**Breadth First Search or BFS for a Graph**

**Breadth First Search (BFS)**is a fundamental**graph traversal algorithm**. It involves visiting all the connected nodes of a graph in a level-by-level manner. In this article, we will look into the concept of **BFS**and how it can be applied to graphs effectively

## Breadth First Search (BFS) for a Graph:

**Breadth First Search (BFS)** is a graph traversal algorithm that explores all the vertices in a graph at the current depth before moving on to the vertices at the next depth level. It starts at a specified vertex and visits all its neighbors before moving on to the next level of neighbors. **BFS** is commonly used in algorithms for pathfinding, connected components, and shortest path problems in graphs.

## Relation between BFS for Graph and BFS for Tree:

Breadth-First Traversal (BFS) for a graph is similar to the [Breadth-First Traversal of a tree](https://www.geeksforgeeks.org/level-order-tree-traversal/).

The only catch here is, that, unlike **trees**, **graphs** may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we divide the vertices into two categories:

* Visited and
* Not visited.

A boolean visited array is used to mark the visited vertices. For simplicity, it is assumed that all vertices are reachable from the starting vertex. BFS **uses** a [**queue data structure**](https://www.geeksforgeeks.org/queue-data-structure/) for traversal.

## Breadth First Search (BFS) for a Graph Algorithm:

Let’s discuss the algorithm for the BFS:

1. **Initialization:**Enqueue the starting node into a queue and mark it as visited.
2. **Exploration:**While the queue is not empty:
   * Dequeue a node from the queue and visit it (e.g., print its value).
   * For each unvisited neighbor of the dequeued node:
     + Enqueue the neighbor into the queue.
     + Mark the neighbor as visited.
3. **Termination:** Repeat step 2 until the queue is empty.

This algorithm ensures that all nodes in the graph are visited in a breadth-first manner, starting from the starting node.

## How Does the BFS Algorithm Work?

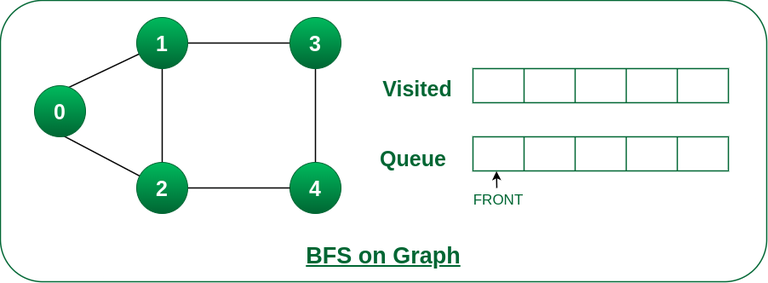
Starting from the root, all the nodes at a particular level are visited first and then the nodes of the next level are traversed till all the nodes are visited.

To do this a queue is used. All the adjacent unvisited nodes of the current level are pushed into the queue and the nodes of the current level are marked visited and popped from the queue.

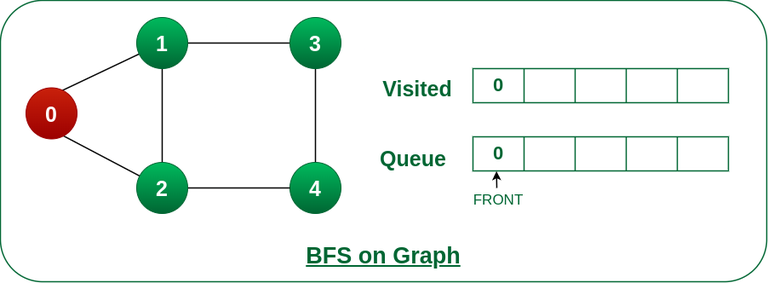
**Illustration:**

Let us understand the working of the algorithm with the help of the following example.

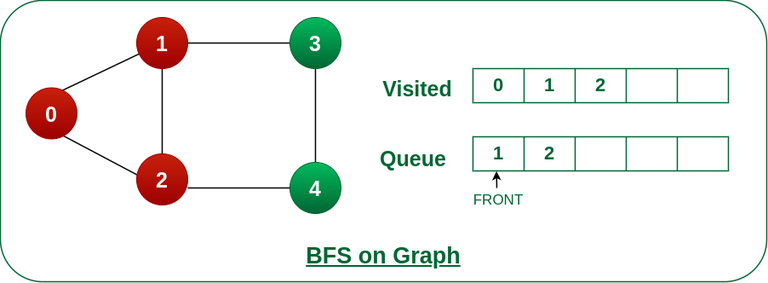
***Step1:****Initially queue and visited array empty.*

**

*Queue and visited arrays are empty initially.*

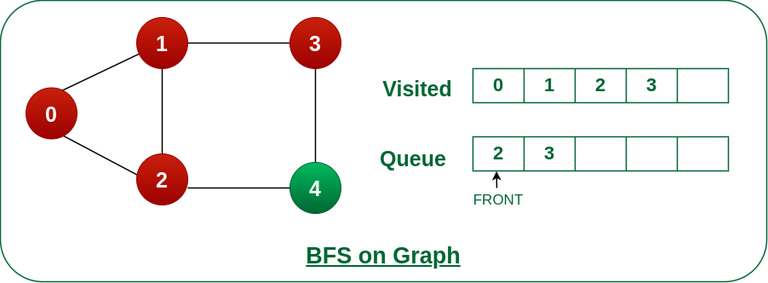
**Step**2: Push node 0 into queue and mark it visited**

*Push node 0 into queue and mark it visited.*

*****Step 3*** *:Remove node 0 from the front of queue and visit the unvisited neighbours and push them into queue.*

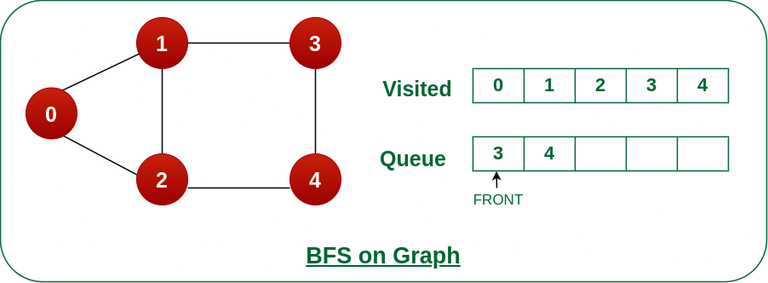
*Remove node 0 from the front of queue and visited the unvisited neighbours and push into queue.*

***Step 4:****Remove node 1 from the front of queue and visit the unvisited neighbours and push them into queue.*

**

*Remove node 1 from the front of queue and visited the unvisited neighbours and push*

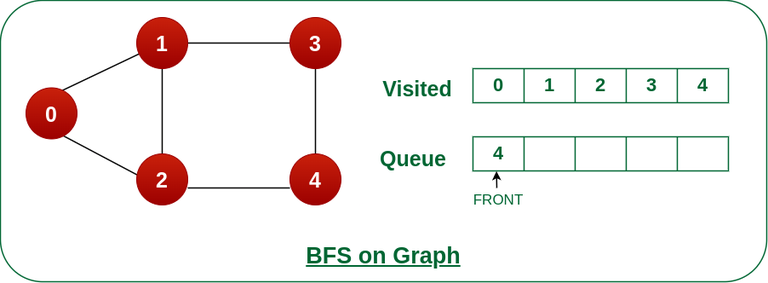
***Step 5:****Remove node 2 from the front of queue and visit the unvisited neighbours and push them into queue.*

**

*Remove node 2 from the front of queue and visit the unvisited neighbours and push them into*

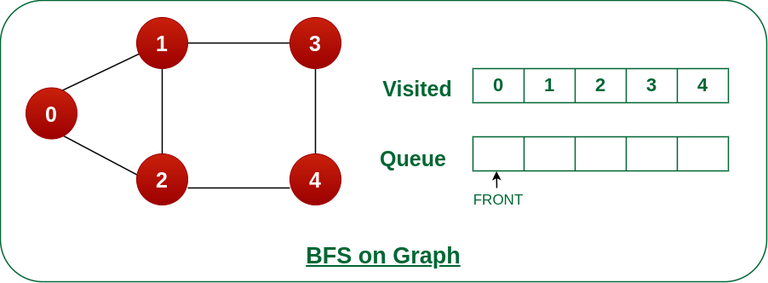
***Step 6:****Remove node 3 from the front of queue and visit the unvisited neighbours and push them into queue.   
As we can see that every neighbours of node 3 is visited, so move to the next node that are in the front of the queue.*

*neighbours of node 3 is visited, so move to the next node that are in the front of the queue.*

**

*Remove node 3 from the front of queue and visit the unvisited neighbours and push them into queue.*

***Steps 7:****Remove node 4 from the front of queue and visit the unvisited neighbours and push them into queue.   
As we can see that every neighbours of node 4 are visited, so move to the next node that is in the front of the queue.*

**

*Remove node 4 from the front of queue and visit the unvisited neighbours and push them into queue.*

*Now, Queue becomes empty, So, terminate these process of iteration.*

***Algorithm:***

*BFS Algorithm Steps*

*Initialize:*

*Set STATUS of all nodes to 1 (ready state).*

*Enqueue Starting Node:*

*Start BFS by enqueueing the starting node A.*

*Set its STATUS to 2 (waiting state).*

*Process Nodes:*

*While the queue is not empty:*

*Dequeue a node N.*

*Process N (e.g., print it, store it, etc.).*

*Set its STATUS to 3 (processed state).*

*Enqueue Neighbors:*

*For each neighbor M of N:*

*If M is in ready state (STATUS = 1):*

*Enqueue M.*

*Set M's STATUS to 2 (waiting state).*

*Repeat steps 3 and 4 until the queue is empty.*

*Exit:*

*End the algorithm once all nodes have been processed.*

*Explanation*

*Status States:*

*STATUS = 1: Ready state (node is ready to be processed).*

*STATUS = 2: Waiting state (node is in the queue waiting to be processed).*

*STATUS = 3: Processed state (node has been processed).*