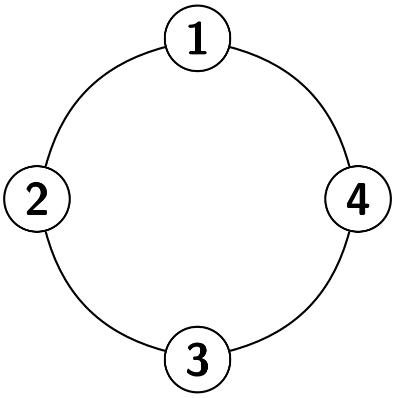
# Assignment-II

## National Institute of Technology Silchar

| | |
|---|---|
| Subject Code: CS-201 | Subject: Data Structures |
| Semester: 3rd B.Tech. | Department: CSE |
| Section: A | Due Date: 8th September 2023 |

1. Given the following graph-



The graph can be represented using an adjacency matrix as shown below-

| | **0** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 1 | 0 | 1 |
| **2** | 0 | 1 | 0 | 1 | 0 |
| **3** | 0 | 0 | 1 | 0 | 1 |
| **4** | 0 | 1 | 0 | 1 | 0 |

The adjacency matrix can be represented using a 2D array. Conventionally, the matrix cell can be 1 byte or 4 bytes each. On the contrary, we would like to reduce memory wastage. The cell of the matrix should occupy a single bit since it represents either 0 or 1. You should not use more than one bit in each matrix cell. Write a program to store a graph using an adjacency matrix given above. Implement the following functions-
*getNeighbor(n)*: It prints all the neighbors of node $n$. For instance, the neighbor of 3 is 2 and 4. The expected time complexity is $O(d)$ where $d$ is the degree of node $n$.
*insertEdge(x,y):* Insert an edge or connection between node x and y. For instance, insertEdge(1,3) establishes a connection between nodes 1 and 3. The expected time complexity is $O(1)$.
*deleteEdge(x,y):* Delete an edge between x and y. For instance, deleteEdge(3,4) deletes the connection between 3 and 4. The expected time complexity is $O(1)$.
*isNeighbor(x,y):* It returns true if y is neighbor of x. The expected time complexity is $O(1)$.
*Path(x,y):* Determine whether there exist a path from x to y or not. If it exists, print all the nodes in the path.
*maxDegree():* It returns the maximum connection of a given graph.
*minDegree():* It returns the minimum connection of a given graph.
*avgDegree():* It returns the average connection of a given graph.

******** END ********