****

**Recorrido BFS**

El recorrido BFS comienza en el nodo 0 y explora todos los nodos vecinos antes de moverse a los vecinos de los vecinos. El orden de visita sería:

1. Comenzamos en el nodo 0.
2. Visitamos los nodos 1 y 2 (conexiones desde 0).
3. Visitamos los nodos 3, 4, 5 y 6 (conexiones desde 1 y 2).
4. Visitamos los nodos 7 y 8 (conexiones desde 3 y 4).

El orden de visita sería: 0, 1, 2, 3, 4, 5, 6, 7, 8.

import java.util.\*;

public class BFS {

private int V; // Número de vértices

private LinkedList<Integer> adj[]; // Lista de adyacencia

BFS(int v) {

V = v;

adj = new LinkedList[v];

for (int i = 0; i < v; ++i)

adj[i] = new LinkedList();

}

void addEdge(int v, int w) {

adj[v].add(w); // Añadir w a la lista de v

}

void BFS(int s) {

boolean visited[] = new boolean[V];

LinkedList<Integer> queue = new LinkedList<Integer>();

visited[s] = true;

queue.add(s);

while (queue.size() != 0) {

s = queue.poll();

System.out.print(s + " ");

for (int n : adj[s]) {

if (!visited[n]) {

visited[n] = true;

queue.add(n);

}

}

}

}

public static void main(String args[]) {

BFS g = new BFS(9);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 3);

g.addEdge(1, 4);

g.addEdge(2, 5);

g.addEdge(2, 6);

g.addEdge(3, 7);

g.addEdge(4, 8);

System.out.println("Recorrido BFS comenzando desde el nodo 0:");

g.BFS(0);

}

}