The Breadth-First Search (BFS) algorithm is a graph traversal method used to explore nodes and edges of a graph in a layer-by-layer manner. It is commonly used in applications like finding the shortest path in an unweighted graph, solving puzzles, and traversing trees.

**How BFS Works:**

1. **Initialization**:
   * Start with a source node and mark it as visited.
   * Use a queue data structure to keep track of nodes to explore.
2. **Processing**:
   * Dequeue a node from the front of the queue.
   * Explore all its unvisited neighbors, marking them as visited and enqueuing them.
3. **Repeat**:
   * Continue the process until the queue is empty.

**Steps of BFS:**

1. Start with the source node.
2. Mark the source node as visited and enqueue it.
3. While the queue is not empty:
   * Dequeue a node, process it, and enqueue all its unvisited neighbors.
4. End when all reachable nodes are visited.

**BFS Pseudocode:**

def bfs(graph, start):

visited = set() # Set to keep track of visited nodes

queue = [] # Initialize a queue

queue.append(start) # Enqueue the start node

visited.add(start) # Mark start as visited

while queue:

node = queue.pop(0) # Dequeue a node

print(node) # Process the node (e.g., print it)

# Explore neighbors

for neighbor in graph[node]:

if neighbor not in visited:

queue.append(neighbor)

visited.add(neighbor)

**Characteristics of BFS:**

* **Time Complexity**:
  + O(V+E)O(V + E)O(V+E), where VVV is the number of vertices and EEE is the number of edges.
* **Space Complexity**:
  + O(V)O(V)O(V), for the queue and visited list.
* **Type**:
  + BFS is a complete and level-order traversal in graphs or trees.

**Example:**

For a graph:

mathematica

Copy code

A: [B, C]

B: [A, D, E]

C: [A, F]

D: [B]

E: [B]

F: [C]

If starting at A, BFS traversal would visit nodes in this order:

Copy code

A -> B -> C -> D -> E -> F