Guía de Actividades Práctico-Experimentales Nro. 007

1. Datos Generales

Asignatura	Estructura de datos
Ciclo	3 A
Unidad	2
Integrantes	- Cael Alejandro Soto Castillo - Cristhian Alexander Dávila Sari - Martina Alejandra Maldonado Machuca - Fernando Sebastian Patiño Diaz
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: int 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, precondición 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→1→3→2, claves: 3 (primera/última) y predicado val<3.
  - Registrar índices/nodos esperados y observados.
  - Evidencias: tabla con entradas, método y salida.

Cierre

- **Discusión: cuándo conviene secuencial vs binaria; centinela en "no encontrado".**
  Después de implementar y probar estos métodos de búsqueda, vimos que la búsqueda secuencial es la más adecuada cuando los datos no están ordenados o cuando trabajamos con listas enlazadas, ya que en una SLL no se puede acceder directamente a posiciones específicas. En cambio, la búsqueda binaria sólo conviene en arreglos ordenados, donde sí es posible calcular el índice medio de manera eficiente y obtener un mejor rendimiento. Respecto al centinela, comprobamos que ayuda a reducir comparaciones dentro del bucle, pero requiere verificar al final si el elemento realmente estaba en el arreglo o si sólo se encontró por efecto del centinela. Esta verificación es clave sobre todo cuando la clave no se encuentra, para evitar falsos positivos.
- **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.

- TABLA CON CASOS:

Key	Dataset	Tipo de búsqueda	Resultado
2	A (mezclado)	Secuencial	1
2	A (mezclado)	Centinela	1
2	A (mezclado)	Primera ocurrencia	1
2	A (mezclado)	Última ocurrencia	99
2	A (mezclado)	Binaria	24
2	B (descendente)	Secuencial	-1
2	B (descendente)	Centinela	-1
2	B (descendente)	Primera ocurrencia	-1
2	B (descendente)	Última ocurrencia	-1
2	B (descendente)	Binaria	-1
2	C (ascendente)	Secuencial	1
2	C (ascendente)	Centinela	1
2	C (ascendente)	Primera ocurrencia	1
2	C (ascendente)	Última ocurrencia	1
2	C (ascendente)	Binaria	1
2	D (casi ordenado)	Secuencial	91
2	D (casi ordenado)	Centinela	91
2	D (casi ordenado)	Primera ocurrencia	91
2	D (casi ordenado)	Última ocurrencia	91
2	D (casi ordenado)	Binaria	-1
2	A (mezclado)	SLL todas las ocurrencias	[1,2,8,12,13,16,17,18,24,25,29,33,37,38,42,51,54,59,71,75,78,87,94,99]
2	A (mezclado)	SLL primera ocurrencia	1
2	A (mezclado)	SLL última ocurrencia	99
2	B (descendente)	SLL todas las ocurrencias	[]
2	B (descendente)	SLL primera ocurrencia	-1
2	B (descendente)	SLL última ocurrencia	-1
2	C (ascendente)	SLL todas las ocurrencias	[1]
2	C (ascendente)	SLL primera ocurrencia	1
2	C (ascendente)	SLL última ocurrencia	1
2	D (casi ordenado)	SLL todas las ocurrencias	[91]
2	D (casi ordenado)	SLL primera ocurrencia	91
2	D (casi ordenado)	SLL última ocurrencia	91
5	A (mezclado)	Secuencial	-1
5	A (mezclado)	Centinela	-1
5	A (mezclado)	Primera ocurrencia	-1
5	A (mezclado)	Última ocurrencia	-1
5	A (mezclado)	Binaria	-1
5	B (descendente)	Secuencial	-1
5	B (descendente)	Centinela	-1
5	B (descendente)	Primera ocurrencia	-1
5	B (descendente)	Última ocurrencia	-1
5	B (descendente)	Binaria	-1
5	C (ascendente)	Secuencial	4
5	C (ascendente)	Centinela	4
5	C (ascendente)	Primera ocurrencia	4
5	C (ascendente)	Última ocurrencia	4
5	C (ascendente)	Binaria	4
5	D (casi ordenado)	Secuencial	70
5	D (casi ordenado)	Centinela	70
5	D (casi ordenado)	Primera ocurrencia	70
5	D (casi ordenado)	Última ocurrencia	70
5	D (casi ordenado)	Binaria	-1
5	A (mezclado)	SLL todas las ocurrencias	[]
5	A (mezclado)	SLL primera ocurrencia	-1
5	A (mezclado)	SLL última ocurrencia	-1
5	B (descendente)	SLL todas las ocurrencias	[]
5	B (descendente)	SLL primera ocurrencia	-1
5	B (descendente)	SLL última ocurrencia	-1
5	C (ascendente)	SLL todas las ocurrencias	[4]
5	C (ascendente)	SLL primera ocurrencia	4
5	C (ascendente)	SLL última ocurrencia	4
5	D (casi ordenado)	SLL todas las ocurrencias	[70]
5	D (casi ordenado)	SLL primera ocurrencia	70
5	D (casi ordenado)	SLL última ocurrencia	70

```
=== DATASET A (mezclado) ===
[ 1 2 2 3 3 1 3 1 2 1 3 3 2 2 3 3 2 2 2 1 3 3 1 3 2 2 1 3 1 2 1 1 3 2 3 3 3 2 2 1 3 1 2 3 1 3 3 3 1 3 3 2 3 1 2 1 3 1 1 2
=== DATASET B (descendente) ===
[ 500 499 498 497 496 495 494 493 492 491 490 489 488 487 486 485 484 483 482 481 480 479 478 477 476 475 474 473 472 471
=== DATASET C (ascendente) ===
[ 1 2 3 4 5 6 7 8 9 10 ]
=== DATASET D (casi ordenado) ===
[ 66 65 14 98 64 19 76 25 51 38 36 74 18 13 57 83 63 43 41 71 47 27 32 79 6 40 55 95 80 67 31 1 82 22 33 94 77 78 12 69 2

=== LISTA A ===
[ 1 → 2 → 2 → 3 → 3 → 1 → 3 → 1 → 2 → 1 → 3 → 3 → 2 → 2 → 3 → 3 → 2 → 2 → 2 → 1 → 3 → 3 → 1 → 3 → 2 → 2 → 1 → 3 → 1 → 2 →
=== LISTA B ===
[ 500 → 499 → 498 → 497 → 496 → 495 → 494 → 493 → 492 → 491 → 490 → 489 → 488 → 487 → 486 → 485 → 484 → 483 → 482 → 481 →
=== LISTA C ===
[ 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 ]
=== LISTA D ===
[ 66 → 65 → 14 → 98 → 64 → 19 → 76 → 25 → 51 → 38 → 36 → 74 → 18 → 13 → 57 → 83 → 63 → 43 → 41 → 71 → 47 → 27 → 32 → 79 →

Ingrese la clave (key) que desea buscar:
```

```
Ingrese la clave (key) que desea buscar: 2

==============================
 BÚSQUEDAS EN ARRAY
==============================

================Dataset A================
Secuencial A = 1
Centinela A = 1
Primera ocurrencia A = 1
Última ocurrencia A = 99
Binaria A = 24

================Dataset B================
Secuencial B = -1
Centinela B = -1
Primera ocurrencia B = -1
Última ocurrencia B = -1
Binaria B = -1

================Dataset C================
Secuencial C = 1
Centinela C = 1
Primera ocurrencia C = 1
Última ocurrencia C = 1
Binaria C = 1

================Dataset D================
Secuencial D = 91
Centinela D = 91
Primera ocurrencia D = 91
Última ocurrencia D = 91
Binaria D = -1
```

```
============================
 BÚSQUEDAS EN SLL
============================

SLL búsqueda A = [1, 2, 8, 12, 13, 16, 17, 18, 24, 25, 29, 33, 37, 38, 42, 51, 54, 59, 71, 75, 78, 87, 94, 99]
SLL primera A = 1
SLL última A = 99

SLL búsqueda B = []
SLL primera B = -1
SLL última B = -1

SLL búsqueda C = [1]
SLL primera C = 1
SLL última C = 1

SLL búsqueda D = [91]
SLL primera D = 91
SLL última D = 91
Desea realizar otra busqueda (key) (s/n):
```

```
Ingrese la clave (key) que desea buscar: 5

==============================
 BÚSQUEDAS EN ARRAY
==============================

================Dataset A================
Secuencial A = -1
Centinela A = -1
Primera ocurrencia A = -1
Última ocurrencia A = -1
Binaria A = -1

================Dataset B================
Secuencial B = -1
Centinela B = -1
Primera ocurrencia B = -1
Última ocurrencia B = -1
Binaria B = -1

================Dataset C================
Secuencial C = 4
Centinela C = 4
Primera ocurrencia C = 4
Última ocurrencia C = 4
Binaria C = 4

================Dataset D================
Secuencial D = 70
Centinela D = 70
Primera ocurrencia D = 70
Última ocurrencia D = 70
Binaria D = -1
```

```
============================
 BÚSQUEDAS EN SLL
============================

SLL búsqueda A = []
SLL primera ocurrencia A = -1
SLL última ocurrencia A = -1

SLL búsqueda total B = []
SLL primera ocurrencia B = -1
SLL última ocurrencia B = -1

SLL búsqueda total C = [4]
SLL primera ocurrencia C = 4
SLL última ocurrencia C = 4

SLL búsqueda total D = [70]
SLL primera ocurrencia D = 70
SLL última ocurrencia D = 70
Desea realizar otra busqueda (key) (s/n):
```

7. Preguntas de Control:

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

  La búsqueda binaria no funciona con listas enlazadas simples aunque esté ordenada porque este tipo de búsqueda necesita acceso aleatorio a los elementos, y en las SLL solo tenemos acceso secuencial. En un arreglo se puede saltar directamente a la posición del medio usando índices, en una lista enlazada simple para llegar al elemento de la mitad se tiene que recorrer todos los nodos desde el inicio uno por uno y si se aplica esto a cada división del algoritmo el rendimiento baja perdiendo toda ventaja de lo que es una búsqueda binaria.

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

  Retornamos inmediatamente cuando encontramos la primera ocurrencia porque así optimizamos el algoritmo, si seguimos buscando después de haber encontrado el elemento estaríamos haciendo comparaciones innecesarias que no mejoran el resultado. En el mejor caso, si el elemento está al principio terminamos rápido, ya que el objetivo del método es encontrar la primera aparición, no todas, así que una vez que la tenemos, no hay razón para seguir procesando el resto de la colección.

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

  La correctitud de la variante centinela se garantiza por dos aspectos principales: primero, al colocar el elemento buscado al final del arreglo, nos aseguramos de que siempre vamos a encontrar algo (o el elemento real o el centinela), lo que elimina la necesidad de verificar los límites en cada iteración. Segundo, después de terminar la búsqueda, verificamos si el elemento encontrado es el centinela o el elemento real, y restauramos el arreglo original si fue modificado. Esto asegura que el resultado sea válido y que el arreglo quede en su estado original.

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

  Para adaptar la búsqueda binaria con duplicados, implementaría dos variantes:

  - **Para la primera ocurrencia:** cuando encuentro el elemento, en lugar de retornar inmediatamente sigo buscando hacia la izquierda para ver si hay otra ocurrencia más. Es decir, actualizo el límite superior para seguir buscando en la

mitad izquierda.

- **Para la última ocurrencia:** similar al anterior, pero cuando encuentro el elemento, sigo buscando hacia la derecha para ver si hay otra ocurrencia más a la derecha, actualizando el límite inferior.

- **Propón dos casos borde que hayan detectado errores en tus pruebas.**

  En nuestras pruebas encontramos dos casos borde que inicialmente causaron problemas:

  - **Arreglo con un solo elemento repetido:** Probamos con un arreglo [7,7,7,7,7] buscando el número 7. Nuestro algoritmo de primera ocurrencia funcionaba bien, pero el de última ocurrencia inicialmente retornaba el primer índice en lugar del último. Tuvimos que ajustar el algoritmo para que siguiera buscando incluso después de encontrar una coincidencia.
  - **Lista enlazada vacía:** Al probar con una lista sin nodos (null), nuestros métodos de búsqueda en SLL daban NullPointerException. Tuvimos que agregar una verificación al inicio de cada método para que si la cabeza es null, retorne inmediatamente null o -1 según corresponda, sin intentar recorrer la lista.

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; precondición 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

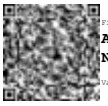
9. Bibliografía

[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