

# Curso 2025-26

## Práctica 1. Búsqueda Inteligente. Satisfacción de restricciones. Sesión 2

## ✓ Clase Variable

- Representa **cada casilla del sudoku**.
- Atributos clave:
  - **fila, columna**: posición en el tablero.
  - **valor**: número asignado (o None si está vacía).
  - **dominio**: lista de valores posibles (1-9, reducida por restricciones).
  - **fijada**: indica si la casilla es parte de la plantilla inicial.
- Permite:
  - Asignar y desasignar valores.
  - Actualizar y restaurar dominios durante el backtracking.

🎯 **Función:** Modelar cada celda como una variable del CSP, con su estado y opciones válidas.

## Backtracking

- Estrategia:
- Construir una solución parcial: asignación parcial que satisface las restricciones de las variables involucradas
- Extender la solución parcial, incluyendo una variable cada vez hasta llegar una solución total
- Si no se puede extender ejecutamos vuelta a atrás:
- cronológico: se elimina la última decisión

## Funcionamiento del Backtracking en Sudoku (Síntesis)

El algoritmo resuelve el sudoku asignando valores a las casillas vacías de forma secuencial, usando un enfoque recursivo:

- 1) Selecciona una casilla vacía (posición  $k$ ) y prueba un valor posible del 1 al 9.
- 2) Comprueba si la asignación es válida: no repite número en la fila, columna o subcuadro.
- 3) Si es válida:

Avanza a la siguiente casilla ( $k + 1$ ).

Si todas están llenas ( $k = n$ ), ha encontrado una solución.

- 4) Si no es válida o no hay más valores posibles:

Retrocede ( $k - 1$ ) y prueba un nuevo valor en la casilla anterior.

- 5) Si se llega al inicio ( $k = 1$ ) sin soluciones, el problema no tiene solución.

# Backtracking

---

```

procedimiento Backtracking( $k, V[n]$ ) ; Llamada inicial: Backtracking(1,  $V[n]$ )
inicio
 $V[k] = \text{Selección}(d_k)$  ; Selecciona un valor de  $d_k$  para asignar a  $x_k$ 
si Comprobar( $k, V[n]$ ) entonces
    si  $k = n$  entonces
        devolver  $V[n]$  ; Es una solución
    si no
        Backtraking( $k + 1, V[n]$ )
    fin si
si no
    si quedan_valores( $d_k$ ) entonces
        Backtraking( $k, V[n]$ )
    si no
        si  $k = 1$  entonces
            devolver  $\emptyset$  ; Fallo
        si no
            Backtraking( $k - 1, V[n]$ )
        fin si
    fin si
fin si
fin Backtracking
  
```

The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The main window displays a Python script named `main.py` for solving a Sudoku puzzle. The script includes imports for `pygame` and `copy`, and defines a 9x9 grid with some pre-filled numbers. A game window titled "Practica 1: Sudoku" is overlaid on the code, showing the current state of the puzzle. The grid is as follows:

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

On the right side of the game window, there are buttons for "Load", "BK", "FC", and "AC3". The terminal at the bottom shows the command prompt and the output of the script, which includes the pygame version and the SDL version.

The screenshot shows a Python IDE with a file explorer on the left, a main editor window, and a terminal at the bottom. The main editor window displays a Python script for a simplified Sudoku game. The script includes imports for pygame and sys, and defines constants for the game board and colors. The terminal shows the output of the script, which is a 9x9 grid of numbers.

**EXPLORADOR**

- EDITORES ABIERTOS
  - main.py
- P1 SOLUCION SIMPLE BT
  - \_\_pycache\_\_
  - m1.txt
  - m2.txt
  - main.py
  - tablero.py
  - variable.py

**main.py**

```

1 # main.py simplificado (solo BK)
2 import pygame
3 import sys
4 from tablero import *
5 from variable import *
6
7 # Constantes para el tablero
8 GREY = (220, 220, 220)
9 NEGRO = (10, 10, 10)
10 GRIS_ACTIVADO = (200, 200, 200)
11 GRIS_NORMAL = (255, 255, 255)
12 BLANCO = (255, 255, 255)
13
14 MARGEN = 5 # margen entre las casillas
15 MARGEN_DERECHO = 10 # margen entre las casillas
16 TAM = 60 # tamaño de las casillas
17 N = 9 # número de filas y columnas
18 VACIA = '0'
19
20 #####
21 # Detecta si se puede poner un número en una casilla
22 #####

```

**Sudoku Simplificado**

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

**PROBLEMAS** **SALIDA**

```

conda activate ent01
/Users/ramonrizoaldegue/ent01/bin/python "/Users/ramonrizoaldegue/Sistemas inteligentes/Sudoku Simplificado/main.py"
Hello from the pygame module!
Cargar tablero
BK

```

**ESQUEMA**

**LÍNEA DE TIEMPO**

**Terminal**

```

+ v ... ^ x
zsh
Python

```

**Status Bar**

Lín. 1, col. 1 Espacios: 4 UTF-8 LF Python 3.10.13 ('ent01': conda) Go Live

## Tareas de la sesión 2 de la Práctica 1

### Implementar Backtracking

- Acceso a variables por posición.
- Funciones para verificar **restricciones** (fila, columna, subcuadro).
- Propagación de restricciones (como eliminar valores de dominios).
- Selección de la siguiente variable no asignada
- Coordina el avance y retroceso del algoritmo.