EIF207 – Estructuras de Datos
Proyecto de programación #1

Prof. M.Sc. Georges E. Alfaro S.

PLANTEAMIENTO DEL PROBLEMA

Una matriz es una tabla de valores, generalmente números. Cada elemento en una matriz se puede localizar por dos enteros que representan su posición horizontal (fila) y vertical (columna). Las matrices cuyos elementos son en su mayoría ceros (o cualquier otro valor arbitrario) se denominan **dispersas** (o **ralas**), de lo contrario, se designan como **densas**. La mayoría de las matrices grandes son dispersas: por ejemplo, las que aparecen en el procesamiento del lenguaje natural (NLP) o en los sistemas de recomendación, porque representan información sobre la interacción entre entidades que rara vez están en contacto entre sí. En la mayoría de las aplicaciones, las matrices no sólo se utilizan para contener información, sino que también se procesan para extraer resultados significativos.

Tipos de matrices dispersas

Hay varias formas diferentes de codificar una matriz dispersa:

Lista de coordenadas (COO): es una lista de tuplas (fila, columna, valor). Probablemente el enfoque más intuitivo, también conocido como formato triplete.

Diccionario de claves (DOK): es un mapa (diccionario) de valores distintos de cero indexados por pares (fila, columna). Internamente, la implementación es básicamente una tabla *hash*, que tiene una complejidad de tiempo $O(n)$ cuando se busca un elemento, aunque puede consumir mucha memoria;

Lista enlazada (LIL): es una matriz irregular de filas, donde cada **fila** es una lista de tuplas **(columna, valor)**. En muchas implementaciones, hay dos listas (índice de columna y valor) que contienen los datos indexados por número de fila. **Para este proyecto, esta es la implementación que se utilizará. Podrán usar una lista simple o doblemente enlazada.**

Fila dispersa comprimida (CSR): la matriz se "aplana por fila", es decir, está representada por tres matrices unidimensionales generalmente llamadas A (o valores), JA (o índices) e IA (o puntero de índice). La primera matriz almacena los valores de elementos distintos de cero y tiene una longitud NNZ. JA almacena el índice de columna de cada elemento en la matriz A y tiene una longitud NNZ. Finalmente, IA almacena el número acumulado de elementos distintos de cero hasta la i-ésima fila. El truco consiste en agregar un elemento inicial de "relleno" IA [0] = 0 y definir todos los demás elementos de forma recursiva de manera que AI [i] = IA [i-1] + número de elementos distintos de cero en el (i-1) -ésima fila de A. Este es el enfoque preferido cuando la matriz se utiliza para la multiplicación. También se conoce como **formato de Yale**.

Columna dispersa comprimida (CSC): igual que CSR pero con índices de columna y fila intercambiados. Este es el enfoque preferido cuando la matriz se utiliza para cortar (*splice*) columnas.

Los primeros tres métodos (COO, DOK, LIL) son útiles si el propósito del programa es crear una matriz y actualizar los elementos con mucha frecuencia, por ejemplo, agregando filas o columnas de forma incremental. En cambio, si el propósito es realizar cálculos como multiplicaciones, se prefieren los dos últimos (CSR, CSC), para lo cual administrar la inserción de nuevos datos es más desafiante ya que requiere muchas operaciones. A menudo se usa una combinación de dos métodos para construir la matriz primero y luego hacer alguna operación sobre ella. Por ejemplo, se puede crear una matriz COO y luego dividirla en columnas con un enfoque CSC.

OBJETIVOS DEL PROYECTO

El objetivo del proyecto es estudiar cómo se puedan utilizar estructuras de datos lineales básicas, ya implementadas, para construir una estructura de datos compleja (no lineal).

DESCRIPCIÓN DEL PROYECTO

Descripción general.

Para este proyecto, se utilizará únicamente la implementación por medio de **listas enlazadas**.

Deberá construirse una biblioteca con una clase genérica que implemente la estructura y las operaciones básicas de **suma**, **transposición** y **multiplicación**. Aunque la clase será genérica, la mayoría de las pruebas se completarán usando valores numéricos (enteros).

El valor de las entradas "vacías" puede ser un valor arbitrario (no necesariamente 0), que se especificará en el constructor de la clase. Deberá definirse un constructor que asuma el valor 0 por defecto (en el caso de valores numéricos) o un valor apropiado según el tipo de elemento base de la matriz.

```
SparseMatrix(int m, int n, T v);
SparseMatrix(int m, int n);

SparseMatrix<T> add(SparseMatrix<T> m);
SparseMatrix<T> transpose();
SparseMatrix<T> multiply(SparseMatrix<T> m);
```

Implemente también los métodos básicos para agregar y acceder a elementos individuales, así como métodos de conversión y comparación:

```
int getRowCount();
int getColumnCount();
void set(int m, int n, T v);
T get(int m, int n);
String toString();
boolean equals(SparseMatrix<T> other);
```

Incluya además un método:

```
SparseMatrix<T> splice(int m0, int m1, int n0, int n1);
```

Para poder extraer de la matriz otra matriz, formada solamente por los elementos que se encuentran dentro de las filas y columnas especificadas:

$$0 \leq m1_1 \leq m_2 < m$$

$$0 \leq n_1 \leq n_2 < n$$

$$A = \begin{vmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \cdots & \cdots & & \cdots \\ a_{m,0} & \cdots & & a_{m-1,n-1} \end{vmatrix}$$

$$splice(A, i, j, p, q) = \begin{vmatrix} a_{i,p} & \cdots & a_{i,q} \\ \cdots & \cdots & \cdots \\ a_{j,p} & \cdots & a_{j,q} \end{vmatrix}$$

CONSIDERACIONES DE IMPLEMENTACIÓN

Las operaciones de suma y multiplicación requieren que las matrices tengan un tamaño que permita la operación (las mismas dimensiones, para la suma; o una cantidad de filas y columnas adecuada para la multiplicación). En este caso, se permitirá que las matrices sean de cualquier tamaño, extendiéndolo como sea necesario para poder efectuar la operación.

Por ejemplo: si se tienen tres matrices:

$$A_{3\times4}, B_{2\times5}, C_{5\times2}$$

Al completar las operaciones indicadas, se construyen matrices de mayor tamaño, de la siguiente manera:

$$A_{3\times4} + B_{2\times5} = A'_{3\times5} + B'_{3\times5}$$

donde las matrices: $A_{3\times4} \rightarrow A'_{3\times5}$, y $B_{2\times5} \rightarrow B'_{3\times5}$, se construyen “agregando” las filas o columnas necesarias. Las filas o columnas que se agreguen simplemente cambian la dimensión de la matriz en el momento de la operación, pero no alteran la estructura de datos internamente.

$$C_{5\times2} \times A_{3\times4} = C'_{5\times3} \times A'_{3\times4}$$

Escriba el programa utilizando el lenguaje de programación **Java** o **C++**. Los programas serán probados de preferencia en la plataforma Linux, pero éstos deberán poderse ejecutar correctamente también en Windows o Mac OS X.

Incluya en el programa las opciones necesarias para poder guardar y recuperar una instancia de la clase (`SparseMatrix`) desde un archivo XML.

Diseñe la arquitectura del programa utilizando UML e incluya los diagramas de clase correspondientes. Si es necesario, incluya otros diagramas, tales como los de secuencia o estado.

ENTREGA Y EVALUACIÓN

El proyecto debe entregarse **por medio del aula virtual, en el espacio asignado para ello**. La entrega es al finalizar la semana 11 (**sábado 24 de octubre de 2021**). No se aceptará ningún proyecto después de esa fecha, ni se admitirá la entrega del proyecto por correo electrónico. El proyecto se puede realizar en grupos de **tres personas, como máximo**. Deberán enviar un correo al profesor indicado la lista de participantes del grupo antes del viernes de la semana 8 (1ero de octubre de 2021).

Incluya comentarios en el código de los programas y describa detalladamente cada una de las clases y métodos utilizados.

Incluya un comentario al inicio de cada archivo fuente, indicando información básica, como se muestra abajo:

```
/**
 *
 * (c) 2021
 * @author Adriana González, Carlos Montero
 * @version 1.0.0 2021-10-24
 *
 * -----------------------------------------------
 * EIF207 Estructuras de Datos
 * 2do ciclo 2021, grupo 01
 * Proyecto 1
 *
 * 12345678 González Abarca, Adriana
 * 87654321 Montero Rodríguez, Carlos
 * -----------------------------------------------
 *
 *
 */
```

En caso de que la aplicación no funcione adecuadamente, efectúe un análisis de los resultados obtenidos, indicando las razones por las cuales el programa no trabaja correctamente, y cuáles son las posibles correcciones que se podrían hacer. Durante la revisión del proyecto, es muy importante poder defender adecuadamente la solución propuesta.

El proyecto se evaluará de acuerdo con la siguiente ponderación:

Diseño de la solución y documentación		**10%**
	Documentación y análisis de resultados:	10%
Funcionalidad		**90%**
	Manejo de documentos (cargar y guardar instancias de la clase):	15%
	Implementación de las clases (representación):	25%
	Implementación de las operaciones (eficiencia y correctitud):	35%
	Implementación de procedimientos de prueba:	15%

Observaciones generales:

- Los proyectos deben entregarse con toda la documentación, diagramas, código fuente y cualquier otro material solicitado.
- Se debe indicar en cada documento el nombre completo y cédula de cada participante del grupo, indicando el nombre del curso, ciclo lectivo y descripción del trabajo que se entrega. Esto incluye comentarios en cada archivo fuente entregado.
- Si los materiales de entrega no están completos, se penalizará hasta un 15% de la nota correspondiente. Asimismo, cualquier trabajo práctico que no sea de elaboración original de los estudiantes (plagio) se calificará con nota 0 (cero) y se procederá como lo indiquen los reglamentos vigentes de la universidad.
- Los trabajos que se reciban después de la fecha señalada para su entrega, **en caso de ser aceptados, serán penalizados con un 30% de la nota por cada día de atraso.**

REFERENCIAS

Data Structures. (23 de septiembre de 2021). Obtenido de BTech Smart Class: http://www.btechsmartclass.com/data_structures/sparse-matrix.html

Di Sipio, R. (25 de mayo de 2021). *A Quick Guide to Operations on Sparse Matrices*. Obtenido de https://medium.com/codex/a-quick-guide-to-operations-on-sparse-matrices-2f8776fab265

Operations on Sparse Matrices. (6 de enero de 2020). Obtenido de Geeks for Geeks: https://www.geeksforgeeks.org/operations-sparse-matrices/

Sparse Matrix. (17 de septiembre de 2021). Obtenido de Wikipedia: https://en.wikipedia.org/wiki/Sparse_matrix