

Sudoku Solver en Haskell

Rodríguez Hernández Alexis Arturo
Sánchez Morales Rodrigo Alejandro

Universidad Nacional Autónoma de México
Facultad de Ciencias

Programación Declarativa, 2018-II

4 de junio de 2018

Tabla de contenidos

1 Introducción

- ¿Qué es el Sudoku?

2 Dificultad de jugar Sudoku

- Reducciones

3 Estrategias

- Tácticas ganadoras

4 Algoritmos Sudoku

- Programación usando restricciones

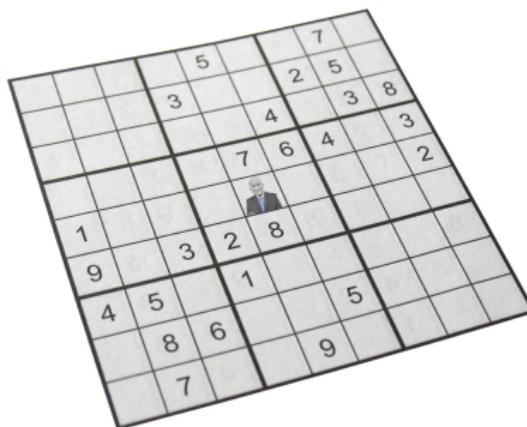
5 Diferentes versiones

6 Implementación

- ¿Cómo resolvimos el problema?

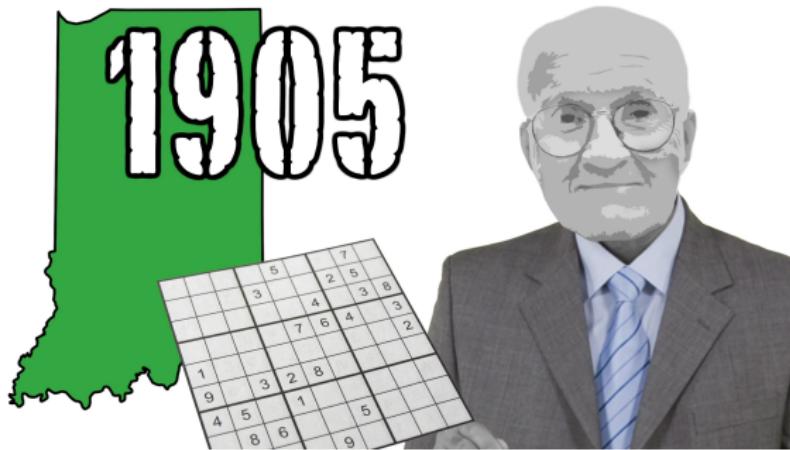
Inicios

Es un juego matemático que se inventó a finales de la década de 1970 por el arquitecto *Howard Garns* nacido en Indianapolis, Indiana (1905-1989). Que originalmente lo llamó "*Number Place*". El cual fue publicado por *Dell Magazines*.



Inicios

Es un juego matemático que se inventó a finales de la década de 1970 por el arquitecto *Howard Garns* nacido en Indianapolis, Indiana (1905-1989). Que originalmente lo llamó "Number Place". El cual fue publicado por *Dell Magazines*.



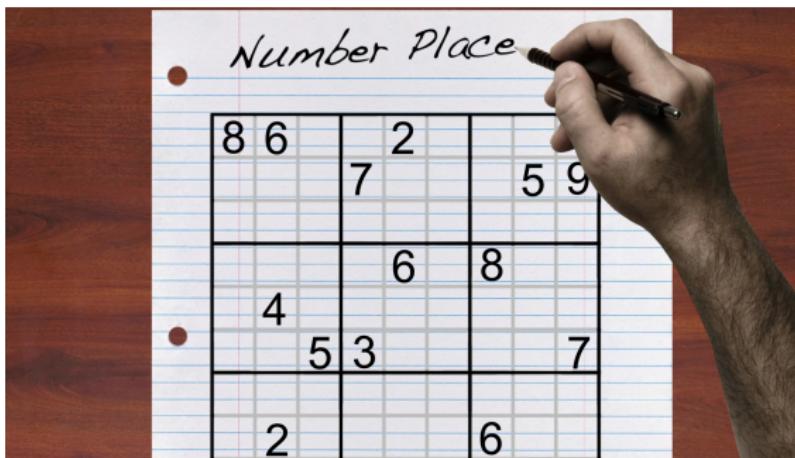
Inicios

Es un juego matemático que se inventó a finales de la década de 1970 por el arquitecto *Howard Garns* nacido en Indianapolis, Indiana (1905-1989). Que originalmente lo llamó "*Number Place*". El cual fue publicado por *Dell Magazines*.



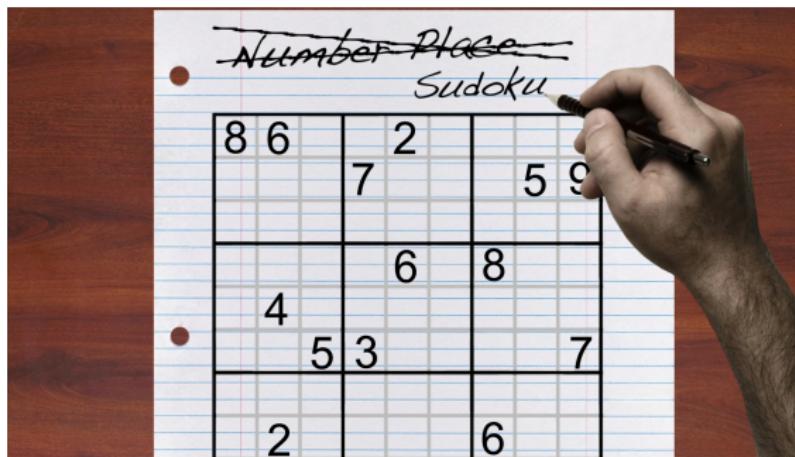
Inicios

Es un juego matemático que se inventó a finales de la década de 1970 por el arquitecto *Howard Garns* nacido en Indianapolis, Indiana (1905-1989). Que originalmente lo llamó "*Number Place*". El cual fue publicado por *Dell Magazines*.



Inicios

Es un juego matemático que se inventó a finales de la década de 1970 por el arquitecto *Howard Garns* nacido en Indianapolis, Indiana (1905-1989). Que originalmente lo llamó "Number Place". El cual fue publicado por *Dell Magazines*.



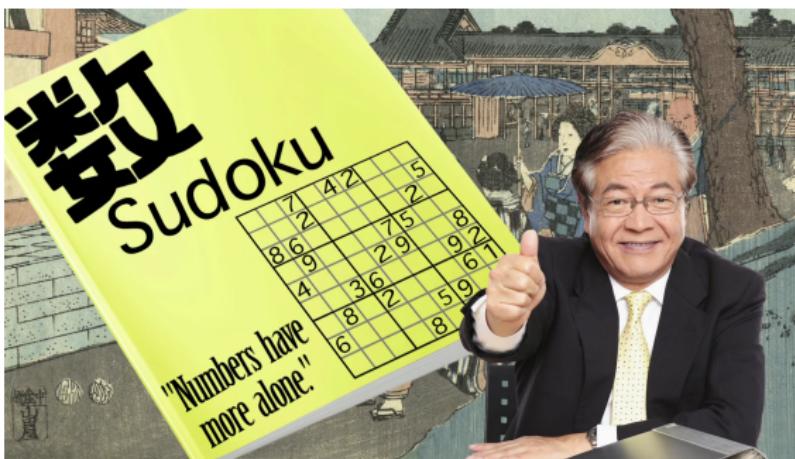
Japón (日本)

En japonés, 数独 (*Sūdoku*). Adquirió popularidad en Japón en la década de 1980. Fue introducido por *Nikoli* y su presidente; *Maki Kaji* lo renombró **Suuji wa dokushin ni kagiru**. (Los dígitos deben ir solos/Los dígitos están limitados a una ocurrencia).



Japón (日本)

En japonés, 数独 (*Sūdoku*). Adquirió popularidad en Japón en la década de 1980. Fue introducido por *Nikoli* y su presidente; *Maki Kaji* lo renombró **Suuji wa dokushin ni kagiru**. (Los dígitos deben ir solos/Los dígitos están limitados a una ocurrencia).



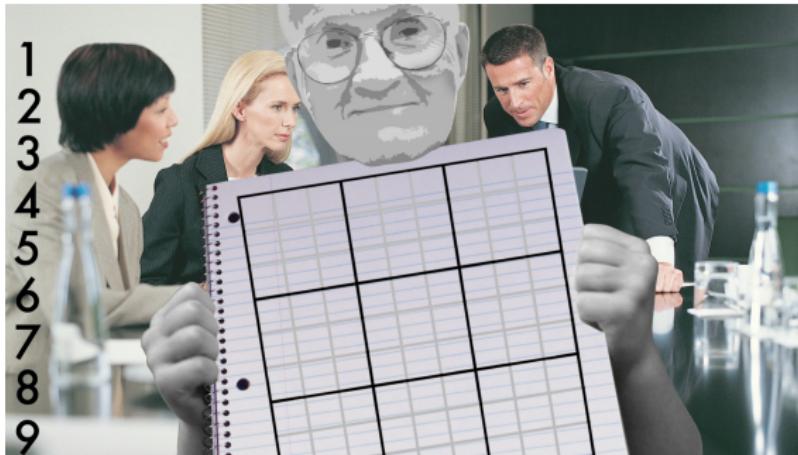
Alrededor del mundo

Se convirtió en un fenómeno mundial cuando se dio a conocer en el ámbito internacional en 2005 cuando numerosos periódicos empezaron a publicarlo en su sección de pasatiempos, tal es el caso de *The Times Newspaper* en Londres.



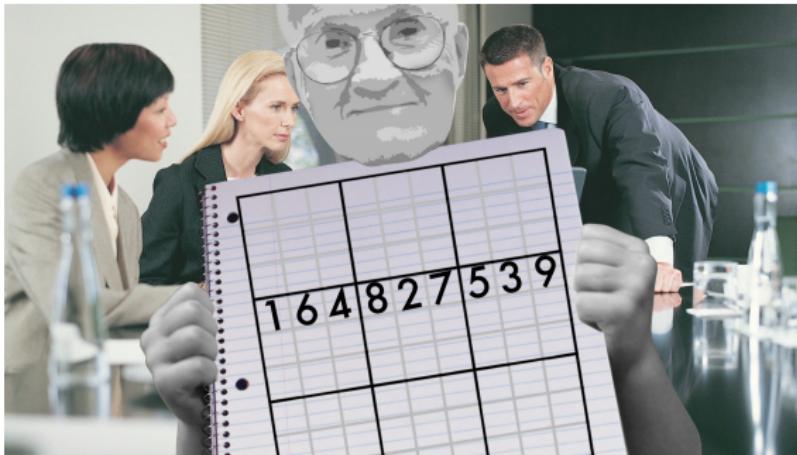
Reglas

El juego consiste en colocar números del 1 al 9 usando cada dígito solamente una vez en toda fila, renglón y bloque.



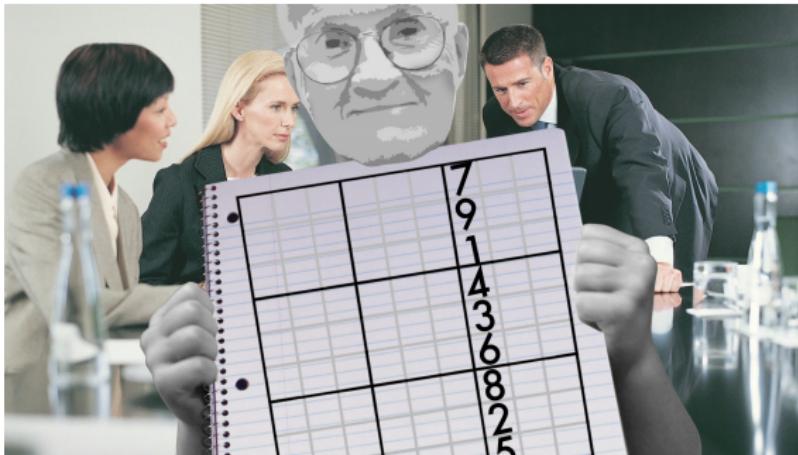
Reglas

El juego consiste en colocar números del 1 al 9 usando cada dígito solamente una vez en toda fila, renglón y bloque.



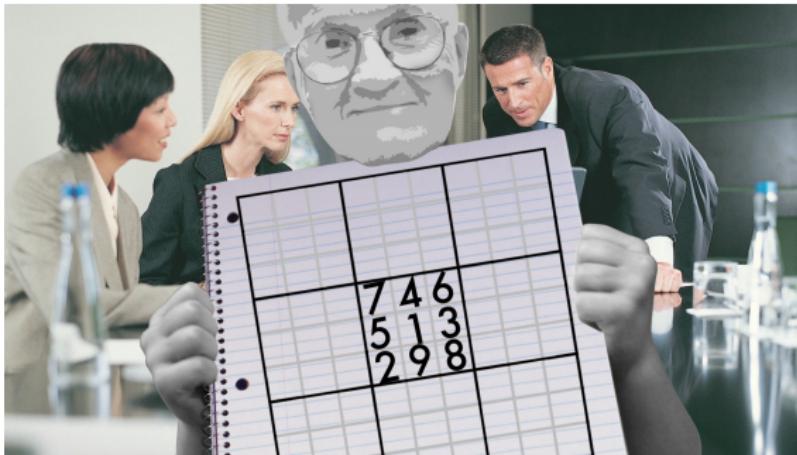
Reglas

El juego consiste en colocar números del 1 al 9 usando cada dígito solamente una vez en toda fila, renglón y bloque.



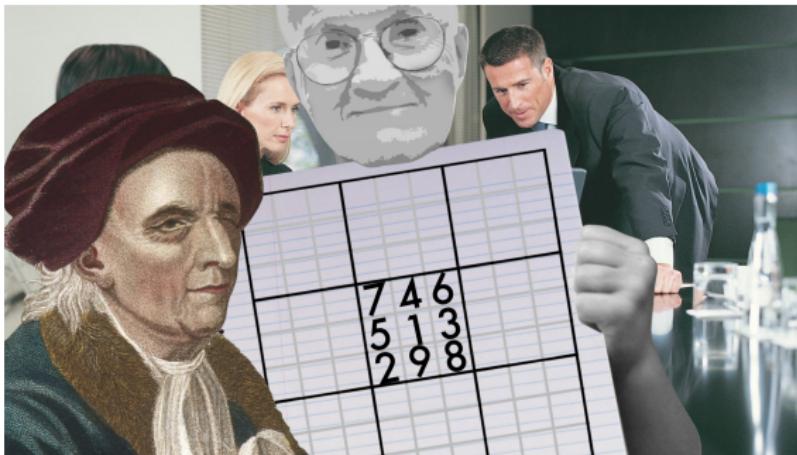
Reglas

El juego consiste en colocar números del 1 al 9 usando cada dígito solamente una vez en toda fila, renglón y bloque.



Reglas

El juego consiste en colocar números del 1 al 9 usando cada dígito solamente una vez en toda fila, renglón y bloque.



7	4	6
5	1	3
2	9	8

Reglas

El juego consiste en colocar números del 1 al 9 usando cada dígito solamente una vez en toda fila, renglón y bloque.



Recuerden a Lucy (Definiciones)

NP

Son los problemas que se puede revisar si una solución es correcta en tiempo polinomial.

NP-Completo

El conjunto de todos los problemas X en NP tales que se existe algún problema Y en NP-Completo tal que podemos reducir Y a X en tiempo polinomial.

NP-Duro

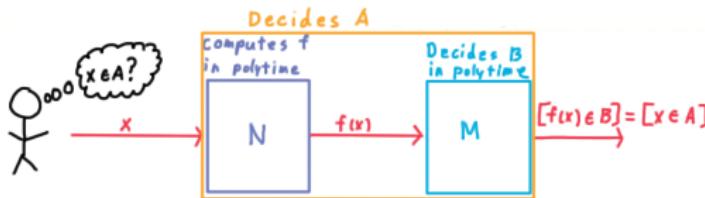
El conjunto de todos los problemas X tales que se existe algún problema Y en NP-Completo tal que podemos reducir Y a X en tiempo polinomial.

¿Cómo probar NP-Completez?

El algoritmo

Para probar que un problema es NP completo debemos probar que está en NP, buscar un problema NP completo y luego reducirlo a nuestro problema en tiempo polinomial.

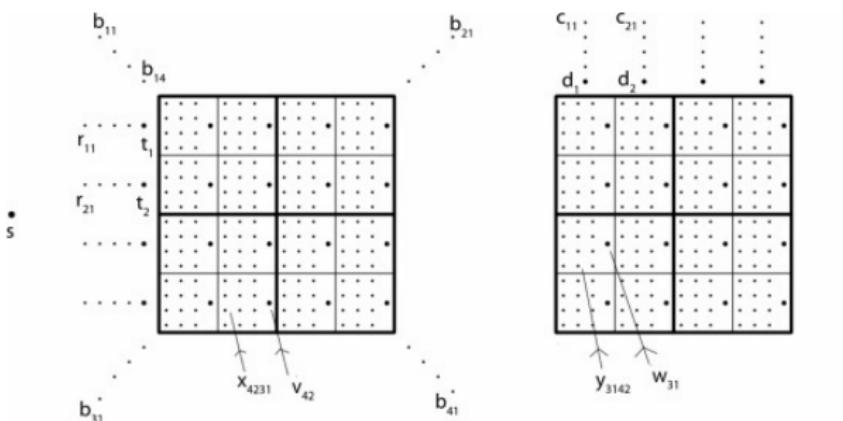
Polynomial Reductions



Reducción de Sudoku al Ciclo Hamiltoniano

De manera rápida

Cada vértice en la gráfica representa la relación entre los números y la posición que ocupan en el sudoku.

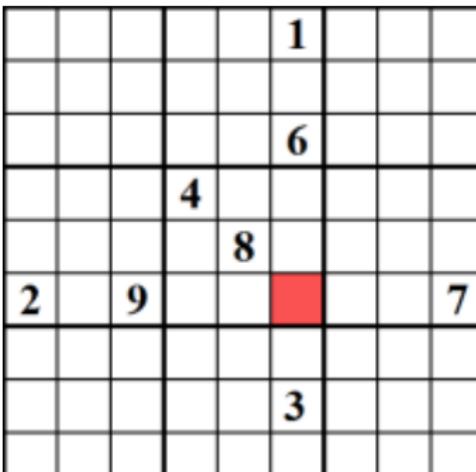


Técnicas para colocar números

Candidato único

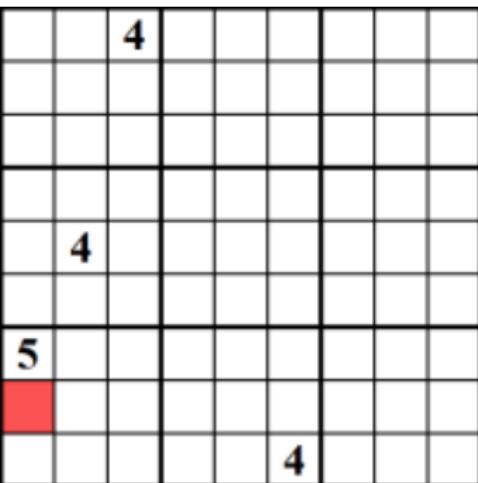
Es cuando una casilla específica puede contener un sólo número.

Esto sucede cuando todos los números aparecen en la fila, columna o bloque específico.



Candidato forzado

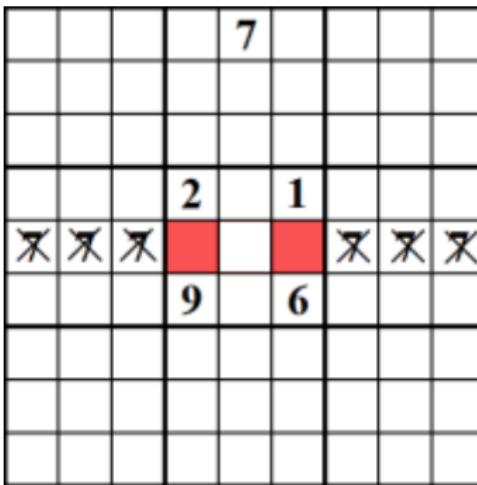
Sabemos que cada bloque, fila y columna en un Sudoku debe contener un número entre 1 y 9. Por lo tanto, si un número puede ser puesto solamente en esa casilla, entonces debe ir ahí.



Técnicas para remover candidatos

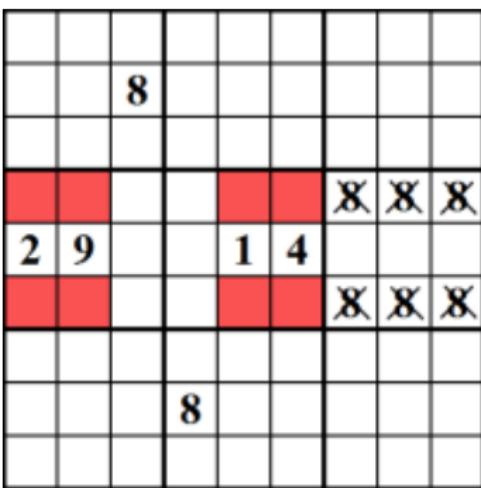
Interacción bloque y columnas/filas

Este método no ayuda a escribir nuevos números, pero ayudará a identificar un número dentro de una fila o columna específica.

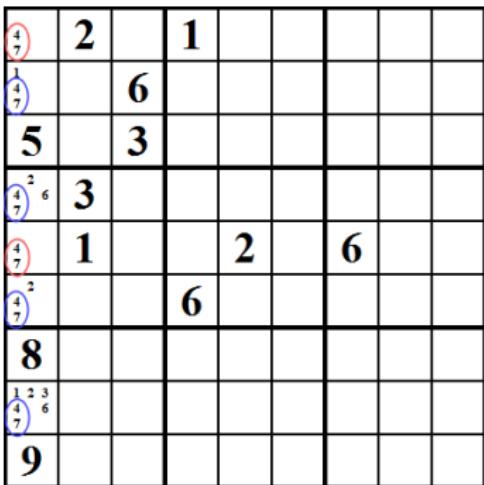


Interacción bloque/bloque

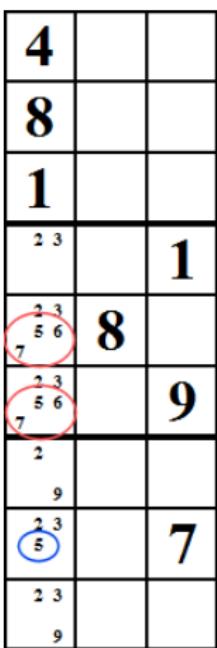
Esta técnica se entiende de manera más intuitiva mirando el siguiente ejemplo:



Subconjunto desnudo



Subconjunto escondido



Backtracking

Consiste en buscar en el árbol de posibilidades, usando DFS, es un algoritmo de fuerza bruta. Aproximadamente existen $6,67 \times 10^{21}$ posibles tableros.

Búsqueda estocástica

Son algoritmos que usan probabilidad y análisis de aleatoriedad, un ejemplo de este tipo de algoritmos son los algoritmos genéticos, el procedimiento estándar sería:

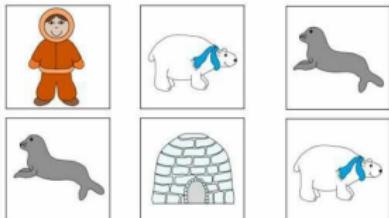
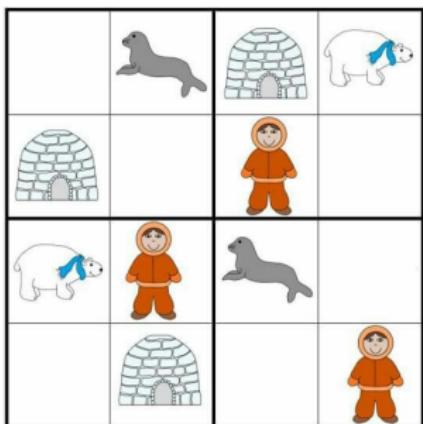
- Asignar números aleatorios a las casillas vacías.
- Calcular el número de errores.
- Intercambiar los números insertados de tal forma que los errores tiendan a cero.

Lógica pura

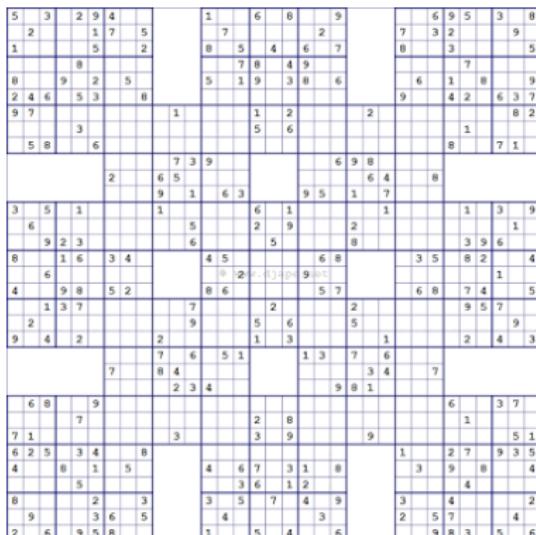
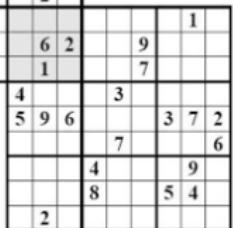
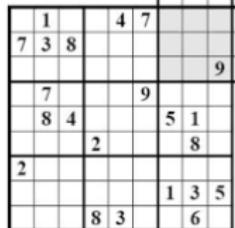
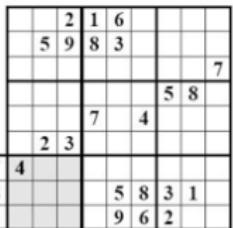
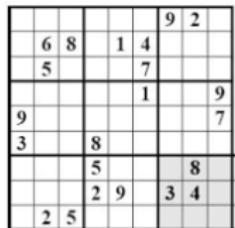
Programación usando restricciones

Consiste en convertir el problema a un conjunto de variables lógicas que representan los estados del problema y sus posibles relaciones, existen sistemas expertos que resuelven estos modelos, como los sistemas de STRIPS.

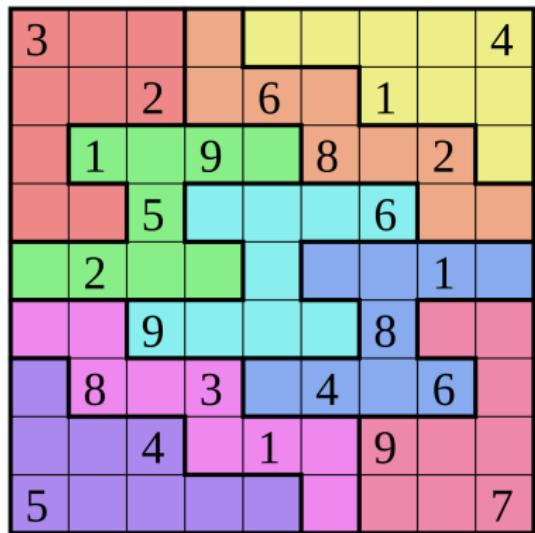
Sudoku Kids



Sudoku Samurai

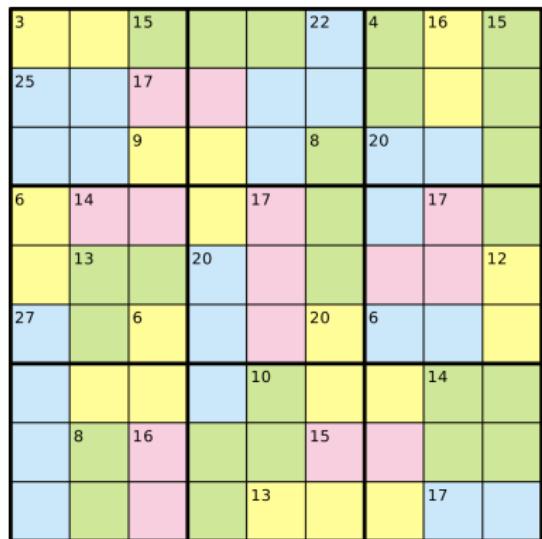


Sudoku Nonomino



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

Sudoku Killer



3		15			22	4	16	15
25		17						
		9			8	20		
6	14			17			17	
	13		20					12
27		6			20	6		
			10				14	
8	16			15				
			13			17		

Wordoku

P	K	R	I	D
D		B		R
B	E		P A	
P		K W A		B
		R K		
A D				
B		E		P
A			E	
E R	P	K B		

ABDEIKPRW

W	P	E	K	A	R	I	B	D
D	I	A	B	W	P	K	E	R
R	B	K	E	I	D	P	A	W
P	E	R	I	K	W	A	D	B
I	W	B	D	P	A	R	K	E
K	A	D	R	B	E	W	P	I
B	K	W	A	E	I	D	R	P
A	D	P	W	R	B	E	I	K
E	R	I	P	D	K	B	W	A

ABDEIKPRW

Cuboku



¿Cómo resolvimos el problema?

Representación del problema

```
-- Las valores en el tablero están representadas por
-- enteros en el rango de 0..9 donde 0 representa "vacío".
type Valor = Int

-- Un cuadro es identificado por un par (fila, columna).
type Casilla = (Int, Int)

-- Un tablero Sudoku es un array 9x9 de valores.
type Tablero = Array Casilla Valor
```

```
-- Convierte una lista de filas de valores a una lista
-- de asociaciones de array.
asociaSudoku :: [[Valor]] -> [(Casilla, Valor)]
asociaSudoku = concatMap asociaFil . zip [0..8]
where
    asociaFil :: (Int, [Valor]) ->[((Int, Int), Valor)]
    asociaFil (fil, val) = asociaCol fil $ zip [0..8] val
    asociaCol :: Int -> [(Int, Valor)] ->[((Int, Int), Valor)]
    asociaCol fil cols = map (\(col, m) -> ((fil, col), m)) cols
```

¿Cómo resolvemos el problema?

```
*Main > asociaSudoku [[5, 3, 4, 6, 7, 8, 9, 1, 2],
[6, 7, 2, 1, 9, 5, 0, 4, 8],
[0, 9, 8, 3, 4, 0, 5, 6, 7],
[8, 5, 9, 7, 6, 1, 4, 2, 3],
[0, 2, 6, 8, 0, 3, 7, 9, 1],
[7, 1, 3, 9, 2, 4, 8, 5, 0],
[9, 6, 1, 5, 3, 7, 2, 8, 4],
[2, 8, 0, 4, 1, 9, 6, 3, 5],
[3, 4, 5, 2, 0, 6, 1, 7, 0]]
```

```
((0,0),5),((0,1),3),((0,2),4),((0,3),6),((0,4),7),((0,5),8),((0,6),9),((0,7),1),((0,8),2)
,((1,0),6),((1,1),7),((1,2),2),((1,3),1),((1,4),9),((1,5),5),((1,6),0),((1,7),4),((1,8),8)
,((2,0),0),((2,1),9),((2,2),8),((2,3),3),((2,4),4),((2,5),0),((2,6),5),((2,7),6),((2,8),7)
,((3,0),8),((3,1),5),((3,2),9),((3,3),7),((3,4),6),((3,5),1),((3,6),4),((3,7),2),((3,8),3)
,((4,0),0),((4,1),2),((4,2),6),((4,3),8),((4,4),0),((4,5),3),((4,6),7),((4,7),9),((4,8),1)
,((5,0),7),((5,1),1),((5,2),3),((5,3),9),((5,4),2),((5,5),4),((5,6),8),((5,7),5),((5,8),0)
,((6,0),9),((6,1),6),((6,2),1),((6,3),5),((6,4),3),((6,5),7),((6,6),2),((6,7),8),((6,8),4)
,((7,0),2),((7,1),8),((7,2),0),((7,3),4),((7,4),1),((7,5),9),((7,6),6),((7,7),3),((7,8),5)
,((8,0),3),((8,1),4),((8,2),5),((8,3),2),((8,4),0),((8,5),6),((8,6),1),((8,7),7),((8,8),0)
]
```

-- Determina si el valor especificado se puede colocar en la
-- posición especificada.

`esValorPosible :: Valor -> Casilla -> Tablero -> Bool`

`esValorPosible m (fil, col) t = noEnFila && noEnColumna && noEnBloque`

`where`

`noEnFila = notElem m $ valoresEnFila t fil`

`noEnColumna = notElem m $ valoresEnColumna t col`

`noEnBloque = notElem m $ valoresEnBloque t (fil, col)`

¿Cómo resolvemos el problema?

*Main > esValorPosible 1 (2, 0)

```
$ tableroSudoku [[5, 3, 4, 6, 7, 8, 9, 1, 2],  
[6, 7, 2, 1, 9, 5, 0, 4, 8],  
[0, 9, 8, 3, 4, 0, 5, 6, 7],  
[8, 5, 9, 7, 6, 1, 4, 2, 3],  
[0, 2, 6, 8, 0, 3, 7, 9, 1],  
[7, 1, 3, 9, 2, 4, 8, 5, 0],  
[9, 6, 1, 5, 3, 7, 2, 8, 4],  
[2, 8, 0, 4, 1, 9, 6, 3, 5],  
[3, 4, 5, 2, 0, 6, 1, 7, 0]]
```

True

¿Cómo resolvemos el problema?

*Main > esValorPosible 7 (2,0)

```
$ tableroSudoku [[5, 3, 4, 6, 7, 8, 9, 1, 2],  
[6, 7, 2, 1, 9, 5, 0, 4, 8],  
[0, 9, 8, 3, 4, 0, 5, 6, 7],  
[8, 5, 9, 7, 6, 1, 4, 2, 3],  
[0, 2, 6, 8, 0, 3, 7, 9, 1],  
[7, 1, 3, 9, 2, 4, 8, 5, 0],  
[9, 6, 1, 5, 3, 7, 2, 8, 4],  
[2, 8, 0, 4, 1, 9, 6, 3, 5],  
[3, 4, 5, 2, 0, 6, 1, 7, 0]]
```

False

¿Cómo resolvemos el problema?

```
-- Regresa una lista de casillas donde el valor es 0
casillasVacias :: Tablero -> [Casilla]
casillasVacias t = [(fil, col) | fil <- [0..8],
                                col <- [0..8],
                                t ! (fil, col) == 0]
```

¿Cómo resolvemos el problema?

```
*Main> casillasVacias $ tableroSudoku [[5, 3, 4, 6, 7, 8, 9, 1, 2],  
[6, 7, 2, 1, 9, 5, 0, 4, 8],  
[0, 9, 8, 3, 4, 0, 5, 6, 7],  
[8, 5, 9, 7, 6, 1, 4, 2, 3],  
[0, 2, 6, 8, 0, 3, 7, 9, 1],  
[7, 1, 3, 9, 2, 4, 8, 5, 0],  
[9, 6, 1, 5, 3, 7, 2, 8, 4],  
[2, 8, 0, 4, 1, 9, 6, 3, 5],  
[3, 4, 5, 2, 0, 6, 1, 7, 0]]  
  
[(1,6),(2,0),(2,5),(4,0),(4,4),(5,8),(7,2),(8,4),(8,8)]
```

¿Cómo resolvimos el problema?

Hagamos pruebas

```
$ ghci sudokuSolver.hs
```

```
*Main> :set +s
```

```
*Main> main
```

```
sudoku0.txt
```

¿Cómo resolvimos el problema?

Referencias



Wikipedia, Sudoku.

(Consultado el 1 de junio, 2018).

<https://en.wikipedia.org/wiki/Sudoku>



Research Gate, La dificultad de jugar Sudoku.

(Consultado el 2 de junio, 2018).

https://www.researchgate.net/publication/228920598_La_dificultad_de_jugar_sudoku



Kristanix, Sudoku Solving Techniques.

(Consultado el 3 de junio, 2018).

<https://www.kristanix.com/sudokuepic/sudoku-solving-techniques.php>

¿Cómo resolvimos el problema?

The end?

