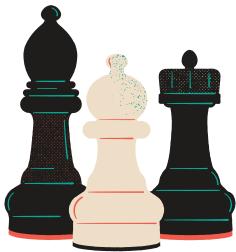


# Actividad 3

## Estructura de Datos

Nombre: Isaac Alejandro Isaias Betance

Actividad: Actividad 3



# Reporte Actividad 3


## Introducción

En esta actividad tuve que aplicar recursividad, algoritmos de backtracking y divide y vencerás. La neta sí estuvo algo pesado, pero me puse a buscar tutoriales en YouTube y de ahí fui agarrando la onda. Al final logré sacar un programa en Java con un menú interactivo donde metí todos los problemas que pedían y hasta algunos extras.

## Desarrollo

### -Problema 1: Fibonacci recursivo

Aquí hice la función con su caso base y el caso recursivo. Bien sencillo pero importante pa' agarrar la base de la recursividad.

```
1  import java.util.*;
2  
3  // hey profe, esta es mi actividad 3 de recursividad
4
5
6  public class Actividad3_Recursividad {
7
8      // 1. Serie de Fibonacci recursiva
9      public static int fibonacci(int n) {
10         if (n <= 1) return n; // caso base
11         return fibonacci(n - 1) + fibonacci(n - 2); // caso recursivo
12     }
13 }
```

```
===== MENU ACTIVIDAD 3 =====
1. Fibonacci recursivo
2. Subset Sum (Suma de subconjuntos)
3. Sudoku recursivo (varias soluciones)
4. Factorial recursivo
5. Torres de Hanoi
6. Problema de las 8 reinas
7. Salir
Elige: 1
Dame un número: 21
Fibonacci de 21 = 10946
```

### -Problema 2: Subset Sum

Armé el algoritmo recursivo para checar si existía un subconjunto que sumara al valor objetivo. Estuvo más tricky, pero lo resolví aplicando divide y vencerás.

```

14 // 2. Subset Sum (suma de subconjuntos)
15 public static boolean subsetSum(int[] conjunto, int n, int objetivo) {
16     if (objetivo == 0) return true; // caso base
17     if (n == 0 && objetivo != 0) return false;
18     if (conjunto[n - 1] > objetivo) return subsetSum(conjunto, n - 1, objetivo);
19     return subsetSum(conjunto, n - 1, objetivo) || subsetSum(conjunto, n - 1, objetivo - conjunto[n - 1]);
20 }
21

```

Elige: 2  
 Objetivo: 20  
 Existe subconjunto? true

### -Problema 3: Sudoku con backtracking

Aquí estuvo lo bueno, implementé el backtracking para que resolviera el tablero y sacara varias soluciones. Me costó porque al inicio no encontraba soluciones, pero ya luego ajusté bien las condiciones y funcionó.

```

22 // 3. Sudoku con backtracking (modo turbo pa que saque varias soluciones)
23 static int N = 9;
24 static int solucionesMax = 3;
25 static int contador = 0;
26
27 public static boolean esValido(int[][] tablero, int fila, int col, int num) {
28     for (int x = 0; x < 9; x++) {
29         if (tablero[fila][x] == num || tablero[x][col] == num) return false;
30     }
31     int startRow = fila - fila % 3, startCol = col - col % 3;
32     for (int i = 0; i < 3; i++) {
33         for (int j = 0; j < 3; j++) {
34             if (tablero[i + startRow][j + startCol] == num) return false;
35         }
36     }
37     return true;
38 }
39
40 public static void resolverSudoku(int[][] tablero, int fila, int col) {
41     if (fila == N - 1 && col == N) {
42         contador++;
43         System.out.println("\nSolución #" + contador + ":");
44         imprimirSudoku(tablero);
45         if (contador >= solucionesMax) return;
46         return;
47     }
48     if (col == N) {
49         fila++;
50         col = 0;
51     }
52     if (tablero[fila][col] != 0) {
53         resolverSudoku(tablero, fila, col + 1);
54         return;
55     }
56     for (int num = 1; num <= 9; num++) {
57         if (esValido(tablero, fila, col, num)) {
58             tablero[fila][col] = num;
59             resolverSudoku(tablero, fila, col + 1);
60             tablero[fila][col] = 0;
61         }
62     }
63 }
64

```

```
Elige: 3
```

```
Solución #1:
```

```
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```

Además le metí un menú interactivo para poder elegir la opción que quiero sin que el programa se cierre.

```
===== MENU ACTIVIDAD 3 =====
1. Fibonacci recursivo
1. Fibonacci recursivo
2. Subset Sum (Suma de subconjuntos)
3. Sudoku recursivo (varias soluciones)
4. Factorial recursivo
4. Factorial recursivo
5. Torres de Hanoi
5. Torres de Hanoi
6. Problema de las 8 reinas
6. Problema de las 8 reinas
7. Salir
Elige: 3
```

```
Solución #1:
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```

```
===== MENU ACTIVIDAD 3 =====
1. Fibonacci recursivo
2. Subset Sum (Suma de subconjuntos)
3. Sudoku recursivo (varias soluciones)
4. Factorial recursivo
5. Torres de Hanoi
6. Problema de las 8 reinas
7. Salir
Elige: █
```

## Conclusión

En resumen, batallé con algunas partes, sobre todo Sudoku y Subset Sum, pero con práctica y tutoriales lo saqué. Aprendí a usar la recursividad directa, indirecta y el backtracking en problemas reales. Al final siento que cumplí con todo lo que pedía la rúbrica y dejé el código con comentarios y organizado a mi estilo.