



TEMA 7

Recursividad

V1.1

Manuel Pereira González

Agenda



- **Introducción**
- Ejemplos
 - Factorial
 - Invertir un Número
 - Resolver Laberinto
- Cuándo utilizar Recursividad
- Algoritmos de Backtracking
 - Las Ocho Reinas
 - Sudoku
- Resumen



Introducción



- Poderosa herramienta de programación
- Alternativa a algoritmos iterativos
- Soluciones elegantes a problemas difíciles de resolver de otro modo
- **Un método es recursivo si contiene invocaciones a sí mismo**
- Una llamada a un método recursivo puede generar una o más invocaciones al mismo método, que a su vez genera otras, ...

Introducción



- Condiciones que debe cumplir un método recursivo:
 - Asegurar que existe una condición de salida, en la que no se producen llamadas recursivas (caso base)
 - Asegurar que se cubren todos los posibles casos entre el base y los no base
 - Cada llamada, en el caso no base, conduce a problemas cada vez más pequeños que terminarán en el caso base

Agenda



- Introducción
- Ejemplos
 - **Factorial**
 - Invertir un Número
 - Resolver Laberinto
- Cuándo utilizar Recursividad
- Algoritmos de Backtracking
 - Las Ocho Reinas
 - Sudoku
- Resumen



Ejemplo: Factorial



- Definición del factorial (definición recursiva):
 - $n! = n * (n-1)!$, para $n > 1$
 - $1! = 1$
- Casos y Soluciones:

Casos	Soluciones
$n = 1$ (caso base)	return 1
$n > 1$	return $n * \text{factorial}(n - 1)$

Ejemplo: Factorial



```
public class Factorial {
    public static void main(String[] args) {
        int variable = 5;
        System.out.println("El factorial de " + variable
            + " es: " + factorial(variable));
    }

    public static int factorial(int n) {
        // factorial(n) = n * factorial (n - 1)
        // factorial(1) = 1
        if(n > 1) {
            // Llamada recursiva
            int tmp = n * factorial(n - 1);
            return tmp;
        } else {
            return 1;
        }
    }
}
```

```
Command Prompt
C:\Practicas\Programacion>java Factorial
El factorial de 5 es: 120
C:\Practicas\Programacion>_
```

Agenda



- Introducción
- Ejemplos
 - Factorial
 - **Invertir un Número**
 - Resolver Laberinto
- Cuándo utilizar Recursividad
- Algoritmos de Backtracking
 - Las Ocho Reinas
 - Sudoku
- Resumen



Ejemplo: Invertir un Número



```
public class Invertir {
    public static void main(String[] args) {
        int numeroAInvertir = 457361;
        invertir(numeroAInvertir);
    }

    public static void invertir(int numero) {
        if(numero < 10) {
            System.out.print(numero);
        } else {
            System.out.print(numero % 10);
            invertir(numero / 10);
        }
    }
}
```

```
Command Prompt
C:\Practicas\Programacion>java Invertir
163754
C:\Practicas\Programacion>
```

Agenda



- Introducción
- Ejemplos
 - Factorial
 - Invertir un Número
 - **Resolver Laberinto**
- Cuándo utilizar Recursividad
- Algoritmos de Backtracking
 - Las Ocho Reinas
 - Sudoku
- Resumen



Agenda



- Introducción
- Ejemplos
 - Factorial
 - Invertir un Número
 - Resolver Laberinto
- **Cuándo utilizar Recursividad**
- Algoritmos de Backtracking
 - Las Ocho Reinas
 - Sudoku
- Resumen



Cuándo utilizar recursividad



- En general, las soluciones recursivas son menos eficientes que las iterativas (coste mayor en tiempo y memoria).
- Consejos:
 - Los algoritmos que por naturaleza son recursivos y donde la solución iterativa es complicada y debe manejarse explícitamente una pila para emular las llamadas recursivas, deben resolverse por métodos recursivos
 - Cuando haya una solución obvia al problema por iteración, debe evitarse la recursividad

Agenda



- Introducción
- Ejemplos
 - Factorial
 - Invertir un Número
 - Resolver Laberinto
- Cuándo utilizar Recursividad
- **Algoritmos de Backtracking**
 - Las Ocho Reinas
 - Sudoku
- Resumen



Algoritmos de Backtracking



- El **backtracking** o **vuelta atrás** es una técnica algorítmica de **resolución general de problemas** mediante una búsqueda sistemática de soluciones.
- Se descompone la tarea a realizar en tareas parciales y se prueba sistemáticamente cada una de estas, que a su vez se descompondrán en subtareas...
- Cuando al elegir una tarea se comprueba que no lleva a una solución, se debe **volver atrás**, y probar con otra

Agenda



- Introducción
- Ejemplos
 - Factorial
 - Invertir un Número
 - Resolver Laberinto
- Cuándo utilizar Recursividad
- Algoritmos de Backtracking
 - **Las Ocho Reinas**
 - Sudoku
- Resumen



Algoritmos de Backtracking: Ocho Reinas



```
private static final int DIMENSION = 8;

public static void main(String[] args) {
    int[][] t = new int[DIMENSION][];
    for(int i=0; i<DIMENSION; i++) {
        t[i] = new int[DIMENSION];
    }
    // Colocar reina en la fila cero
    colocarReina(t, 0);
}

public static boolean colocarReina(int[][] t, int fila) {
    if(fila == DIMENSION) {
        System.out.println("Solución encontrada:");
        imprimir(t);
        return true;
    }
    for(int columna=0; columna<DIMENSION; columna++) {
        if(!amenazada(t, fila, columna)) {
            // Colocamos la reina i-ésima
            t[fila][columna] = fila+1;
            boolean tmp = colocarReina(t, fila+1);
            if(tmp == true) {
                return true;
            }
            // Borrarnos la solución inválida
            t[fila][columna] = 0;
        }
    }
    return false;
}

public static void imprimir(int[][] lab) {
    for(int i=0; i<lab.length; i++) {
        for(int j=0; j<lab[i].length; j++) {
            System.out.print(lab[i][j]);
        }
        System.out.println();
    }
}
```

```
Command Prompt
C:\Practicas\Programacion>java OchoReinas
Solucion encontrada:
10000000
00020000
00000003
00000400
00500000
00000600
07000000
00000000
C:\Practicas\Programacion>
```

Algoritmos de Backtracking: Ocho Reinas



```
public static boolean amenazada(int[][] t, int fila, int columna) {
    // Mira si hay alguna reina en la misma fila o columna
    for(int i=0; i<DIMENSION; i++) {
        if(t[fila][i] != 0 || t[i][columna] != 0) {
            return true;
        }
    }
    // Diagonal arriba a la izquierda
    for(int f=fila, c=columna; f>=0 && c>=0; f--, c--) {
        if(t[f][c] != 0 || t[f][c] != 0) {
            return true;
        }
    }
    // Diagonal arriba a la derecha
    for(int f=fila, c=columna; f>=0 && c<DIMENSION; f--, c++) {
        if(t[f][c] != 0 || t[f][c] != 0) {
            return true;
        }
    }
    // Diagonal abajo a la izquierda
    for(int f=fila, c=columna; f<DIMENSION && c>=0; f++, c--) {
        if(t[f][c] != 0 || t[f][c] != 0) {
            return true;
        }
    }
    // Diagonal abajo a la derecha
    for(int f=fila, c=columna; f<DIMENSION && c<DIMENSION; f++, c++) {
        if(t[f][c] != 0 || t[f][c] != 0) {
            return true;
        }
    }
    return false;
}
```

Agenda



- Introducción
- Ejemplos
 - Factorial
 - Invertir un Número
 - Resolver Laberinto
- Cuándo utilizar Recursividad
- Algoritmos de Backtracking
 - Las Ocho Reinas
 - **Sudoku**
- Resumen



Algoritmos de Backtracking: Sudoku



```

public class Sudoku {
    public static final int DIMENSION = 9;

    public static void main(String[] args) {
        int[][] tablero = new int[][] {
            {0, 7, 0, 0, 0, 0, 0, 0, 8, 0},
            {0, 5, 0, 6, 0, 0, 0, 0, 0, 1},
            {0, 0, 3, 1, 4, 0, 0, 0, 0, 0},

            {9, 0, 6, 0, 5, 0, 3, 0, 0, 0},
            {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
            {0, 0, 5, 0, 2, 0, 1, 0, 7, 0},

            {0, 0, 0, 0, 6, 5, 7, 0, 0, 0},
            {5, 0, 0, 0, 0, 1, 9, 2, 0, 0},
            {0, 4, 0, 0, 0, 0, 0, 1, 0, 0},
        };

        imprimir(tablero);
        if (!resolver(tablero)) {
            System.out.println("El sudoku no tiene solución.");
        }
    }

    public static void imprimir(int[][] tablero) {
        for (int i=0; i<DIMENSION; i++) {
            if (i % 3 == 0) {
                System.out.println();
            }
            for (int j=0; j<DIMENSION; j++) {
                if (j % 3 == 0) {
                    System.out.print(" ");
                }
                System.out.print(tablero[i][j]);
            }
            System.out.println();
        }
    }

    public static boolean resolver(int[][] tablero) {
        for (int i=0; i<DIMENSION; i++) {
            for (int j=0; j<DIMENSION; j++) {
                if (tablero[i][j] != 0) {
                    continue;
                }
                for (int k=1; k<=9; k++) {
                    if (esPosibleInsertar(tablero, i, j, k)) {
                        tablero[i][j] = k;
                        boolean b = resolver(tablero);
                        if (b) {
                            return true;
                        }
                        tablero[i][j] = 0;
                    }
                }
                return false;
            }
        }
        System.out.println("Encontrada solución.");
        imprimir(tablero);
        return true;
    }
}

```

Algoritmos de Backtracking: Sudoku



```

public static boolean esPosibleInsertar(int[][] tablero, int i, int j, int valor) {
    // Mira columna
    for (int a=0; a<DIMENSION; a++) {
        if (a!=i && tablero[a][j] == valor) {
            return false;
        }
    }

    // Mira fila
    for (int a=0; a<DIMENSION; a++) {
        if (a!=j && tablero[i][a] == valor) {
            return false;
        }
    }

    // Mira cuadrado
    int y = (i / 3) * 3;
    int x = (j / 3) * 3;
    for (int a=0; a<DIMENSION/3; a++) {
        for (int b=0; b<DIMENSION/3; b++) {
            if (a!=i && b!=j && tablero[y+a][x+b] == valor) {
                return false;
            }
        }
    }
    return true;
}

```

```

C:\Practicas\Programacion>java Sudoku
070 000 000
058 600 001
003 140 000

906 050 300
000 000 000
005 020 107

000 065 700
300 001 920
040 000 010
Encontrada solución:
671 592 483
450 573 291
293 148 576

916 857 342
724 316 859
835 924 167

189 265 734
367 481 925
542 739 618
C:\Practicas\Programacion>

```

Agenda



- Introducción
- Ejemplos
 - Factorial
 - Invertir un Número
 - Resolver Laberinto
- Cuándo utilizar Recursividad
- Algoritmos de Backtracking
 - Las Ocho Reinas
 - Sudoku
- **Resumen**



Resumen



- | | |
|---|--|
| <ul style="list-style-type: none">▪ Introducción<ul style="list-style-type: none">▪ Método que se invoca a sí mismo▪ Soluciones recursivas vs iterativas▪ Ejemplos<ul style="list-style-type: none">▪ Factorial<ul style="list-style-type: none">▪ $n! = n * (n-1)!$▪ $1! = 1$▪ Invertir un Número▪ Resolver Laberinto | <ul style="list-style-type: none">▪ Cuándo utilizar<ul style="list-style-type: none">▪ Normalmente menos eficiente▪ Recomendado si solución iterativa difícil y problema de naturaleza recursiva▪ Algoritmos de Backtracking<ul style="list-style-type: none">▪ Las Ocho Reinas▪ Sudoku |
|---|--|