Universidad de Costa Rica

Sede Rodrigo Facio

Escuela de Matemática Pura

CA0305 Herramientas de Ciencia de Datos II

**Proyecto individual:**

**Sudoku**

Estudiante:

Laura Villacís Delgado C28386

Profesor:

Luis Alberto Juárez Potoy

I Ciclo 2024

**Introducción**

El sudoku es un juego matemático que pone en práctica habilidades de razonamiento lógico y resolución de problemas. Este surgió a partir del concepto de Cuadrados Latinos que fue desarrollado por Leonhard Euler en el siglo XVIII. Sin embargo, fue hasta el año 1979 que el estadounidense Howard Garns desarrolló el juego como se le conoce hoy en día y lo publicó por primera vez como Number Placement en el libro Dell Pencil Puzzles y en la revista Word Games. En el año 1984 este juego llegó a Japón, donde se popularizó y recibió el nombre Sudoku. Más adelante, en el 2004, este fue publicado en The Times of London, impulsando así su popularidad entre el público. Actualmente, es un juego que se publica en los periódicos de los distintos países y sigue siendo jugado por personas alrededor del mundo.

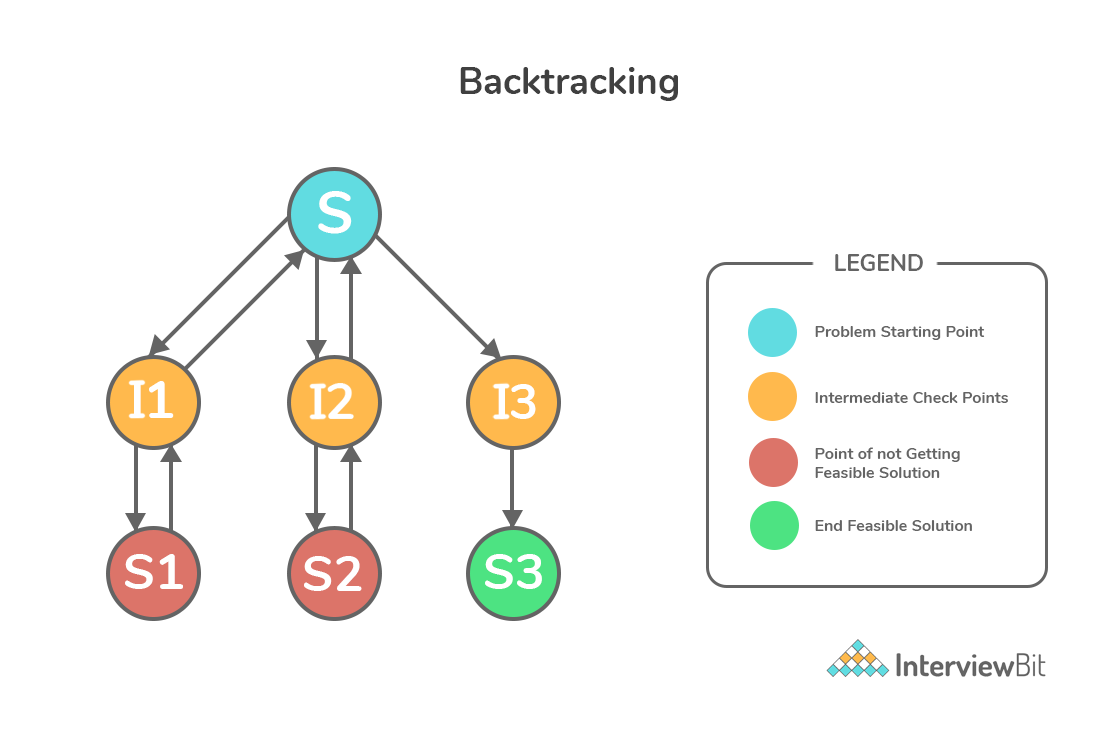
A partir del análisis de este juego es que surge la pregunta: ¿Se puede crear un algoritmo que sea capaz de resolver un Sudoku en el lenguaje de programación Python? Y la respuesta es sí, de hecho, este problema de programación ha sido ampliamente estudiado y existen maneras de solucionarlo utilizando procesos básicos y simples de la programación hasta proceso de aprendizaje profundo (Deep Learning). Este trabajo se enfoca principalmente en el análisis del método Bactracking y un método heurístico de resolución. Asimismo, a partir de los métodos para resolver, se realiza una clase que permita al usuario jugar al Sudoku.

**Desarrollo**

El código consiste en dos clases principales: Sudoku y Juego. La clase Sudoku se encarga de la creación de un objeto que tiene como atributo el tablero y las soluciones, tanto la original como la que se consigue con cada método utilizado. Por otro lado, la clase Juego hereda de la clase Sudoku todos los atributos y no se incluye ninguno nuevo, puesto que no es necesario. A continuación de van a explicar lo que hace cada clase y algunos de sus métodos.

La clase Sudoku tiene como función principal la creación de un tablero de Sudoku y su resolución por dos métodos distintos. Primero, la creación del tablero se realiza mediante un método llamado Backtracking, el cual se va a explicar posteriormente. Durante este proceso, se genera un Sudoku completamente resuelto y que sea una solución válida. Posteriormente, según el nivel que se indique al llamar al método encargado de crear el Sudoku se elimina una cierta cantidad de casillas, entre mayor sea el nivel, mayor cantidad de casillas se van a eliminar para aumentar la dificultad. Una vez obtenido el tablero, este se guarda como el atributo sudoku de la clase.

El primer método de resolución es el denominado Backtracking, el cual es un algoritmo que sirve principalmente para resolver problemas de optimización y combinatoria, en donde la solución está compuesta por distintas variables. Este explora todo el campo de soluciones posibles mediante la creación de un árbol de búsqueda implícito. Entonces, dado un problema en el cual la solución completa consiste en una construcción de una combinación de las variables para satisfacer ciertas restricciones, para la primera decisión, el algoritmo escoge una opción válida dentro de las posibles. Luego, esta elección la analiza más a profundo para ver si se puede llegar o no a una solución válida, y así continúa hasta completar el problema. Si en algún punto, alguna decisión tomada no llega a una solución válida, esto quiere decir que alguna de las decisiones tomadas anteriormente no fue la correcta, por lo que el algoritmo se devolvería e intentaría con otra opción. Si la elección sí lleva a una respuesta válida, entonces esta se guardaría y se continuaría con la búsqueda. Dado un campo de búsqueda finito, siempre se va a encontrar una solución que funcione, ya que la efectividad de este modelo es muy alta. A continuación, se presenta una visualización del funcionamiento de Backtracking:



Entonces, en la imagen es el punto S, de ahí toma la decisión I1, pero esta no llega a una solución viable, por lo que se devuelve, en este caso al punto de inicio, y prueba con I2, pero otra vez no llega a una solución válida. Por consiguiente, se devuelve de nuevo e intenta con I3 y se puede ver que efectivamente se llega a una solución válida. Es importante mencionar que este método hace uso de la recursión, ya que de esta manera se recorre el árbol de búsqueda.

Ahora bien, el otro método utilizado para la resolución se basa en la lógica que una persona utilizaría para resolver el juego. Entonces, se utiliza un método sencillo que vamos a llamar Basic Filler, este consiste en obtener todos los valores válidos para cada casilla según la fila, la columna y el bloque en el que se encuentre, posteriormente se obtiene la intersección de estos tres conjuntos. Si la intersección es solo un número, entonces ese va a ser asignado a la casilla inmediatamente. Este proceso se realiza en un orden fijo, se empieza en la casilla (0,0) y la búsqueda se va realizando de izquierda a derecha dentro de cada columna y luego se mueve a la siguiente fila hacia abajo. Es por esto, que soluciones posteriores permitan asignarle algún valor a la casilla, por lo que este procedimiento se realiza en un ciclo While, hasta que la cantidad de números que se pueden encontrar no varíe.

Una vez realizado este proceso, se utiliza el algoritmo Backtracking para terminar de resolver el tablero, ya que Basic Filler en ciertos casos no es capaz de llegar a una solución completa. Esto porque a veces se llega a casillas que no se pueden establecer solamente a partir de las restricciones directas del juego.

Con respecto a la clase Juego, esta hereda todos los atributos de la clase Sudoku y los utiliza para crear un juego, entonces hay dos modalidades: con vidas y sin vidas. Ambas modalidades consisten en que en la consola se va a pedir el número de casilla que se quiere modificar, entonces se debe insertar en formato ‘fila,columna’ y posteriormente indicar el valor que se quiere insertar. Es importante aclarar que la fila y columna se empiezan a contar desde cero, y se cuenta de derecha a izquierda y de arriba hacia abajo, por lo que el formato de las casillas se debe de ingresar de esa manera.

La primera modalidad consiste en que el usuario tiene 3 vidas, entonces cada vez que falle se le indica que se falló y se le resta una vida. Al momento en que el usuario cometa 3 errores, entonces ya se termina el juego y perdió. Por el otro lado, para el juego sin vidas, el usuario puede poner todos los valores que quiera y cada vez que complete el tablero se le va a preguntar si ya terminó, si indica que sí, entonces se evalúa el sudoku y se indica si llegó o no a la solución correcta.

Asimismo, se probó un código de Deep Learning encontrado en la web para resolver el Sudoku. Pero, un algoritmo así necesita un alto nivel computacional, con el cual no se contaba, por lo que se dificultó llegar a resultados contundentes acerca del funcionamiento de este. Para esto se necesitaron de las librerías Keras y Tensorflow las cuales son especializadas en el aprendizaje profundo.

Con respecto a las librerías, no se utilizó alguna librería en específico para el funcionamiento del código en general ya que se trabajó con listas y los métodos integrados de Python. Únicamente se importaron las siguientes tres librerías: “copy”, “random” y “Fore” de “colorama”. “Copy” se utilizó para hacer copias por completo de ciertos atributos de manera que no se cambiara el original. “Random” funcionó para generar valores aleatorios para generar los nuevos sudokus. Y finalmente “Fore” se utilizó para imprimir los números en rojo cuando el usuario se equivocaba al ingresar algún valor.

**Resultados y Recomendaciones**

Después de terminado el código y realizado distintas pruebas, se llegó a que el método de Backtracking encontró las soluciones correctas durante todas las pruebas realizadas, lo cual apoya la teoría de que este es un método efectivo. Sin embargo, ese método no es eficiente ya que tiene que buscar entre todas las soluciones posibles y encontrar las válidas. Por lo que, se recomendaría algún otro método más eficiente para resolver el Sudoku. En esa línea, a la resolución del sudoku se han aplicado técnicas de búsqueda estocásticas. Esto es mencionado en el trabajo llamado “Stochastic Optimization Approaches For Solving Sudoku” realizado por Meir Perez and Tshilidzi Marwala, en el cual probaron distintos métodos y concluyeron que sí se lograba hacer el proceso más eficiente.

Asimismo, con el método heurístico, después de realizar pruebas, se concluyó que aplicando solo el método Basic Filler, los sudokus de nivel 1 y nivel 2 se resolvieron por completo durante todas las pruebas e incluso algunas veces se lograron resolver de nivel 3. Así, se concluye que el método mencionado, a pesar de ser sencillo, sirve para resolver sudokus de un bajo nivel. Ahora bien, para los niveles 3, 4 y 5 la solución no fue hallada, pero sí se logró rellenar algunas casillas. A la hora de usar Basic Filler y Backtracking, la solución sí fue hallada en todos los casos, además el tiempo que se tarda hacer los dos métodos es bastante parecida. Sin embargo, en general, cuando se usa Basic Fill y Backtracking el tiempo es menor a que si solo se usara Backtracking, lo cual indica que en ciertos casos el método Basic Filler reduce el espacio de búsqueda del Backtracking y optimiza el proceso.

Con respecto al método de Deep Learning, no se pudo llegar a resultados contundentes porque el entrenamiento del modelo no se realizó de manera adecuada. Esto porque no se tenían los recursos necesarios ni el conocimiento suficiente para poder aplicarlo de manera correcta. Se recomienda entonces un mayor estudio de estas librerías para poder usar el modelo con los datos y probar el nivel de efectividad de este. Sin embargo, para este caso en específico del sudoku, no hace falta un método que cueste tanto a nivel computacional, ya que se logró llegar a soluciones con un método más sencillo: Backtracking.

Como conclusión, la programación de un algoritmo para resolver un sudoku es un buen ejercicio para poner en práctica habilidades de programación y de razonamiento lógico. Esto porque al programar la lógica con la que se resuelven estos juegos implica sistematizar un proceso que las personas realizan de forma casi automática, por lo que se eleva el nivel de dificultad. Asimismo, se muestra que un problema puede tener diferentes enfoques y dependiendo del principal objetivo entonces se elige el método más adecuado.

**Referencias**

Anand, M. (2023, 27 julio). How to think of Backtracking. *Medium*. <https://medium.com/@mrinalanand/how-to-think-of-backtracking-61b71e518d2a>

Arias, A., & Arias, A. (2024, 2 febrero). Algoritmos para resolver sudokus - Damavis Blog. *Damavis Blog - Data - Machine Learning - Visualization*. <https://blog.damavis.com/algoritmos-para-resolver-sudokus/>

GeeksforGeeks. (2023, 14 diciembre). *Sudoku Solver using TensorFlow*. GeeksforGeeks. <https://www.geeksforgeeks.org/sudoku-solver-using-tensorflow/>

Perez, M y Marwala, T (s.f.) *Stochastic Optimization Approaches For Solving Sudoku.* <https://arxiv.org/pdf/0805.0697>

PyPI (2022, 25 octubre). *colorama*. <https://pypi.org/project/colorama/>

Sudoku.com (s.f.) *La historia de Sudoku*. <https://sudoku.com/es/como-jugar/la-historia-de-sudoku/>

Tarunk. (s. f.). *GitHub - tarunk04/sudoku-backtracking-visualizer: A simple program to solve sudoku using a backtracking algorithm and visualize the working of the backtracking algorithm in real-time. Also playable!* GitHub. <https://github.com/tarunk04/sudoku-backtracking-visualizer>

*2.4 Backtracking | Programación, refactoriza tu mente*. (s. f.). <https://docs.jjpeleato.com/algoritmia/backtracking>