**Manual of code**

**Lab 6.2**

**Explanation of the Code**

This code implements a Breadth-First Search (BFS) algorithm to traverse a graph. BFS is a method for exploring a graph level by level, starting from a given node. The code uses a queue (implemented as a list) to keep track of nodes to visit next, ensuring that nodes are processed in the order they are discovered**.**

**How the Code Works**

1. The bfs Function

The bfs function performs the Breadth-First Search. It takes two inputs:

* graph: A dictionary representing the graph. Each key is a node, and the value is a list of its neighboring nodes.
* start: The starting node for the traversal.

****

**2. Data Structures Used**

* visited: A set to keep track of nodes that have already been visited. This ensures that each node is processed only once.
* queue: A list that acts as a queue (First-In-First-Out or FIFO) to store nodes that need to be visited.

****

**3. Initialization**

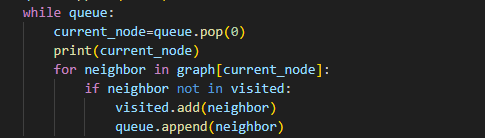
* Mark the start node as visited and add it to the queue.

****

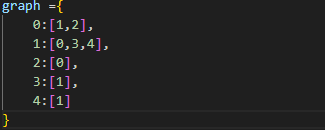
**4. Traversing the Graph**

The algorithm works as follows:

1. While the queue is not empty:
   * Remove the first node from the queue (this simulates the FIFO behavior of a queue).
   * Print the node (or process it in some way).
   * Add its unvisited neighbors to the end of the queue and mark them as visited.

****

**5. The Graph**

The graph is represented as a dictionary where each key is a node, and the value is a list of its neighboring nodes. Here's the graph used in the code**: **

**6. Running the BFS**

The BFS is started from node 0:

****

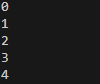
Step-by-Step Execution

Here’s what happens when the BFS is run:

1. Start at node 0:
   * Mark 0 as visited.
   * Add 0 to the queue.
2. Process node 0:
   * Print 0.
   * Add its neighbors 1 and 2 to the queue.
3. Process node 1:
   * Print 1.
   * Add its neighbors 3 and 4 to the queue.
4. Process node 2:
   * Print 2.
   * Node 2 has no unvisited neighbors.
5. Process node 3:
   * Print 3.
   * Node 3 has no unvisited neighbors.
6. Process node 4:
   * Print 4.
   * Node 4 has no unvisited neighbors.

**Output**

The output of the BFS traversal starting from node 0 is:

****