UTEM

Trabajo 1
Análisis de Algoritmos

Integrantes:
Javier Nanco Becerra
Constanza Olivos Fernández

Profesor:
Luis Herrera

Fecha de entrega: 28/04/2024

Índice

Introducción

La problemática consiste en solucionar el taller 1 de análisis de algoritmo que tiene como objetivo crear un tablero y colocar piezas de domino de forma que todas las columnas y filas estén interrumpidas por al menos una pieza de domino, donde cada pieza ocupa dos casillas del tablero a lo cual durante el informe lo llamaremos corte. El producto del número de filas por columnas debe ser un número par para que pueda existir solución al problema. El objetivo es encontrar todas las soluciones posibles para este problema a través del algoritmo de Backtracking, el cual es explorar todas las posibles soluciones por medio de una búsqueda en profundidad de manera recursiva, descartando aquellas que no satisfacen las restricciones del problema.

Desarrollo

Decidimos realizar este trabajo en Python. En primer lugar, Python ofrece una amplia gama de bibliotecas que nos podían ser útiles en la creación del proyecto, y si bien es cierto que en nuestra lógica no utilizamos las librerías, si lo hicimos para la medición de tiempo y la creación del Excel. Además es bastante simple el manejo de arreglos y matrices, lo cual es fundamental para nuestro trabajo. Junto con esto, notamos que Python tiene un rendimiento superior en operaciones recursivas en comparación con otros lenguajes. Esto es muy necesario, ya que nuestro proyecto implica una gran cantidad de cómputo.

Empezamos haciendo un diagrama de flujo para planificar cómo íbamos a realizar el código, como se ve en la primera imagen. Después, consideramos algunas restricciones importantes, como asegurarnos de que las piezas no se superpongan ni salgan del tablero, como se muestra en la segunda imagen. En la última figura, nos aseguramos de que cada fila y columna del tablero tuviera al menos un corte.

El programa empieza pidiendo al usuario el tamaño del tablero, debe ingresar a través de la consola el valor de las filas y columnas. Posterior a esto es necesario verificar que filas por columnas den como resultado un número par y enseguida llenar la matriz con ceros. Luego, el programa llama a la función llenar_tablero() para colocar las piezas. Primero verifica si el lugar donde queremos poner la pieza está vacío y además no es la última fila. En caso de ser verdadero, pasa al siguiente paso y verifica si podemos colocar la pieza horizontalmente. Si podemos, coloca la pieza en el tablero y marca los cuadrados ocupados con el número de la pieza. Si no podemos colocar la pieza horizontalmente, borra lo que intentó y prueba si la pieza se puede mover verticalmente. Después, añadimos condiciones con las restricciones y llamamos a llenar_tablero() agregando en 1 más el valor de la pieza siguiente hasta que el tablero esté lleno correctamente, con al menos un corte en cada fila y columna.

Finalmente, el programa imprime el tiempo que tardó en hacer el Backtracking. También crea un Excel con todas las matrices que fueron solución al problema, para que podamos verlas y verificarlas de una manera más legible.

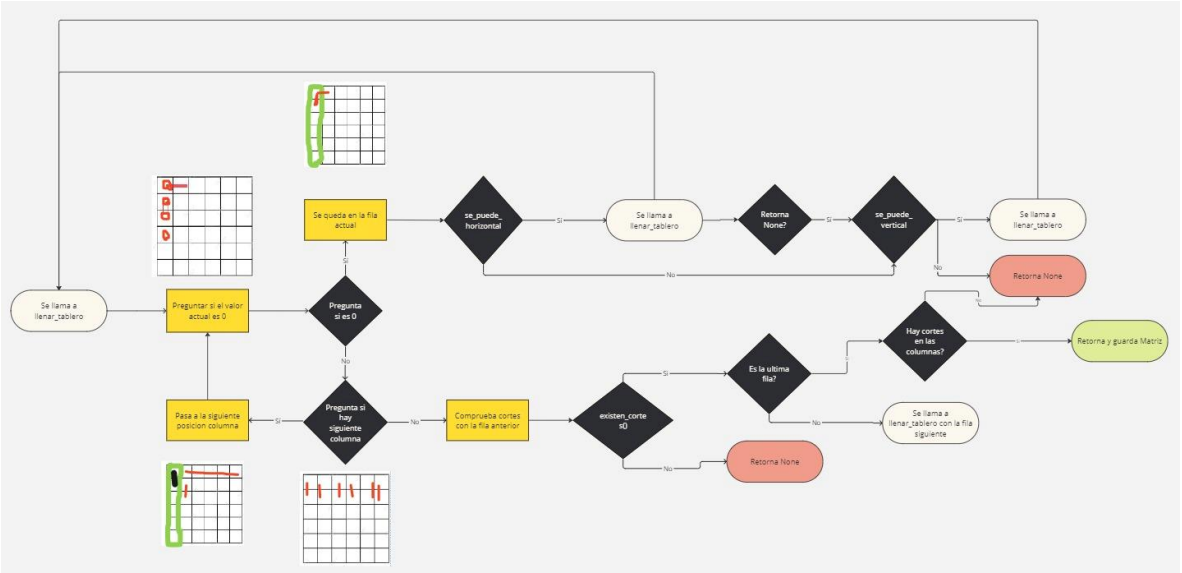

Ilustración 1: Diagrama general

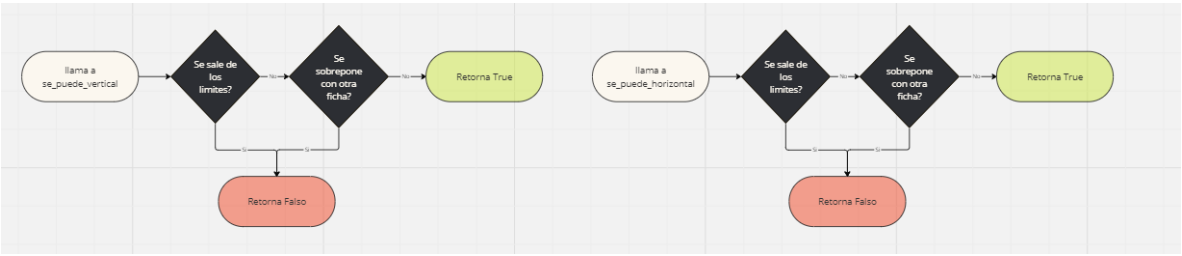

Ilustración 2: Diagrama con restricciones

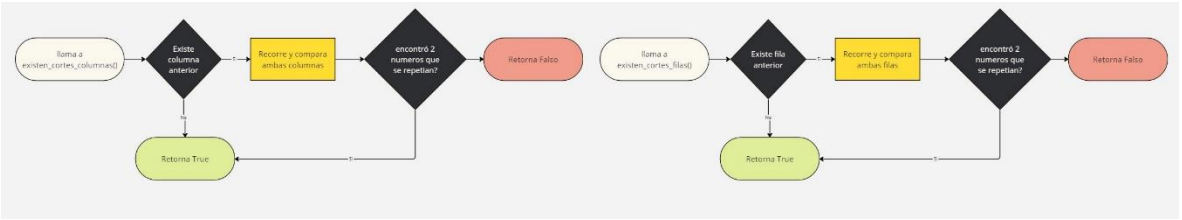

Ilustración 3: Diagrama para verificar cortes

Restricciones

Las restricciones que tomamos en cuenta para la realización del Backtracking son:

1. El producto de filas por columnas debe ser par: Esta restricción se verifica en la función crear_tablero_vacio(), donde se solicita al usuario que ingrese el número de filas y columnas, y se verifica si el producto de estos es par.

2. No sobre poner piezas: Esta restricción se verifica en la función comprueba_ceros(), que garantiza que las posiciones donde se colocan las piezas estén vacías antes de colocarlas.

3. Las piezas no deben salirse del tablero: Esta restricción se verifica en las funciones es_movimiento_valido_H() y es_movimiento_valido_V(), que comprueban que los movimientos horizontales y verticales sean válidos y que no se salgan del tablero.

4. Debe haber cortes en las filas y columnas: Esta restricción se verifica en las funciones existen_cortes_columnas() y existen_cortes_filas(), las cuales revisan que al finalizar las filas o columnas existan cortes.

Problemas encontrados

Todas las matrices cuadradas menores a 8x8 no tiene solución o todas las matrices menores a 6x5, ya que no se pueden cortar todas sus filas y columnas.

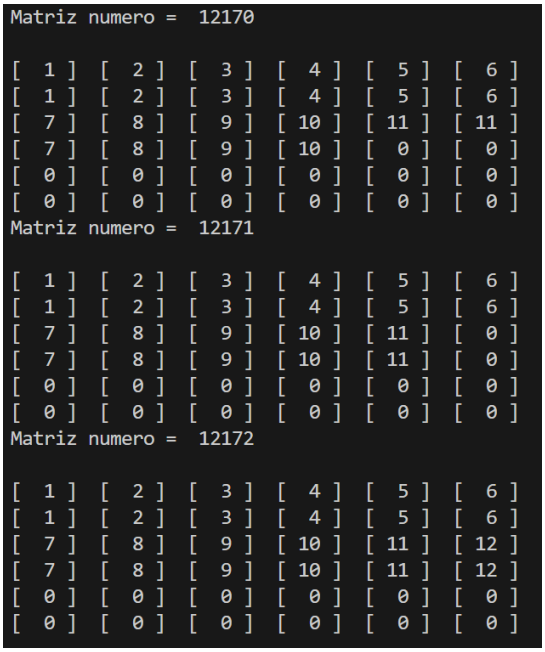

Ilustración 4: Matriz 6x6 inicio vertical

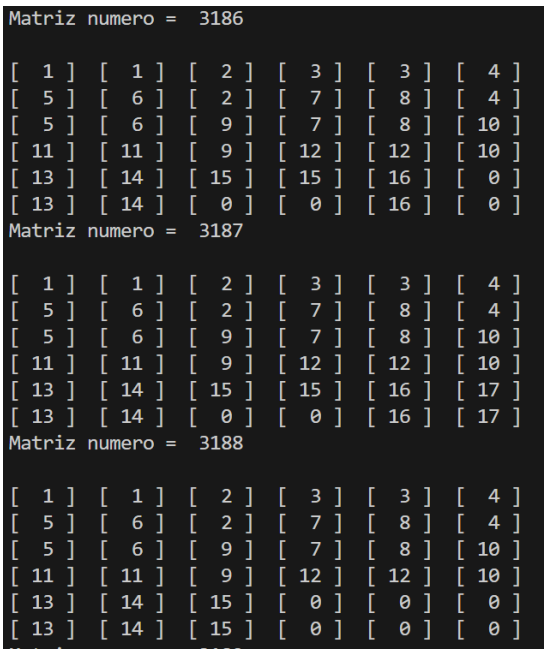

Ilustración 5: Matriz 6x6 inicio horizontal

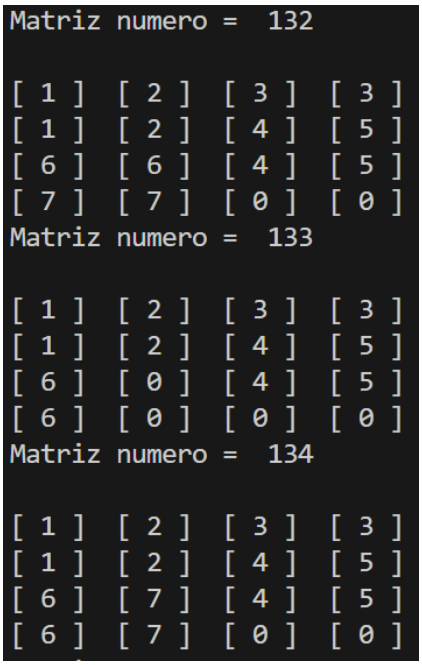

Ilustración 6: Matriz 4x4 inicio vertical

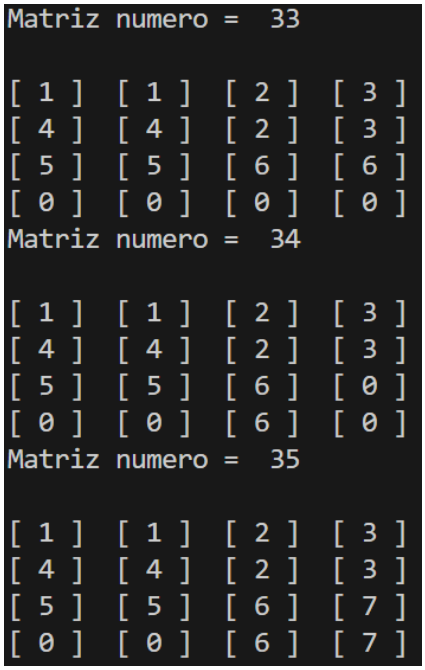

Ilustración 7: Matriz 4x4 inicio horizontal

Estos son ejemplos de tableros cuadrados más pequeños que 8x8. Los dos primeros son de 6x6 y los siguientes dos son de 4x4. Después de intentar encontrar una solución para el 6x6 durante unos 0.019 segundos aproximadamente y 12172 intentos, y para el 4x4 con 162 intentos en 0.000993 segundos, no se pudo encontrar una matriz donde todas las columnas y filas estuvieran cortadas después de probar a llenar la matriz con la primera pieza tanto vertical como horizontalmente.

Cuando empezamos a hacer el diagrama de flujo, no nos pareció tan difícil al principio. Pero a medida que avanzábamos, surgieron problemas. Estábamos usando la técnica de fuerza bruta para buscar soluciones, pero nos costaba integrar el Backtracking. Sin embargo, a medida que seguimos adelante, empezamos a entenderlo mejor y a implementar correctamente las funciones recursivas. Logramos hacer que el programa volviera atrás y dejara un espacio vacío si la opción de la pieza no era viable, antes de intentar con otra opción.

Inicialmente, intentamos considerar todos los posibles movimientos de la pieza, como se puede ver en la ilustración numero 8 donde la pieza amarilla es cuando era vertical y la celeste cuando era horizontal y todas las piezas a sus alrededores son los movimientos posibles, pero no pudimos hacerlo, ya que el programa se demoraba mucho al probarlos todos. Por eso, al final decidimos limitarnos a movimientos horizontales y verticales. También nos costó bastante implementar el corte de columnas. Al principio, el código solo verificaba los cortes en las filas y, si no se cumplían, volvía atrás y lo corregía. Pero para las columnas no hicimos eso, así que tuvimos que agregar un verificador de columnas. Al final, tuvimos nos aseguramos de que la matriz no contenga ceros y que todas las filas y columnas estuvieran cortadas en la matriz antes de pasar a buscar la próxima solución.

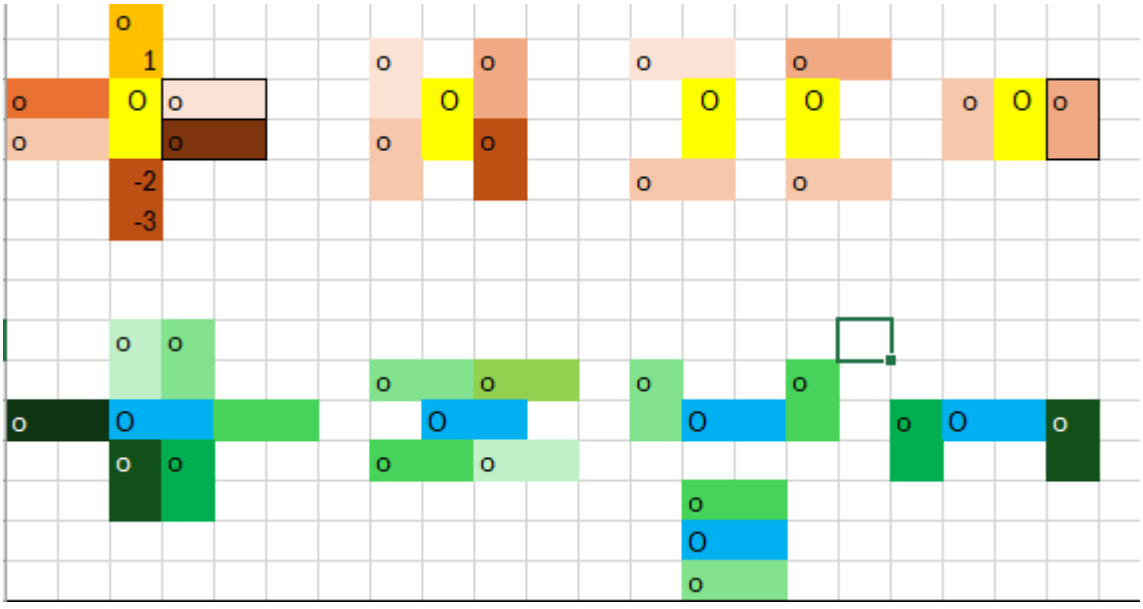

Ilustración 8: Movimiento de piezas

El backtracking es una técnica de búsqueda exhaustiva que explora todas las posibles soluciones de un problema. Sin embargo, la eficiencia del backtracking puede depender de varios factores, incluidos los siguientes:

Restricciones del problema: Las restricciones del problema pueden limitar el número de soluciones válidas y, por lo tanto, influir en la cantidad de configuraciones que el algoritmo de backtracking debe explorar.

Estructura del tablero: La forma en que se estructura el tablero y se colocan las piezas puede afectar la cantidad de soluciones posibles. En algunos casos, la disposición del tablero puede limitar el número de configuraciones válidas.

Eficiencia del algoritmo: La eficiencia del algoritmo de backtracking también puede influir en la capacidad para explorar todas las soluciones posibles en un tiempo razonable. Un algoritmo de backtracking bien implementado y optimizado puede encontrar un gran número de soluciones en un tiempo razonable, mientras que un enfoque menos eficiente puede ser más lento o incluso inmanejable para problemas grandes .

En resumen, si bien el backtracking tiene el potencial de explorar todas las soluciones posibles, la cantidad de soluciones encontradas puede variar según los factores mencionados anteriormente. En última instancia, el número de soluciones que se encuentren depende de la naturaleza del problema y de cómo se implemente y ejecute el algoritmo de backtracking, por lo que es muy probable que existan variaciones en las respuestas a la misma problemática.

Resultados

Diagrama de cómo funciona el Backtracking

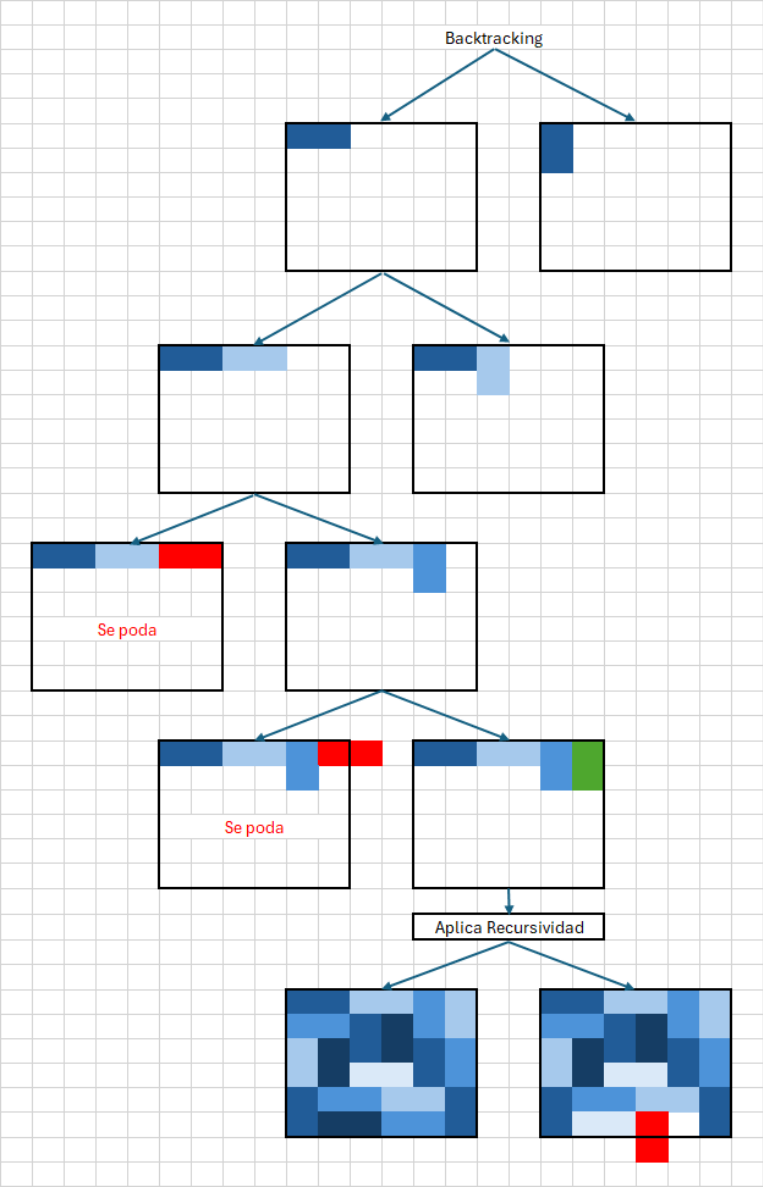

Ilustración 9: Diagrama Backtracking

En la ilustración número 9 se ejemplifica como se finalizan las salidas y se van podando las ramas que no nos llevan a la solución correcta.

Este es un ejemplo de una matriz 9x8, la cual se demoró 123.476 segundos, aproximadamente 2 minutos, en encontrar 555473 soluciones. Se logra ver que luego de 20000 matrices pasa a crear la siguiente hoja de Excel para seguir llenándola con las matrices soluciones.

Matriz 551537							
1	2	3	4	5	6	6	7
1	2	3	4	5	8	9	7
10	11	11	12	12	8	9	13
10	14	15	16	16	17	17	13
18	14	15	19	20	21	22	22
18	23	24	19	20	21	25	26
27	23	24	28	29	29	25	26
27	30	30	28	31	31	32	33
34	34	35	35	36	36	32	33
Matriz 551538							
1	2	3	4	5	6	6	7
1	2	3	4	5	8	9	7
10	11	11	12	12	8	9	13
10	14	15	16	16	17	17	13
18	14	15	19	20	21	22	22
18	23	24	19	20	21	25	26
27	23	24	28	29	29	25	26
27	30	30	28	31	32	32	33
34	34	35	35	31	36	36	33
Matriz 551539							
1	2	3	4	5	6	6	7
1	2	3	4	5	8	9	7
10	11	11	12	12	8	9	13
10	14	15	16	16	17	17	13
18	14	15	19	20	21	22	22
18	23	24	19	20	21	25	26
27	23	24	28	29	29	25	26
27	30	30	28	31	32	33	33
34	34	35	35	31	32	36	36

Ilustración 10: Matriz 9x8

170171	Matriz 555471							
170172	1	2	3	4	5	6	7	7
170173	1	2	3	4	5	6	8	9
170174	10	11	12	13	14	15	8	9
170175	10	11	12	13	14	15	16	17
170176	18	19	19	20	21	21	16	17
170177	18	22	22	20	23	24	25	25
170178	26	27	28	28	23	24	29	29
170179	26	27	30	31	31	32	32	33
170180	34	34	30	35	35	36	36	33
170181								
170182	Matriz 555472							
170183	1	2	3	4	5	6	7	7
170184	1	2	3	4	5	6	8	9
170185	10	11	12	13	14	15	8	9
170186	10	11	12	13	14	15	16	17
170187	18	19	20	20	21	21	16	17
170188	18	19	22	23	23	24	24	25
170189	26	26	22	27	28	29	29	25
170190	30	31	31	27	28	32	33	33
170191	30	34	34	35	35	32	36	36
170192								
170193	Matriz 555473							
170194	1	2	3	4	5	6	7	7
170195	1	2	3	4	5	6	8	9
170196	10	11	12	13	14	15	8	9
170197	10	11	12	13	14	15	16	17
170198	18	19	20	20	21	21	16	17
170199	18	19	22	23	23	24	25	25
170200	26	26	22	27	28	24	29	29
170201	30	31	31	27	28	32	32	33
170202	30	34	34	35	35	36	36	33
170203								

Sheet19 Sheet20 Sheet21 Sheet22 Sheet23 Sheet24 Sheet25 Sheet26 **Sheet27**

Ilustración 11: Hoja de excel de 9x8

Conclusiones

El proyecto abordó de manera efectiva el desafío de encontrar todas las soluciones posibles para el problema planteado, utilizando el algoritmo de Backtracking en Python. Se destacó la elección de Python como lenguaje de programación debido a su amplia gama de bibliotecas útiles y su rendimiento superior en operaciones recursivas, aspecto crucial para este proyecto.

El informe proporcionó una descripción detallada del proceso de desarrollo, desde la planificación inicial con diagramas de flujo hasta la implementación de las restricciones necesarias para garantizar la validez de las soluciones generadas. Se discutieron las dificultades encontradas durante el desarrollo, como la gestión de movimientos de piezas y la verificación de cortes en filas y columnas, y se explicaron las soluciones adoptadas para superar estos desafíos.

Si bien el Backtracking ofrece la capacidad de explorar exhaustivamente todas las soluciones posibles, la cantidad de soluciones encontradas puede variar según los factores mencionados. En última instancia, la implementación y ejecución del algoritmo de Backtracking influyen en el número de soluciones encontradas para un problema dado.

Además, se presentaron ejemplos concretos de matrices de tamaño reducido que demostraron la incapacidad de encontrar soluciones para ciertos casos, lo que resalta la importancia de comprender las limitaciones del algoritmo utilizado y las restricciones del problema.

Finalmente, se proporcionó una visualización del funcionamiento del algoritmo de Backtracking a través de un diagrama, lo que contribuyó a una comprensión más clara de su operación interna y su capacidad para explorar exhaustivamente todas las posibles soluciones.

En resumen, el informe brindó una visión completa del proceso de desarrollo y las consideraciones clave involucradas en la implementación del algoritmo de Backtracking para resolver el problema planteado, demostrando un enfoque riguroso y metodológico para abordar desafíos algorítmicos complejos.

Bibliografías

- Diagrama de flujo: https://miro.com/welcomeonboard/Tkp3RnNkNW00cUxCSWowcjZpVW9LVWp3eW13TU5leTJVUkMxOWp5SHlpQXJrUHNRcDk4SmZLcDFFTlljTE8xSXwzNDU4NzY0NTQwMTAxMjMyMjU4fDI=?share_link_id=420552079715

- Explicación Backtracking: https://www.youtube.com/watch?v=-bjTb0o6EAQ&t=639s&ab_channel=SantiagoFiorino

- Explicación Backtracking: https://www.youtube.com/watch?v=KjEmT74Z1wk