****

**Recorrido DFS**

El recorrido DFS comienza en el nodo 0 y explora lo más profundo posible a lo largo de cada rama antes de retroceder. El orden de visita sería:

1. Comenzamos en el nodo 0.
2. Visitamos el nodo 1 (conexión desde 0).
3. Visitamos el nodo 3 (conexión desde 1).
4. Visitamos el nodo 7 (conexión desde 3).
5. Retrocedemos al nodo 1 y visitamos el nodo 4.
6. Visitamos el nodo 8 (conexión desde 4).
7. Retrocedemos al nodo 0 y visitamos el nodo 2.
8. Visitamos el nodo 5 (conexión desde 2).
9. Visitamos el nodo 6 (conexión desde 2).

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

import java.util.\*;

public class DFS {

private int V; // Número de vértices

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

DFS(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 DFSUtil(int v, boolean visited[]) {

visited[v] = true;

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

for (int n : adj[v]) {

if (!visited[n])

DFSUtil(n, visited);

}

}

void DFS(int v) {

boolean visited[] = new boolean[V];

DFSUtil(v, visited);

}

public static void main(String args[]) {

DFS g = new DFS(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 DFS comenzando desde el nodo 0:");

g.DFS(0);

}

}

Recorrido DFS comenzando desde el nodo 0:

0 1 3 7 4 8 2 5 6