Breadth-First Search (BFS) Algorithm

General Explanation of BFS:

BFS is a fundamental graph traversal algorithm used to explore nodes and edges of a graph in a layer-wise manner. The algorithm starts at a given node (the root) and explores all of its neighboring nodes. Then it moves to the next layer and continues this process until all nodes have been visited, or a particular goal has been found.

Steps of BFS:

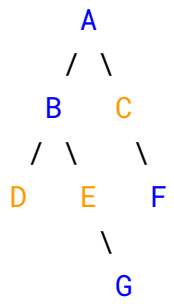
1. **Start with a queue** that holds the starting node.
2. **Mark the node as visited** and add it to the queue.
3. **While the queue is not empty**, perform the following:
    - Dequeue the front node.
    - Check if this node is the goal. If yes, return the solution.
    - For each of its neighboring nodes, if they have not been visited, mark them as visited and enqueue them.
4. **Repeat** this process until the queue is empty or the goal is found.

BFS Characteristics:

- **Time complexity**: O(V+E), where V is the number of vertices (nodes), and E is the number of edges.
- **Space complexity**: O(V) due to the storage of nodes in the queue.

Example Diagram for BFS:

Let's assume a graph where BFS starts from node A and explores the following nodes in this order:



1. **Queue**: Initially contains just the root node A.
2. **First Layer**: A is explored, and its neighbors B and C are added to the queue.
3. **Second Layer**: B and C are dequeued one by one, and their neighbors (D, E, F) are added.

4. **Third Layer**: The algorithm continues by exploring D, E, F, and finally G.

Applying BFS to the Water Jug Problem

The app\Algorithms\WaterJugBSFSolver.php file implements BFS to solve the Water Jug Problem, where you need to measure a specific amount of water using two jugs of given capacities. The algorithm explores all possible states of the two jugs and actions (fill, empty, transfer) until it finds a solution or determines there is no solution.

Problem Overview:

- Two water jugs with capacities X liters and Y liters.
- You need to measure exactly Z liters.

The algorithm:

1. Starts with both jugs empty.
2. Tries all possible actions (fill, empty, transfer) to transition from one state to another.
3. Uses BFS to explore each new state layer by layer (all possible jug states are considered).
4. Continues this process until the exact Z liters is found in one of the jugs.

Key Components in the PHP Code:

1. **Queue**: `$this->queue[] = [0, 0, []];` – The queue starts with both jugs being empty.
2. **State Transitions**: The algorithm attempts actions like filling, emptying, and transferring water between the jugs. For each state, it checks whether this new state has been visited or not.
3. **Termination**: If either jug reaches the desired amount, the BFS terminates and returns the series of actions leading to the solution.

How BFS Works in the Code:

1. **Initial State**: `[0, 0]` – both jugs are empty.
2. **Explore States**:
    - Fill jug X.
    - Fill jug Y.
    - Empty jug X.
    - Empty jug Y.
    - Transfer water from jug X to jug Y.
    - Transfer water from jug Y to jug X.

3. Each of these actions leads to a new state that is added to the queue.
4. **Termination Condition**: If at any point one of the jugs holds exactly Z liters, the algorithm stops and outputs the solution.

Diagram for BFS in the Water Jug Problem:

```
Initial State: (0, 0) - Both jugs empty
|
|-- Fill Jug X: (X, 0)
|-- Fill Jug Y: (0, Y)
|-- Transfer X -> Y: (X-a, a)
|-- Transfer Y -> X: (a, Y-a)
|
Repeat for each new state until Z liters is found in either jug.
```

Each action represents a transition to a new state in the graph of possible water configurations, and BFS ensures that the shortest series of actions is found.