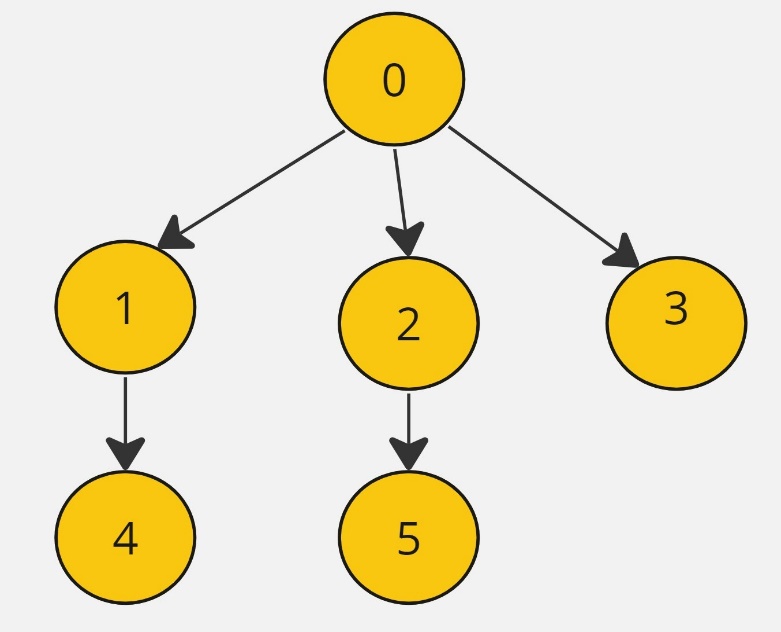
Breadth First Search Traversal of Graph [CodeStudio](https://www.codingninjas.com/studio/problems/bfs-in-graph_973002)

The program employs Breadth-First Search (BFS) traversal to explore the graph efficiently, even when there are disconnected components. The bfsOfGraph function initializes a queue for BFS, marking visited nodes and exploring neighbors in a level-wise manner.

Example:



Output: 0, 1, 2, 3, 4, 5

**AddEdge function:**

* **Purpose:**
  + Populates the graph's adjacency list based on the provided edge list.
* **Explanation:**
  + Iterates through each edge in the **edges** vector.
  + For each edge, extracts the source vertex **u** and iterates over the connected vertices.
  + Adds an edge from **u** to **v** in the adjacency list.
  + If the graph is undirected, adds an edge from **v** to **u** as well.
* **Time Complexity:**
  + **O(E), where E is the number of edges in the input vector.**
* **Space Complexity:**
  + **O(E), where E is the number of edges. Each edge results in the creation of one or two entries in the adjacency list.**

**Approach 1: Function to print the BFS Traversal of the Graph**

* **Purpose:**
  + Performs Breadth-First Search (BFS) traversal on the graph.
* **Explanation:**
  + Initializes a visited map to track visited nodes.
  + Initializes a **bfsTraversal** vector to store the result of BFS traversal.
  + Initializes a queue (**bfsQueue**) for BFS.
  + Iterates through each node in the graph (necessary for handling disconnected components).
  + For each unvisited node, starts BFS from that node.
  + In the BFS loop, explores neighbors of the current node, adding them to the queue if not visited.
  + Constructs a temporary vector (**temp**) representing the BFS traversal component.
  + Adds the component to the overall result.
* **Time Complexity:**
  + **O(V + E), where V is the number of vertices and E is the number of edges. Accounts for the traversal of all vertices and edges, even in the presence of disconnected components.**
* **Space Complexity:**
  + **O(V + E), where V is the number of vertices and E is the number of edges. Includes space for the visited map, BFS traversal result vector, and queue.**

**Overall Time and Space Complexity:**

* **Overall Time Complexity:**
  + **Using addEdge: O(V + E) for both connected graphs and graphs with disconnected components.**
  + **Without addEdge (only BFS): O(V) for the BFS traversal alone.**
* **Overall Space Complexity:**
  + **O(V + E), considering the adjacency list, visited map, and auxiliary data structures in the bfsOfGraph function.**