Seguimiento Análisis de algoritmos 2.2

Alejandro Giraldo Herrera

Institución universitaria EAM

52410: Ingeniería de software

Ing. Fernando Tovar Herrera

17 de Abril, 2023

1. Pseudocódigo: Actualizar el algoritmo diseñado en clase para Ordenamiento y búsqueda, así: Código fuente.

Código por si el enlace no funciona:

```
# Programa que busca un valor en un arreglo de datos dados
# después de organizar este.

"""Función de solicitud de datos"""
def solitar_datos() -> tuple:
    datos: list[int] = []
    print("Datos del arreglo:")
    while True:
        numero = input("Ingrese un número. (Ingrese un valor diferente a
un número para terminar)\nNúmero: ")
        if numero.isdigit():
            datos.append(int(numero))
        else:
            break

    while True:
        a_buscar = input("Ingrese el número a buscar: ")
        if a_buscar.isdigit():
            a_buscar = int(a_buscar)
            break

    return datos, a_buscar

"""Función encargada de ordenar."""
def ordenar(arreglo: list[int], largor: int) -> None:
    if largor <= 1:
        return

    # Ordena el último elemento del arreglo
    ordenar(arreglo, largor - 1)
    # Último valor de la lista ordenado
    ultimo: int = arreglo[largor - 1]
    # Iterador que controlará el flujo a continuación, su valor es el
penúltimo valor
    iterador: int = largor - 2

    # Mientars que el iterador no sea negativo y sea mayor que el
último,
```

```
    # se corren los espacios hacia la derecha
    while iterador >= 0 and arreglo[iterador] > ultimo:
        arreglo[iterador + 1] = arreglo[iterador]
        iterador -= 1

    # Coloca el valor en su posición correcta
    arreglo[iterador + 1] = ultimo
"""Función para buscar un valor indicado en una lista dada."""
def busqueda(arreglo: list[int], dato: int, izquierda: int, derecha:
int) -> int:
    # Dado que la parte derecha es menor que la izquierda
    # el dato no ha sido encontrado.
    if izquierda > derecha:
        return -1

    indice_del_medio: int = (izquierda + derecha) // 2
    elemento_del_medio: int = arreglo[indice_del_medio]

    # Se retorna el índice del dato del medio si este es el dato
buscado.
    if elemento_del_medio == dato: return indice_del_medio
    if dato < elemento_del_medio:
        return busqueda(arreglo, dato, izquierda, indice_del_medio - 1)
    else:
        return busqueda(arreglo, dato, indice_del_medio + 1, derecha)

"""Función main encargada de arrancar la aplicación."""
def main() -> None:
    datos, dato_a_buscar = solitar_datos()
    print(f"Arreglo original: {str(datos)}")
    ordenar(datos, len(datos))
    print(f"Datos  ordenados: {str(datos)}")
    encontrado: int = busqueda(datos, dato_a_buscar, 0, len(datos) - 1)
    mensaje: str
    if encontrado == -1:
        mensaje = f"El dato no se ha encontrado."
    else:
        mensaje = f"El dato se ha encontrado en la posición
{encontrado}"

    print(mensaje)

# Condición que se encarga de llamar el método main dada la
```

```
# condincición que se ejecute el módulo principal.
if __name__ == "__main__":
    main()
```

2. Herramientas de software.

2.1. Con excel:

2.1.1. Cómo ordenar un array (una columna).

R/ Para ordenar un array en excel, debemos de seguir los siguientes pasos:

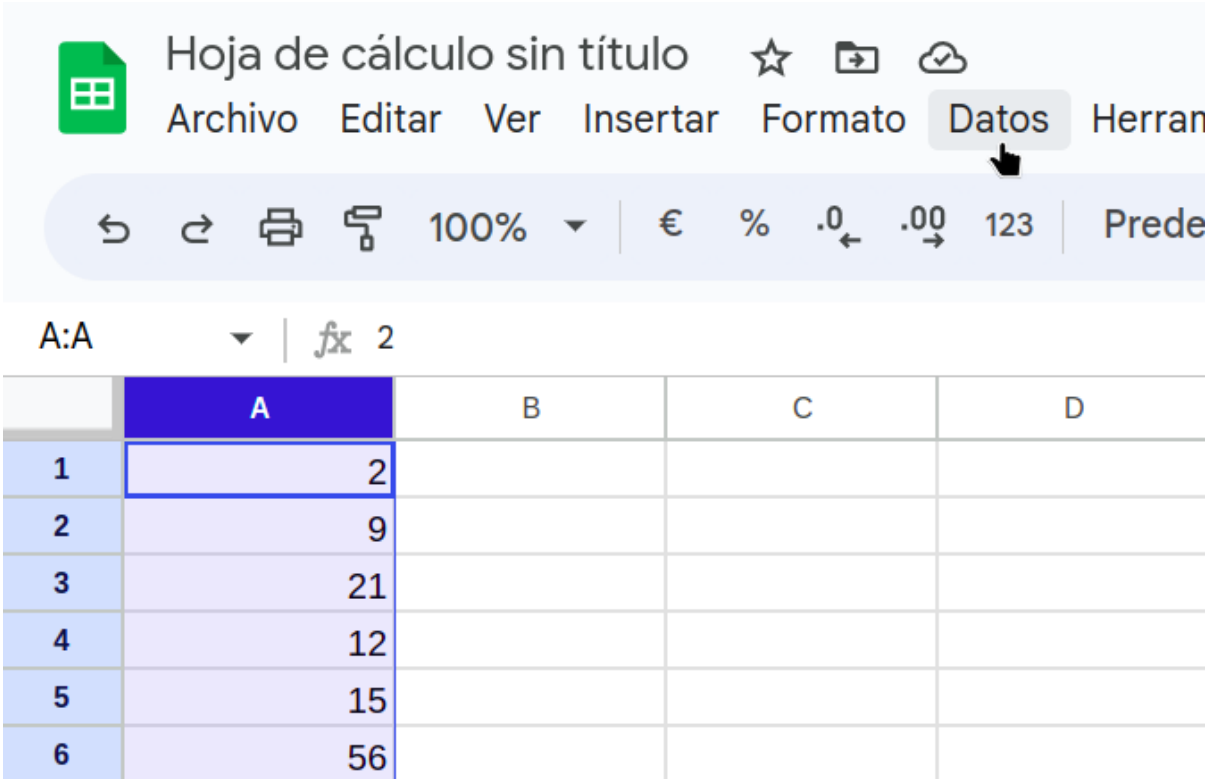
1. Agregamos los datos en una sola columna:

	A
1	2
2	9
3	21
4	12
5	15
6	56

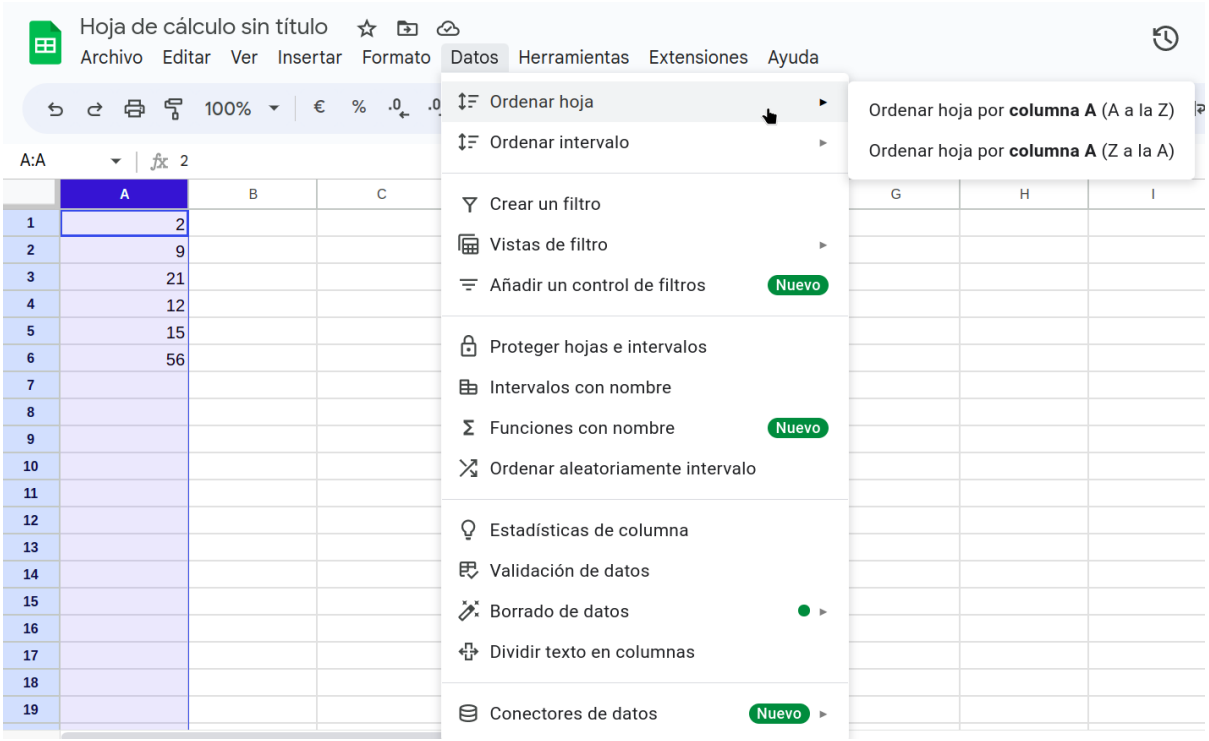
2. Seleccionamos la letra de la columna para seleccionar todos los datos en esa columna:

	A
1	2
2	9
3	21
4	12
5	15
6	56

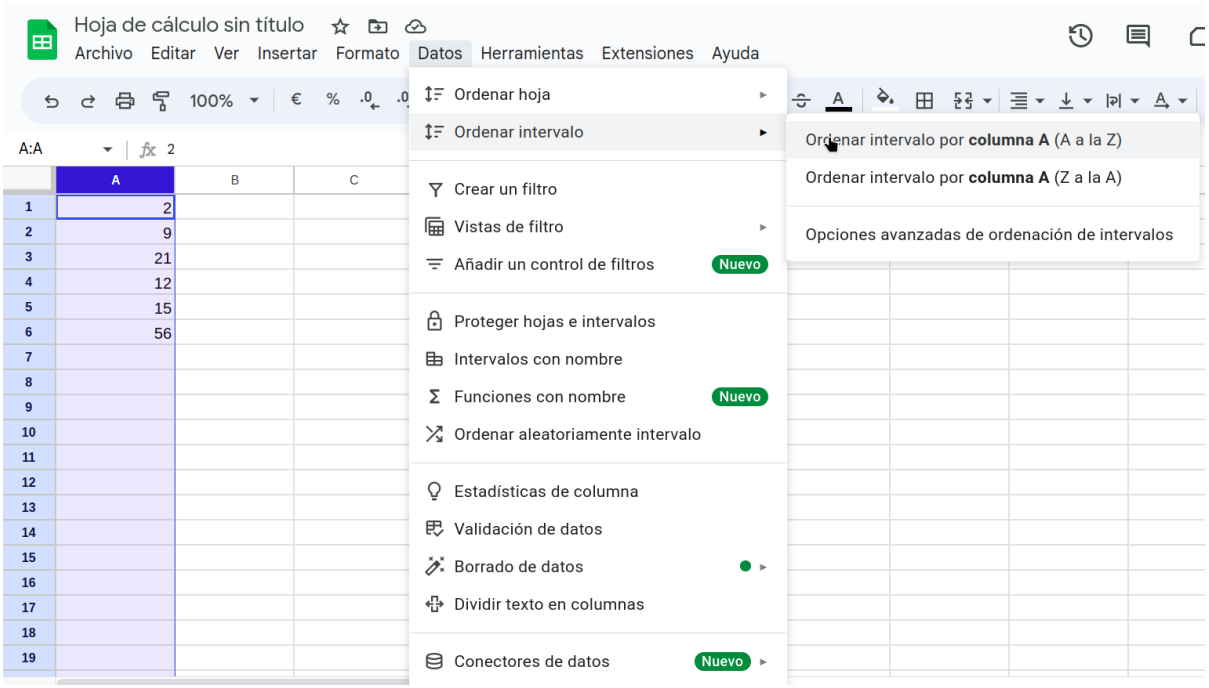
3. Seleccionamos la pestaña de datos:



4. Seleccionamos la opción de Ordenar intérvalo:



5. Seleccionamos ordenar intérvalo por columna A(A a la Z):



6. Veremos que los datos se han ordenado:

	A
1	2
2	9
3	12
4	15
5	21
6	56

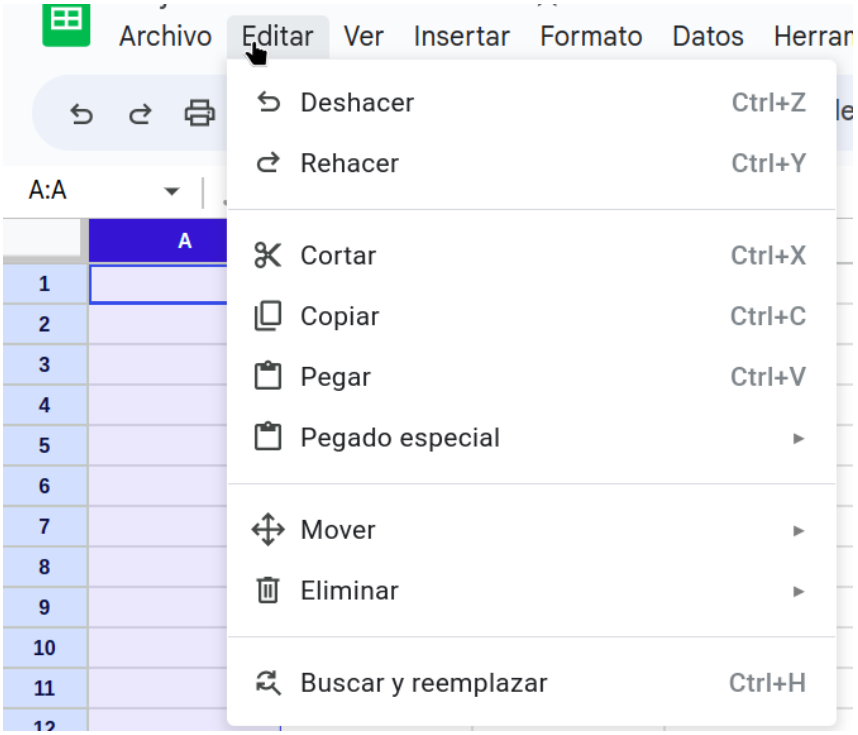
2.1.2. Cómo buscar un dato en una array (una columna):

R/ Para buscar un elemento en el array de datos ya ordenado hacemos lo siguiente:

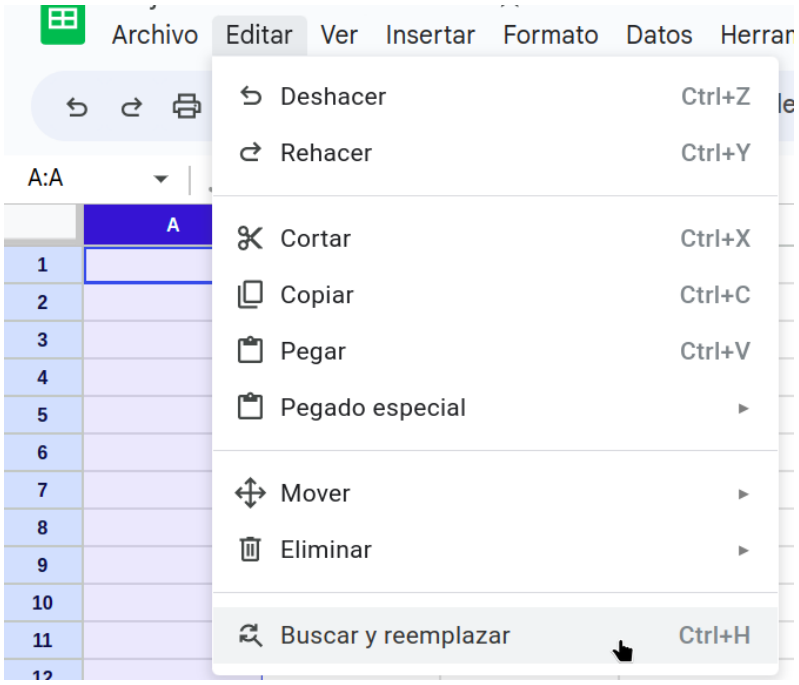
1. Seleccionamos la columna seleccionando su letra:

	A
1	2
2	9
3	12
4	15
5	21
6	56

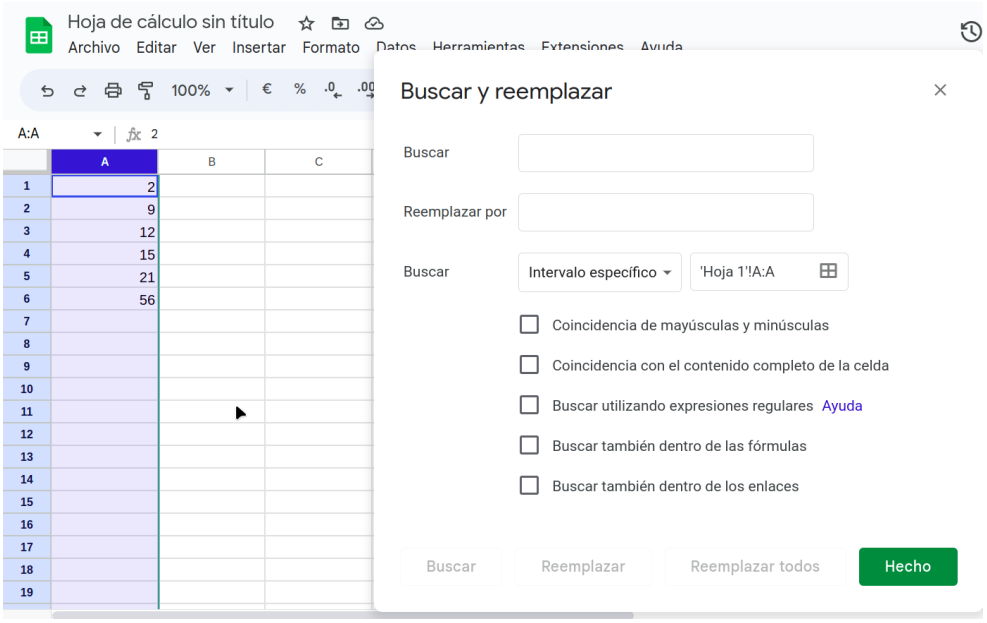
2. Seleccionamos la opción de Editar:



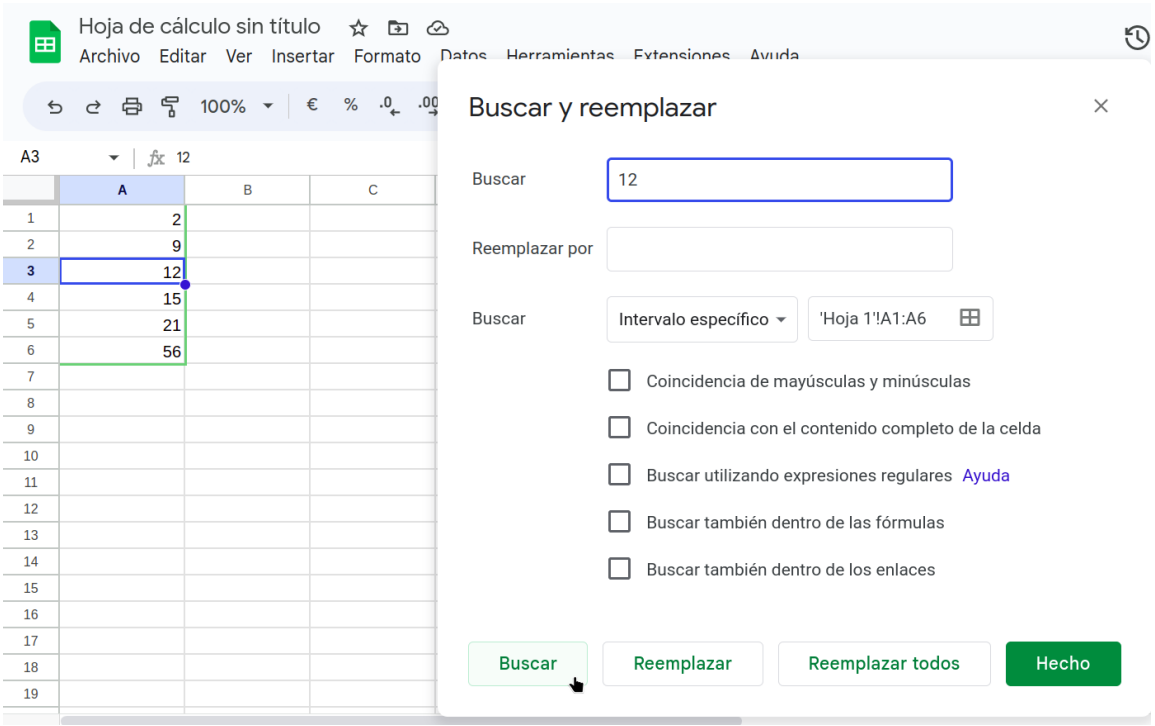
3. Seleccionamos la opción de Buscar y Reemplazar:



4. Ingresamos el valor a buscar:



5. Al darle buscar, nos mostrará todas las coincidencias:



2.2. Con java: Pequeño código fuente (sin ciclos en el ordenamiento y en la búsqueda) para:

2.2.1. Cómo ordenar un array: Código fuente

```
import java.util.ArrayList;
import java.util.Arrays;

// Explicación de métodos que proporciona Java para el
ordenamiento de datos.
public class Ordenar {
```

```
        public static void main(String[] args) {
            // Se puede definir un arreglo estático y ordenarlo con el
metodo
            // sort de la clase Arrays
            int[] datos = { 2, 9, 21, 12, 15, 56 };
            Arrays.sort(datos);
            String datosOrdenados = Arrays.toString(datos);
            System.out.println("Datos ordenados del array estático: "
+ datosOrdenados); // Retorna [2, 9, 12, 15, 21, 56]

            // O directamente se puede definir un Arraylist y acceder
al
            // método sort de la clase ArrayList
            ArrayList<Integer> datos2 = new ArrayList<>();
            datos2.add(2);
            datos2.add(9);
            datos2.add(21);
            datos2.add(12);
            datos2.add(15);
            datos2.add(56);
            datos2.sort(null);
            System.out.println("Datos ordenados del array dinámico: "
+ datos2); // Retorna [2, 9, 12, 15, 21, 56]
        }

    }
```

2.2.2. Cómo buscar un dato en un array (o lista): Código fuente

```
import java.util.ArrayList;
import java.util.Arrays;

public class Busqueda {

    public static void main(String[] args) {
        // Se puede definir un arreglo estático y buscar con el
metodo
        // contains de la clase Arrays
        int[] datos = { 2, 9, 21, 12, 15, 56 };
        int datoABuscar = 9;
        System.out.println(Arrays.binarySearch(datos,
datoABuscar)); // Retorna true 1. Ha sido encontrado.

        // También se puede definir un ArryLis y acceder
directamente
        // a su método de búsqueda.
        ArrayList<Integer> datos2 = new ArrayList<>();
        datos2.add(2);
        datos2.add(9);
        datos2.add(21);
        datos2.add(12);
        datos2.add(15);
        datos2.add(56);
        System.out.println(datos2.contains(datoABuscar)); //
Retorna true. Ha sido encontrado.
    }
}
```