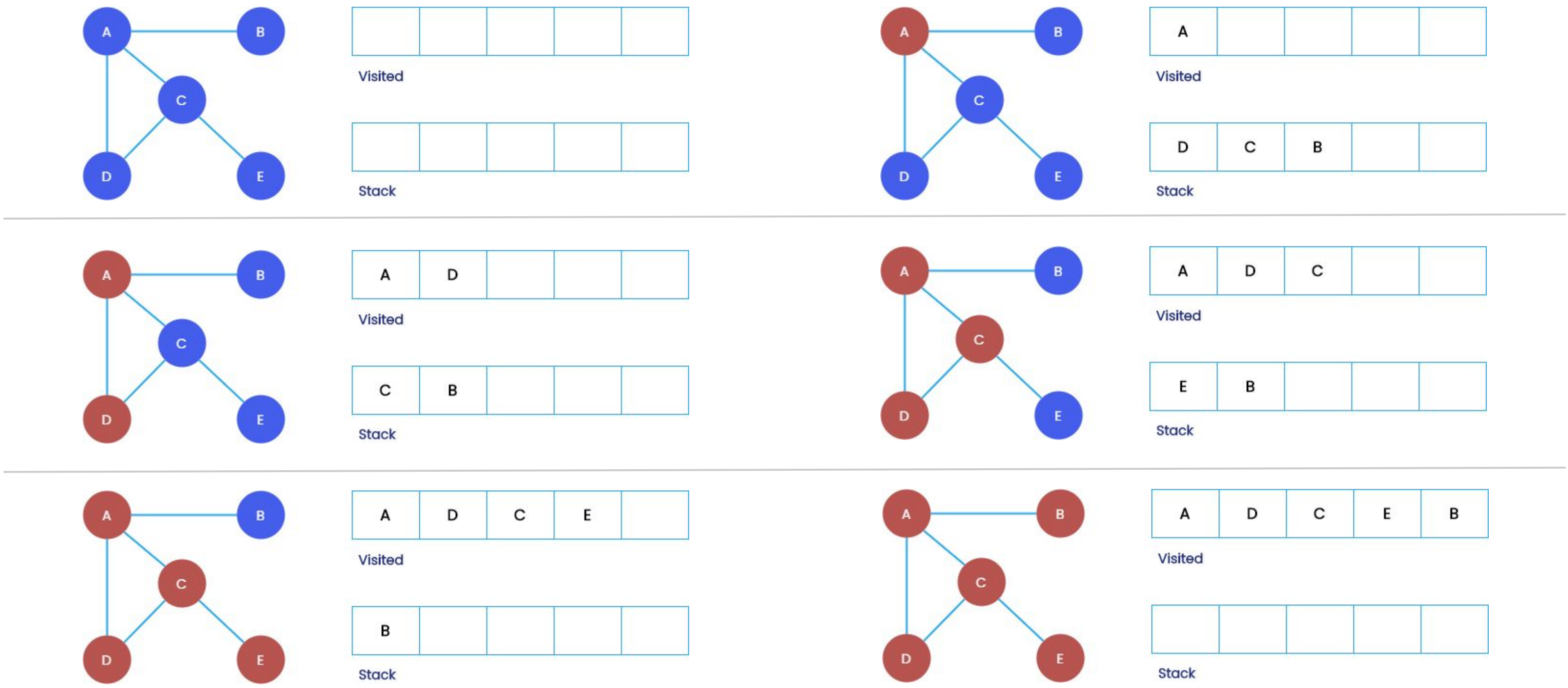
Depth First Search(DFS) and Breadth First Search(BFS).

Depth First Search (DFS)

- In DFS method, the travel starts from a node then follows its outgoing edges.
- Within successors, it travels their successors and so on.
- The search goes deeper into the search space till no successor is found.
- DFS implementation puts each nodes of the graph in following lists
  - Visited
  - Not Visited (Stack)
- This marking is used to avoid cycles while traversing.

```
Algorithm: DFS
 1.   Push Starting Node A on the Stack.
 2.   Repeat steps 3 and 4 until stack is empty.
 3.   Take the top item N from the Stack and add it to the
      Visited List.
 4.   Push all unvisited neighbours of N onto Stack.
 5.   Exit.
```

Depth First Search (DFS): An Example

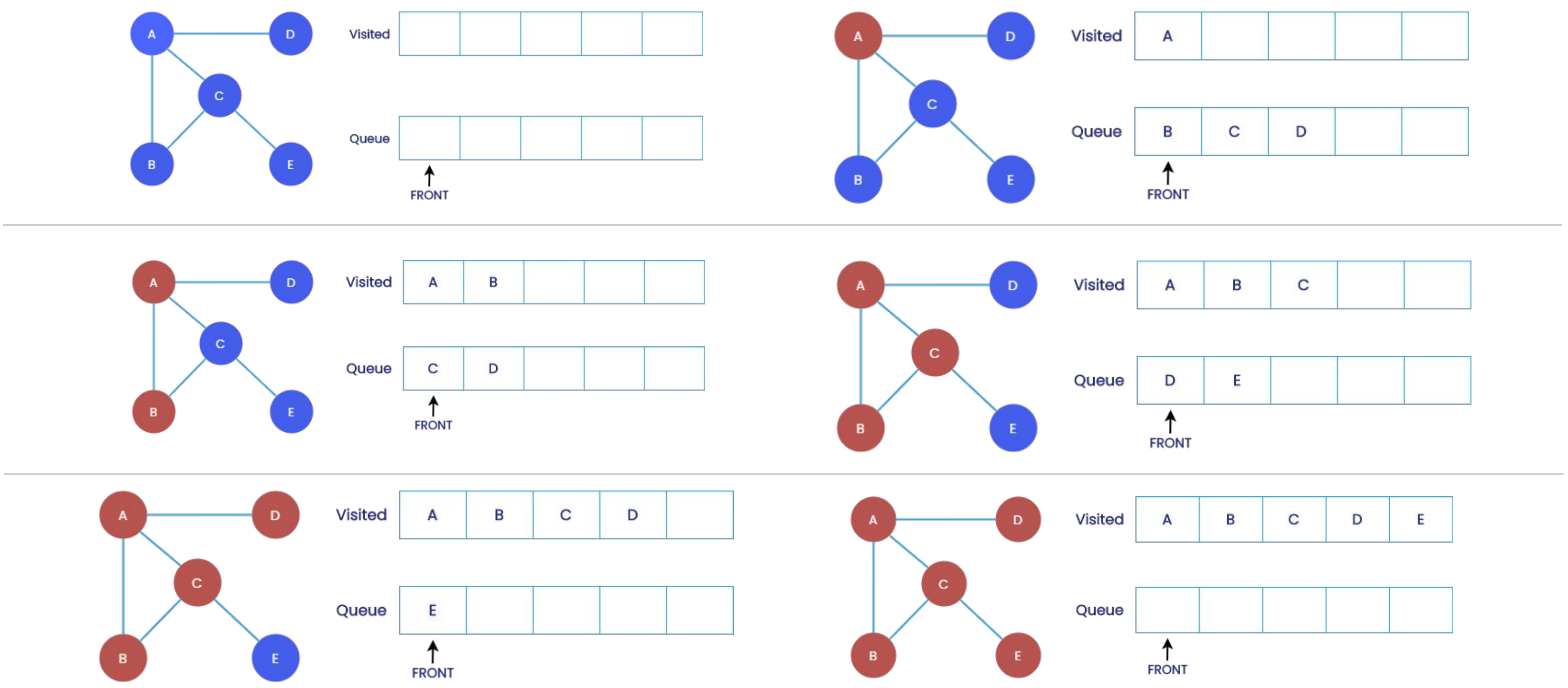


Breadth First Search (BFS)

- In BFS method, the travel starts from a node then carries onto its adjacent nodes.
- It traverse all the sibling nodes within a level and then moves to next level.
- This continues until all the vertices are visited.
- DFS implementation puts each nodes of the graph in following lists
  - Visited
  - Not Visited (Queue)
- This marking is used to avoid cycles while traversing.

```
Algorithm: BFS
 1.   Put the starting node A to the back of the Queue.
 2.   Repeat Steps 3 and 4 until Queue is empty.
 3.   Remove the front node N of Queue and add it to the
      Visited List.
 4.   Add all unvisited neighbours of N to the rear of the
      Queue.
 5.   Exit
```

Breadth First Search (BFS): An Example

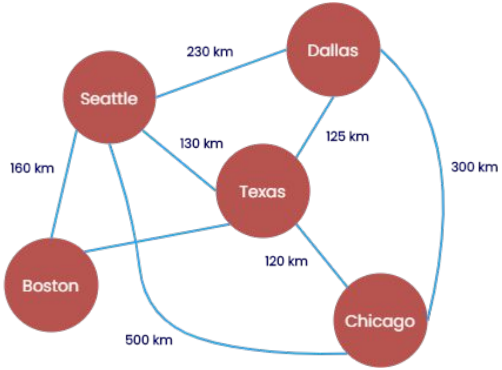


03

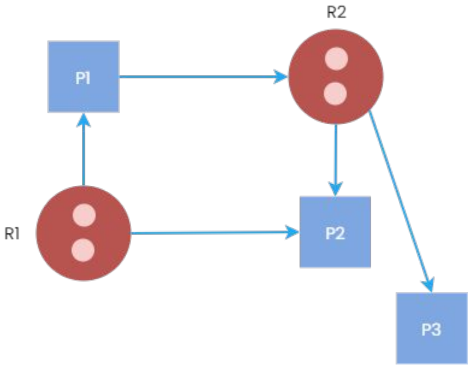
Applications

Applications of Graph data structure.

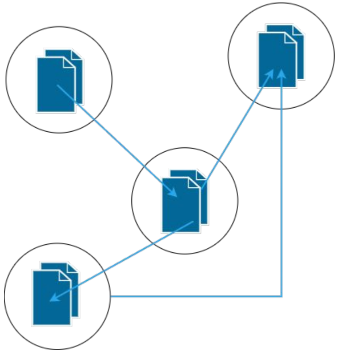
Applications



Google Map as a Graph



Resource Allocation Graph



World Wide Web as a Graph

- In Facebook, Users, Pages, Groups, etc. - nodes and edges - connection/relation.
- Path optimization Algorithms.
- Scientific Computations.
- Recommendation Algorithms.

Study at Home

Notes
Learn topics better with a little less effort utilizing notes by subject experts.

Syllabus
View or download syllabus for Bsc Computer Science under Calicut University.

Video Lessons
Better learning experience through video lectures from skilled personnel.

Question Papers
Familiarize with exam patterns through our library of previous year question papers.

Quiz
Test your knowledge of topics through various quizzes.

Solved Questions
Vast collection of questions and answers related to each topic.

Programming Laboratory
Practicals made easy with complete list of tested programs.

Software Downloads
Practice programming by downloading and installing respective softwares.

Teaching Resources
Find PPTs and related resources for teaching aid here.

teachics.

The Computer Science Learning Platform.