



ESCUELA POLITÉCNICA NACIONAL

INGENIERÍA DE SOFTWARE

**Estructura de Datos y Algoritmos I
ICCD343**

CASOS DE RECURSIVIDAD

**Alumnos: Juan Mateo Quisilema, Fernando
Huilca, Sebastián Ramos.**

PROFESORA: Dra. Mayra CARRION

FECHA DE ENTREGA: 01 de junio de 2024



RECURSIVIDAD

A. PROBLEMA: Investigar y demostrar su conocimiento en recursividad.

VIDEO:

https://youtu.be/XXi59C7_MJk.

Presente 4 casos interesante de recursividad con llamadas de un subprograma a otro subprograma, o varias recursividades en una clase)

a. Algoritmo1:

1. ALGORITMO BusquedaBinariaRecursiva
2. ENTRADA: Lista ordenada de elementos, valor a buscar, inicio, fin
3. SALIDA: Índice del valor si se encuentra, -1 si no se encuentra
4. INICIO
5. SI inicio > fin ENTONCES
6. RETORNAR -1 // Caso base: valor no encontrado
7. FIN SI
8. $\text{medio} \leftarrow (\text{inicio} + \text{fin}) \text{ DIV } 2$
9. SI Lista[medio] = valor ENTONCES
10. RETORNAR medio // Caso base: valor encontrado
11. FIN SI
12. SI Lista[medio] > valor ENTONCES
13. RETORNAR BusquedaBinariaRecursiva(Lista, valor, inicio, medio - 1) // Buscar en la mitad izquierda
14. SINO
15. RETORNAR BusquedaBinariaRecursiva(Lista, valor, medio + 1, fin) // Buscar en la mitad derecha
16. FIN SI
17. FIN

b. Implementación:



ESCUELA POLITÉCNICA NACIONAL FACULTAD DE INGENIERÍA DE SISTEMAS

```
1 import javax.swing.*;
2
3 public class BusquedaBinaria {
4     private int[] listaAux; // Declaración del arreglo donde se almacenarán los números
5
6     public BusquedaBinaria(int dimension) {
7         this.listaAux = new int[dimension]; // Inicialización del arreglo con la dimensión dada
8         JOptionPane.showMessageDialog(parentComponent: null, message: "** RECUERDE QUE SI LA LISTA NO ESTA ORDENADA EL ALGORITMO NO FUNCIONARÁ **"); // Mensaje de advertencia
9         inicializarArreglo(); // Llamada al método para llenar el arreglo
10    }
11
12    private void inicializarArreglo() {
13        for (int i = 0; i < listaAux.length; i++) {
14            int dataAux = Integer.parseInt(JOptionPane.showInputDialog("Ingrese el elemento " + (i + 1) + ":")); // Solicitud de entrada para cada elemento del arreglo
15            listaAux[i] = dataAux; // Asignación del elemento al arreglo
16        }
17    }
18
19    public void implementarBusquedaBinaria(int datoABuscar) {
20        int indiceAux = buscarIndiceDeSolucion(indiceMedio: this.listaAux.length / 2, datoABuscar); // llamada al método recursivo de búsqueda binaria
21        if (indiceAux == -1) {
22            JOptionPane.showMessageDialog(parentComponent: null, message: "El elemento no se encontró :3"); // Mensaje si el elemento no se encuentra
23            return;
24        }
25        JOptionPane.showMessageDialog(parentComponent: null, message: "El valor buscado está en la posición: " + (indiceAux + 1)); // Mensaje con la posición del elemento encontrado
26    }
27 }
```

c. Resultado de ejecución:

```
22 // Ejecutar la acción correspondiente a la opción seleccionada
23 switch (opcion) {
24     case 1:
25         int dimension = Integer.parseInt(JOptionPane.showInputDialog("Ingrese la dimensión de la lista:"));
26         BusquedaBinaria busquedaBinaria = new BusquedaBinaria(dimension);
27         int valorABuscar = Integer.parseInt(JOptionPane.showInputDialog("Ingrese el valor a buscar:"));
28         busquedaBinaria.implementarBusquedaBinaria(valorABuscar);
29         break;
30     case 2:
31         int numeroAux = Integer.parseInt(JOptionPane.showInputDialog("Ingrese el número a sumar sus datos:"));
32         SumaDigitosDeUnNumero sumaDigitosDeUnNumero = new SumaDigitosDeUnNumero();
33         sumaDigitosDeUnNumero.sumarDigitos(numeroAux);
34         break;
35     case 3:
36         int numeroAux = Integer.parseInt(JOptionPane.showInputDialog("Ingrese el número de los dígitos:"));
37         SumaDigitosDeUnNumero sumaDigitosDeUnNumero = new SumaDigitosDeUnNumero();
38         sumaDigitosDeUnNumero.sumarDigitos(numeroAux);
39         break;
40 }
```

Input

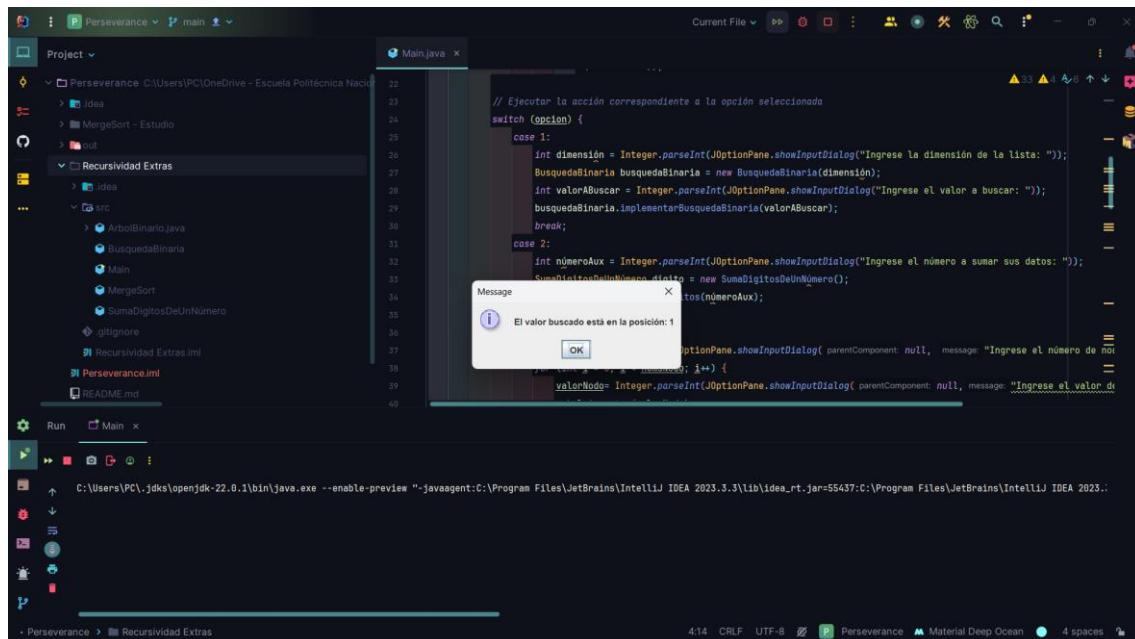
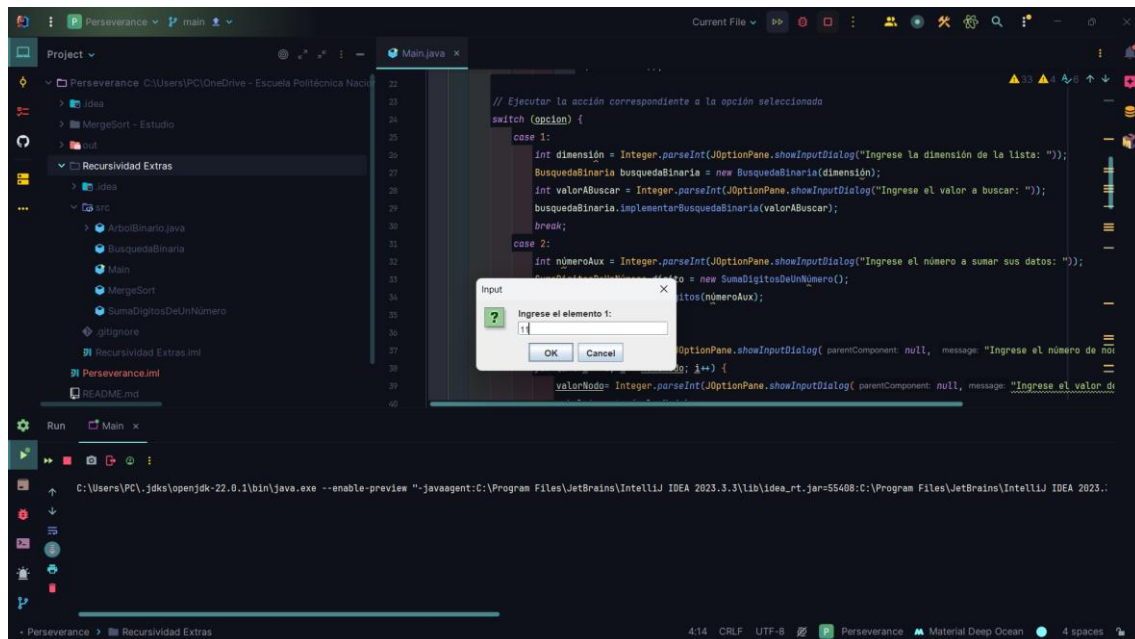
Ingrese la dimensión de la lista:

4

OK Cancel



ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS



d. Algoritmo2:

1. ALGORITMO SumaDeDigitosRecursiva
2. ENTRADA: Número entero n
3. SALIDA: Suma de los dígitos de n
4. INICIO
5. SI $n = 0$ ENTONCES
6. RETORNAR 0 // Caso base: Si el número es 0, la suma de sus dígitos es 0
7. SINO
8. $ULTIMO_DIGITO \leftarrow n \bmod 10$ // Obtener el último dígito de n



e. Implementación:

f. Resultados ejecución:

The screenshot shows an IDE with a Java project named 'Perseverance'. The main file 'Main.java' is open, displaying a program that interacts with the user to sort an array. The code includes methods for inserting, printing, and sorting an array. An 'Input' dialog box is open, prompting the user to enter a number to sum their data, with the value '123' entered. The code also includes a message dialog box that says 'Agregar código Java.'.

```

40      arbol.insertar(valorNode);
41    }
42    arbol.imprimirArbol();
43    break;
44    case 4:
45      int tamaño;
46      tamaño = Integer.parseInt(JOptionPane.showInputDialog("Ingrese el tamaño de su arreglo: "));
47      int[] arreglo = inicializarArreglo(tamaño);
48      StringBuilder arregloInicial = new StringBuilder("----- ARREGLO INICIAL ----- \n\n");
49      devolverStringArreglo(arreglo, arregloInicial);
50      mergeSort.mergeSort(arreglo, left: 0, right: arreglo.length - 1);
51      StringBuilder arregloOrdenado = new StringBuilder("----- ARREGLO ORDENADO ----- \n\n");
52      devolverStringArreglo(arreglo, arregloOrdenado);
53      JOptionPane.showMessageDialog(parentComponent: null, message: arregloInicial.toString() + "\n\n" + arregloOrdenado.toString());
54
55      JOptionPane.showMessageDialog(parentComponent: null, message: "Agregar código Java.");
56
57      JOptionPane.showMessageDialog(parentComponent: null, message: "Agregar código Java.");
58

```



ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

g. Algoritmo3:

ALGORITMO InsertarNodoArbolRecursivo

ENTRADA: Nodo nodoActual, valorNodo a insertar

SALIDA: Nodo actualizado con el nuevo valor insertado

INICIO

SI nodoActual es null ENTONCES

RETORNAR NuevoNodo(valorNodo) // Caso base: creación de un nuevo
nodo con el valor especificado

FIN SI

SI valorNodo < nodoActual.valorNodo ENTONCES

nodoActual.izquierdo ←
InsertarNodoArbolRecursivo(nodoActual.izquierdo, valorNodo) // Insertar
en el subárbol izquierdo

SINO SI valorNodo > nodoActual.valorNodo ENTONCES

nodoActual.derecho ←
InsertarNodoArbolRecursivo(nodoActual.derecho, valorNodo) // Insertar en
el subárbol derecho

FIN SI

RETORNAR nodoActual

FIN



ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

h. Implementación:

```
import javax.swing.JOptionPane;

class Nodo{
    /**
     * Esta clase define un nodo del árbol binario con un valor y referencias a los nodos
     * izquierdo y derecho.
     */
    int valorNodo;
    Nodo izquierdo;
    Nodo derecho;
    public Nodo(int valorNodo){
        this.valorNodo= valorNodo;
        this.izquierdo=null;
        this.derecho=null;
    }
}

public class ArbolBinario {
    /**
     * Esta clase define un árbol con una raíz inicialmente nula
     */
    private Nodo raiz;
    public ArbolBinario(){
        this.raiz=null;
    }

    public void insertar(int valorNodo){
        raiz= insertarRecursivo(raiz, valorNodo);
    }

    public Nodo insertarRecursivo (Nodo nodoActual, int valorNodo){ //Inserta recursivamente un valor en el nodo
        if(nodoActual==null){
            return new Nodo(valorNodo);
        }
        if(valorNodo<nodoActual.valorNodo){
            nodoActual.izquierdo= insertarRecursivo(nodoActual.izquierdo, valorNodo);
        }else if (valorNodo> nodoActual.valorNodo){
            nodoActual.derecho= insertarRecursivo(nodoActual.derecho, valorNodo);
        }
        return nodoActual;
    }

    public void imprimirArbol(){
        imprimirArbolRecursivo(raiz);
    }

    public void imprimirArbolRecursivo(Nodo nodoActual){ //Imprime el arbol de manera recursiva
        if(nodoActual!=null){
            imprimirArbolRecursivo(nodoActual.izquierdo);
            JOptionPane.showMessageDialog(null, nodoActual.valorNodo+" ");
            imprimirArbolRecursivo(nodoActual.derecho);
        }
    }
}
```

i. Resultados ejecución:

Entrada

?

RECURSIVIDAD CASOS EXTRAS

1. Búsqueda Binaria .
2. Suma de dos dígitos?? xd.
3. Árboles.
4. no se aun.
0. Salir.

3

Aceptar Cancelar



Entrada

×

?

Ingrese el número de nodos que tendrá el árbol

2

Aceptar

Cancelar

Entrada

×

?

Ingrese el valor del nodo 1:

1

Aceptar

Cancelar

Entrada

×

?

Ingrese el valor del nodo 2:

2

Aceptar

Cancelar

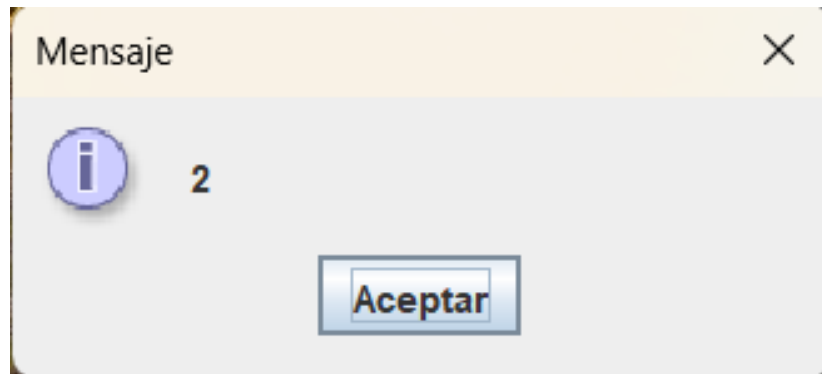
Mensaje

×

i

1

Aceptar



j. Algoritmo4: Merge Sort

1. INICIO
2. SI $\text{left} < \text{right}$ ENTONCES
3. $\text{mid} \leftarrow (\text{left} + \text{right}) / 2$
4. // Ordenar la primera mitad
5. MergeSort(array, left, mid)
6. // Ordenar la segunda mitad
7. MergeSort(array, mid + 1, right)
8. // Combinar las dos mitades ordenadas
9. Merge(array, left, mid, right)
10. FIN SI
11. FIN

k. Implementación:



ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

```
Main.java MergeSort.java x
1 public class MergeSort {
2
3     // Función principal que implementa Merge Sort
4     public void mergeSort(int[] array, int left, int right) {
5         if (left < right) { //Caso base
6             int mid = (left + right) / 2;
7
8             // Ordena la primera y segunda mitad
9             mergeSort(array, left, mid);
10            mergeSort(array, left: mid + 1, right);
11
12            // Combina las dos mitades ordenadas
13            merge(array, left, mid, right);
14        }
15    }
16
17    // Función para combinar dos mitades
18    public void merge(int[] array, int left, int mid, int right) {
19        // Encuentra el tamaño de los subarrays a combinar
20        int n1 = mid - left + 1;
21        int n2 = right - mid;
22
23        // Crea arrays temporales
24        int[] leftArray = new int[n1];
25        int[] rightArray = new int[n2];
26
27        // Copia los datos a los arrays temporales
28        System.arraycopy(array, left, leftArray, destPos: 0, n1);
29        System.arraycopy(array, srcPos: mid + 1, rightArray, destPos: 0, n2);
30
31        // Índices iniciales de los subarrays y del array combinado
32        int i = 0, j = 0, k = left;
33
34        // Combina los arrays temporales de vuelta en el array original
35        while (i < n1 && j < n2) {
36            if (leftArray[i] <= rightArray[j]) {
37                array[k] = leftArray[i];
38                i++;
39            } else {
40                array[k] = rightArray[j];
41                j++;
42            }
43            k++;
44        }
45
46        // Copia los elementos restantes de leftArray, si hay alguno
47        while (i < n1) {
48            array[k] = leftArray[i];
49            i++;
50            k++;
51        }
52
53        // Copia los elementos restantes de rightArray, si hay alguno
54        while (j < n2) {
55            array[k] = rightArray[j];
56            j++;
57            k++;
58        }
59    }
60 }
```

I. Resultados ejecución:



ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Input

?

RECURSIVIDAD CASOS EXTRAS

1. Búsqueda Binaria .

2. Suma de dos dígitos.

3. Árboles.

4. Merge Sort

0. Salir.

4

OK

Cancel

Input

?

Ingrese el tamaño de su arreglo:

5

OK

Cancel

Input

?

Ingrese el número del índice 0/4:

12

OK


Cancel



ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Input

×




Ingrese el número del índice 4/4:

OK

Cancel

Message

×



_____ ARREGLO INICIAL _____

12 | 13 | 11 | 5 | 7

_____ ARREGLO ORDENADO _____

5 | 7 | 11 | 12 | 13

OK