**Practical No. 4 Breadth First Search**

**Objective:**

Write a Program to Implement Breadth First Search using Python.

# Practical Significance

Breadth-First Search (BFS) has practical significance in various fields and applications due to its unique characteristics. Here are some practical applications:

Network Routing and Broadcasting:

In computer networks, BFS is often used to discover neighboring nodes and determine the shortest path for routing.

It is also employed in broadcasting information across a network efficiently.

Web Crawling:

Search engines use BFS to crawl the web and index pages. Starting from a seed page, BFS explores links level by level, ensuring a systematic and comprehensive traversal.

Puzzle Solving:

BFS is used in puzzle-solving scenarios, such as the famous "Eight Puzzle" or "Fifteen Puzzle," to find the shortest sequence of moves to reach the goal state.

Maze Solving:

BFS can be applied to solve mazes by finding the shortest path from the start to the exit. It guarantees the discovery of the shortest path when the maze has uniform edge weights.

Robotics and Autonomous Vehicles:

BFS is employed in robotics and autonomous vehicle navigation to explore and map unknown environments systematically.

Optimizing Data Structures:

BFS is often used in optimizing data structures like trees and graphs, ensuring efficient access and retrieval of information.

Game Development:

BFS can be applied in game development for tasks such as pathfinding, where it helps in finding the shortest path for characters or objects.

Database Querying:

BFS is used in certain database querying scenarios to explore relationships and dependencies between different entities.

In summary, BFS is a versatile algorithm with practical applications across various domains, providing an efficient way to explore and analyze relationships in interconnected systems.

# Minimum Theoretical Background

**BFS Algorithm:**

Input:

Graph G represented as an adjacency list, starting vertex start, and goal vertex goal.

Initialization:

Create an empty set visited to keep track of visited vertices. Create a deque queue and enqueue the start vertex.

Add the start vertex to the visited set.

BFS Loop:

While the queue is not empty:

Dequeue a vertex current\_vertex from the front of the queue. Print or process current\_vertex.

If current\_vertex is equal to the goal vertex:

Print a message indicating that the goal state is reached. Return, indicating that the goal state is reached.

For each neighbor neighbor of current\_vertex in the graph: If neighbor is not in the visited set:

Enqueue neighbor to the back of the queue. Add neighbor to the visited set.

Output: Print a message indicating that the goal state is not reached if the loop completes without returning.

# Exercise

* 1. Write a Program to Implement Breadth First Search without goal state using Python.
  2. Write a Program to Implement Breadth First Search with goal state using Python.