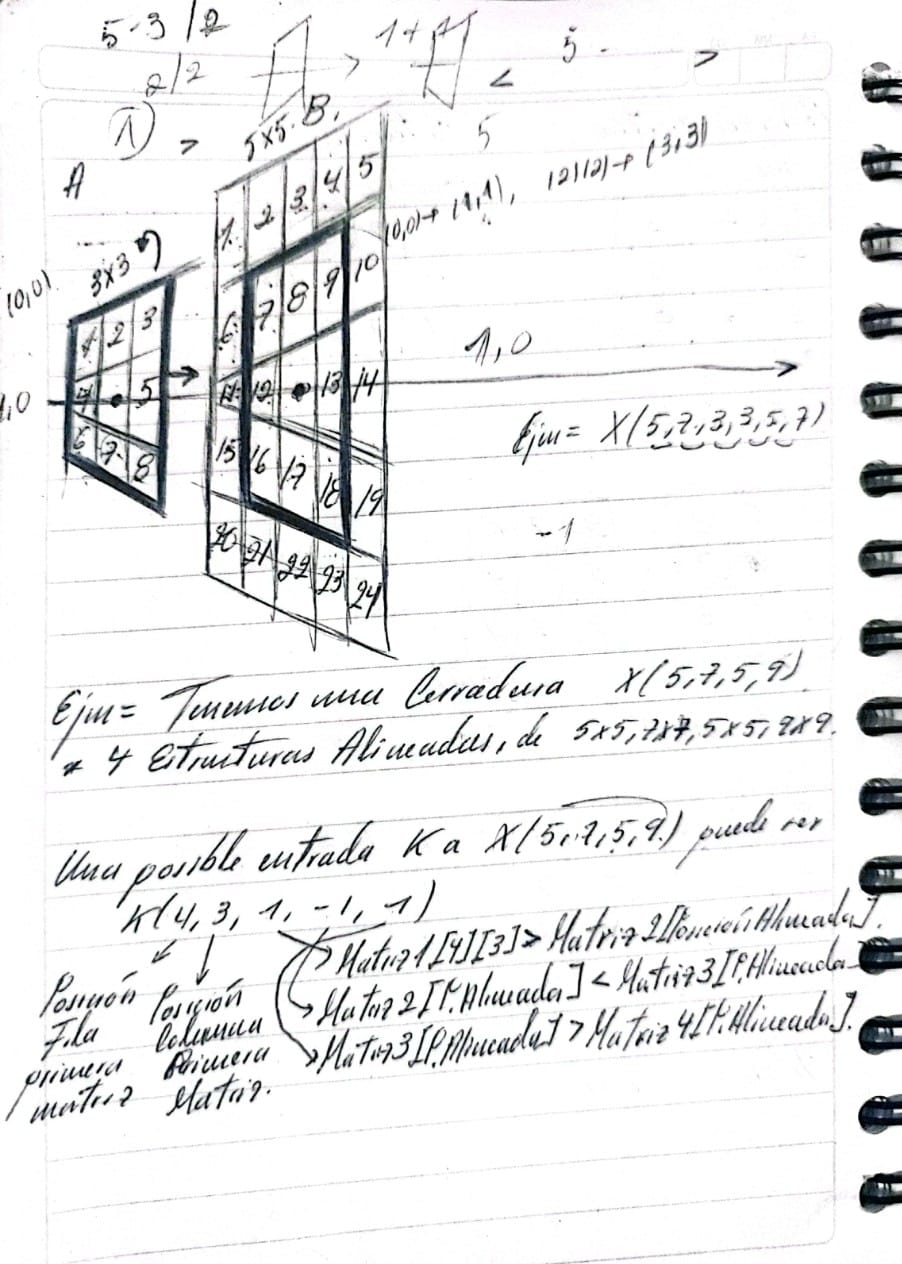
|  |  |
| --- | --- |
|  | **Informática II Semestre 2024 – 1**  **Desafío 1. Marzo 21 de 2024** |

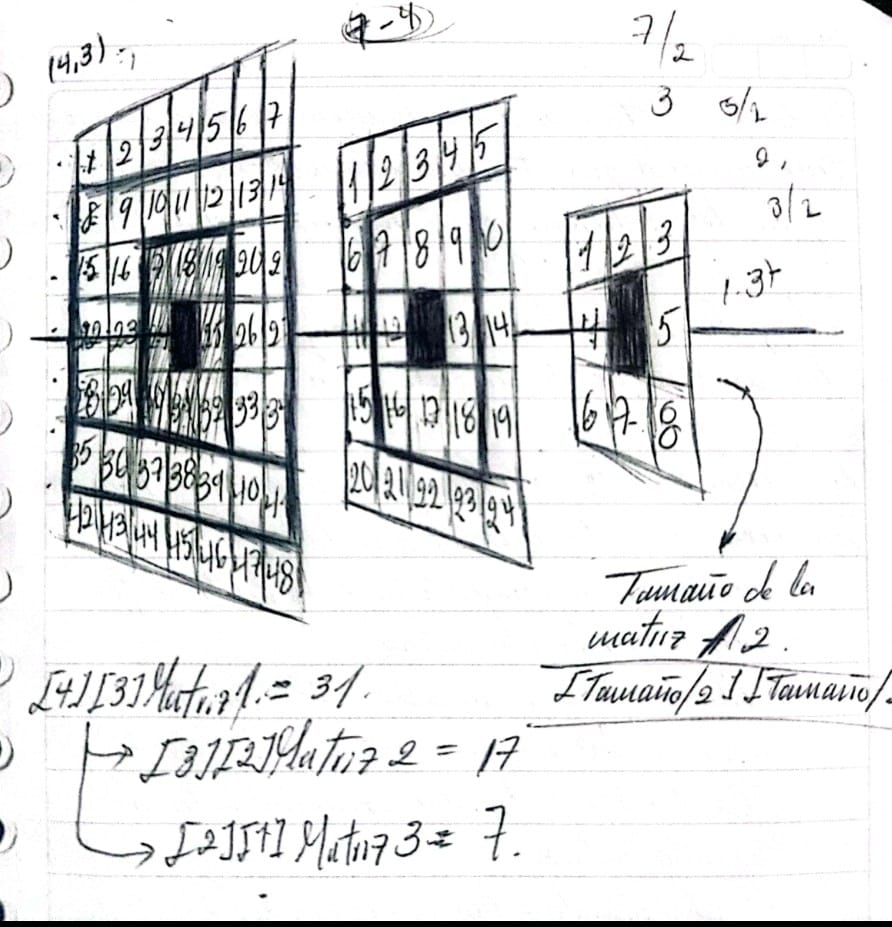
Puntos del informe

* + 1. Análisis del problema y consideraciones para la alternativa de solución propuesta.
    2. Esquema donde describa las tareas que usted definió en el desarrollo de los algoritmos.
    3. Algoritmos implementados.
    4. Problemas de desarrollo que afrontó.
    5. Evolución de la solución y consideraciones para tener en cuenta en la implementación.

A) Implementación del problema  
  
Para este desafío planteado, primero, debemos tener una cerradura con varias estructuras, en la cual, si nos dan una “llave”, tenemos que tratar de buscar “el candado” que abra con esa “llave”.

Para saber cuántas estructuras o matrices debe tener el “candado”, tenemos que ver cuantas condiciones nos dan en la llave, es decir, por ejemplo, en la llave que nos dan:  
K(4,3,1,-1,1): Como nos dicen, las 2 primeras revelan la primera matriz, y las otras 3, nos dicen las condiciones que deben de tener para que este “candado” se abra.  
Por lo cual, el candado va a tener: #condiciones + 1, es decir:  
numero de estructuras = número de condiciones + 1.  
  
Ahora bien, para llenar estas matrices menos el de la posición de la mitad, necesitaremos 2 bucles for, en la cual rellenen estas posiciones empezando desde la 1 hasta terminar, menos, la posición de la mitad, la cual es:  
posición central = [tamaño de la matriz / 2][tamaño de la matriz / 2].   
  
Por ahora, no sabemos cuales matrices son las que utilizaremos, es decir, podrían ser de forma ordenada y ascendente o descendente, o podrían totalmente al azar, pero esto sería una pésima práctica.  
  
Ahora bien, tenemos otro problema, se nos dijo que las matrices están alineadas a través del centro, entonces como saber, dependiendo de la primera posición de la matriz, que posición de la siguiente matriz le corresponde.  
  
Para eso, estuvimos pensando y analizando a través de graficas:





primero, debíamos asegurarnos de verificar los límites de la segunda matriz para determinar si una posición dada en la primera matriz tiene una posición correspondiente válida en la segunda matriz.  
  
para esto implementamos una función de tipo buleano, el cual nos permitirá ver si los limites son correctos:  
  
bool esCompatible(int fila, int columna, int tamano\_primera, int tamano\_segunda) {

// Verificar que la posición dada esté dentro de los límites de la segunda matriz

if (fila >= 0 && columna >= 0 && fila < tamano\_segunda && columna < tamano\_segunda) {

return true; // La posición dada en la primera matriz es compatible con la segunda matriz

}

return false; // La posición dada no es compatible con la segunda matriz

}

Ahora bien, falta lo mas importante, saber en qué posición de la siguiente matriz va a quedar dependiendo de la posición dada en la primera matriz:

Para esto, decidí dividirlo en dos.  
Por un lado esta cuando la primera matriz evaluada es mas pequeña que la segunda.  
  
para esto tenemos las siguientes formulas:  
  
fila\_segunda= fila\_primera + (tamaño\_segunda – tamaño\_primera)/2.  
  
columna\_segunda= columna\_primera + (tamaño\_segunda – tamaño\_primera)/2.  
  
  
Para esto, primero, calculamos cuántas filas deben moverse hacia abajo la matriz pequeña para estar centrada en la matriz grande. Esto se hace tomando la diferencia de tamaños entre las dos matrices y dividiéndola entre 2 para centrarla. A eso le sumamos la fila de la posición dada en la matriz pequeña.

Por ejemplo, si la matriz grande es de tamaño 9x9 y la matriz pequeña es de tamaño 5x5, la diferencia de tamaño es 9- 5 = 4. Dividimos esto entre 2 para obtener 2, que es la cantidad de filas que la matriz pequeña debe moverse hacia abajo para estar centrada en la matriz grande. Si la posición dada en la matriz pequeña es la fila 2, entonces la fila correspondiente en la matriz grande sería 2 + 2 = 4. Si las filas son enteras, redondearíamos hacia abajo para obtener la fila 4 en la matriz grande.  
  
  
De manera similar, calculamos cuántas columnas debe moverse hacia la derecha la matriz pequeña para estar centrada en la matriz grande. Esto también se hace tomando la diferencia de tamaños entre las dos matrices y dividiéndola entre 2. A eso le sumamos la columna de la posición dada en la matriz pequeña.

Siguiendo el ejemplo anterior, si la posición dada en la matriz pequeña es la columna 3, entonces la columna correspondiente en la matriz grande sería 3 + 2 = 5. Nuevamente, redondearíamos en caso de hacerlo, hacia abajo para obtener la columna 5 en la matriz grande.

En resumen, estas fórmulas calculan la posición en la matriz grande que corresponde a una posición dada en la matriz pequeña, manteniendo ambas matrices centradas por el centro una respecto a la otra.  
Esto, solo cuando tenemos primero una matriz pequeña y la siguiente una mas grande que esta

Ahora bien, necesitamos otras “formulas” para saber la posición respectiva en el caso contrario, es decir, cuando la primera matriz es más grande que la siguiente.

Para esto igualmente tenemos las siguientes formulas:  
  
fila\_segunda= (fila\_primera – (tamaño\_primera – tamaño\_segunda)/2)  
  
columna\_segunda= (columna\_primera – (tamaño\_primera – tamaño\_segunda)/2)  
  
  
Supongamos que tenemos una matriz grande y una matriz pequeña, ambas cuadradas, y queremos encontrar la posición en la matriz pequeña que corresponde a una posición dada en la matriz grande, usamos casi que el mismo razonamiento con las anteriores.

Primero, calculamos cuántas filas debe moverse hacia arriba la matriz grande para estar centrada en la matriz pequeña. Esto se hace tomando la diferencia de tamaños entre las dos matrices y dividiéndola entre 2. Luego, a eso le restamos la fila de la posición dada en la matriz grande.

Por ejemplo, si la matriz grande es de tamaño 9x9 y la matriz pequeña es de tamaño 5x5, la diferencia de tamaño es 9 - 5 = 4. Dividimos esto entre 2 para obtener 2, que es la cantidad de filas que la matriz grande debe moverse hacia arriba para estar centrada en la matriz pequeña. Si la posición dada en la matriz grande es la fila 8, entonces la fila correspondiente en la matriz pequeña sería 2 - (8 - 5) = 2 - 3 = -1. En este caso, al ser una posición negativa, significa que la posición correspondiente en la matriz pequeña no existe en términos enteros, lo que indica que la posición dada en la matriz grande no está alineada con la matriz pequeña en este contexto.

De manera similar, calculamos cuántas columnas debe moverse hacia la izquierda la matriz grande para estar centrada en la matriz pequeña. Esto también se hace tomando la diferencia de tamaños entre las dos matrices y dividiéndola entre 2. Luego, a eso le restamos la columna de la posición dada en la matriz grande.

Siguiendo el ejemplo anterior, si la posición dada en la matriz grande es la columna 9, entonces la columna correspondiente en la matriz pequeña sería 2 - (9 - 5) = 2 - 4 = -2. Al igual que en el caso de las filas, una posición negativa indica que la posición correspondiente en la matriz pequeña no existe en términos enteros.

Ahora bien, pensando en esto, nos podemos dar cuenta de que, con esto, no es necesario esta función:  
  
bool esCompatible(int fila, int columna, int tamano\_primera, int tamano\_segunda) {

// Verificar que la posición dada esté dentro de los límites de la segunda matriz

if (fila >= 0 && columna >= 0 && fila < tamano\_segunda && columna < tamano\_segunda) {

return true; // La posición dada en la primera matriz es compatible con la segunda matriz

}

return false; // La posición dada no es compatible con la segunda matriz

}

Ya que solo con ponerle un condicional de que si la fila\_siguiente o columna\_siguiente es menor a 0   
((fila siguiente<0)||(columna\_siguiente<0)), esto quiere decir que se sale de los limites.  
  
Luego de esto, tenemos que hacer creo que lo mas importante, que seria rotar las diferentes estructuras o matrices en los 4 estados, el neutro, el estado 1, el 2 y el 3.   
  
para el neutro no tenemos que hacer nada, ya que es la misma matriz, ahora bien, para el estado neutro, analizamos la matriz y llegamos a la conclusión para poder rotarla 90 grados en sentido anti horario.

Ejm: tenemos una matriz cuadrada de tamaño 3x3.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 |  | 5 |
| 6 | 7 | 8 |

Ahora, le sacamos la transpuesta, esto, en términos de código:

Esta sería la original:  
\*(\*(puntero\_matriz+i)+j)  
  
y esta la transpuesta:  
\*(\*(puntero\_matriz+j)+i)  
  
es decir, se cambian las columnas por las filas y las filas por las columnas, quedando la matriz de la siguiente manera:

|  |  |  |
| --- | --- | --- |
| 1 | 4 | 6 |
| 2 |  | 7 |
| 3 | 5 | 8 |

Ahora bien, notemos que queremos llegar a esto:

|  |  |  |
| --- | --- | --- |
| 3 | 5 | 8 |
| 2 |  | 7 |
| 1 | 4 | 6 |

Como notamos, simplemente lo que nos falta, es cambiar la posición de las filas, la primera debería ser la última, la segunda la penúltima y así sucesivamente.   
En términos de código, la fila 0, debería ser la (n-1), y la (n-1) debería ser la fila 0, etc.  
  
Por lo que la matriz nos queda en el estado 1:

|  |  |  |
| --- | --- | --- |
| 3 | 5 | 8 |
| 2 |  | 7 |
| 1 | 4 | 6 |

Ahora, para llegar al estado2 y 3, donde hay que rotarla 180 grados y 270 grados, notamos que es hacer el mismo procedimiento, ya que se están rotando 90 grados siempre en sentido anti horario, por lo que para el estado 2 simplemente es hacer 2 veces el procedimiento del estado 1 con la matriz neutra. Lo mismo con el estado 3, simplemente, repetir 3 veces el procedimiento del estado 1 con la matriz neutra.  
  
   
b) Esquema donde describa las tareas que usted definió en el desarrollo de los algoritmos.

1. Cuatro funciones para la matriz (neutra, 1, 2, 3,)
2. Una función que busque los valores dentro de las matrices o sea un valor en específico
3. Una función que reciba los valores (A, B, C Y D) y los compare para ver si pueden abrir el candado
4. Una función que almacene los posibles valores de x
5. Una función que verifique que los valores de k pueden abrir el candado o sea x