ALGORÍTMICA

Material de apoyo pedagógico

Segunda Etapa

Prof. Carlos Alberto Benítez Brítez

Bachillerato Técnico en

Informática

Colegio Nacional de E.M.D. Dr. Raúl Peña - "Multi"

Caacupé - 2024

Arrays, arreglos, vectores, matrices en JAVA.

Actividades propuestas (de repaso).

PARTE TEORICA

BUSCA LA INFORMACIÓN EN INTERNET *O LEE EL PDF SOBRE ARRAY* Y RESPONDE LOS SIGUIENTES PLANTEAMIENTOS

1. ¿Qué es un array o arreglo en JAVA?
2. ¿Qué tipo de datos se puede almacenar en un array en JAVA?
3. ¿Cómo se define un array unidimensional, arreglo o vector en JAVA?
4. ¿Se puede definir un array y darle valor inicial al mismo tiempo en JAVA?
5. ¿Después de definir un array se puede inicializar en JAVA?
6. ¿En un array se puede almacenar distintos tipos de datos en JAVA? ¿Por qué?
7. ¿Cómo se identifica un elemento de un array en JAVA?
8. ¿Cómo se accede a un elemento de un array en JAVA?
9. ¿Cuál es el valor inicial del índice en un array en JAVA?
10. ¿Cuál es el valor final del índice en un array en JAVA?
11. ¿Cuál es el método para saber la longitud de un array en JAVA?
12. ¿Cómo se puede recorrer un array unidimensional en JAVA?

PARTE PRACTICA

Desarrolla un programa que implemente métodos para trabajar con matrices bidimensionales y las utilice en un programa principal para:

- ✓ Solicitar al usuario las dimensiones de la matriz (filas y columnas).
- ✓ Validar que las dimensiones sean números enteros positivos menores que 11.
- ✓ Declarar una matriz bidimensional de enteros con las dimensiones especificadas por el usuario.
- ✓ Cargar la matriz con números aleatorios entre 35 y 75.
- ✓ Imprimir la matriz.
- ✓ Calcular y mostrar el promedio de los elementos del triángulo superior.
- ✓ Calcular y mostrar el promedio de los elementos del triángulo inferior.

Métodos a implementar:

- ✓ cargar_matriz(matriz, filas, columnas): Carga la matriz con números aleatorios entre 35 y 75.
- ✓ imprimir_matriz(matriz, filas, columnas): Imprime los elementos de la matriz.
- ✓ promedio_TS(matriz, filas, columnas): Calcula y retorna el promedio de los elementos del triángulo superior.
- ✓ promedio_TI(matriz, filas, columnas): Calcula y retorna el promedio de los elementos del triángulo inferior.

Ejemplo de ejecución:

Ingrese el número de filas y de columnas: **4**

Matriz generada:

42	68	59	47
38	63	72	56
51	45	61	37
64	52	40	74

Promedio del triángulo superior: **56.5**
Promedio del triángulo inferior: **48.3**

Nota: La matriz debe ser cuadrada (misma cantidad de filas y columnas).
En una matriz bidimensional, se consideran elementos del **triángulo superior** aquellos que se encuentran por encima de la diagonal principal, mientras que los elementos del **triángulo inferior** se encuentran por debajo de la misma.

Desarrolla un programa que implemente métodos para trabajar con matrices bidimensionales y las utilice en un programa principal para:

- ✓ Solicitar al usuario las dimensiones de la matriz (filas y columnas).
- ✓ Validar que las dimensiones sean números enteros positivos menores que 21.
- ✓ Declarar una matriz bidimensional de enteros con las dimensiones especificadas por el usuario.
- ✓ Cargar la matriz con números aleatorios entre 15 y 50.
- ✓ Imprimir la matriz completa.
- ✓ Solicitar al usuario una fila específica (dentro del rango válido de filas).
- ✓ Calcular y mostrar el mayor valor, el menor valor y el promedio de los elementos en la fila indicada.

Métodos a implementar:

- ✓ cargar_matriz(matriz, filas, columnas): Carga la matriz con números aleatorios entre 15 y 50.
- ✓ imprimir_matriz(matriz, filas, columnas): Imprime los elementos de la matriz.
- ✓ obtener_mayor(matriz, filas, columnas, fila_seleccionada): Calcula y retorna el mayor valor en la fila especificada (fila_seleccionada).
- ✓ obtener_menor(matriz, filas, columnas, fila_seleccionada): Calcula y retorna el menor valor en la fila especificada (fila_seleccionada).
- ✓ calcular_promedio(matriz, filas, columnas, fila_seleccionada): Calcula y retorna el promedio de los elementos en la fila especificada (fila_seleccionada).

Ejemplo de ejecución:

Ingrese el número de filas: **4**
Ingrese el número de columnas: **5**

Matriz generada:

42 68 59 47 64

38 63 72 56 52

51 45 61 37 40
64 52 40 74 67
Seleccione una fila (desde 1 hasta 4): 2

Mayor valor en la fila **2**: **72**
Menor valor en la fila **2**: **38**
Promedio de la fila **2**: **56.2**

Arrays, arreglos, vectores, matrices en JAVA.

Actividades propuestas (de repaso).

PARTE PRACTICA

1. **Suma y resta de matrices:** Dadas dos matrices del mismo tamaño, escribe un programa que calcule y muestre la suma y la resta de estas matrices.

Ejemplo: Dadas las matrices:

A = [[1, 2, 3], [4, 5, 6]]

B = [[7, 8, 9], [10, 11, 12]]

La suma de A y B sería:

A + B = [[8, 10, 12], [14, 16, 18]]

Y la resta:

A - B = [[-6, -6, -6], [-6, -6, -6]]

Consideraciones:

- ✓ **Dimensiones:** Ambas matrices deben tener las mismas dimensiones para poder sumarlas o restarlas.
- ✓ **Elemento a elemento:** La suma y la resta se realizan elemento a elemento.

2. **Multiplicación de una matriz por un escalar:** Dada una matriz y un número escalar, escribe un programa que calcule el producto de la matriz por el escalar.

Ejemplo: Dada la matriz A y el escalar 2:

A = [[1, 2, 3], [4, 5, 6]]

El producto de A por 2 sería:

2 * A = [[2, 4, 6], [8, 10, 12]]

Consideraciones:

- ✓ **Cada elemento:** Se multiplica cada elemento de la matriz por el escalar.

3. **Transpuesta de una matriz:** Dada una matriz, escribe un programa que calcule y muestre su matriz transpuesta.

Ejemplo: Dada la matriz A:

A = [[1, 2, 3], [4, 5, 6]]

La transpuesta de A sería:

A^T = [[1, 4], [2, 5], [3, 6]]

Consideraciones:

- ✓ **Intercambio de filas y columnas:** Las filas de la matriz original se convierten en columnas en la matriz transpuesta.

4. **Matriz identidad:** Escribe un programa que genere una matriz identidad de un tamaño dado.

Ejemplo: Una matriz identidad de tamaño 3x3 es:

I = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]

Consideraciones:

- ✓ **Diagonal principal:** Los elementos de la diagonal principal son 1, y el resto son 0.

5. **Diagonal principal:** Dada una matriz cuadrada, escribe un programa que imprima los elementos de su diagonal principal.

Ejemplo: Dada la matriz A:

A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

La diagonal principal de A es: [1, 5, 9]

Consideraciones:

- ✓ **Elementos en la diagonal:** La diagonal principal de una matriz cuadrada está formada por los elementos que tienen el mismo índice de fila y columna.

6. **Análisis de Ventas en una Tienda**: Imaginemos que tenemos una tienda que vende tres tipos de productos electrónicos: teléfonos móviles, tablets y computadoras portátiles. Estos productos se venden en cuatro ciudades diferentes: Asunción, Ciudad del Este, Encarnación y Caacupé. Queremos analizar las ventas de estos productos para tomar decisiones estratégicas.

Matriz de ventas:

Podemos representar las ventas en una matriz de 3 x 4, donde cada fila corresponde a un producto y cada columna a una ciudad.

	Asunción	Ciudad del Este	Encarnación	Caacupé
Móviles	1000	800	1200	900
Tablets	500	600	400	300
Portátiles	800	900	700	

Preguntas a responder:

- ✓ Ventas totales por producto: ¿Cuántos teléfonos móviles, tablets y portátiles se vendieron en total?
- ✓ Ventas totales por ciudad: ¿Cuántas unidades se vendieron en cada ciudad?
- ✓ Producto más vendido: ¿Cuál es el producto que más se vendió en general?
- ✓ Ciudad con más ventas: ¿En qué ciudad se vendieron más productos en total?
- ✓ Producto más vendido por ciudad: ¿Cuál es el producto más vendido en cada ciudad?
- ✓ Ventas promedio por producto: ¿Cuál es el promedio de ventas de cada producto?
- ✓ Ventas promedio por ciudad: ¿Cuál es el promedio de ventas en cada ciudad?

Estructura de datos y organización de archivos

Estructura de datos. PILAS

Actividades propuestas.

Previa lectura y análisis del Capítulo 4 – Paginas del 57 al 77 – Introducción a la estructura de datos, del libro “Estructura de datos y organización de archivos” segunda edición Mary E. S. Loomis, para poner en práctica lo aprendido, completa las sentencias que se proponen a continuación.

1.1. ¿A qué estructuras de datos pertenecen las pilas?
1.2. ¿Qué es una pila?
1.3. ¿Cómo se representa gráficamente una pila?
1.4. Escribe ejemplos de pilas en la vida real
1.5. Escribe ejemplos de aplicaciones de pilas
1.6. ¿Cuáles son las operaciones sobre pilas?
1.7. ¿Por dónde se realizan las operaciones de inserción y supresión en una pila?
1.8. Dibuja una pila e indique la dirección de entrada y salida de los elementos
1.9. ¿Cuál es el tope de una pila vacía?
1.10. ¿Cuántos elementos hay en una pila de nueva creación?

Estructura de datos y organización de archivos

Estructura de datos. COLAS

Actividades propuestas.

Previa lectura y análisis del Capítulo 5 – Paginas del 78 al 98 – Introducción a la estructura de datos, del libro “Estructura de datos y organización de archivos” segunda edición Mary E. S. Loomis, para poner en práctica lo aprendido, completa las sentencias que se proponen a continuación.

1.11. ¿A qué estructuras de datos pertenecen las **COLAS**?
1.12. ¿Qué es una **COLA**?
1.13. ¿Cómo se representa gráficamente una **COLA**?
1.14. Escribe ejemplos de **COLAS** en la vida real
1.15. Escribe ejemplos de aplicaciones de **COLAS**
1.16. ¿Cuáles son las operaciones sobre **COLAS**?
1.17. ¿Por dónde se realizan las operaciones de inserción y supresión en una **COLA**?
1.18. Dibuje una **COLA** e indique la dirección de entrada y salida de los elementos
1.19. ¿Cuántos elementos hay en una **COLA** de nueva creación?
1.20. ¿Cuál es la diferencia entre una **PILA** y una **COLA**?

Estructura de datos y organización de archivos

Estructura de datos. LISTAS LIGADAS

Actividades propuestas.

Previa lectura y análisis del Capítulo 6 – Paginas del 99 al 138 – Introducción a la estructura de datos, del libro "Estructura de datos y organización de archivos" segunda edición Mary E. S. Loomis, para poner en práctica lo aprendido, completa las sentencias que se proponen a continuación.

1. ¿A qué estructuras de datos pertenecen las listas ligadas?
2. ¿Qué es una lista ligada?
3. ¿Cuáles son los tipos de listas?
4. ¿Cuáles son las diferencias entre listas ligadas, listas circulares ligadas, listas doblemente ligadas?
5. ¿Cómo se sabe que una lista está vacía?
6. ¿Cuáles son los costos de una lista ligada?
7. ¿Cuáles son las operaciones básicas en una lista ligada?

REPRESENTA EN FORMA GRÁFICA

1. Una lista ligada vacía
2. Una lista ligada de 4 elementos
3. Una lista circular vacía
4. Una lista circular ligada de 5 elementos
5. Una lista doblemente ligada vacía
6. Una lista doblemente ligada de 3 elementos

Estructura de datos y organización de archivos

Estructura de datos. GRAFOS

Actividades propuestas.

Previa lectura y análisis del Capítulo 7 – Paginas del 139 al 166 – Introducción a la estructura de datos, del libro "Estructura de datos y organización de archivos" segunda edición Mary E. S. Loomis, para poner en práctica lo aprendido, completa las sentencias que se proponen a continuación.

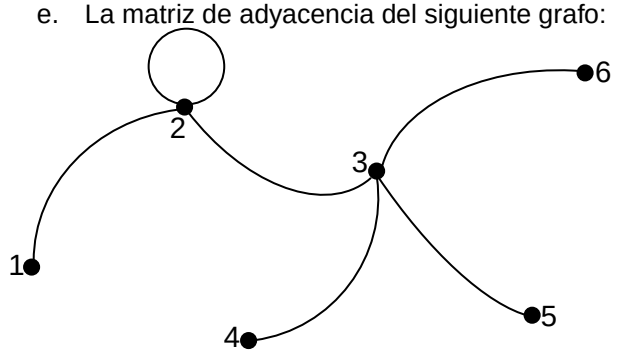
Escribe el concepto de:

a. Grafo.
b. Aplicaciones de los grafos.
c. ¿A qué tipo de estructura de datos corresponde un grafo?
d. Nodos o vértices.

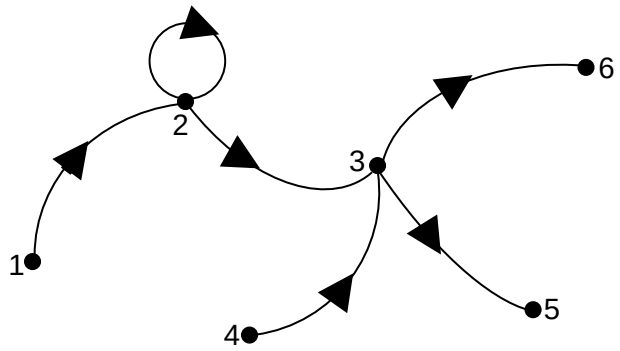
e. Arista o arcos.
f. Bucles.
g. Orden del grafo.
h. Grafo de orden cero.
i. Grafo simple.
j. Multígrafo.
k. Grafo conexo.
l. Grafo no conexo.
m. Trayectoria de un grafo.
n. Longitud de la trayectoria.
o. Grafos dirigidos.
p. Dirección del grafo dirigido.
q. Grado interno de un nodo.
r. Grado externo de un nodo.
s. Grado del nodo.
t. Matriz de adyacencia.
u. Representación de la matriz de adyacencia para un grafo dirigido.
v. Representación de la matriz de adyacencia para un grafo no dirigido.
w. Recorrido de grafos.
x. Recorrido en amplitud.
y. Recorrido en profundidad.

Representa en forma gráfica:

a. Un grafo dirigido.
b. Un grafo no dirigido.
c. Un grafo acíclico.
d. Un grafo que muestre las distancias entre ciudades.
e. La matriz de adyacencia del siguiente grafo:



f. La matriz de adyacencia del siguiente grafo:



Estructura de datos y organización de archivos

Estructura de datos. Árboles generales y binarios

Actividades propuestas.

Previa lectura y análisis del Capítulo 8 – Paginas del 167 al 210 – Introducción a la estructura de datos, del libro "Estructura de datos y organización de archivos" segunda edición Mary E. S. Loomis, para poner en práctica lo aprendido, completa las sentencias que se proponen a continuación.

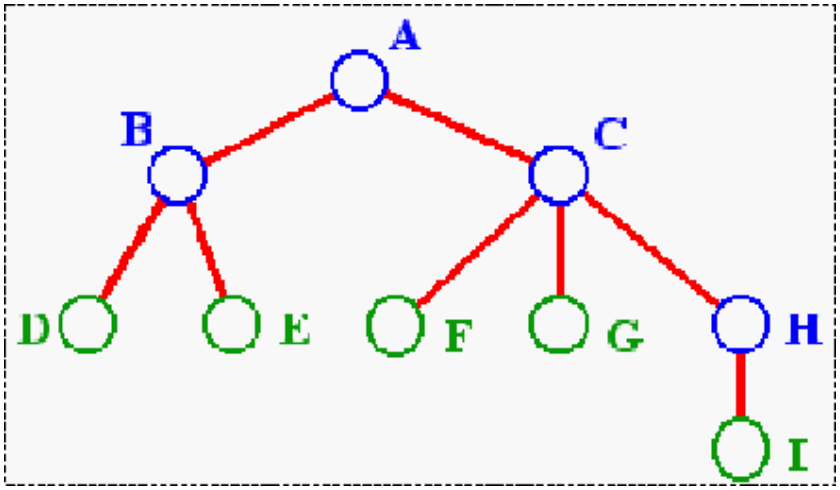
Escribe el concepto de:

a. Árbol general.
b. Raíz.
c. Subárboles.
d. Nodo hijo.
e. Nodos hermanos.
f. Nodos hojas.
g. Nivel del árbol.
h. Altura del árbol.
i. Peso del árbol.
j. Bosque.
k. Aplicaciones de los árboles.
l. ¿A qué tipo de estructura de datos corresponde un árbol?
m. Formas gráficas de representar árboles generales y binarios.
n. Árbol binario.
o. Árboles binarios similares.
p. Árboles binarios equivalentes.
q. Árbol binario completo.
r. Árbol binario casi completo.
s. Características o propiedades que distinguen un árbol binario de un árbol general.
t. Recorrido de árboles.
u. Recorrido en pre-orden
v. Recorrido en en-orden
w. Recorrido en post-orden

Representa en forma gráfica:

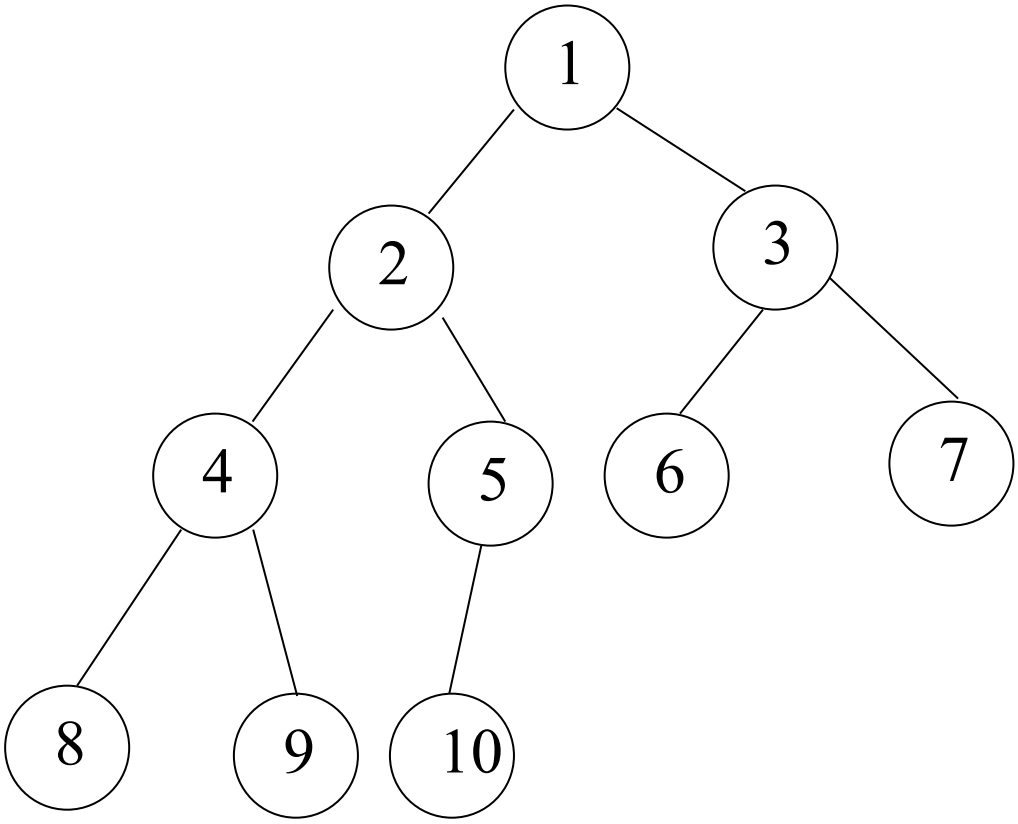
a. Un árbol general.
b. Un árbol binario.
c. Árboles binarios que sean similares.
d. Árboles binarios que sean equivalentes.

Indica las partes del siguiente árbol



a. Nodo raíz:
b. Padre de D y E:
c. El primo de C:
d. Los hijos de B:
e. Los nodos hojas:
f. La profundidad (Nivel) de E:
g. La altura del árbol:

Indica el recorrido del siguiente árbol:



a. En-orden:
b. Pre-orden:
c. Post-Orden

Estructura de datos y organización de archivos

Métodos de Ordenación en Arrays con Java

Introducción

Los métodos de ordenación son algoritmos fundamentales en la programación que permiten organizar los elementos de una estructura de datos, como un array, de manera ascendente o descendente. En Java, existen numerosos métodos de ordenación, cada uno con sus propias características, ventajas y desventajas. En este material, exploraremos algunos de los métodos de ordenación más comunes y sus implementaciones en Java.

Características, Ventajas y Desventajas de los Métodos de Ordenación

0. Método de la Burbuja

- ✓ **Características:** Compara elementos adyacentes e intercambia aquellos que están en el orden incorrecto. Repeta este proceso hasta que no se produzcan más intercambios.
- ✓ **Ventajas:** Fácil de implementar y entender.
- ✓ **Desventajas:** Ineficiente para grandes conjuntos de datos, ya que realiza muchas comparaciones innecesarias.

1. Método de Ordenación por Selección

- ✓ **Características:** Encuentra el elemento mínimo en cada pasada y lo coloca en su posición correcta.
- ✓ **Ventajas:** Realiza menos intercambios que la burbuja.
- ✓ **Desventajas:** El número de comparaciones es similar al de la burbuja.

2. Método de Ordenación por Inserción Directa

- ✓ **Características:** Construye una secuencia ordenada un elemento a la vez. Inserta cada nuevo elemento en su posición correcta dentro de la secuencia ordenada.
- ✓ **Ventajas:** Eficiente para conjuntos de datos pequeños y parcialmente ordenados.
- ✓ **Desventajas:** Puede ser lento para conjuntos de datos grandes.

3. Método de Ordenación por Inserción Binaria

- ✓ **Características:** Similar a la inserción directa, pero utiliza búsqueda binaria para encontrar la posición correcta del elemento a insertar.
- ✓ **Ventajas:** Más eficiente que la inserción directa para conjuntos de datos grandes.
- ✓ **Desventajas:** Requiere un array ordenado inicialmente.

4. Método de Ordenación Shell

- ✓ **Características:** Una variante de la inserción directa que compara elementos distantes al principio y luego reduce gradualmente la distancia.
- ✓ **Ventajas:** Más eficiente que la inserción directa para muchos tipos de datos.
- ✓ **Desventajas:** La elección del incremento inicial puede afectar el rendimiento.

5. Método de Ordenación Heap Sort

- ✓ **Características:** Construye un montículo (heap) a partir de los datos y luego extrae repetidamente el elemento máximo del montículo.
- ✓ **Ventajas:** Eficiente en el peor de los casos.
- ✓ **Desventajas:** Requiere espacio adicional para el montículo.

Comparación de Métodos de Ordenación

La eficiencia de un algoritmo de ordenación se suele medir en términos de:

- ✓ **Tiempo de ejecución:** Cuánto tiempo tarda el algoritmo en ordenar un conjunto de datos de un determinado tamaño.
- ✓ **Espacio utilizado:** Cuánta memoria adicional requiere el algoritmo además del espacio para los datos originales.

La complejidad algorítmica nos permite analizar el comportamiento de un algoritmo a medida que el tamaño de la entrada aumenta. Se expresa utilizando la notación O grande (O grande).

Tabla comparativa:

Método de Ordenación	Mejor caso (O)	Peor caso (O)	Caso promedio (O)	Espacio adicional	Estable
Burbuja	n	n^2	n^2	1	Sí
Selección	n^2	n^2	n^2	1	No
Inserción directa	n	n^2	n^2	1	Sí
Inserción binaria	n log n	n^2	n^2	1	Sí
Shell	n log n	n^2	Depende del incremento	1	No
HeapSort	n log n	n log n	n log n	n	No
QuickSort	n log n	n^2	n log n	log n	No
MergeSort	n log n	n log n	n log n	n	Sí

Explicación de las columnas:

- ✓ **Mejor caso:** El tiempo de ejecución cuando los datos están en el mejor orden posible para el algoritmo (por ejemplo, ya están ordenados).
- ✓ **Peor caso:** El tiempo de ejecución cuando los datos están en el peor orden posible para el algoritmo.
- ✓ **Caso promedio:** El tiempo de ejecución promedio para una entrada aleatoria.
- ✓ **Espacio adicional:** La cantidad de memoria extra que requiere el algoritmo además del array original.
- ✓ **Estable:** Un algoritmo de ordenación es estable si mantiene el orden relativo de elementos iguales en la secuencia ordenada.

Análisis:

- ✓ **Burbuja, Selección, Inserción directa:** Estos métodos son simples de implementar pero tienen una complejidad en el peor caso de $O(n^2)$, lo que los hace ineficientes para grandes conjuntos de datos.
- ✓ **Inserción binaria, Shell:** Son mejoras sobre la inserción directa y pueden ser más eficientes en algunos casos, pero aun así tienen una complejidad en el peor caso de $O(n^2)$.
- ✓ **HeapSort, QuickSort, MergeSort:** Estos métodos tienen una complejidad promedio de O(n log n), lo que los hace mucho más eficientes que los anteriores para grandes conjuntos de datos.
- ✓ **QuickSort:** Es generalmente el algoritmo de ordenación más rápido en la práctica, pero su tiempo de ejecución puede degradarse a $O(n^2)$ en el peor caso.

- ✓ **MergeSort:** Es estable y garantiza un tiempo de ejecución de O(n log n) en todos los casos, pero requiere espacio adicional lineal.

Cuando usar qué método:

- ✓ **Conjuntos de datos pequeños:** La simplicidad de los métodos de burbuja, selección e inserción directa puede ser suficiente.
- ✓ **Conjuntos de datos grandes y ordenados parcialmente:** La inserción binaria puede ser una buena opción.
- ✓ **Conjuntos de datos grandes y desordenados:** HeapSort, QuickSort o MergeSort son las mejores opciones.

Factores a considerar:

- ✓ **Tamaño del conjunto de datos:** Para conjuntos de datos pequeños, la diferencia de eficiencia entre los algoritmos puede ser insignificante.
- ✓ **Orden inicial de los datos:** Algunos algoritmos se benefician de datos parcialmente ordenados.
- ✓ **Requisitos de memoria:** Si la memoria es limitada, es importante considerar el espacio adicional requerido por cada algoritmo.
- ✓ **Estabilidad:** Si es importante mantener el orden relativo de elementos iguales, se deben elegir algoritmos estables.

En resumen:

La elección del método de ordenación adecuado depende de una variedad de factores, incluyendo el tamaño del conjunto de datos, la orden inicial de los datos, los requisitos de memoria y si se necesita un algoritmo estable. Comprender la complejidad algorítmica y las características de cada método te permitirá tomar decisiones informadas al desarrollar tus aplicaciones.

Actividad propuesta.

Busca los métodos de ordenación de la tabla y por último codifica en JAVA con interfaz gráfica.

Contenido

Actividad: Desarrollo de un Sistema Web para la Gestión de Contenido Educativo

Objetivo: Crear una página web dinámica que muestre y administre contenido relacionado con estructuras de datos, basado en el material pedagógico proporcionado (de todas las clases anteriores). Usarán PHP para la lógica de backend, MySQL para la gestión de datos y Bootstrap 5 para el diseño.

Requisitos Técnicos

- ✓ **Lenguajes y tecnologías:** PHP, MySQL, HTML, CSS, JavaScript, Bootstrap 5
- ✓ **Entorno:** Servidor local
- ✓ **Editor:** Cualquiera que prefieran (Visual Studio Code)

Instrucciones Paso a Paso

1. Preparación del Entorno

1. **Instala un servidor local**: Usa XAMPP para configurar el entorno de desarrollo.
2. **Configura la base de datos en MySQL**:
   - ✓ Crea una base de datos llamada estructurasDatos.
   - ✓ Define las siguientes tablas:
     - ❖ **contenido**:
       - ➢ id (INT, AUTO_INCREMENT, PRIMARY KEY)
       - ➢ titulo (VARCHAR, 255)
       - ➢ descripcion (TEXT)
       - ➢ tipo (ENUM, valores: "Array", "Pila", "Cola", "Lista", "Grafo", "Árbol", "Ordenación")
       - ➢ archivo (VARCHAR, 255) - para almacenar rutas de archivos adicionales si es necesario
     - ❖ **usuarios** (para administrar):
       - ➢ id (INT, AUTO_INCREMENT, PRIMARY KEY)
       - ➢ nombre (VARCHAR, 100)
       - ➢ email (VARCHAR, 100, UNIQUE)
       - ➢ password (VARCHAR, 255)

Las tablas son a modo de ejemplos, puedes agregar todas las tablas con los campos necesarios.

2. Diseño de la Base de Datos

1. **Estructura y normalización:** Estructurar bien la base de datos para evitar redundancias.
2. **Inserción de datos iniciales:** Agrega ejemplos basados en el contenido del PDF, como temas de arrays, pilas, colas, etc.

3. Creación del Frontend con Bootstrap 5

1. **Diseño de interfaz principal:**
   - ❖ Usa **tarjetas (cards)** para mostrar cada concepto de estructura de datos.
   - ❖ Implementa un **navbar** para la navegación entre secciones: Arrays, Pilas, Colas, Listas, Grafos, Árboles, Ordenación.
   - ❖ Crea un **layout responsivo** con Bootstrap Grid System para asegurar compatibilidad en dispositivos móviles.

2. **Página de inicio (index.php):**
   - ❖ Muestra una introducción y una lista de categorías.
   - ❖ Cada tarjeta debe tener un botón que lleve a una página de detalles.

4. Backend en PHP

1. **Conexión a la base de datos:**
   - ✓ Configura la conexión usando un archivo config.php para almacenar las credenciales de la base de datos.
   - ✓ Implementa clases o funciones para CRUD (Create, Read, Update, Delete) de la tabla contenido.
2. **Operaciones CRUD:**
   - ✓ **Añadir nuevo contenido:** Página que permita insertar nuevos temas de estructuras de datos.
   - ✓ **Modificar contenido:** Interfaz para editar detalles de cada entrada.
   - ✓ **Eliminar contenido:** Confirmación para eliminar registros.
   - ✓ **Mostrar contenido:** Usa Bootstrap para formatear la presentación.

5. Diseño de Interfaz para el Contenido

1. **Página de visualización (contenido.php):**
   - ✓ Visualiza detalles completos de un concepto, mostrando título, descripción y archivos adicionales (si los hay).
   - ✓ Incluye botones para **Editar** y **Eliminar** si el usuario tiene permisos.
2. **Formulario de gestión (gestion.php):**
   - ✓ Formulario para añadir o modificar temas.
   - ✓ Validación en el frontend (JavaScript) y backend (PHP).
   - ✓ Muestra mensajes de error justo debajo de los campos relevantes en caso de validaciones fallidas.

6. Autenticación y Seguridad

1. **Sistema de inicio de sesión y registro (login.php, register.php):**
   - ✓ Los usuarios deben registrarse y autenticarse para agregar o modificar contenido.
   - ✓ Usa password_hash para almacenar contraseñas y password_verify para validarlas.
2. **Control de acceso:**
   - ✓ Solo los usuarios autenticados pueden acceder a la gestión de contenido.
   - ✓ Implementa middleware simple para verificar sesiones de usuario.

7. Estilo y Diseño Responsivo con Bootstrap

1. **Interactividad:**
   - ✓ Añade **modales** para agregar y editar contenido sin recargar la página.
   - ✓ Usa **carouseles** o **acordeones** para secciones con mucho contenido.
2. **Colores y temas:**
   - ✓ Escoge una paleta de colores que sea profesional y legible.
   - ✓ Asegúrate de que todos los componentes sean accesibles para personas con discapacidades visuales.

8. Extensiones Opcionales

1. **Búsqueda dinámica:** Implementa un sistema de búsqueda en la página principal para filtrar conceptos.
2. **Archivos adjuntos:** Permite a los usuarios cargar documentos PDF o imágenes adicionales que se mostrarán junto con el contenido.
3. **Estadísticas:** Muestra estadísticas simples como el número de conceptos por categoría.

9. Pruebas y Validación

1. **Verifica la funcionalidad:** Prueba todas las operaciones CRUD para asegurar que funcionan correctamente.
2. **Revisión de diseño:** Asegúrate de que la página sea **responsiva** y **amigable** en dispositivos móviles.
3. **Pruebas de seguridad:** Revisa que no haya **vulnerabilidades comunes** como SQL Injection.

10. Entrega y Evaluación

1. **Entrega del código:** Los alumnos deben presentar el código completo, organizado en carpetas claras para frontend, backend y base de datos.
2. **Informe:** Incluye una pequeña documentación explicando la estructura del proyecto, decisiones de diseño y capturas de pantalla.
3. **Presentación:** Cada grupo debe presentar su trabajo en clase, destacando las características implementadas y cómo lograron interactividad y diseño responsivo.

Evaluación

- ✓ **Estructura y organización del código**: 30%
- ✓ **Funcionalidad completa del sistema**: 30%
- ✓ **Uso de Bootstrap y diseño visual**: 20%
- ✓ **Seguridad y validación de datos**: 10%
- ✓ **Documentación y presentación**: 10%