Sudoku Solver

Intelligent Systems Inc.

Mario Elder Flores 21841245
Lourdes Zamora 21511351
Diego Ramos 21951033

Objetivos

1. **Generar** configuraciones válidas de Sudoku (Soluciones), de las cuales se pueden remover/ocultar números al azar para generar juegos garantizados a tener una solución
2. Crear un algoritmo básico que pueda **resolver Sudokus** regulares (9x9) con un número variable de números faltantes (40%).
3. Permitir crear y resolver Sudokus de **Diferentes tipos** (4x4), (9x9) y (16x16)
4. Crear una **visualización** de cómo se resuelve el sudoku paso a paso

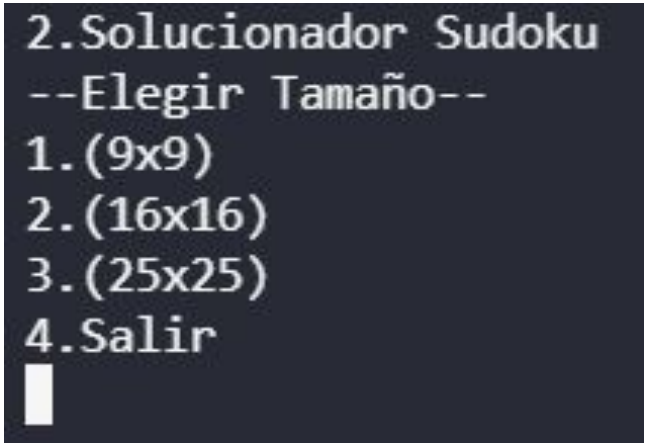
¿Que se logró, que no se ha logrado?

- Generador de tableros de distintos tamaños = 100%
- Interfaz de Visualización = Cheque
- Sistema Inteligente capaz de resolver el tablero = 99% fe (Sigue en desarrollo)

Va atrasado el sprint…

Métodos de implementación – Generación de Tableros Sudoku

Para la generación de tableros Sudoku con solución se empleó una interfaz simple en donde permite la selección de qué tamaño de tablero desea el usuario crear con ayuda del programa.
Ejemplo:



Métodos de implementación – Generación de Tableros Sudoku

Al seleccionar el tamaño de tablero que se desea se llama a la misma función que se encarga de crear el arreglo bidimensional que representa el tablero Sudoku en este caso. En esta función se utiliza un patrón que permite que todo tablero generado por el programa sea válido.

```
def createBoard(base):
    side   = base*base

    def pattern(r,c): return (base*(r%base)+r//base+c)%side

    def shuffle(s): return sample(s,len(s))
    rBase = range(base)
    rows  = [ g*base + r for g in shuffle(rBase) for r in shuffle(rBase) ]
    cols  = [ g*base + c for g in shuffle(rBase) for c in shuffle(rBase) ]
    nums  = shuffle(range(1,base*base+1))

    board = [ [nums[pattern(r,c)] for c in cols] for r in rows ]
    removeNumbers(side,board)
```

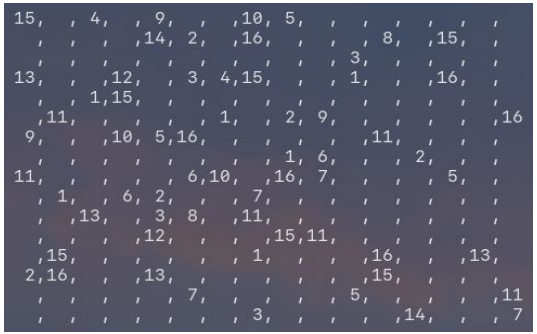
Métodos de implementación – Generación de Tableros Sudoku

Para finalizar el proceso de generación de tableros, se agarra el arreglo bi-dimensional con que representa el tablero y se le eliminan números de forma tal que represente un problema común de Sudoku mientras mantiene su capacidad de resolución y al final este paso se almacena en un csv con el tamaño del tablero como nombre.

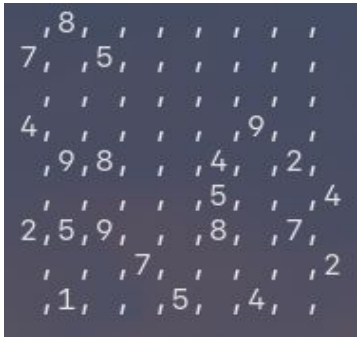
```
def removeNumbers(side,board):
    squares = side*side
    empties = squares * 3//4
    for p in sample(range(squares),empties):
        board[p//side][p%side] = 0

    numSize = len(str(side))
    filename = "sudoku_"+str(side)+"x"+str(side)+".csv"
    with open(filename, 'w') as f:
        with redirect_stdout(f):
            for line in board: print(",".join(f"{n or ' ':{numSize}}" for n in line))
```

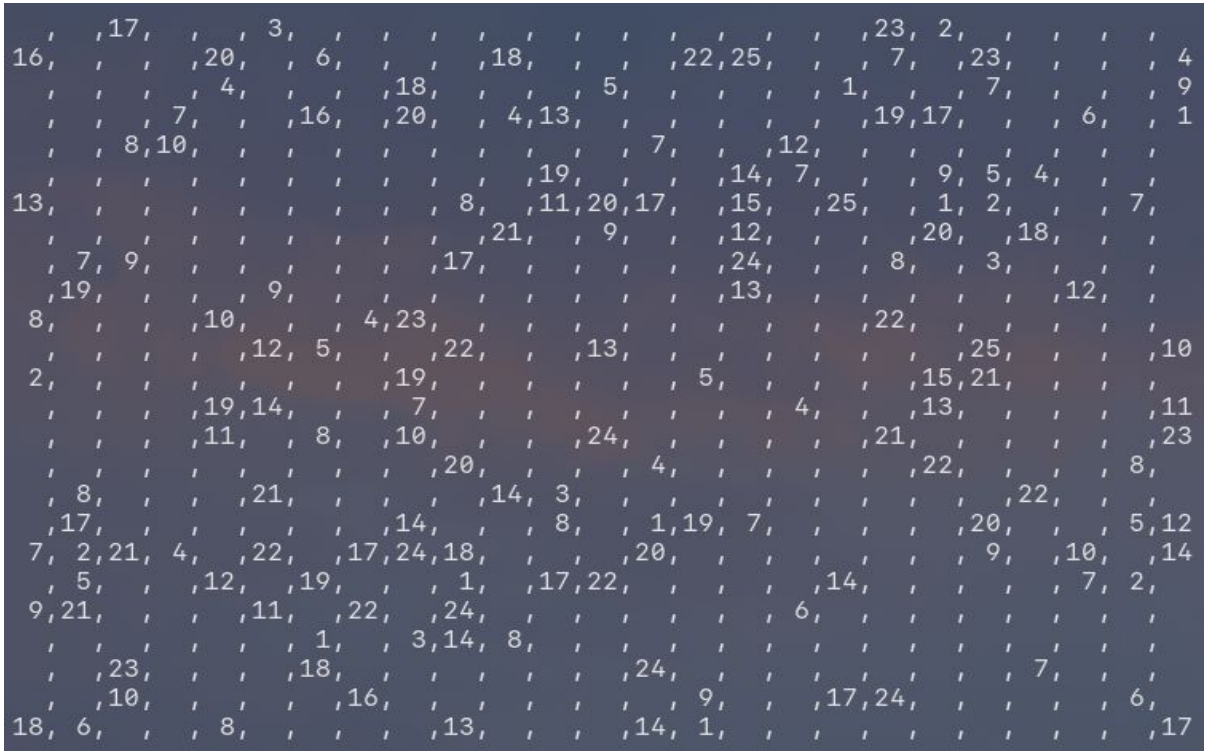
Métodos de implementación
Tableros Generados - Ejemplos



16x16



9x9



25x25

Métodos de implementación
Visualización

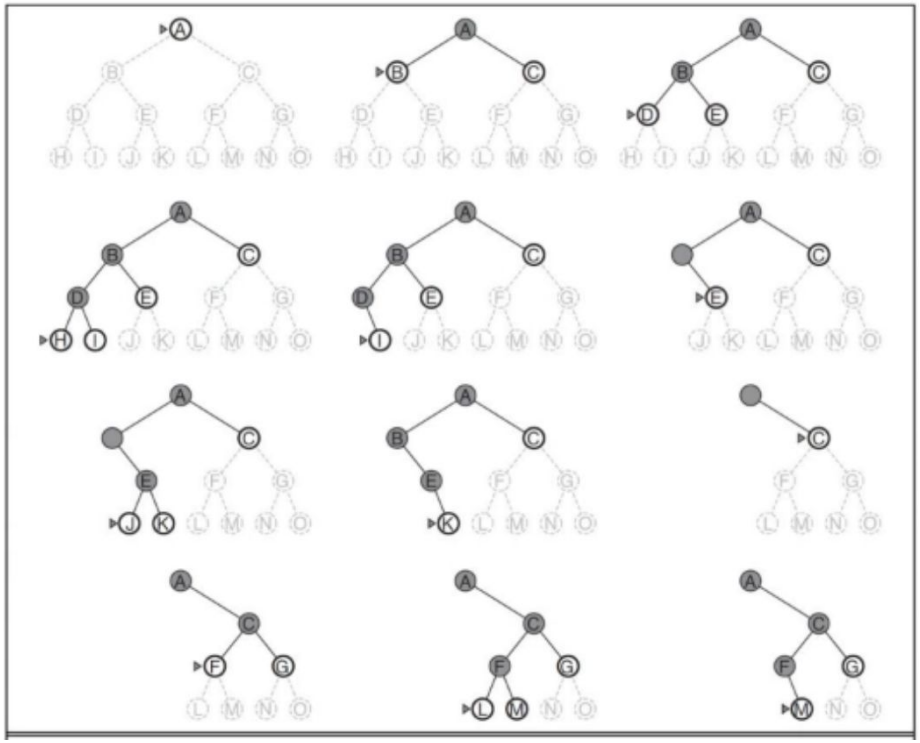
Métodos de implementación
Resolución - Propagación de Restricciones

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

```
4   6   7   | 1   8   9   | 5   3   2
15  3   589 | 2   45  7   |189 468 169
15  2   589 | 3   45  6   |189 48  7
------------+-------------+------------
8   7   3   | 69  2   1   | 4   5   69
9   4   26  | 5   37  8   | 17  26  136
256 1   256 | 69  37  4   |789 268 369
------------+-------------+------------
26  8   26  | 7   9   5   | 3   1   4
3   9   4   | 8   1   2   | 6   7   5
7   5   1   | 4   6   3   | 2   9   8
```

P. Norvig, “Solving every Sudoku puzzle,” Accessed: Mar. 13, 2018. [Online]. Available: http://norvig.com/sudoku.html

Métodos de implementación
Resolucion - DFS (con heuristica)



P. Norvig, “Solving every Sudoku puzzle,” Accessed: Mar. 13, 2018. [Online]. Available: http://norvig.com/sudoku.html

Herramientas externas y Hardware

Librerias/Modulos utilizados:

- Pygame (visualización)
- re (Manejo de expresiones Regulares)
- copy (Para hacer copias de profundas de arreglos)
- math (Funciones matemáticas)
- sys (Interfaz con la CLI)

Lenovo Thinkpad T480

CPU: Intel i7-8550U 4-Core

Base: 1.8 GHz

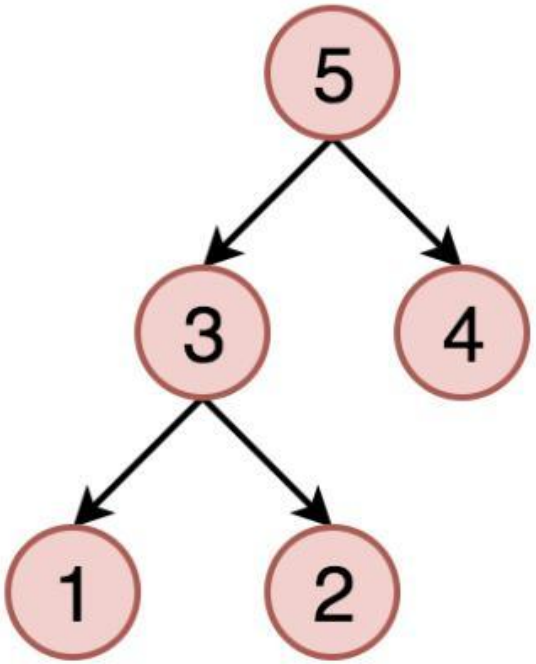
Turbo: 4.000GHz

RAM: 24GB DDR4

GPU: Intel UHD Graphics 620

Demo

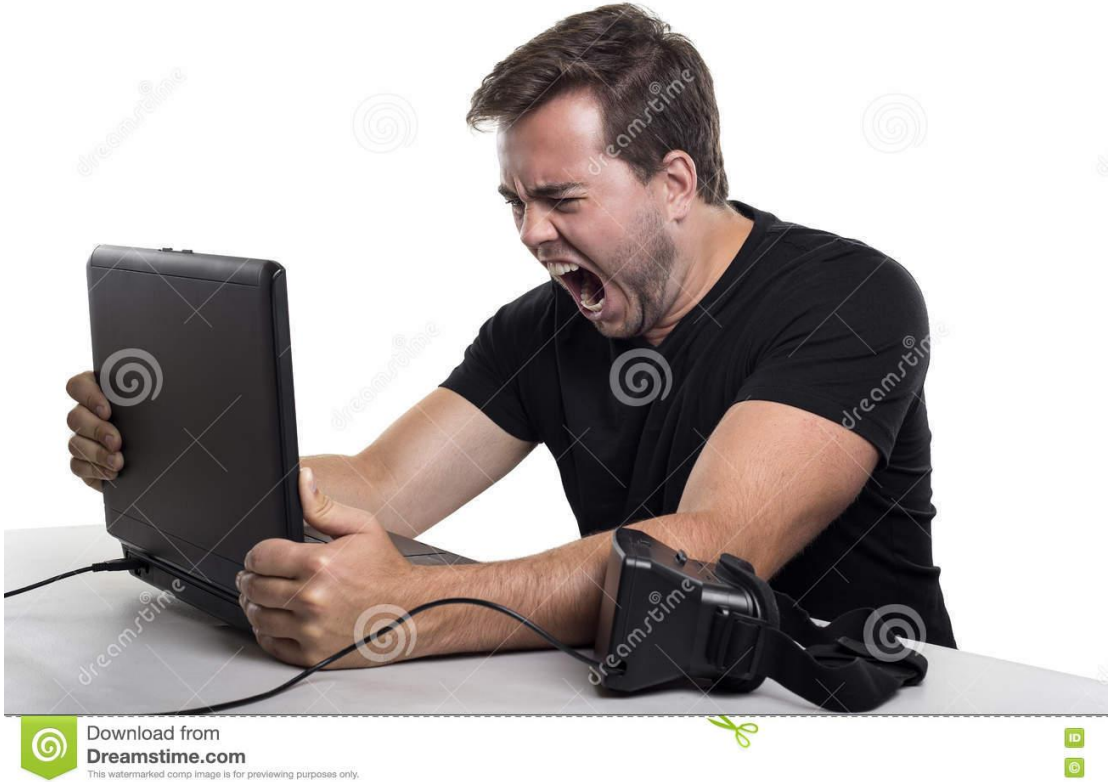
Datos Preliminares



El DFS recorrió **1196 nodos** en alrededor de **15.74 ms** antes de tronar

Esto puede o no ser prometedor… ya que el tablero resultante antes de quebrar puede no estar cerca de la solución. Fuera este el caso, significa que el dfs puede volver a comenzar la búsqueda desde el nodo raíz. Esto facilmente puede duplicar estas metricas o incluso incrementar el orden de magnitud.

En Conclusión... El proyecto sigue en desarrollo...



Hemos aprendido y ganado experiencia!

"Programming isn't about what you know; it's about what you can figure out.”

- *Chris Pine*

Muchas Gracias!