**Artificial Intelligent (Lab)**

**Task # 05**

**A purple circle with dots and lines

AI-generated content may be incorrect.**

**Submitted To: Sir Rasikh Ali**

**Submitted By: Shumaila Maryam**

**Roll no: SU92-BSDSM-F24-062**

**Submitted On: 25-September-25**

**Section: BSDS-3A**

**Department of Software Engineering, Superior University, Lahore**

**Question # 01:**

**BFS with Queue**

**Introduction:**  
This Python program implements the Breadth-First Search (BFS) algorithm to find a goal node within a graph. BFS systematically explores a graph by visiting all of a node's neighbors before moving to the next level, making it ideal for finding the shortest path in unweighted graphs. This program visualizes the traversal by printing each step of the process.

**Why I Made This:**  
The purpose of building this program was to:

* Gain a practical understanding of graph traversal algorithms.
* Learn the specific mechanics of the Breadth-First Search strategy.
* Practice implementing a queue data structure to manage the search order.
* Understand the role of a visited list in preventing redundant checks and infinite loops.
* Create a clear and reusable tool for visualizing how BFS explores a graph.

**How It Works:**

**Step 1: Initialization**

1. The program begins with a graph represented as a dictionary.
2. A visited list is created to store nodes that have already been processed.
3. A queue is created, which acts as a to-do list for nodes to visit. The start node is added to it first.

**Step 2: Level-by-Level Traversal**

1. The program iterates through each node in the queue.
2. For each node, it first checks if it is in the visited list. If not, the node is marked as visited and its name is printed.
3. The program then checks if the current node is the goal. If it is, a success message is printed, and the search concludes.

**Step 3: Queueing the Neighbors**

1. If the current node is not the goal, the program gets its list of neighbors (children) from the graph.
2. It loops through the neighbors and adds each one that has *not* already been visited to the end of the queue.
3. This process ensures that the search expands outward one level at a time.

**Summary:**  
This program provides a clear demonstration of the Breadth-First Search algorithm. By using a queue to manage the order of exploration, it guarantees that the graph is traversed in a level-by-level fashion. This implementation is a foundational example for understanding graph theory and is essential for solving problems related to network analysis, pathfinding, and web crawling.

**Advantages and Disadvantages:**

**Advantages:**

* **Finds the Shortest Path:** In an unweighted graph, BFS is guaranteed to find the shortest path from the start node to the goal node.
* **Completeness:** If a solution exists, BFS will always find it. It will not get stuck in an infinite path like a simple Depth-First Search might.

**Disadvantages:**

* **High Memory Usage:** BFS requires storing all nodes at the current level in the queue. For a graph with a large number of nodes at each level (a high branching factor), this can consume a significant amount of memory.
* **Can Be Slow:** If the goal node is located very deep within the graph, BFS can be slower than other algorithms because it must first explore every node at every level before it.

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.