# Análisis, Desarrollo y Evolución de Algoritmos en Informática 2.

## a. Análisis del problema y consideraciones para la alternativa de solución propuesta.

Para abordar el problema, dedicamos tiempo a una lectura exhaustiva del archivo desafío 1. Nuestro objetivo fue comprender a fondo el problema antes de proceder con la elaboración de cualquier diagrama de flujo o estrategia de solución. Enfocamos nuestra atención en comprender las complejidades del problema y evaluar las posibles alternativas de solución de manera integral.

Una de las principales estrategias que adoptamos fue dividir todo el programa en funciones. Esto nos permitió realizar un trabajo colaborativo más efectivo y organizado. Además, nos enfocamos en comprender cómo utilizar Git y GitHub de manera adecuada antes de comenzar con la implementación de las funciones.

Hasta el momento, hemos creado una función para cada tarea específica. Por ejemplo, una función para crear la matriz, otra para rotar la matriz y otra para determinar la posición que se está buscando en la matriz rotada, basándonos en su fila y columna. Estamos avanzando a un ritmo sólido y hemos logrado comprender muy bien el problema en su totalidad.

## b. Esquema donde describa las tareas que usted definió en el desarrollo de los algoritmos.

Las tares que definimos en cada algoritmo radican de lo que se vio en clase acerca de arreglos, memoria dinámica, punteros y demás temas que son indispensables para el desarrollo de esta práctica, además complementándolo muy bien con lo que sea ha estudiado por nuestra propia cuenta

1. Definir la función saber\_dimension\_matriz():

* Me recibe como parámetro un puntero a entero el cual va a ser la clave ingresada por el usuario
* Dependiendo de los elementos en las primeras dos posiciones del arreglo de la clave, se determina el mayor
* Si el valor de fila es mayor que el de columna, la dimensión de la matriz será igual al valor de fila si fila es impar. Si fila es par, la dimensión de la matriz será fila + 1
* Si el valor de columna es mayor que el de fila, la dimensión de la matriz será igual al valor de columna si columna es impar. Si columna es par, la dimensión de la matriz será columna + 1.
* La función nos retorna un entero que indica el tamaño de la matriz cuadrada

1. Definir la función matrices():

* Cada una recibe dos enteros como parámetro, para determinar el tamaño de la matriz y cual será su rotación.
* Implementan algoritmos para rotar la matriz en 90, 180 y 270 grados en sentido antihorario respectivamente
* Retorna un doble puntero

1. Definir función buscar\_posicion ():

* Esta función me recibe como parámetro un puntero que tiene la dirección de memoria del primer valor de la clave la cual esta en un arreglo dinámico y el otro parámetro es un entero el cual nos determina si la siguiente matriz es mayor, igual o de menor tamaño.
* Haciendo las debidas operaciones para que también apunte a la segunda dirección de memoria la cual tiene almacenada el valor de la columna
* Y después se pone en marcha un algoritmo el cual nos determina cual es el valor de la fila y columna de la siguiente matriz
* Por último, nos retorna un puntero al arreglo dinámico (cabe recalcar que se debe liberar la memoria una vez utilizada)

1. Definir comparar\_matrices0, comparar\_matrices\_menos y comparar\_matrices1 ():

* Recibe como parámetros 2 punteros dobles correspondientes a las 2 matrices a comparar, y 2 punteros que guardan las posiciones que se quiere comparar en ambas matrices
* Se obtienen las coordenadas de la posición a comparar de cada matriz a través de los punteros clave y clave2, donde clave[0] y clave[1] representan la fila y columna de la primera matriz (ptr1), y clave2[0] y clave2[1] representan la fila y columna de la segunda matriz (ptr2).
* Compara los elementos en las posiciones especificadas de ambas matrices. Si las condiciones se cumplen, la función devuelve true, de lo contrario, devuelve false

1. imprimir\_matriz():

* Utiliza dos bucles for para recorrer cada fila y columna de la matriz.
* Para cada elemento en la matriz, utiliza matriz[i][j] para acceder al valor en la fila i y columna j.
* Imprime el valor de cada elemento de la matriz seguido de un tabulador (\t), lo que crea un formato de salida tabular.
* Después de imprimir todos los elementos de una fila, imprime un salto de línea (\n) para pasar a la siguiente fila.

1. Definir liberar\_memoria():

* Utiliza un bucle for para recorrer cada fila de la matriz.
* Dentro del bucle, utiliza delete ptr[i] para liberar la memoria asignada para cada fila de la matriz.
* Después de liberar la memoria de todas las filas, utiliza delete[] ptr para liberar la memoria del puntero que apunta a las filas de la matriz.

1. Definir es\_en\_anillo\_exterior ():

* Recibe como parámetros un entero que es el tamaño de la matriz y un puntero a entero de un arreglo con la posición a validar.
* Si la posición corresponde a los extremos de la matriz (fila 1, fila n, columna 1 o columna n) retorna true, de lo contrario, retorna false.

1. Definir candado, candado\_menor, candado\_igual ():

* Recibe como parámetros un entero que indica la dimensión de la matriz original, 2 punteros con las posiciones a comparar en ambas matrices y una referencia a entero.
* Inicia con un for con i = 0 hasta i<4, dado que 3 es la última rotación.
* Se crea las matrices de acuerdo con la dimensión original (siempre se empieza a comparar la matriz original con si misma).
* Se compara las matrices de acuerdo con el parámetro en la clave
* Se imprimen las matrices.
* Se realiza la verificación, depende de ésta si se rompe el ciclo o continua.
* Dado el caso de que no cumpla, se aumenta el tamaño de la dimensión en 2 para mantenerla impar.
* Si cumple, el entero referenciado toma el valor de i y se rompe el ciclo.
* Dado el caso de que la comparación sea menor, se realiza otra verificación con la función es\_en\_anillo\_exterior(). Dado el caso de que sea verdadero y se hayan realizado todas las rotaciones, se imprime un mensaje sobre la imposibilidad de crear una matriz que cumpla la condición y retorna un puntero nulo.
* La función retorna un puntero a un arreglo que contiene la dimensión, rotación de la nueva matriz y la posición del elemento que se comparó.

c. Algoritmos implementados.

|  |  |
| --- | --- |
| **FUNCIÓN** | **DESCRIPCIÓN** |
| saber\_dimension\_matriz | La función determina la dimensión de la matriz para asegurar que sea lo suficientemente grande para contener tanto la fila como la columna proporcionada, considerando que, si alguno de los valores es par, se incrementa en 1 para asegurar una dimensión impar y tener un punto central en la matriz. |
| buscar\_posicion | Recibe como parámetros un puntero clave que apunta a un arreglo de enteros con la posición de un elemento en una matriz (fila y columna), y un entero siguiente\_matriz que indica si la siguiente matriz es mayor, igual o de menor tamaño. Luego, la función calcula y devuelve un puntero que apunta a un arreglo de enteros con la posición del elemento en la siguiente matriz. |
| imprimir\_matriz | Recorre la matriz y la imprime en la consola de manera ordenada, separando cada elemento por tabuladores y cada fila por saltos de línea.  Es importante tener en cuenta que la matriz se trata como una matriz constante en esta función, lo que significa que no se pueden modificar los valores de la matriz dentro de la función imprimir\_matriz. Esto es útil cuando se quiere asegurar que la función de impresión no altere la matriz original. |
| comparar\_matrices0 | Verifica si los elementos en las posiciones especificadas de dos matrices son iguales. Si lo son, devuelve true; de lo contrario, devuelve false. |
| comparar\_matrices1 | Verifica si el elemento de la siguiente matriz en la posición específica es mayor que el elemento en la matriz anterior. Si es así, devuelve true; de lo contrario, devuelve false. |
| comparar\_matrices\_menos | Verifica si el elemento de la siguiente matriz en la posición específica es menor que el elemento en la matriz anterior. Si es así, devuelve true; de lo contrario, devuelve false. |
| liberar\_memoria | Se encarga de liberar toda la memoria asignada dinámicamente para una matriz creada mediante asignación de memoria dinámica. |
| matrices | Esta función crea la matriz recibiendo un entero n que es la dimensión de la matriz y otro que indica la rotación. Genera un arreglo dinámico con los números del 1 al n\*n-1 y lo rellena de izquierda a derecha desde la fila 1 a la fila n (si la rotación es 0), de abajo hacia arriba desde la columna 1 hasta la columna n (si la rotación es 1), de derecha a izquierda desde la fila n hasta la fila 1 (si la rotación es 2) o de arriba abajo desde la columna n hasta la columna 1 (si la rotación es 3), dejando siempre en el centro de la matriz el 0 (elemento neutro). Retorna un doble puntero. |
| candado | Esta función se encarga de crear un arreglo correspondiente al candado para la clave ingresada cuando la condición es 1. Integra las funciones desarrolladas en el programa. Retorna un puntero con la nueva dimensión, rotación, y la posición (fila, columna). |
| candado\_menor | Esta función se encarga de crear un arreglo correspondiente al candado para la clave ingresada cuando la condición es -1. Integra las funciones desarrolladas en el programa. Retorna un puntero con la nueva dimensión, rotación, y la posición (fila, columna). Utiliza la función es\_en\_anillo\_menor para validar, dado el caso de que sea verdadero y ya se realizaron todas las rotaciones, imprime un mensaje manifestando la imposibilidad de crear un candado para dicha condición y retorna un puntero nulo. |
| candado\_igual | Esta función se encarga de crear un arreglo correspondiente al candado para la clave ingresada cuando la condición es 0. Integra las funciones desarrolladas en el programa. Retorna un puntero con la igual dimensión, rotación, y posición (fila, columna) de la matriz anterior. |
| es\_en\_anillo\_exterior | Verifica si la posición se encuentra en los bordes de la matriz |

## d. Problemas de desarrollo que afrontó.

Durante el desarrollo de nuestro proyecto, nos hemos encontrado con varios desafíos significativos. Uno de los principales ha sido la formulación de la lógica para crear matrices rotadas, Además, hemos experimentado dificultades al utilizar algunas funciones específicas de Git y GitHub. Si bien estas herramientas son fundamentales para el trabajo colaborativo y el control de versiones, su aprendizaje y dominio requieren práctica y comprensión profunda.

A pesar de estos desafíos, consideramos que esta experiencia ha sido muy chévere. La aplicación práctica de los conceptos aprendidos en clase, combinada con la resolución de problemas reales, contribuye significativamente a nuestra formación como futuros ingenieros, de igual manera se estará actualizando el documento de acuerdo con los avances y dificultades que se tengan.

Encontramos problemas relacionados con las posibles soluciones cuando el elemento a comparar se encuentra en el anillo exterior de la matriz, dado que, si se desea buscar una matriz con un valor menor a éste, en dicha posición, si la matriz de la misma dimensión no cumple, ninguna otra lo hará, por tanto no es posible construir un candado para esa clave.

El tema relacionado con la gestión de memoria fue también un inconveniente, pero supimos afrontarlo de manera rápida y sencilla, sin complicaciones, dado que nuestro código siempre estuvo lo más organizado posible y evitamos cualquier enredo que nos ocasionara pérdida de memoria.

## e. Evolución de la solución y consideraciones para tener en cuenta en la implementación.

Nuestro progreso en la resolución del problema ha sido bastante satisfactorio hasta el momento. Tanto mi compañero como yo tenemos una comprensión clara de los conceptos involucrados, lo que nos ha permitido avanzar de manera eficiente en el desarrollo de las funcionalidades requeridas.

En el momento de este commit, la mayoría de las funcionalidades ya están implementadas, lo que indica que estamos progresando a un ritmo sólido.

Además, se estará actualizado el informe con las funcionalidades que se agregaran a lo largo de la evolución del proyecto.

En cuanto a las consideraciones de implementación, es crucial tener un manejo adecuado de las validaciones en nuestro código. Esto implica verificar y asegurar que los datos de entrada sean correctos y estén dentro de los rangos esperados, evitando así posibles errores o comportamientos inesperados.

Asimismo, es importante familiarizarnos y utilizar de manera eficiente las funcionalidades que hemos desarrollado a lo largo de esta semana. Esto incluye aprovechar al máximo las capacidades de las funciones creadas, asegurándonos de su correcto funcionamiento y optimización, nuestro objetivo principal es completar y entregar el desafío antes del tiempo propuesto.

Tuvimos que realizar cambios en el código, existieron ciertos errores que parchamos de manera correcta y el programa ya funciona correctamente. Integramos las funciones de rotación y creación de matriz en una sola así como creamos 2 funciones más para las comparaciones, de tal forma que el código sea de lo más entendible.

Creamos las funciones candado, que implementan el resto de funciones y retornan finalmente la solución al desafío, el candado con las dimensiones de las matrices y sus respectivas rotaciones.