Departamento de Ingeniería de Sistemas y Computación
Diseño y análisis de algoritmos
ISIS-1105

DOCUMENTO ANÁLISIS P2

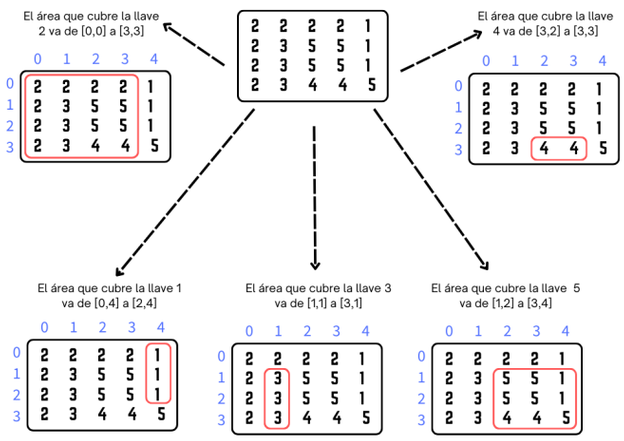
Juan David Rios Nisperuza, 202215787, jd.riosn1@uniandes.edu.co

Laura Julieth Carretero Serrano, 202214922, l.carretero@uniandes.edu.co

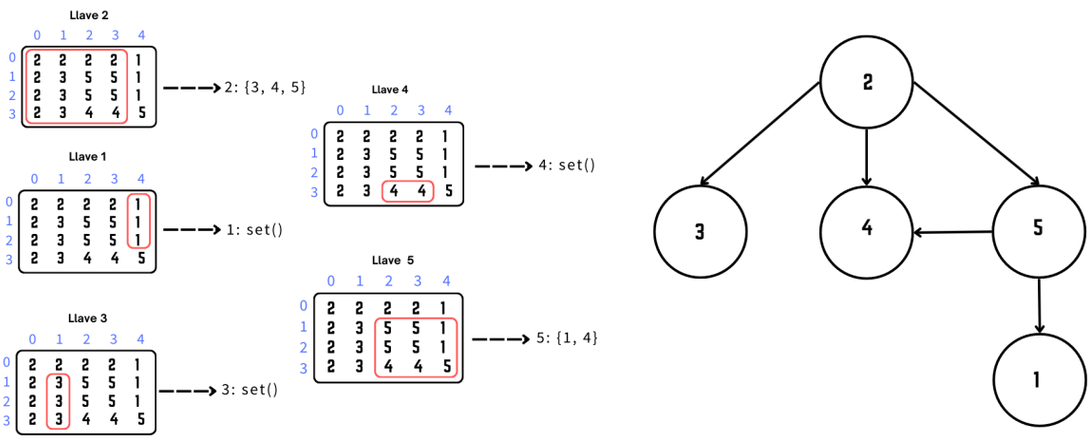
1. Algoritmo de solución

La función “encontrar_esquinas_submatrices” recibe una matriz como entrada y encuentra las esquinas superior izquierda e inferior derecha de cada bloque contiguo de un mismo valor en la matriz. Utiliza un diccionario para almacenar estas esquinas por cada valor en la matriz. Luego, la función “obtener_valores_encerrados” toma la matriz original y el diccionario de esquinas para determinar qué otros valores están encerrados dentro de cada submatriz. Se construye un grafo dirigido que captura las dependencias entre los valores.

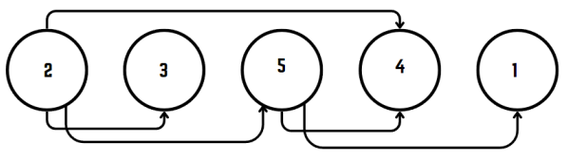
Encontrar_esquinas_submatrices:



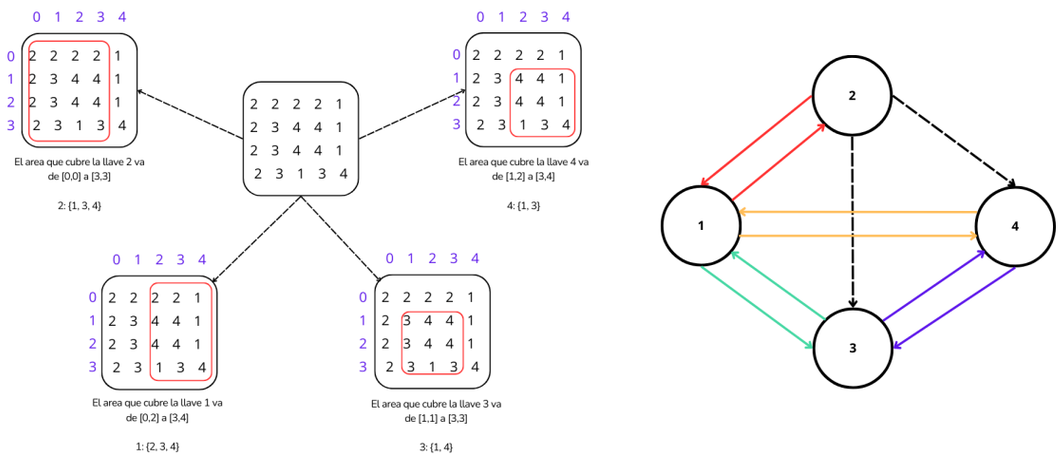
Obtener_valores_encerrado:



Posteriormente, se aplica un algoritmo de ordenamiento topológico para establecer un orden lógico de apertura de las cajas, considerando las dependencias entre los valores. Este algoritmo utiliza un diccionario para almacenar los grados de entrada de cada nodo en el grafo. Si el grafo contiene ciclos, se retorna `None` indicando que no se pueden abrir las cajas dado que se producen colisiones. Por otro lado, si no hay ciclos, se retorna la secuencia de nodos en orden topológico.



Ejemplo para el cual el algoritmo implementado arrojará “NO SE PUEDE”:



Para este ejemplo podemos apreciar que se producen ciclos, por tanto el algoritmo implementado retornara NONE e imprimirá en consola “NO SE PUEDE”.

Por último, la función solve lo que hace es coordinar el proceso de encontrar las esquinas, identificar los valores encerrados, aplicar el ordenamiento topológico y finalmente, imprimir el resultado, ya sea el orden de apertura de las cajas o la indicación de que no es posible abrirlas.

2. Análisis de complejidades espacial y temporal

Complejidad espacial: La complejidad espacial del código es $O(m\ x\ n)$ debido a que la matriz de entrada, que tiene dimensiones $m\ x\ n$. Todas las demás estructuras de datos, como diccionarios y listas, dependen directamente del tamaño de esta matriz. En el peor caso, cuando cada elemento de la matriz es único, la cantidad de memoria utilizada crece de manera proporcional al tamaño total de la matriz. Por lo tanto, la complejidad espacial general del código está esencialmente determinada por la representación de la matriz.

Complejidad temporal: La complejidad temporal de esta solución se estima en $k\ x\ m\ x\ n$, y se puede fundamentar de la siguiente manera: en el código, se realiza un bucle para encontrar los bordes de cada llave k veces. Dentro de este bucle, se busca los valores dentro del área de cada llave mediante un recorrido de $m\ x\ n$. Dado que estos bucles están anidados, la complejidad total es $k\ x\ m\ x\ n$.
Adicionalmente, se incorpora la complejidad del orden topológico al utilizar un grafo de listas adyacentes, que tiene una complejidad de O(V + E), donde V es el número de vértices y E es el número de aristas. Si sumamos esta complejidad al triple bucle anterior, obtenemos el peor caso, que sería el triple recorrido de $k\ x\ m\ x\ n$ más la complejidad del orden topológico. La complejidad temporal total se compone de las operaciones dentro de los bucles y la complejidad del orden topológico, lo que resulta en O$((k\ x\ m\ x\ n)\ +\ (V\ +\ E))$ lo cual se resume en O $(k\ x\ m\ x\ n)$

3. Respuestas a los escenarios de comprensión de problemas algorítmicos

Escenario 1. Se puede aplicar la misma llave dos veces

(i) Nuevos retos propuestos: En el contexto de permitir la aplicación repetida de una misma llave, surgen desafíos sustanciales en la determinación de las esquinas de las submatrices, la gestión de la relación entre valores encerrados y la construcción del orden topológico. La premisa inicial de que las esquinas de las submatrices se determinan exclusivamente por el valor de la llave en la matriz se ve cuestionada. Para abordar este desafío, sería necesario implementar una estructura de datos más compleja que permita rastrear múltiples instancias de esquinas de submatrices asociadas a un mismo valor de llave. La lógica para determinar los valores encerrados por cada llave se complica al considerar la posibilidad de aplicar la misma llave dos veces. La relación entre los valores encerrados se vuelve más intrincada, exigiendo una gestión más elaborada de las instancias de una misma llave. La solución podría requerir un enfoque más sofisticado para comprender y manejar las complejidades en la relación entre valores y sus múltiples instancias. La construcción del grafo y el orden topológico, que originalmente asumen que cada valor de llave representa un nodo único, necesitan ajustes sustanciales. La adaptación a múltiples instancias de un mismo valor implica la necesidad de modificar las estructuras de datos y la lógica subyacente, desafiando la simplicidad inicial del enfoque empleado en la solución original. En conjunto, estos desafíos resaltan la complejidad añadida al considerar la aplicación repetida de una misma llave en el escenario propuesto.

(ii) Cambios en la Solución: En la adaptación de la solución para permitir la aplicación de la misma llave dos veces, se plantea la necesidad de realizar cambios fundamentales en la estructura de datos que rastrea las esquinas de las submatrices. En lugar de mantener un único conjunto de esquinas por valor de llave, se propone la implementación de un mapa de listas. Cada lista dentro del mapa podría subdividirse para representar las dos posibles áreas que se pueden utilizar al abrir la caja con la misma llave en diferentes instancias. Esta modificación permitiría una representación más precisa y flexible de las esquinas asociadas a cada valor de llave. Asimismo, la lógica para determinar los valores encerrados por cada llave debe ser actualizada para abordar la complejidad adicional en la relación entre estos valores. Se sugiere incorporar una lógica que considere el tamaño del conjunto resultante del área encerrada. En caso de que el conjunto sea considerablemente grande, podría aplicarse una subdivisión en dos áreas más pequeñas. Por el contrario, si el conjunto es demasiado pequeño, la regla de subdivisión podría no aplicarse. Este enfoque proporciona una mayor adaptabilidad para gestionar diversos escenarios de aplicaciones de llaves, teniendo en cuenta la posibilidad de aplicar la misma llave en múltiples ocasiones.

Escenario 2. Se pueden diseñar estructuras en forma de ele (L)
(i) Nuevos retos propuestos: La incorporación de estructuras en forma de "L" introduce desafíos adicionales en la búsqueda de esquinas y valores encerrados en el código existente. Actualmente diseñado para identificar esquinas de submatrices rectangulares, el código requeriría modificaciones sustanciales para detectar y manejar las formas más complejas que podrían surgir con estas nuevas estructuras. Además, la lógica para obtener valores encerrados debe adaptarse a la posibilidad de que las estructuras en forma de "L" tengan partes que se extienden en direcciones no contiguas, demandando un enfoque más flexible en la verificación de vecinos y la determinación de valores adyacentes.
(ii) Cambios en la Solución: La incorporación de estructuras en forma de "L" requeriría ajustes significativos en el código existente. En la función “encontrarEsquinasSubmatrices”, sería necesario realizar modificaciones para identificar esquinas de formas más complejas, permitiendo una mayor flexibilidad que no se limite exclusivamente a rectángulos o cuadrados. Esto podría implicar la necesidad de identificar si estas estructuras en forma de "L" constan de tres esquinas, lo cual agregaría complejidad al proceso de detección. En la función “obtenerValoresEncerrados”, los ajustes deberían abordar la variabilidad en las formas de las estructuras en forma de "L", incorporando una lógica que, en caso de identificar el valor como "L", realice un conteo sin recorrer la matriz completa, optimizando así el proceso de obtención de valores adyacentes. Estos cambios específicos son esenciales para adaptar el código de manera efectiva a la presencia de estas formas más complejas.