



# Guía de Actividades Práctico-Experimentales Nro. 008

## 1. Datos Generales

<b>Asignatura</b>	Estructura de datos
<b>Ciclo</b>	3 A
<b>Unidad</b>	3
<b>Resultado de aprendizaje de la unidad</b>	Entiende los principios básicos de las operaciones en árboles, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad.
<b>Título de la Práctica</b>	Búsqueda en Java: Secuencial y Binaria
<b>Nombre del Docente</b>	Andrés Roberto Navas Castellanos
<b>Fecha</b>	Jueves 15 de enero
<b>Horario</b>	07h30 – 10h30
<b>Lugar</b>	Aula
<b>Tiempo planificado en el Sílabo</b>	3 horas

## 2. Objetivo(s) de la Práctica:

- Implementar correctamente los algoritmos DFS (profundidad) y BFS (anchura) en Java.
- Verificar conectividad, niveles (distancias en no ponderados) y detección básica de ciclos, en grafos dirigidos y no dirigidos.

## 3. Materiales y reactivos:

- Datasets.

## 4. Equipos y herramientas

- JDK OpenJDK (obligatorio).
- IDE: Visual Studio Code (extensión “Extension Pack for Java”) o IntelliJ IDEA Community.
- Sistema de control de versiones: Git; repositorio en GitHub.
- EVA/Moodle institucional: para entrega de evidencias.
- Herramientas de documentación: README Markdown, editor ofimático (Google Docs/LibreOffice/Word).



## 5. Procedimiento / Metodología

Enfoque metodológico: ABPr (Aprendizaje Basado en Proyectos).

### Inicio

- Presentación del objetivo y criterios de éxito.
- Formación de equipos (3-4) y revisión de la rúbrica.
- Creación de repo Git.
- Lineamientos de uso responsable de IA.

### Desarrollo

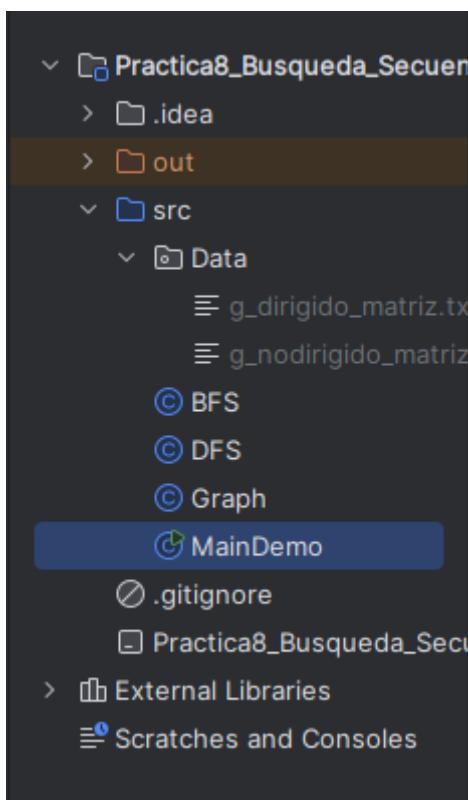
- Implementar ‘Graph’ con ‘nVertices’, bandera ‘directed’ y ‘List<List<Integer>> adj’. Cargar desde archivo y ‘addEdge’.
- BFS: ‘List<Integer> bfs(int s)’ con ‘Queue<Integer>’ (ArrayDeque), vector ‘dist[]’ y vector ‘parent[]’ opcional.
- DFS: ‘List<Integer> dfs(int s)’ recursivo u opción con ‘Stack<Integer>’. (Opcional: tiempos de entrada/salida).
- Probar en ‘g\_dirigido.txt’ y ‘g\_nodirigido.txt’ para distintos orígenes; registrar orden y niveles (BFS).
- Extender: componentes (no dirigido) o conjuntos alcanzables (dirigido) repitiendo recorridos desde no visitados.
- Plus: detectar ciclo (criterio no dirigido/dirigido).

### Cierre

- Comparar órdenes de visita de DFS y BFS.
- Discutir usos: BFS para caminos mínimos no ponderados; DFS para exploración y ciclos/topología (con variantes).

## 6. Resultados esperados:

- Código ‘Graph’, ‘BFS’, ‘DFS’, ‘MainDemo’ con ejecución reproducible.





**UNL**

Universidad  
Nacional  
de Loja  
1859

**FEIRNNR - Carrera de Computación**

```
import java.util.*;  
  
public class BFS { 2 usages  ↳ Ismael  
  
    private Graph graph;  3 usages  
    private int[] dist;  7 usages  
  
    public BFS(Graph graph) { 1 usage  ↳ Ismael  
        this.graph = graph;  
        this.dist = new int[graph.getNumVertices()];  
        Arrays.fill(dist, val: -1);  
    }  
  
    public List<Integer> traverse(int start) { 1 usage  ↳ Ismael  
        boolean[] visited = new boolean[graph.getNumVertices()];  
        Queue<Integer> queue = new ArrayDeque<>();  
        List<Integer> order = new ArrayList<>();  
  
        visited[start] = true;  
        dist[start] = 0;  
        queue.add(start);  
  
        while (!queue.isEmpty()) {  
            int u = queue.poll();  
            order.add(u);  
  
            for (int v : graph.getAdj(u)) {  
                if (!visited[v]) {  
                    visited[v] = true;  
                    dist[v] = dist[u] + 1;  
                    queue.add(v);  
                }  
            }  
        }  
        return order;  
    }  
}
```



**UNL**

Universidad  
Nacional  
de Loja  
1859

**FEIRNNR - Carrera de Computación**

```
import java.util.*;  
  
public class DFS {  
  
    private Graph graph;  
    private boolean[] visited;  
    private List<Integer> order;  
  
    public DFS(Graph graph) {  
        this.graph = graph;  
        visited = new boolean[graph.getNumVertices()];  
        order = new ArrayList<>();  
    }  
  
    public List<Integer> traverse(int start) {  
        dfsRecursive(start);  
        return order;  
    }  
  
    private void dfsRecursive(int u) {  
        visited[u] = true;  
        order.add(u);  
  
        for (int v : graph.getAdj(u)) {  
            if (!visited[v]) {  
                dfsRecursive(v);  
            }  
        }  
    }  
}
```





**UNL**

Universidad  
Nacional  
de Loja

1859

**FEIRNNR - Carrera de Computación**

===== MENÚ DE GRAFOS =====

1. Grafo Dirigido
2. Grafo No Dirigido
3. Nuevo Grafo
0. Salir

Seleccione una opción:

Seleccione una opción: **1**

Ingresar el vértice de origen (1 a 10): **1**

== RESULTADOS DESDE EL VÉRTICE 1 ==

BFS: [1, 2, 3, 9, 5, 6, 7, 10, 8]

Distancias BFS:

Vértice 1 -> 0  
Vértice 2 -> 1  
Vértice 3 -> 1  
Vértice 4 -> -1  
Vértice 5 -> 2  
Vértice 6 -> 2  
Vértice 7 -> 2  
Vértice 8 -> 3  
Vértice 9 -> 1  
Vértice 10 -> 2

DFS: [1, 2, 3, 7, 5, 10, 8, 9, 6]

== RESULTADOS DESDE EL VÉRTICE 1 ==

BFS: [1, 2, 4, 8, 10, 5, 6, 3, 9, 7]

Distancias BFS:

Vértice 1 -> 0  
Vértice 2 -> 1  
Vértice 3 -> 2  
Vértice 4 -> 1  
Vértice 5 -> 2  
Vértice 6 -> 2  
Vértice 7 -> 2  
Vértice 8 -> 1  
Vértice 9 -> 2  
Vértice 10 -> 1

DFS: [1, 2, 5, 4, 3, 6, 7, 10, 8, 9]

**LINK GIT:**

[https://github.com/Mzero11/Practica8\\_Busqueda\\_Secuencial\\_Binaria](https://github.com/Mzero11/Practica8_Busqueda_Secuencial_Binaria)



**UNL**

Universidad  
Nacional  
de Loja  
1859

**FEIRNNR - Carrera de Computación**

- Tabla con orden de visita y dist[] de BFS por dataset y origen.

**GRAFO DIRIGIDO**

ORDEN:

1,2,3,9,5,6,7,10,8

Vértice	Ordes BFS	Distancia
<b>1</b>	1	0
<b>2</b>	2	1
<b>3</b>	3	1
<b>4</b>	-	-1
<b>5</b>	5	2
<b>6</b>	6	2
<b>7</b>	7	2
<b>8</b>	9	3
<b>9</b>	4	1
<b>10</b>	8	2

**GRAFO NO DIRIGIDO**

ORDEN:

1,2,4,8,10,5,6,3,9,7

Vértice	Ordes BFS	Distancia
<b>1</b>	1	0
<b>2</b>	2	1
<b>3</b>	8	2
<b>4</b>	3	1
<b>5</b>	6	2
<b>6</b>	7	2
<b>7</b>	10	2
<b>8</b>	4	1
<b>9</b>	9	2
<b>10</b>	5	1

- Capturas de salida de consola; breve informe (1–2 págs.).



**UNL**

Universidad  
Nacional  
de Loja  
1859

**FEIRNNR - Carrera de Computación**

## 7. Preguntas de Control:

- **¿Por qué BFS entrega distancias mínimas en aristas no ponderadas? ¿Qué cambia si hay pesos?**

Porque visita los nodos por niveles, y cada arista tiene el mismo costo.

BFS ya no sirve; se usa **Dijkstra** u otro algoritmo de caminos mínimos.

- **¿Qué estructuras auxiliares necesita cada algoritmo y por qué?**

**BFS** Cola y arreglo de visitados.

**DFS** Pila (o recursión) y arreglo de visitados.

- **¿Cómo obtener componentes conexas con DFS/BFS?**

Ejecutar DFS o BFS desde cada nodo no visitado; cada ejecución es una componente.

- **¿Cómo detectas ciclo en no dirigidos con DFS? ¿Y en dirigidos?**

Si se visita un nodo ya visitado que no es el padre, hay ciclo.

Si durante DFS se visita un nodo en la pila de recursión, hay ciclo.

- **¿Cuándo preferirías matriz de adyacencia en lugar de listas?**

En grafos pequeños o densos o cuando se necesita saber rápido si existe una arista.



**UNL**

Universidad  
Nacional  
de Loja  
1859

FEIRNNR - Carrera de Computación

## 8. Evaluación

Criterio	4 – Excelente	3 – Bueno	2 – Básico	1 – Insuficiente	Pts
<b>BFS</b>	Correcto; orden y dist[] válidos; código claro	Funciona con detalles menores	Parcial/errores	No funcional	3.0
<b>DFS</b>	Correcto; rec/iter; ciclo básico	Funciona con detalles menores	Parcial/errores	No funcional	3.0
<b>Pruebas evidencias</b> /	Tablas completas + capturas	Aceptables	Escasas	Nulas	2.0
<b>Informe conclusiones</b> /	Claras: cuándo usar DFS/BFS	Aceptables	Superficiales	Nulas	1.5
<b>Calidad código</b>	Organizado; README breve	Aceptable	Pobre	Deficiente	0.5

## 9. Bibliografía

[1] OpenDSA Project, "Graph Traversals: BFS and DFS," Virginia Tech, 2021–2024.

[2] P. W. Bible and L. Moser, An Open Guide to Data Structures and Algorithms. PALNI Open Press, 2023.

[3] S. S. Skiena, The Data Structures and Algorithms Design Manual, 3rd ed., Springer, 2020.

[4] Oracle, "Java SE 17–21 Collections Framework: Queue, Deque, Stack patterns," 2021–2025.

## 10. Elaboración y Aprobación

<b>Elaborado por</b>	Andrés R Navas Castellanos <b>Docente</b>	 Firmado electrónicamente por: <b>ANDRES ROBERTO NAVAS CASTELLANOS</b> Validar únicamente con FirmaEC
<b>Revisado por</b> <b>Solo si es realizado en laboratorios</b>	Luis Sinche <b>Técnico Docente</b>	No Aplica
<b>Aprobado por</b>	Edison L Coronel Romero <b>Director de Carrera</b>	 Firmado electrónicamente por: <b>EDISON LEONARDO CORONEL ROMER</b> Validar únicamente con FirmaEC