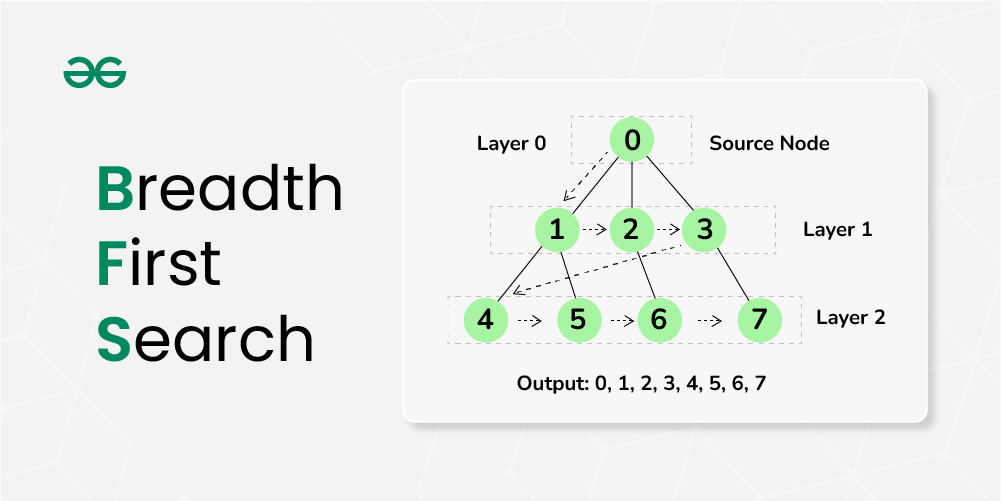
Kodirov Bekhzod Group A – 85476549

Asude Ergüven Group A - 65365616

Parallel BFS Project Report

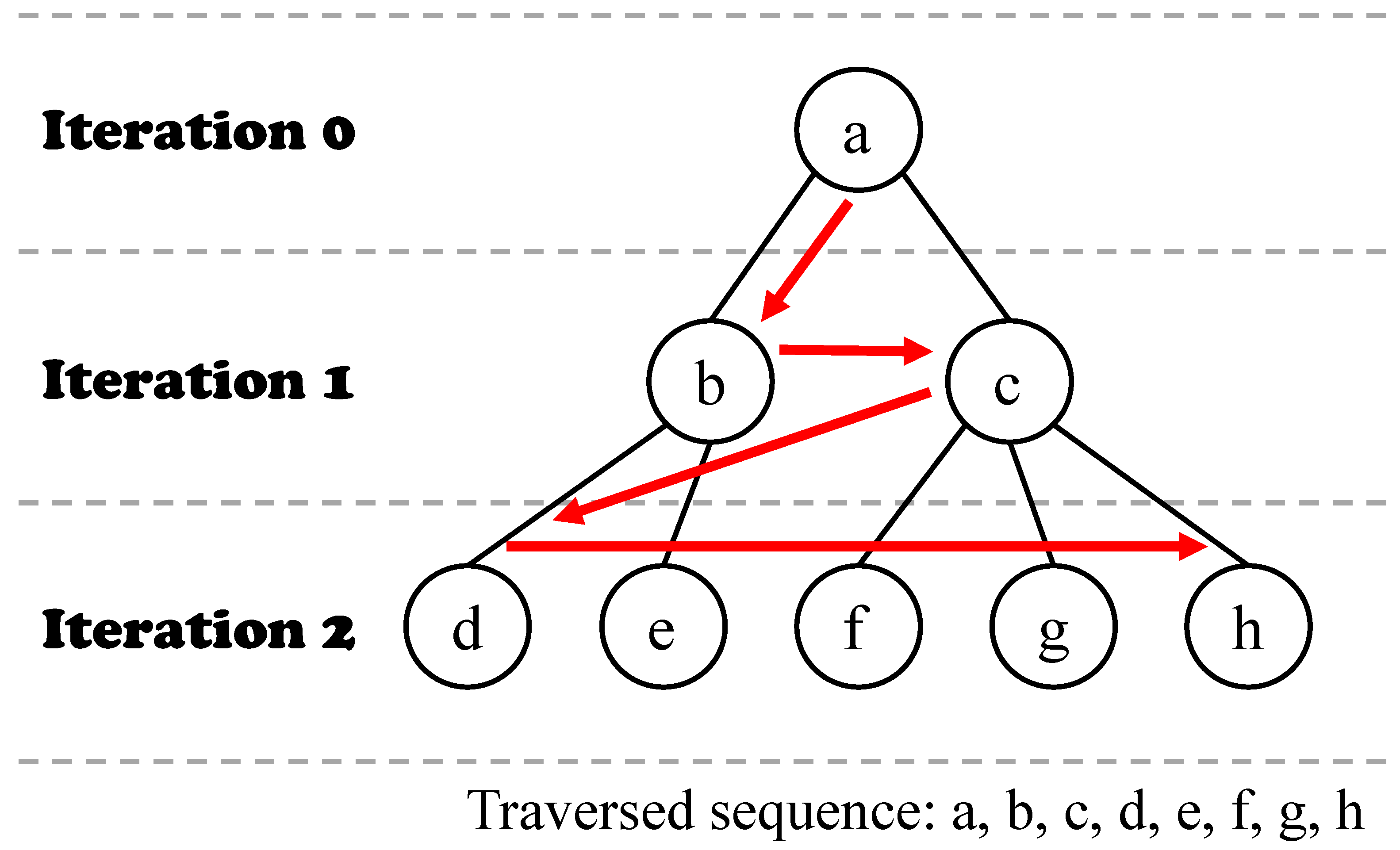
# 1. Explain the Algorithm

Breadth-First Search (BFS) is a graph traversal algorithm used to explore nodes and edges of a graph starting from a specific node. It works by visiting all the nodes at the current depth level before moving on to nodes at the next depth level. BFS uses a queue to keep track of nodes to be explored, ensuring that the closest nodes are visited first.



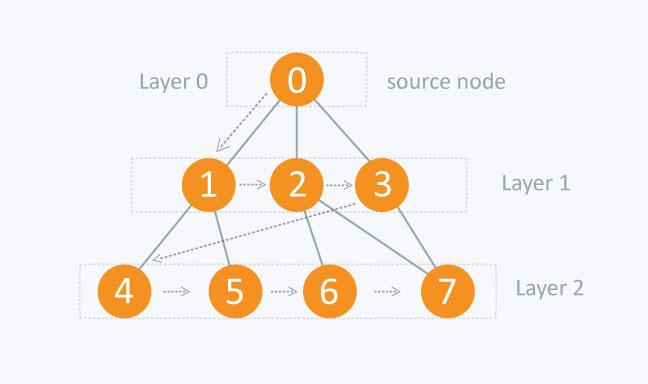
# 2. How the Code Runs in a Serial Way

In the serial implementation of BFS, we start from a given node, mark it as visited, and enqueue it. We then enter a loop where we dequeue the front node, visit all its unvisited neighbors, mark them as visited, and enqueue them. This process continues until the queue is empty, ensuring that all nodes at the current level are processed before moving to the next level.



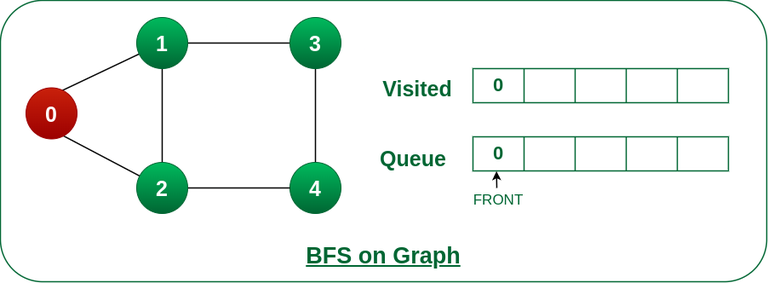
# 3. How the Code Runs in a Parallel Way

In the parallel implementation of BFS using OpenMP, we start similarly by initializing a queue with the start node and marking it as visited. However, instead of processing nodes one by one, we divide the work among multiple threads. Each thread processes a subset of the nodes at the current level, visiting their neighbors and marking them as visited. Local queues are used by each thread to store the nodes to be visited in the next level. At the end of each level, these local queues are merged back into the global queue. This approach allows multiple nodes to be processed simultaneously, speeding up the traversal.

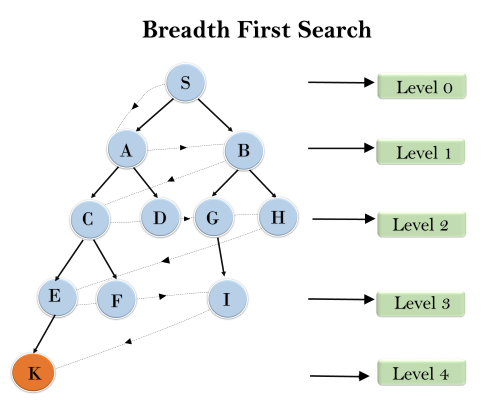


**Steps in Parallel BFS:**

1. **Initialization**:
   * Enqueue the start node A to the global queue.
   * Mark A as visited.
   * Example: Queue: [A]



1. **Iteration 1** (Parallel Processing):
   * All threads collaborate to process nodes at the current level.
   * Thread 1 dequeues A, visits it, and enqueues its neighbors B and C to its local queue.
   * Example: Thread 1 Local Queue: [B, C]
2. **Merge Local Queues**:
   * Merge all local queues back into the global queue.
   * Example: Global Queue: [B, C]
3. **Iteration 2** (Parallel Processing):
   * Threads process nodes B and C in parallel.
   * Thread 1 dequeues B, visits it, and enqueues its neighbors D and E to its local queue.
   * Thread 2 dequeues C, visits it, and enqueues its neighbors F and G to its local queue.
   * Example: Thread 1 Local Queue: [D, E], Thread 2 Local Queue: [F, G]
4. **Merge Local Queues**:
   * Merge all local queues back into the global queue.
   * Example: Global Queue: [D, E, F, G]
5. **Repeat**:
   * Continue this process until the global queue is empty.



# 4. Critical Sections and Synchronization

In the parallel BFS implementation, critical sections are used to protect shared resources, such as the global queue and the visited nodes array, from concurrent access by multiple threads. The synchronization ensures that no two threads can modify these resources at the same time, preventing data races. OpenMP directives like ***#pragma omp critical*** are used to define these critical sections. The method used to parallelize the work involves dividing the nodes among multiple threads, allowing each thread to process a portion of the graph simultaneously, and then merging the results.