Proyecto Final

Fundamentos y Estructuras de Programación

Departamento de Ingeniería de Sistemas Pontificia Universidad Javeriana

Profesores: Carlos Alberto Ramírez Restrepo

Gerardo Sarria

ALEJANDRO MEZA

JUAN CAMILO PEÑA

GERMÁN ANDRÉS CAYCEDO M.

Santiago de Cali, junio 7 de 2017

DETALLES DE LA TRES IMPLEMENTACIONES

1. El formato de vector coordenado que nosotros implementamos para nuestra primera implementación en términos generales consta de una estructura compuesta por tres campos de datos tipo vector donde en la primera se guarda en orden el valor del dato extraído de la matriz, en segundo lugar en el segundo campo se guarda en el mismo orden la posición de la fila de donde se extrajo el dato guardado en el anterior campo y por último en el tercero se guarda la posición de la columna de donde se extrajo el dato inicial. Los tres vectores tienen tamaño *n.*

2. El formato de vector comprimido que nosotros implementamos fue el de comprimido por filas y este muy parecido al anterior consta de una estructura compuesta por tres campos de datos tipo vector donde en el primero se guarda el valor del dato extraído de la matriz en orden, en el segundo campo de guarda la posición de la columna de donde se extrajo el dato y por último en el tercer campo se guarda la posición de donde empiezan los valores de cada fila en el vector anterior de columnas. Los dos primeros vectores tienen tamaño *n* y el ultimo tiene tamaño *n+1*.

3. El formato de listas enlazadas por fila que fue el que nosotros implementamos consta de una estructura compuesta por un vector el cual guardara las listas y otra estructura de listas enlazadas las cuales es donde se guarda el dato y representara las filas y el vector es el que representa las columnas. En los nodos de las listas se almacenan dos datos, el primero es el dato a guardar que se extrae de la matriz y en el segundo se guarda la posición de la columna de donde estaba el dato; la posición de la fila de donde se extrajo el dato es representada por la posición del vector en la que está la lista alojada. El tamaño del vector y el de la lista es *n.*

ANÁLISIS DE COMPLEJIDAD

1. La complejidad de Crear Matriz Completa es de $O(n^2)$ ya que en ella se encuentran dos ciclos de *for* anidados para leer la leer la matriz dispersa y así crear la nueva matriz completa entonces las veces que se va a ejecutar la función en total con los dos ciclos va a ser n^2. Para los tres formatos la complejidad es la misma, cuando cambia es muy poco.

2. La complejidad de Obtener Matriz Completa es de $O(n^2)$ muy parecida a la anterior ya que esta también cuenta con dos ciclos de *for* anidados donde cada uno se va a ejecutar *n* y *n+1* veces respectivamente por lo cual al hacer el cálculo nos da en términos generales que el número de veces que se ejecutará esta función será n^2. Para los tres formatos la complejidad es la misma, cuando cambia es muy poco y no afecta la complejidad general.

3. La complejidad de Obtener elemento para Vector coordenado y para Vector comprimido es muy similar y es de O(n), es lineal porque al ser vectores solo cuentan con un ciclo el cual va recorrerlos n veces hasta que encuentren la posición que buscaban y se extraiga el elemento buscado. Pero para Lista Enlazadas es un poco diferente ya que primero se tiene que recorrer el vector donde se guardan las listas y después se recorre la lista buscando el elemento deseado para lo cual se requiere un ciclo el cual recorre la lista arrojada por la fila buscada lo que nos da que la complejidad de esta función para este TAD es de O(n).

4. La complejidad de Obtener fila para Vector coordenado y para Vector comprimido es de O(n). El lineal ya que al ser vectores al recorrerlos en búsqueda del valor el máximo de veces que se va a hacer será n veces y así extraer el elemento. Para las Lista enlazadas es un poco diferente ya que se llega al vector en la posición deseada de fila y después se recorre la lista retornando los valores en las columnas que se encuentran en la fila deseada, lo cual nos da que la complejidad de esta operación también en términos generales en lineal O(n) aunque se necesitan un poco de pasos más.

5. La complejidad de Obtener columna para Vector coordenado y para Vector comprimido en lineal O(n) ya que el ciclo que se encargara de buscar los valores asociados a la columna buscada solo se desarrollara máximo n veces hasta encontrar todos los valores buscados. Para listas enlazadas es diferente ya que ya se necesita otro ciclo, para que uno recorra el vector retornando las listas y otro para que busque dentro de las listas la columna deseada y retorne el valor asociado a esa columna que se busca, al tener dos ciclos anidados ya la complejidad cambia y es ahora O(n^2) para esta función en listas enlazadas.

6. La complejidad para Obtener Fila Dispersa para Vector coordenado y para vector comprimido es lineal O(n) muy parecido al de obtener fila con el breve cambio de que al haber ceros cambia el número de datos y ahora va a ser $2n$ lo que sigue siendo de igual manera lineal. Para listas enlazadas es lo mismo no cambia y va a continuar siendo lineal con el máximo de veces de $2n$.

7. La complejidad para Obtener Columna dispersa para Vector Coordenado y para Vector Comprimido es muy similar a la de Obtener Columna y sigue siendo lineal, aunque ahora con los ceros son más los datos que toca recorrer y analizar en términos generales la complejidad sigue siendo O(n), así sea ya $2n$ el número de veces máximo que se va a ejecutar cada ciclo igual es lineal. Para listas enlazadas también es muy similar, aunque el número de datos a analizar ya es $2n$ también al ser solo los dos ciclos anidados en términos generales la complejidad sigue siendo O(n^2).

8. La complejidad para Obtener número de elementos para Vector coordenado y para Vector Comprimido siempre va a ser el máximo de números que hay en el arreglo que en el caso más completo va a ser n ya que toca recorrer todo el vector, pero al ser vector va a ser solo un ciclo lineal, por lo tanto, la complejidad va a ser O(n) para estos vectores. Para listas va ser un poco diferente ya que va a tocar recorrer el vector que guarda las listas y además cada una de las listas lo que nos lleva usar dos ciclos anidados para el conteo de todos los elementos que se encuentran en el por lo cual la complejidad para esta función en este TAD va a ser O(n^2).

9. La complejidad para Modificar Posición para para Vector coordenado y para Vector Comprimido va a ser lineal ya que lo único que se necesita es recorrer los vectores buscando el dato a modificar de posición y después añadirlo en la posición deseada por lo cual se necesitan dos ciclos, pero estos no están anidados, lo cual no eleva la complejidad significativamente, en términos generales va a ser en promedio *2n* el número de ejecuciones por lo cual nos lleva a que la complejidad va a ser O(*n*). Para listas va a ser un poco más eficiente ya que al ya saber la fila no nos toca recorrer todo el vector que guarda las listas, pero de igual forma si nos va a tocar recorrer toda la lista buscando el valor a modificar lo cual nos da una complejidad de O(*n*).

10. La complejidad de Suma de Matrices para Vectores Coordenados y Vectores Comprimidos va a ser O(*n*). Ya que lo que necesitamos es recorrer los vectores y sumar los valores que estos retornen por lo cual el máximo de veces que se va a desarrollar este ciclo va a ser n veces lo cual nos da una complejidad lineal. Para Listas enlazadas es diferentes ya que para recorrer todo el vector e ir recorriendo las listas y que estas nos retornen los valore que se encuentran dentro de ellas y así sumarlos se necesitan dos ciclos anidados lo que nos eleva el número de ejecuciones y nos da una complejidad de O(n^2).

11. La complejidad para Producto Matriz Vector para Vectores Coordenados y Vectores Comprimidos va a ser del estilo lineal ya que se necesita un ciclo que recorra los vectores asociados con la matriz y otro que recorra el vector a multiplicar, pero estos son ciclos separados que van a generar un número de ejecuciones del estilo *2n,* pero nunca dejaran de ser lineal por lo cual su complejidad será O(*n*). Para Listas enlazadas va a ser un poco diferente, aunque el proceso es el mismo, pero para recorrer todo este tipo de dato se necesitan dos ciclos anidados los cuales elevan la complejidad y nos va a dar que en términos generales la complejidad para esta función en este TAD va a ser O(n^2).

12. La complejidad para Matriz Transpuesta para Vectores Coordenados y Vectores Comprimidos va a ser lineal ya que es máximo de veces que se va a ejecutar será *n* al leer todos los valores y solo será necesario un ciclo lo cual nos arroja en términos generales una complejidad O(*n*). Pero para listas enlazadas va a ser diferente ya que al recorrer el vector y las listas dentro del vector ya no va a ser suficiente un solo ciclo si no que se necesitará otro ciclo anidado que va recorriendo las listas alojadas en el vector por lo cual ahora la complejidad de esta función para este TAD va a ser en términos generales O(n^2).

CONCLUSIONES Y ASPECTOS POR MEJORAR

1. La primera conclusión a la que llegamos fue ver la forma tan interesante donde un mismo dato se puede representar en varios tipos de datos dependiendo de la necesidad o de la limitación de recursos que se tenga sin perder información ni otro tipo de cosas si no modelándolo desde otro tipo de dato abstracto.

2. Otra conclusión a la que se llego fue al ver como un mismo dato al ser representado en otro TAD puede cambiar su agilidad frente a varias operaciones planteadas sabiendo que en los diferentes TAD estaban alojados los mismos datos, pero complejidad podía cambiar radicalmente así se viera desde la misma aplicación y todo cambiaba por el TAD en el que se modelaba.

3. La otra conclusión a la que llegamos y que nos dimos cuenta que fue muy útil es que dependiendo de las limitaciones que tengamos y de las aplicaciones que vayamos a usar se puede ver como un TAD es más eficiente que otro para una misma operación y como eso puede cambiar para la siguiente operación permitiéndonos esto ver qué tipo de TAD nos puede servir más para implementar y modelar un dato dependiendo de sus aplicaciones y permitiéndonos hacer unas implementaciones mas productivas, limpias y eficientes.

4. También vimos como así dos tipos de implementaciones parecieran muy similares y su complejidad también sea muy similar como el vector coordenado y el vector comprimido al usarse en grandes cantidades de datos y en algunas funciones se puede hacer la diferencia en su aplicación y toca ser muy cuidadoso en la selección de que estructura usar sabiendo que en un futuro puede hacer un cambio significativo.

5. Se pudo evidenciar a manera de conclusión también como mientras en algunas operaciones no se veía muy eficiente las listas enlazadas por filas, cuando se tiene la fila a buscar pasa a ser bastante eficiente, se puede decir que más que las otras implementaciones y puede ser de gran utilidad cuando se tienen ciertos datos como las filas.

6. Por mejorar nos dimos cuenta que al tener varias implementaciones y una cantidad razonable de operaciones para cada implementación debíamos haber asignado más tiempo al trabajo de ellas para que la calidad del trabajo fuera mejor y esto también aplica para el cálculo de la complejidad el cual también requería bastante tiempo para hacerse de la mejor forma.

7. Otro aspecto por mejorar es el de trabajar más unidos el grupo ya que al varias veces no coincidir los horarios nos afectaba el trabajar juntos en un mismo espacio y nos llevaba a delegar tareas por separado para cada miembro del grupo.

8. Para terminar, nos dimos cuenta que debíamos mejorar en la consulta a algún docente o profesor de la materia ya que a través del desarrollo del proyecto surgieron una cantidad razonable de dudas y varias veces no nos acercamos a solucionarlas, llevando a que se nos complicara el trabajo en el proyecto.