

**Green University of Bangladesh**

**Department of Computer Science and Engineering (CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Spring, Year:2025), B.Sc. in CSE (Day)**

**LAB REPORT NO: 02**

**Course Title: Artificial Intelligence Lab**

**Course Code: CSE 316 Section: 221-D4**

**Student Details**

|  |  |
| --- | --- |
| Name | ID |
| ARMAN HOSSAIN | 221002624 |

**Lab Date : 01 Mar 2025**

**Submission Date : 06 Mar 2025**

**Course Teacher’s Name : Mahjabin Rahman Oishe**

|  |  |
| --- | --- |
| **Lab Report Status**  **Marks: …………………………………** | **Signature:.....................** |
| **Comments:..............................................** | **Date:..............................** |
|  |  |

1. **TITLE OF THE LAB EXPERIMENT**

* Write a python code to perform bfs traversal on the given graph and print the order of the nodes.

1. **OBJECTIVES/AIM**

* To understand and implement the Breadth-First Search (BFS) algorithm
* To explore graph traversal techniques using a queue
* To enhance problem-solving skills by working with graph data structures
* To optimize code efficiency in handling directed graphs

1. **PROCEDURE**

**Breadth-First Search Implementation:**

* Represent the graph using an adjacency list.
* Create a queue to keep track of nodes to visit.
* Use a set to store visited nodes to avoid reprocessing.
* Start from a given node, add it to the queue, and explore its neighbors level by level.
* Continue the process until all reachable nodes are visited.

1. **IMPLEMENTATION**

**Breadth-First Search Code:**

from collections import deque

class BFS:

    def \_\_init\_\_(self, graph, source):

        self.graph = graph

        self.source = source

        for row in self.graph:

            print(row)

        print("Source node is: ",self.source)

    def st\_bfs(self):

        queue = deque()

        visited\_node = set()

        queue.append(self.source)

        visited\_node.add(self.source)

        print("BFS Traverse:", self.source, end=" ")

        while queue:

            node = queue.popleft()

            for i in range(len(self.graph[node])):

                if self.graph[node][i] == 1 and i not in visited\_node:

                    queue.append(i)

                    visited\_node.add(i)

                    print(i, end=" ")

graph = [

    [0, 1, 1, 0, 0, 0],

    [0, 0, 1, 1, 0, 0],

    [0, 0, 0, 1, 0, 0],

    [0, 0, 0, 0, 1, 0],

    [1, 1, 0, 0, 0, 1],

    [0, 0, 0, 0, 0, 0]

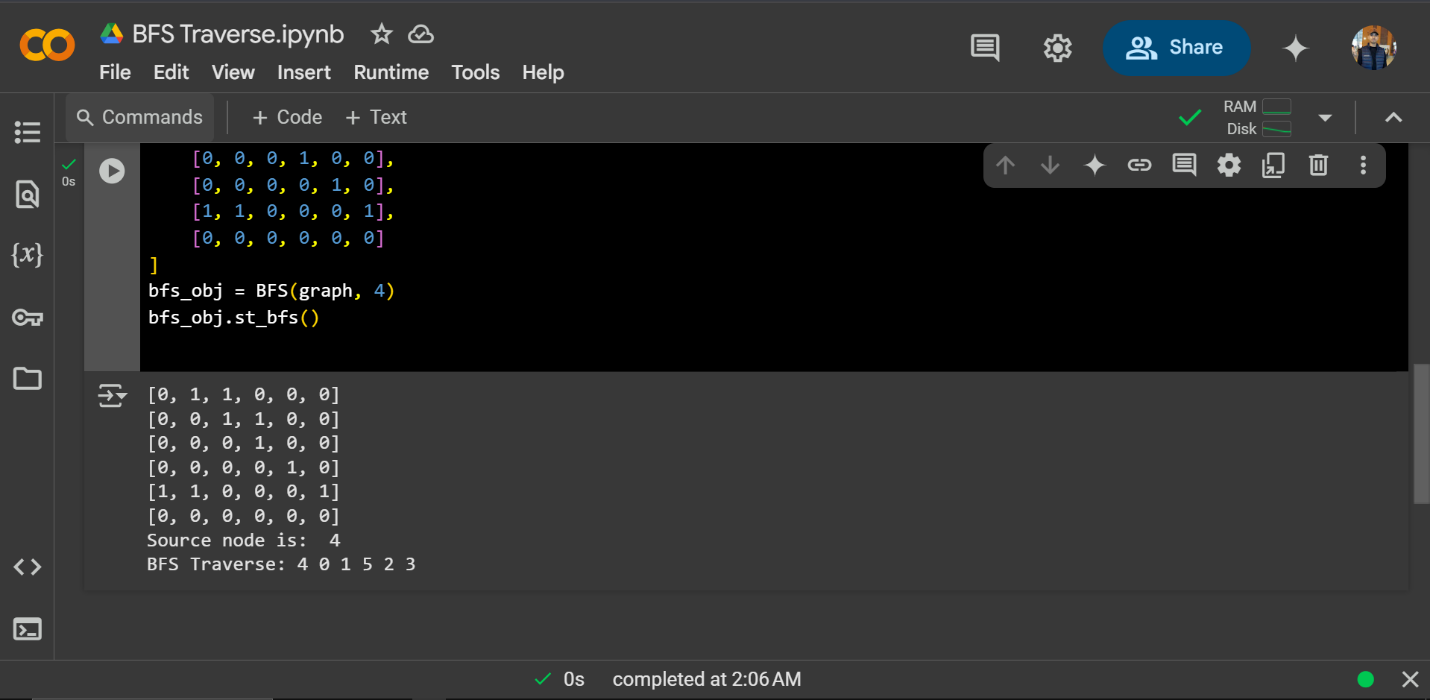
]

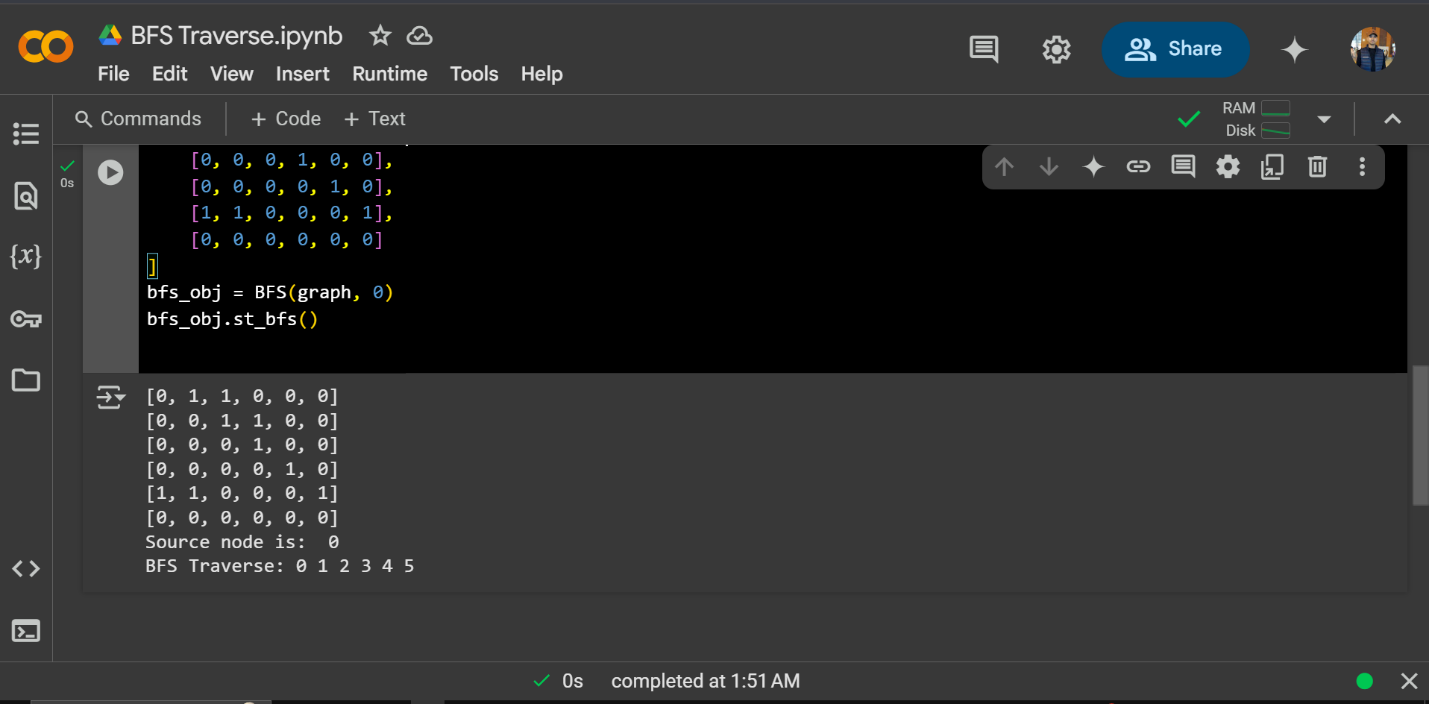
bfs\_obj = BFS(graph, 0)

bfs\_obj.st\_bfs()

**5. Test Result / Output**

**Breadth-First Search Output:**





**6. ANALYSIS AND DISCUSSION**

* The BFS algorithm effectively explores the graph level by level using a queue.
* A set is used to track visited nodes, preventing infinite loops.
* The algorithm ensures that each node is processed once, making it efficient with a time complexity of O(V + E), where V is the number of vertices and E is the number of edges.
* The BFS traversal for the given graph starting from node 0 results in the sequence: 0 2 1 3 4 5.

**6. SUMMARY:**

This experiment demonstrated the implementation of the Breadth-First Search (BFS) algorithm in Python. By representing a graph using an adjacency list and using a queue for traversal, we efficiently explored the nodes. This experiment helped in understanding graph traversal techniques and improving problem-solving skills in handling directed graphs in Python.