

Guía de Actividades Práctico-Experimentales Nro. 007

1. Datos Generales

Asignatura	Estructura de datos
Ciclo	3 A
Unidad	2
Resultado de aprendizaje de la unidad	Aplica los métodos de ordenación y búsqueda en la resolución de problemas, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad.
Título de la Práctica	Búsqueda en Java: Secuencial y Binaria
Nombre del Docente	Andrés Roberto Navas Castellanos
Fecha	Jueves 27 de noviembre
Horario	07h30 – 10h30
Lugar	Aula
Tiempo planificado en el Sílabo	3 horas

2. Objetivo(s) de la Práctica:

- Implementar correctamente las variantes canónicas de búsqueda secuencial y búsqueda binaria en Java.
- Validar con casos borde, y justificar cuándo aplicar cada método según la estructura de datos (arreglo vs SLL).

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

- Paso 1. Primera ocurrencia (array y SLL)
 - Arrays: `indexOfFirst(int[] a, int key)` → retornar al primer match.
 - SLL: `Node findFirst(Node head, int key)` → retornar nodo al primer match.
 - Casos borde: vacío, uno solo, duplicados (en índice 0, medio, final).
- Paso 2. Última ocurrencia (array y SLL)
 - Arrays: una pasada guardando last actualizado; o de atrás hacia adelante.
 - SLL: una pasada guardando Node last.
 - Casos: sin apariciones, todas las posiciones coinciden.
- Paso 3. `findAll` por predicado (array y SLL)
 - Arrays: `List<Integer> findAll(int[] a, IntPredicate p)`
 - SLL: `List<Node> findAll(Node head, Predicate<Node> p)`
 - Predicados sugeridos: "par", "`==key`", "`< umbral`".
 - Salida: lista de índices (array) / nodos (SLL).
- Paso 4. Secuencial con centinela (solo arrays)
 - Técnica: guardar el último elemento, escribir key al final, bucle sin chequeo de límites, restaurar último, decidir si fue hallazgo real o por centinela.
 - Comparar comparaciones realizadas vs. variante clásica.
- Paso 5. Búsqueda binaria (arrays ordenados)
 - `int binarySearch(int[] a, int key)` (iterativa).
 - Cuidados: $mid = low + (high - low) / 2$, precondition de arreglo ordenado.
 - Opcional (plus): `lowerBound/upperBound` para primera/última con duplicados.
- Paso 6. Pruebas y verificación
 - Ejecutar `SearchDemo` con:
 - Arrays: A, B, C, D; claves: 7, 5, 2, 42 (no está).
 - SLL: $3 \rightarrow 1 \rightarrow 3 \rightarrow 2$, claves: 3 (primera/última) y predicado `val < 3`.
 - Registrar índices/nodos esperados y observados.
 - Evidencias: tabla con entradas, método y salida.

```
===== PRUEBAS DE VERIFICACIÓN (DEMO) =====
```

```
===== PRUEBAS SOBRE ARREGLO: A =====
```

```
Contenido → [1, 3, 7, 7, 9]
```

Array	Clave	First	Last	FindAll
A	7	2	3	[2, 3]
A	5	-1	-1	[]
A	2	-1	-1	[]
A	42	-1	-1	[]

```
===== PRUEBAS SOBRE ARREGLO: B =====
```

```
Contenido → [5, 5, 5, 8]
```

Array	Clave	First	Last	FindAll
B	7	-1	-1	[]
B	5	0	2	[0, 1, 2]
B	2	-1	-1	[]
B	42	-1	-1	[]

```
===== PRUEBAS SOBRE ARREGLO: C =====
```

```
Contenido → [2, 4, 6, 8]
```

Array	Clave	First	Last	FindAll
C	7	-1	-1	[]
C	5	-1	-1	[]
C	2	0	0	[0]
C	42	-1	-1	[]

```
===== PRUEBAS SOBRE ARREGLO: D =====
```

```
Contenido → [10, 20, 30]
```

Array	Clave	First	Last	FindAll
D	7	-1	-1	[]
D	5	-1	-1	[]
D	2	-1	-1	[]
D	42	-1	-1	[]

```
===== PRUEBAS SOBRE ARREGLO: VACÍO =====
```

```
Contenido → []
```

Array	Clave	First	Last	FindAll
VACÍO	7	-1	-1	[]
VACÍO	5	-1	-1	[]
VACÍO	2	-1	-1	[]
VACÍO	42	-1	-1	[]

```
===== PRUEBAS CON LISTA ENLAZADA (SLL) =====
```

```
Contenido de la lista SLL:
```

```
3 → 1 → 3 → 2
```

Operación	Resultado	Posición
First(3)	3	0
Last(3)	3	2
Valores < 3	[1, 2]	N/A

```
✓ Pruebas finalizadas correctamente.
```

Cierre

- Discusión: cuándo conviene secuencial vs binaria; centinela en “no encontrado”.

Se realizó una discusión entre los 4 participantes del grupo y se llegó a lo siguiente: Algunos compañeros estuvieron de acuerdo en que secuencial es simple y puede funcionar sobre cualquiera, además es el punto clave ya que el intentar binaria en SLL para llegar al medio se debe realizar paso a paso lo que hace que sea ineficiente por lo que en SLL es conveniente usar un algoritmo secuencial. Por tanto, lo ideal es que en una colección no ordenada se use el método secuencial mientras que en una ordenada con índices se utilice binaria, aunque también se discutía sobre que secuencial no tiene recorridos repetidos, aunque si es implementado sin las debidas precauciones puede caer en null y por ello los casos borde se notan más reveladores mediante listas. En el caso de centinela se coincidió en que es beneficioso cuando el valor no se encuentra en la colección y es necesario el uso de key para asegurar que el bucle no salga del rango, aunque eso no lo hace más rápido que binaria porque sigue teniendo $O(n)$ lo que hace que recorra casi todo el array por lo que se puede optimizar la verificación del límite, pero no es posible optimizar la cantidad de comparaciones de búsqueda.

- Completar README e informe con evidencias y decisiones.

6. Resultados esperados:

- ZIP con src/ (implementaciones y SearchDemo).
- Tabla (o CSV) con casos: colección, clave/predicado, método, salida.
- README (1 pág.): cómo compilar/ejecutar; casos bordes; notas sobre precondiciones.
- (Opcional +) Comparación de comparaciones entre secuencial clásica vs centinela.

7. Preguntas de Control:

- **¿Por qué la binaria no es adecuada para SLL aunque esté ordenada?**

No es adecuada debido a que en una lista ordenada se debe recorrer la lista desde el nodo inicial, nodo por nodo hasta que se llegue al elemento central donde el tamaño se da desde $O(N)$ lo que provoca que se deba acceder de forma repetida al nodo medio haciendo que aplicar binaria se vuelva ineficiente.

- **En primera ocurrencia, ¿por qué se retorna en cuanto se encuentra?**

Porque se ha cumplido el objetivo del algoritmo haciendo que sea eficiente evitando la redundancia ya que, el seguir buscando solo incrementaría el tiempo sin llevar a ningún aporte.

- **¿Qué garantiza la correctitud de la variante centinela?**

Se garantiza la correctitud evitando falsos positivos en la última posición, restableciendo el valor original siempre sin alterar los resultados reales

- **¿Cómo adaptarías la binaria para duplicados (primera/última)?**

Cuando en primera aparición $a[\text{mind}] == \text{key}$ en vez de devolver mind se va a continuar buscando en dirección a la parte izquierda del arreglo y cuando $a[\text{mind}] == \text{key}$ en última aparición se continúa buscando hacia la parte derecha.



- **Propón dos casos bordes que hayan detectado errores en tus pruebas.**
- Arreglo o lista vacía
- Presencia de duplicados

8. Evaluación



Criterio	4 – Excelente	3 – Bueno	2 – Básico	1 – Insuficiente	Pts
Secuencial (first/last/findAll)	Correctos; manejan bordes; código claro	Detalles menores	Parcial	No funcional	3.0
Centinela (arrays)	Implementado y explicado; comparación de comparaciones	Implementado	Parcial	No	2.0
Binaria (arrays)	Correcta; cuidado con mid; precondition validada	Detalles menores	Parcial	No	2.5
Evidencias (tabla/README)	Completas y reproducibles	Aceptables	Escasas	Nulas	1.5
Calidad de código	Organización y nombres adecuados	Aceptable	Pobre	Deficiente	1.0



9. Bibliografía

- [1] P. W. Bible and L. Moser, An Open Guide to Data Structures and Algorithms. PALNI Open Press, 2023.
- [2] OpenDSA Project, "Searching and Sorting Modules," Virginia Tech, 2021–2024 (REA con visualizaciones).
- [3] Oracle, "Java SE 17–21 Documentation: Arrays.binarySearch, Comparator y patrones de búsqueda," 2021–2025.

10. Elaboración y Aprobación

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