Escuela de Computación

Ingeniería en Computación

IC-1803 - Taller de Programación

Resolvedor de Sudokus

Fernando González

2021075114

Ana Laura Mora

2021066978

Pablo Sandí

2021120523

Prof. Eddy Ramírez Jiménez

Mayo 2021

Resumen ejecutivo

El tercer proyecto del semestre consistió en la creación de un programa que recibiera un Sudoku y devolviera la resolución del mismo. Dicha entrada podría ser de un tamaño de $n \times n$ siempre y cuando fuese un cuadrado perfecto. Además, se asumía que el Sudoku ingresado era válido y tenía una única solución.

Antes de realizar el programa se contaba con un algoritmo general utilizado para resolver problemas de *backtracking*, el cual hace uso de una pila y posee dos funciones fundamentales: una que indica cuándo se llega a la solución (y en este caso detiene el ciclo debido a que hay una única respuesta) y otra que, al no haber encontrado una solución aún, inserta las otras posibles soluciones en la pila.

El programa creado contó con múltiples funciones que son explicadas con más detalle en la sección *Descripción de la solución*, pero al referirnos a las fundamentales que tiene el algoritmo general, la función que detenía el ciclo era muy simple: si la matriz que contenía al sudoku ya no tenía ceros, entonces ya se había encontrado la salida correspondiente. De no ser así, entonces se debía proceder a buscar las posibilidades que irían a la pila.

Esta última parte no fue producto de una única función, si no varias apoyándose entre sí para lograr el objetivo. Básicamente, se creó una función encargada de validar si un número podía ser utilizado en una casilla (al revisar la fila, columna y región), esta era utilizada por otra función para encontrar las posibilidades de una casilla y una más se encargaba de determinar cuál era la casilla con menos posibilidades para que luego se le asignara a cada posible sudoku las posibilidades encontradas. De esta manera en la pila se insertaban copias de las matrices con el sudoku para encontrar el camino que llevara a la respuesta.

Las pruebas fueron hechas con sudokus de dos tamaños. En *Resultados de pruebas* se muestran los tiempos y soluciones de sudokus 9×9. Las pruebas de 16×16 no se incluyeron en el documento por temas de márgenes.

Introducción

El proyecto cuenta con dos secciones principales, una de ellas es la documentación. Esta está compuesta por distintos puntos: primero la portada, seguida del resumen ejecutivo, el cual presenta una breve explicación del proyecto (los objetivos, algunos antecedentes y las especificaciones del problema). Después seguiría el marco teórico, en donde se aborda el tema de una manera más histórica ya que se centra en la historia del lenguaje y sus propiedades. Al llegar a una parte más intermedia se encuentra la descripción de la solución, donde se explica con detalles el algoritmo y los problemas que se hayan podido presentar en el transcurso. Posteriormente estarían los resultados de pruebas, los cuales tienen una dependencia directa con el código del trabajo, ya que consiste en evaluar con distintos casos el rendimiento del mismo, y ya por último, se encuentran las conclusiones y aprendizajes. De parte de las conclusiones, se da una observación de los resultados de prueba y del proyecto de manera generalizada y en los aprendizajes se plasma el conocimiento adquirido en el trabajo de manera individual.

Para finalizar, la otra sección que completa el proyecto es el algoritmo, este consistía en realizar un programa que lograra la resolución de un sudoku empezando desde el punto en donde se encontrara la menor cantidad de posibilidades para colocar un valor en una casilla. Es fundamental recalcar que para esto se tenía que contar con un conocimiento previo de los estudiantes sobre el manejo de listas y matrices, además de las pilas que fueron de mucha importancia para la realización del algoritmo del "backtracking", también se cumplió con los rubros que especificaba el programa, como lo eran, la realización de funciones descritas anteriormente, la variedad de tamaño de sudoku y la eficiencia del algoritmo.

Marco teórico

A lo largo del curso y para trabajar en este proyecto se utilizó el lenguaje de programación Python. Para muchas personas es común que este sea su primer contacto con algún lenguaje de este tipo y no es coincidencia, en el 2020 Python ocupó el segundo puesto entre los lenguajes de programación más utilizados del año según un reporte realizado por GitHub.[1]

Python fue creado a finales de la década de los ochenta por el informático holandés Guido Van Rossum, aunque su primera publicación fue en 1991. Es un lenguaje de programación interpretado, de propósito general y de código abierto que se caracteriza por la simplicidad de su sintaxis, lo cual lo hace fácil de entender y aprender, esto lo ha convertido en uno de los lenguajes más usados de la última década, especialmente para quienes están iniciando a programar.

Su popularidad en la industria de computación también se debe a que cuenta con una gran cantidad de funciones, módulos y bibliotecas ya incluidas, lo cual ofrece muchas ventajas si se saben utilizar de manera correcta. Además de esto, es multiplataforma, por lo que su código puede ser ejecutado en diversos sistemas operativos, como Linux, MacOS o Windows. Al ser de propósito general, Python puede ser utilizado para distintas disciplinas, como la creación de aplicaciones móviles o para computadoras, desarrollo de páginas web, juegos y hasta inteligencia artificial. [2]

Otra de las herramientas utilizadas para llevar a cabo este proyecto fueron los IDE PyCharm e Ideone. Un entorno de desarrollo integrado (o IDE por sus siglas en inglés), es un programa que permite la escritura, edición y compilación de código fuente, entre otras cosas. Son especialmente útiles debido a que la mayoría se encargan de ir analizando partes del código a como se escribe, son capaces de detectar ciertos errores y la mayoría resaltan la sintaxis del código por lo que hacen de programar una experiencia más amena.[3] PyCharm es un IDE creado por la empresa JetBrains específicamente para Python, actualmente es uno de los entornos de desarrollo integrado más

populares para este lenguaje. Mientras que Ideone es un IDE en línea que permite trabajar con una gran variedad de lenguajes, no solamente Python.

En estos últimos proyectos se investigó sobre un sistema de control de versiones llamado Git. Este software fue desarrollado por el ingeniero en sistemas Linus Torvalds y está pensando especialmente para facilitar el trabajo en equipo a la hora de programar, debido a que permite llevar el control de los cambios que se realizan en el código fuente mediante un historial de las versiones. Git es una herramienta rápida y potente, de software libre y utiliza un sistema de trabajo con ramas.

Git es sumamente conveniente para el trabajo de proyectos en equipo por la forma en que permite organizar el código y su evolución, además de la manera en que permite crear nuevas ramas para probar o experimentar con ideas en ambientes de trabajo que no afecten a la rama principal, pero si se quiere añadir al proyecto lo que se trabajó en este ambiente también es posible hacerlo.[4]

Descripción de la solución

La primera función que se creó para la solución del proyecto fue la de "entrada", la cual requirió de la creación de una variable global que leyera la primera línea del sudoku para saber cuantas líneas tendría que leer luego de esto, así la función podría crear una sola matriz con todas las líneas ingresadas por el usuario. Esto debido a que en la entrada no se indica de cuánto por cuánto era el sudoku. Al terminar este proceso la función llamaba a "resolverSudoku".

Esta próxima función, la cual se puede denominar como la principal ya que se encargaba de llamar a casi todas las funciones creadas para resolver el sudoku, iniciaba introduciendo a la pila el sudoku enviado por "entrada". En ella se creó el ciclo *while* característico del algoritmo de *backtracking* visto en clase. Dentro del ciclo, además del tope de la pila, se declaró una variable llamada "posibilidades" que utilizaba una función para obtener la casilla con menos posibilidades del sudoku. Dicha función al encontrar una casilla con un cero mediante dos ciclos *for*, llamaba a una función auxiliar de nombre "obtenerPosibilidades", la cual será explicada en brevedad. Esto permitía determinar con cuantas posibilidades contaba la casilla vacía. Si solamente tenía 1, inmediatamente retornaba la posición de la casilla y su única opción. De ser más, entonces la función comparaba la cantidad de posibilidades con las de las otras casillas para así devolver la posición y opciones de la casilla que contara con menos posibilidades.

La función "obtenerPosibilidades" utilizaba un *for* para pasar por los números del 1 hasta el tamaño del sudoku y al verificar si el número era válido con la función "contradicción" insertaba en una lista los números que podrían utilizarse en la casilla. Esta última mencionada al recibir el sudoku, la posición de una casilla y un número por insertar, pasaba por medio de ciclos por la columna, fila y región en la que la casilla se encontraba. Si en alguna ya estaba el número por insertar, retornaba verdadero, de lo contrario retornaba falso. Revisar las filas y las columnas fue simple, la parte compli-

cada fue la lógica detrás de revisar las regiones.

Al revisar estos bloques se intentó con distintas operaciones, especialmente con sumas y restas. El problema es que al utilizar sumas y restas no se logró generalizar una operación, ya que esta debía cambiar cada vez que se recibía una casilla en una región diferente, así que se pasó a intentar con divisiones y multiplicaciones. De esta manera se descubrió que al aplicar una división entera sobre la posición en la fila o en la columna de la casilla, el cociente resultante era el valor exacto para que al ser multiplicado por el tamaño de las regiones diera el inicio de estas. Claramente, si al inicio se le sumaba el tamaño de las regiones, daría el final, es decir, ya se tenían los valores por los que se debía iterar.

Luego de esta explicación, se puede proseguir con la función principal. Dentro del ciclo se indicó que si el sudoku trabajado era solución entonces debía imprimirse y el *while* se detenía. La función que determinaba si un sudoku estaba resuelto, simplemente verificaba que no quedaran más ceros en la matriz. En el caso de que aún no fuese solución, se verificaba si había un sudoku que tuviera una única posibilidad y, de ser así, se agregaba a la pila. De lo contrario, se pasaba por otro ciclo que por cada posible sudoku, creaba una copia de este, asignaba las posibilidades de sus filas y columnas y lo agregaba a la pila. De esta manera el programa iteraba hasta encontrar la solución del sudoku ingresado.

Resultados de pruebas

Casos de prueba		
Entrada	**Salida**	**Tiempo**
5 3 0 0 7 0 0 0 0	5 3 4 6 7 8 9 1 2	
6 0 0 1 9 5 0 0 0	6 7 2 1 9 5 3 4 8	
0 9 8 0 0 0 0 6 0	1 9 8 3 4 2 5 6 7	
8 0 0 0 6 0 0 0 3	8 5 9 7 6 1 4 2 3	
4 0 0 8 0 3 0 0 1	4 2 6 8 5 3 7 9 1	0.04s
7 0 0 0 2 0 0 0 6	7 1 3 9 2 4 8 5 6	
0 6 0 0 0 0 2 8 0	9 6 1 5 3 7 2 8 4	
0 0 0 4 1 9 0 0 5	2 8 7 4 1 9 6 3 5	
0 0 0 0 8 0 0 7 9	3 4 5 2 8 6 1 7 9	
0 0 0 3 0 1 0 9 0	7 4 6 3 8 1 2 9 5	
5 0 0 6 0 0 1 0 0	5 3 8 6 9 2 1 4 7	
0 0 9 0 0 0 0 0 8	2 1 9 4 5 7 3 6 8	
0 0 0 7 0 0 0 3 0	8 9 1 7 6 5 4 3 2	
0 7 4 2 0 8 0 0 0	6 7 4 2 3 8 9 5 1	0.2s
0 0 5 0 0 9 0 0 6	3 2 5 1 4 9 7 8 6	
0 0 0 5 0 0 0 0 0	4 8 7 5 2 3 6 1 9	
0 6 0 0 0 0 0 7 3	9 6 2 8 1 4 5 7 3	
0 0 0 0 7 0 8 0 0	1 5 3 9 7 6 8 2 4	
0 0 0 0 7 0 1 0 0	3 6 9 2 7 5 1 8 4	
0 7 0 0 0 0 0 5 9	1 7 2 4 6 8 3 5 9	
0 0 4 3 0 9 0 0 0	8 5 4 3 1 9 6 2 7	
0 3 7 0 5 6 0 4 0	2 3 7 1 5 6 9 4 8	
0 1 0 0 3 0 0 0 2	4 1 8 9 3 7 5 6 2	0.08s
0 9 0 8 0 0 7 0 0	5 9 6 8 4 2 7 3 1	
0 0 0 0 0 0 2 0 5	9 4 1 6 8 3 2 7 5	
0 0 0 0 0 0 0 1 0	7 2 3 5 9 4 8 1 6	
6 0 5 0 2 0 4 0 0	6 8 5 7 2 1 4 9 3	

Casos de prueba		
Entrada	**Salida**	**Tiempo**
8 0 0 0 0 0 0 0 0 0 0 3 6 0 0 0 0 0 0 7 0 0 9 0 2 0 0 0 5 0 0 0 7 0 0 0 0 0 0 0 4 5 7 0 0 0 0 0 1 0 0 0 3 0 0 0 1 0 0 0 0 6 8 0 0 8 5 0 0 0 1 0 0 9 0 0 0 0 4 0 0	8 1 2 7 5 3 6 4 9 9 4 3 6 8 2 1 7 5 6 7 5 4 9 1 2 8 3 1 5 4 2 3 7 8 9 6 3 6 9 8 4 5 7 2 1 2 8 7 1 6 9 5 3 4 5 2 1 9 7 4 3 6 8 4 3 8 5 2 6 9 1 7 7 9 6 3 1 8 4 5 2	2.58s
0 0 0 0 0 0 2 0 0 0 8 0 0 0 7 0 9 0 6 0 2 0 0 0 5 0 0 0 7 0 0 6 0 0 0 0 0 0 0 9 0 1 0 0 0 0 0 0 0 2 0 0 4 0 0 0 5 0 0 0 6 0 3 0 9 0 4 0 0 0 7 0 0 0 6 0 0 0 0 0 0	9 5 7 6 1 3 2 8 4 4 8 3 2 5 7 1 9 6 6 1 2 8 4 9 5 3 7 1 7 8 3 6 4 9 5 2 5 2 4 9 7 1 3 6 8 3 6 9 5 2 8 7 4 1 8 4 5 7 9 2 6 1 3 2 9 1 4 3 6 8 7 5 7 3 6 1 8 5 4 2 9	0.67s
3 0 0 0 0 0 0 0 0 6 5 0 0 1 0 0 7 0 0 0 4 2 0 7 5 0 0 0 0 0 9 0 0 0 0 0 0 0 0 7 0 0 0 0 9 0 9 0 0 6 3 0 4 2 1 7 5 0 3 0 0 6 4 0 4 0 0 0 0 0 0 5 0 0 8 0 0 0 0 0 0	3 2 7 6 8 5 4 9 1 6 5 9 3 1 4 2 7 8 8 1 4 2 9 7 5 3 6 5 6 2 9 4 8 7 1 3 4 8 3 7 2 1 6 5 9 7 9 1 5 6 3 8 4 2 1 7 5 8 3 2 9 6 4 2 4 6 1 7 9 3 8 5 9 3 8 4 5 6 1 2 7	0.04s

Conclusiones

Al realizarse diversas pruebas se comprobó la funcionalidad del programa, probando con sudokus de diferente dificultad y obteniendo respuestas satisfactorias con todos ellos. Al lado de probar la funcionalidad del programa también se pudo comprobar el tiempo de ejecución de este y se observó cómo, dependiendo del sudoku, este puede llegar a durar más que con el resto. De cualquier manera, consideramos que se consiguió obtener un buen tiempo de ejecución en general.

Con todo lo conseguido se puede afirmar que el programa fue un éxito y, más allá del programa, el proyecto en general ya que de igual manera se logró cumplir con la documentación. Todos los participantes del grupo cumplieron con sus respectivas labores que se fueron asignando a lo largo de distintas reuniones y no se presentó ningún inconveniente a lo largo de la realización del proyecto. Consideramos que le debemos el éxito en el proyecto a la debida división del proyecto en partes, esto fue un elemento fundamental que facilitó en gran manera la realización de este.

Finalmente consideramos que se cumplió con el objetivo general del proyecto el cual era demostrar capacidad para usar matrices, listas y pilas, ya que sin conocimientos previos en el ámbito de listas y matrices se hubiera complicado enormemente la realización del proyecto. Los conocimientos previos nos ayudaron a comprender correctamente el problema principal y a partir de ahí encontrar su solución correspondiente, a la vez que también tuvimos la oportunidad de empezar a practicar el uso de pilas.

Aprendizajes

Fernando

En particular, me llamó mucho la atención la gran utilidad que tiene el "backtracking" en la programación (más que todo que exista un solo algoritmo para la solución de este tipo de problemas). De parte del proyecto, se puede decir que conseguí tener un mayor manejo de las pilas y, me quedo la experiencia el programar un algoritmo que ya venía dentro de una de las funciones del mismo Python. Me gustaría agregar que, con respecto a mis compañeros, me parece que se trabajó de una manera óptima y por lo mismo es que se hizo medianamente sencillo el llegar a una solución del trabajo.

Ana

Este proyecto fue una buena manera de reforzar los conceptos que se deben tener claros con las matrices y las listas para poder manipular ambas estructuras, pero además fue una gran introducción a las pilas. Ya en clase el profesor mencionó cómo aprender del backtracking nos abre la puerta a las soluciones de una cantidad enorme de problemas, así que considero sumamente útil el haber empezado a familiarizarnos con esta técnica. Además, el proceso de trabajar en un proyecto se vuelve más ameno cuando se tiene compañeros dispuestos a trabajar realmente en equipo, aportar ideas y dar críticas constructivas.

Pablo

Personalmente puedo rescatar la oportunidad de experimentar con el backtracking, considero esto de bastante utilidad para complementar lo visto en clase relacionado al tema, a su vez, el practicar con matrices y listas también lo considero de gran utilidad para mantener el tema fresco e incluso ayudar a reforzarlo. Nuevamente, al volver a trabajar en equipo considero

que se puede aprender bastante al ver trabajar a otras personas, desde su forma de orientar el problema hasta su forma de encontrarle una respuesta, halló interesante como los métodos pueden variar tanto de una persona a otra y aprender de las virtudes de cada uno para fortalecer el propio trabajo individual.

Referencias

[1] GitHub, "The state of the octoverse." [Online]. Available: https://octoverse.github.com/

[2] javaTpoint, "Python history." [Online]. Available: https://www.javatpoint.com/python-history

[3] R. Hat, "El concepto de ide." [Online]. Available: https://www.redhat.com/es/topics/middleware/what-is-ide

[4] J. C. Rubio, "Qué es git y para qué sirve," Feb 2019. [Online]. Available: https://openwebinars.net/blog/que-es-git-y-para-que-sirve/