**VALIDAR SUDOKU**

El algoritmo “validar\_sudoku” se encarga de, como lo dice su nombre, validar si un sudoku de tamaño n está bien solucionado o no. Recordemos que para que un sudoku sea válido, debe tener en una fila, columna y región, todos los números del 1 hasta n sin repetirse. Esto tiene que pasar sin excepción en todas las filas, columnas y regiones que halla en el tablero.

El algoritmo empieza con un for, recorriendo todas las filas del sudoku, verificando que no se repita ningún número y que ninguno tenga el valor 0, ya que, de ser así, significaría que todavía no se ha completado al 100% el sudoku. De forma similar, se hace lo mismo con todas las columnas y luego con las regiones. Al final, el algoritmo retornara true de ser valido el sudoku.

A medida que el tamaño del tablero aumenta, el tiempo de ejecución del algoritmo aumenta, ya que se tienen que verificar más filas, columnas y regiones.

**RESOLVER SUDOKU**

El algoritmo “resolver\_sudoku”, resuelve cualquier sudoku de tamaño n. La resolución de este se hace por medio de la técnica Backtracking, en el cual se hace lo siguiente:

1. Primero se necesita de un algoritmo llamado “numero\_valido”, con el cual se verifica si un número (el que se quiera poner en una casilla vacía) es válido ponerlo en la determinada casilla vacía. Para ello, se verifica la fila, columna y región de la celda en cuestión, en busca de un número igual al que queramos poner.
2. Segundo, se necesita de un algoritmo llamado “iniciar\_celdas”, el cual almacenará en un vector todas las celdas vacías que hay en el tablero. Una vez almacenadas, mira por cada una los posibles candidatos que puede tener la celda y los almacena. Para ello, se necesita de un struct llamado Celda, en el cual se guarda la fila, columna y un vector con los posibles candidatos de la celda en cuestión. Luego de tener todos los candidatos de cada celda vacía, se procede a ordenar el vector en función del número de candidatos de cada celda, de menor a mayor; es decir, irán primero las que tengan menos candidatos.

Una vez se sabe de los algoritmos necesarios, ya se puede entender la resolución del sudoku. El algoritmo que lo resuelve se usa de forma recursiva, donde se recorre cada celda vacía y se prueba con los candidatos que tiene. Y se va probando en cada celda vacía y por ello se utiliza la técnica del backtracking, ya que va a llegar un punto donde no es posible poner ningún número; entonces el algoritmo se tendrá que devolver, cuantas veces sea necesario, y probar con otro número de los candidatos de la celda.

Si se encuentra una solución, el algoritmo retornará true; de lo contrario, retornará false con un mensaje de que no hay solución para el sudoku en cuestión.

A medida que el tamaño del tablero aumenta, la ejecución del algoritmo aumenta exponencialmente, ya que se tendrán que almacenar más celdas vacías, candidatos posibles y se tendrá que llamar recursivamente muchas más veces.

**OTROS ALGORITMOS**

El programa principal, lo que hace es ofrecer un menú que provee diferentes opciones: resolver, validar, generar o imprimir el sudoku. Para ello, se crearon los siguientes algoritmos:

* sobrescribir\_archivo: sobrescribe en el archivo “Entrada.txt” borrando y añadiendo un sudoku.
* leer\_archivo: lee desde el archivo de texto “Entrada.txt” el sudoku que se va a procesar. Como el sudoku está en formato diferente en este archivo de entrada, lo que se hace en transcribirlo de manera que se pueda leer bien en el programa.
* imprimir\_sudoku: imprime el sudoku con su tamaño respectivo.
* encontrar\_uno: encuentra si hay celdas vacías con solo un candidato.
* resolver\_uno: para que la resolución con backtracking sea mas eficiente, primero se resolverán todas las casillas que tengan solo un candidato posible, de esa manera, se borraran esos candidatos de la fila, columna y región en la que se encuentre la celda con un solo candidato.
* borrar\_candidatos: borra el candidato seleccionado de la fila, columna y región deseada.