EL ROMPECABEZAS DE SUDOKU

El Sudoku se ha convertido en un pasatiempo muy popular y ha comenzado a aparecer en casi todos los periódicos nacionales junto con el crucigrama. Las reglas básicas del juego son muy simples, debemos llenar cada cuadrado en una cuadrícula de 9 × 9 con uno de los dígitos 1-9. Estamos limitados por reglas que establecen que no pueden aparecer dos del mismo número en la misma fila o columna, o dentro del mismo cuadro 3 × 3 (ver figura).

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

Figura 1 Un sudoku simple.

El conocimiento del dominio es la clave para estas técnicas de resolución de problemas. Podemos usar lo que sabemos acerca de cómo los humanos resuelven el problema para crear una heurística para elegir primero la rama más prometedora del árbol de decisión, evitando así (con suerte) el recorrido de enormes árboles.

NOTAS SOBRE LA RESOLUCIÓN DEL SUDOKU USANDO LA BÚSQUEDA EN PROFUNDIDAD

Aunque se garantizar que resolverá cualquier Sudoku, aunque podemos afirmar que estas técnicas no son la mejor manera de resolver sudokus. Sin embargo, lo que podemos decir sobre los primeros métodos de profundidad es que podría ser una de las formas más simples y genéricas de resolver sudokus.

El primer algoritmo de búsqueda profunda recorre un "árbol de decisión". El árbol de decisión no es una estructura de datos en sí mismo, es solo una forma práctica de visualizar la forma en que el algoritmo recorre el espacio del problema.

Para el ejemplo visto en la figura 1, comenzaríamos en el primer cuadrado libre en el tablero, fila 0 (la fila superior) columna 1 (la segunda columna de la izquierda). Este es el primer cuadrado vacío en el tablero.

Hay varios números que podemos usar para este cuadrado: 1, 7 y 9; Todos los demás símbolos romperían las reglas del Sudoku. Por lo tanto, hay tres ramas para investigar, ver figura 2.

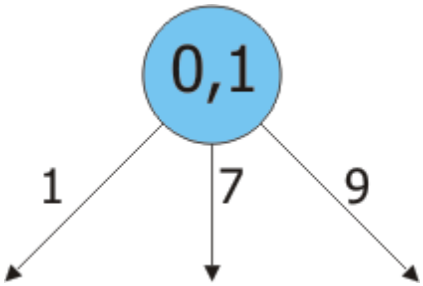

Figura 2 Primer nivel del árbol de decisión

El primer algoritmo de profundidad intenta resolver el problema atravesando el lado izquierdo del árbol de decisión, por lo que investigamos la primera rama colocando un 1 en el cuadrado 0,1.

Una vez que hayamos calculado un número para el primer cuadrado (0,1), debemos intentar encontrar un símbolo que vaya al cuadrado 0,2 (el segundo cuadrado libre a medida que recorremos el tablero). En este punto tenemos dos opciones, colocando un 7 o un 9 en el cuadrado.

Por lo tanto, el árbol se ramifica en dos aquí; tomamos la rama de la izquierda colocando un 7 en el cuadrado 0,2 , ver figura 3.

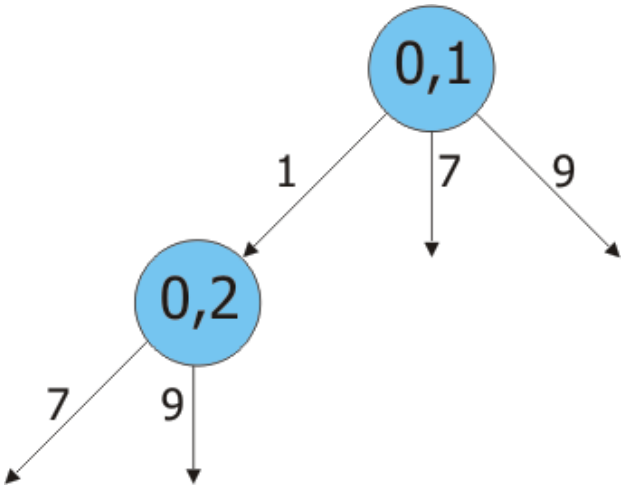

Figura -3 Segundo nivel del árbol de decisión

La Figura 4 muestra el siguiente paso a través del árbol. Tenemos dos opciones para el cuadrado 0,3: 5 y 9. Seleccionamos 5 y continuamos.

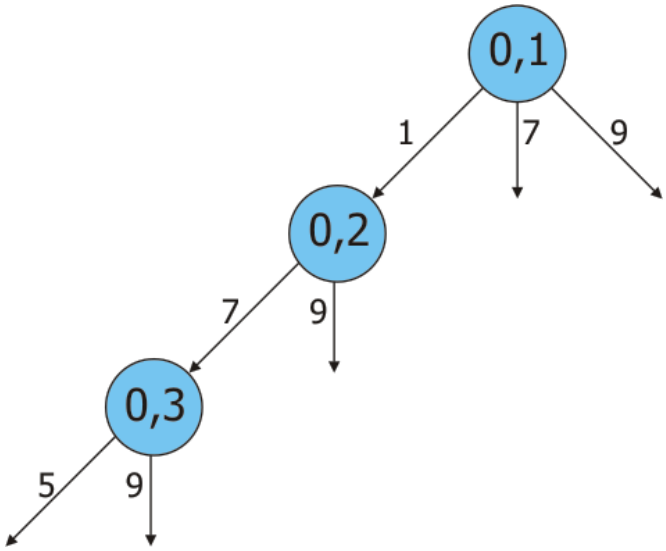

Figura 4 Tercer nivel del árbol de decisión

En algún momento sucederá una de dos cosas, ya sea ...

- Hemos llenado cada cuadro del tablero y, por lo tanto, hemos resuelto el Sudoku.
- Llegamos a un cuadrado donde no podemos colocar ningún símbolo. En este caso, retrocedemos volviendo al cuadrado anterior e intentando el siguiente símbolo posible.

La figura 5 nos muestra retroceder. El algoritmo ha investigado todas las ramas posibles con 1 en el primer cuadrado y ninguno condujo a una solución. Por lo tanto, intentamos nuevamente con un 7 en el primer cuadrado.

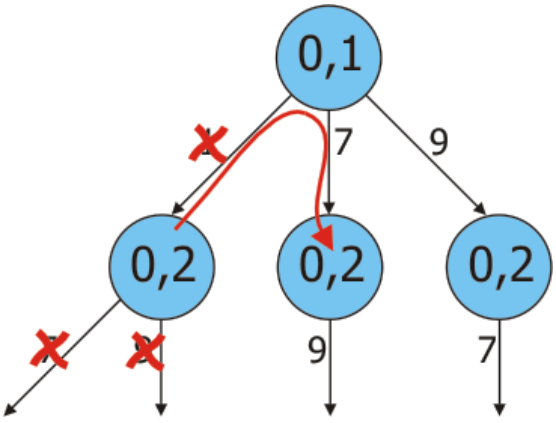

Figura 5 Retroceso a través del árbol de decisión

La solución (que se muestra en la figura 6) en realidad tiene un 1 en el primer cuadrado. El retroceso que se muestra en la figura 5 se muestra como un ejemplo, solo para darle una idea de lo que sucede: no ocurre en esta parte del árbol para este problema en particular.

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

Figura 6 Solución al problema de ejemplo

Estas notas están inspiradas en uno de sus artículos de Peter Norvig, Director de Investigación de Google. Te recomiendo que leas su artículo antes de ‘entrar en harina’.

1. (https://norvig.com/sudoku.html).

Otra referencia, que sería muy recomendable leer es el artículo de Fernando Rodrigues Junior, el cual este documento compara los diferentes enfoques de búsqueda no informada a través de la aplicación en un problema de Sudoku Puzzle.

2. (https://es.slideshare.net/FernandoJunior52/bdfsdfspaper)

OBJETIVO

Realizar un programa en **Racket** que resuelva el problema del Sudoku, con los algoritmos de *Búsqueda en Profundidad* y, también, *Búsqueda en Anchura*, imprimiendo por pantalla la solución encontrada.

El trabajo se podrá elaborar por equipos de una, dos o tres personas, del mismo grupo de laboratorio. En el momento de la exposición y defensa, que se indicará con antelación, deberán estar presentes todos los miembros del equipo y uno o varios de los miembros, elegido por los profesores, tendrá que hacerse cargo de toda o parte de la defensa.

El trabajo se presentará físicamente, en un informe impreso que contenga comentada detalladamente la discusión y formulación del problema y del método de resolución empleado, así como el código Racket correspondiente bien documentado. Además, se entregará una copia del documento en formato "pdf" y el código de la aplicación en formato "rkt" en la actividad del aula virtual creada a tal efecto.

Los profesores proporcionarán un fichero con una serie de Sudokus a resolver.

VALORACIÓN

En la solución aportada por los alumnos, se tendrán en cuenta los aspectos siguientes:

- Positivamente
  - Usar solo funciones *core* de Racket (no usar require)
  - Uso de funciones de primer orden (*map, fold, filter)* cuando sea posible.
  - Uso de bucle *for* o variantes *for/and, for/or*, etc. cuando sea posible.
  - Uso de recursión, cuando no sea posible mediante resolver mediante *funciones de primer orden* o mediante el uso de *for*.
  - Imprimir secuencia de pasos para la resolución del sudoku
  - Imprimir el sudoku con subdivisiones de cuadrantes.
  - Comprobar si el sudoku a resolver es válido, antes de empezar a resolverlo.
  - Realizar test unitarios para un número significativo de funciones.

- Negativamente, por ir contra la filosofía de la programación funcional.
  - Uso de bucles tradicionales de la librería "control" de Racket.
  - Uso de asignaciones (funciones que terminen en *set!)*. Se exime de este requisito al uso de tablas hash.

ENTREGA POR BLACKBOARD:

1. Ensayo explicativo del trabajo realizado. (*pdf*)
2. Código fuente de la solución aportada. (*rkt*, con comentarios)

FECHA DE ENTREGA:

- **5 de diciembre de 2019 es la fecha límite a las 14:00**. Recibidos los trabajos y establecido el número de grupos, se publicará un calendario con la hora de exposición para cada grupo.