# AI LAB TASK 6

## 1. BFS without Queue & without Node

**Initialization**:

A `visited` set is used to track nodes that have been visited.

`current_level` is a list that starts with the `start_node` and keeps track of the nodes to be explored at the current level.

**Outer loop**:

While `current_level` has nodes, it explores them.

For each node in `current_level`, it prints the node and checks its neighbors.

**Neighbor exploration**:

If a neighbor hasn't been visited, it is added to the `visited` set and appended to the `next_level` list (which will become the new set of nodes to explore in the next iteration).

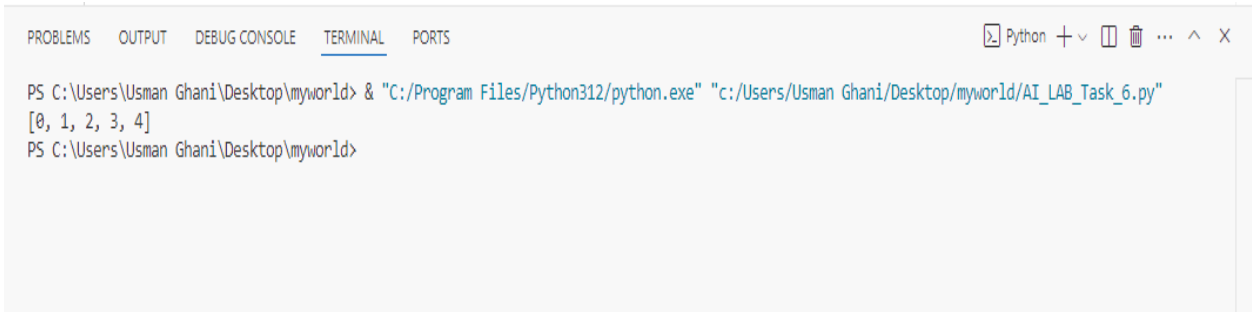
**Next level**:

After processing all nodes in `current_level`, the list `current_level` is updated to `next_level`, which holds all unvisited nodes for the next BFS level.

**Termination**:

- The process continues until there are no more nodes to explore (i.e., when `current_level` becomes empty).

This approach simulates BFS without using a queue by manually managing the levels using lists (`current_level` and `next_level`).



```
PS C:\Users\Usman Ghani\Desktop\myworld> & "C:/Program Files/Python312/python.exe" "c:/Users/Usman Ghani/Desktop/myworld/AI_LAB_Task_6.py"
[0, 1, 2, 3, 4]
PS C:\Users\Usman Ghani\Desktop\myworld>
```

## 2. BFS with Queue & Node

**Node Class**:

`Node` object has a `value` (node identifier) and a list of `neighbors` (adjacent nodes). `add_neighbor()` method adds a neighboring node to the `neighbors` list.

**BFS Function** (`bfs_with_queue_and_node`):

**Visited Set**: Tracks which nodes have been visited.

**Queue**: A deque is used to store nodes to explore. Initially, it starts with the `start_node`.

**BFS Traversal**:

The loop continues until the queue is empty.

The node at the front of the queue is dequeued and processed. If it hasn't been visited, its value is printed, and it is added to the `visited` set.

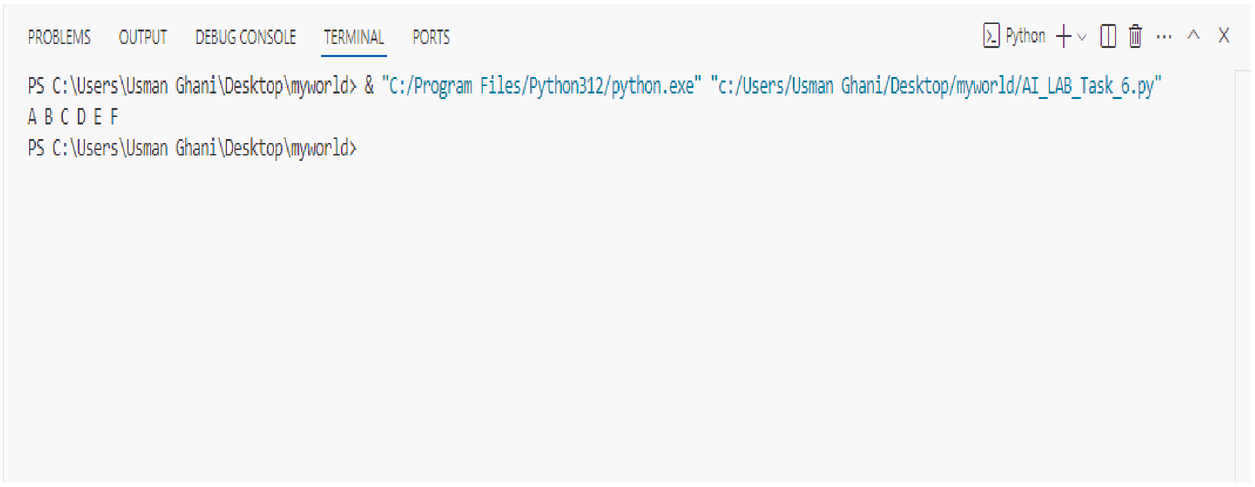
The unvisited neighbors of the current node are enqueued for future exploration.

**Graph Setup**:

Nodes are created (A, B, C, D, E, F), and edges (connections between nodes) are established using `add_neighbor()`.

**Execution**:

BFS is initiated from node `A`, and the algorithm explores all connected nodes, printing them in BFS order.



```
PS C:\Users\Usman Ghani\Desktop\myworld> & "C:/Program Files/Python312/python.exe" "c:/Users/Usman Ghani/Desktop/myworld/AI_LAB_Task_6.py"
A B C D E F
PS C:\Users\Usman Ghani\Desktop\myworld>
```