PROYECTO 2- IPC2- SECCIÓN D

201907636– Carlos Roberto Rangel Castillo

Resumen

Esta investigación tiene como objetivo la descripción de las matrices dispersas y listas ortogonales, siempre evidenciando la importancia de ambos, su funcionamiento y sus diferentes formas de representación, utilizando el lenguaje Python el cual es el requerido en este proyecto, este tipo de TDA se utiliza para la optimización a una gran escala los cuales son aplicables en la teoría de grafos, redes, combinatorias, métodos numéricos, etc. Este proyecto tiene como finalidad el aprendizaje de la implementación de listas ortogonales o la matriz dispersa, el manejo y uso de una interfaz gráfica usando la librería Tkinter y el uso de graphviz. El uso de esas herramientas ayudó a la realización de este proyecto de una manera en la cual se agilizo su resolución mediante los temas y herramientas ya mencionados anteriormente.

Palabras clave

1. TDA
2. Lista Ortogonal
3. Matriz Dispersa
4. Lista Doblemente Enlazada
5. Nodo

Abstract

This research aims to describe sparse matrices and orthogonal lists, always showing the importance of both, their operation and their different forms of representation, using the Python language which is required in this project, this type of TDA is used for optimization on a large scale which are applicable in the theory of graphs, networks, combinatorics, numerical methods, etc. The purpose of this project is to learn the implementation of orthogonal lists or the sparse matrix, the management and use of a graphical interface using the Tkinter library and the use of graphviz. The use of these tools helped to carry out this project in a way that expedited its resolution through the topics and tools already mentioned above.

Keywords

1. *TDA*
2. *Orthogonal list*
3. *Sparse Matrix*
4. *Doubly Linked List*
5. *Node*

Introducción

El uso de la matriz dispersa o ya sea una matriz creada con una lista ortogonal es para la optimización a gran escala, el análisis escala, el ahorro de memoria, la secuencia de búsqueda de datos, eliminación de los mismos e inclusive agregar datos. En una matriz dispersa se maneja de forma óptima agregar datos ya que en donde se quiere agregar un dato se crea un nuevo nodo y se ingresa el dato nuevo que se desea agregar ahí mismo. La lista ortogonal es perfecta para la secuencia de búsqueda de datos gracias a sus cuatro apuntadores los cuales permiten que se pueda recorrer toda la matriz de una forma rápida.

Desarrollo del tema

Las matrices dispersas son ampliamente usadas en la computación científica, especialmente en la optimización a gran escala, análisis estructural y de circuitos, dinámica de fluidos computacionales y en general en solución numérica de ecuaciones diferenciales parciales; otras áreas de interés en donde se pueden aplicar la representación dispersa son la teoría de grafos, teoría de redes, la combinatoria, los métodos numéricos, entre otros.

Dada su frecuente aparición existen múltiples paquetes de software que permiten su implementación y operación como son Sparspak, el Yale Sparse Matrix package, algunas rutinas de la librería de subrutinas de Harwell, además del paquete de software Matlab, entre otros.

Dado que este tipo de matrices ocurren tan naturalmente a través de los años se han desarrollado distintos métodos para representarlas en un computador, de manera que sean más eficientes en su computación y almacenamiento.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \end{bmatrix}$$

Figura 1. Matriz dispersa.

Otra definición que se le puede atribuir a lo que es una matriz dispersa es: Son matrices en las cuales existen gran cantidad de valores nulos o ceros. El aprovechamiento de este conocimiento permite reducir el costo computacional de las operaciones que se pueden realizar sobre estas matrices, así como el costo espacial para el almacenamiento de la información. Las matrices dispersas permiten el procesamiento de grandes volúmenes de información de conjuntos de datos no densos. En muchos casos el manejo de estos volúmenes de datos sin tener en cuenta la densidad de estos, se

vuelve impracticable, pues operaciones como la multiplicación de matrices, se vuelve un proceso altamente costoso desde el punto de vista computacional, tanto temporal como espacial. Muchas implementaciones sobre este tipo de matrices han sido llevadas a cabo con estructuras como listas y arreglos dinámicos, los cuales mejoran el procesamiento de matrices dispersas, pero no logran aprovechar todas las potencialidades de estas matrices, pues solo tienen en cuenta una sola dimensión dispersa. En este trabajo se propone una implementación de matrices dispersas en el lenguaje de programación Java empleando árboles balanceados para su implementación. Se comparan los resultados con otras implementaciones de matrices dispersas.

Listas Ortogonales

Una lista ortogonal es aquella en la que sus nodos se encuentran encadenados por cuatro ligas, es decir, cada nodo se encuentra doblemente ligado en forma horizontal, y cada nodo se encuentra doblemente ligado en forma vertical. Una lista ortogonal puede ser implementada como lineal o circular. Este tipo de listas se puede utilizar para representar matrices.

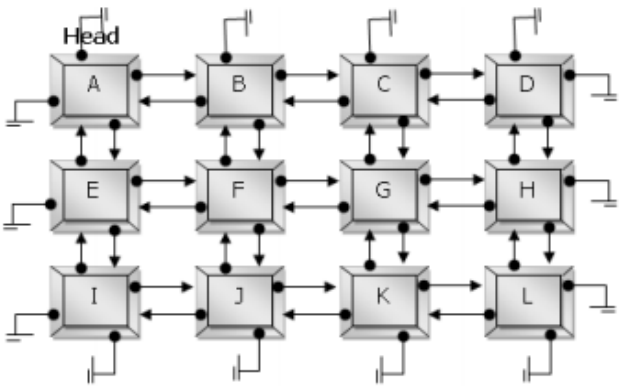

Figura 2. Lista ortogonal.

La característica principal de una lista ortogonal lineal es que las ligas de los últimos nodos apuntan hacia el valor nulo. El nodo de una lista ortogonal debe contener como mínimo cinco campos: uno para almacenar la información y cuatro para guardar la dirección de memoria hacia el siguiente, anterior, arriba y abajo de la lista. En la figura se puede apreciar la estructura del nodo para una lista ortogonal.

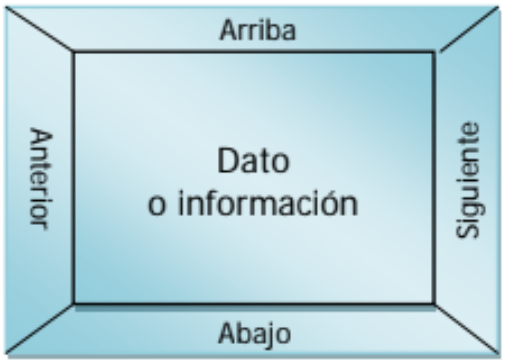

Figura 3. Nodo de una lista ortogonal.

Lista doblemente enlazada

En ciencias de la computación, una lista doblemente enlazada es una estructura de datos que consiste en

un conjunto de nodos enlazados secuencialmente. Cada nodo contiene tres campos, dos para los llamados enlaces, que son referencias al nodo siguiente y al anterior en la secuencia de nodos, y otro más para el almacenamiento de la información (en este caso un entero).

El enlace al nodo anterior del primer nodo y el enlace al nodo siguiente del último nodo, apuntan a un tipo de nodo que marca el final de la lista, normalmente un nodo centinela o puntero null, para facilitar el recorrido de la lista. Si existe un único nodo centinela, entonces la lista es circular a través del nodo centinela.

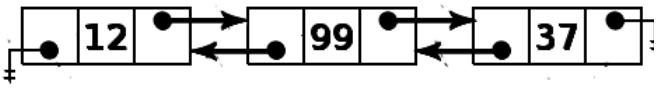

Figura 4. Lista Doblemente Enlazada.

El doble enlace de los nodos permite recorrer la lista en cualquier dirección. Mientras que agregar o eliminar un nodo en una lista doblemente enlazada requiere cambiar más enlaces que en estas mismas operaciones en una lista enlazada simple, las operaciones son más simples porque no hay necesidad de mantener guardado el nodo anterior durante el recorrido, ni necesidad de recorrer la lista para hallar el nodo anterior, la referencia al nodo que se quiere eliminar o insertar es lo único necesario.

El almacenamiento con nexos en sólo un sentido tiene el problema de no permitir recorrido hacia atrás, pero si se agrega un segundo apuntador se puede lograr este objetivo. Pero tienen el problema de requerir más espacio de memoria.

La ventaja que proporciona esta estructura es la operación de búsqueda e inserción, recorriendo hacia atrás en forma inmediata.

Una implementaci.on de un nodo de una lista doblemente enlazada se muestra en el siguiente código, donde nuevamente se asume que los elementos son
cadenas de caracteres.

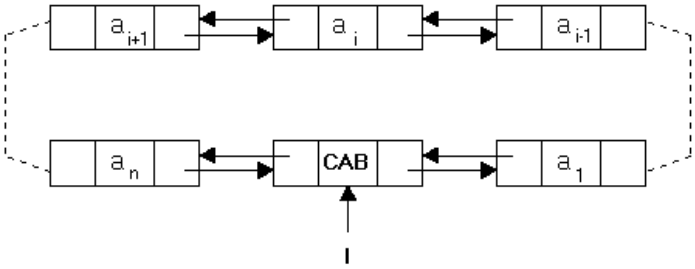

Figura 5. Visualización a fondo de una lista doblemente enlazada.

Los nodos de una lista doblemente enlazada tienen nada más dos apuntadores que es al siguiente nodo y al anterior, la diferencia con una lista ortogonal es que se puede describir la anteriormente mencionada como la unión de dos listas doblemente enlazadas para formar una matriz.

Las cuales unas listas doblementes enlazadas son las columnas y las otras representan las filas, así es como se forma una ortogonal con cuatro apuntadores que son arriba, abajo, derecha e izquierda.

Además que los apuntadores de la derecha de los últimos nodos de cada fila son nulos, los de la

izquierda de los nodos iniciales son nulos, los de abajo de la última fila son nulos y los de arriba de la primera fila son nulos.

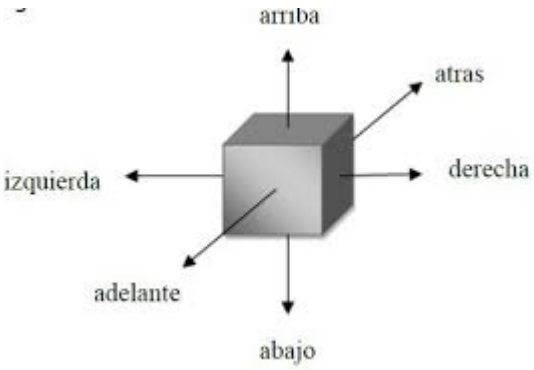

Figura 6. Nodo de tres niveles.

Este nodo es un caso especial, como se puede ver es un nodo es tercera dimensión si así se quiere ver ante el lector, lo especial y lo que identifica a este nodo es que posee seis apuntadores los cuales son: Izquierda, derecha, atrás, adelante, arriba y abajo.

En lo que resta de esta página se estará hablando y profundizando sobre la matriz dispersa.

Ventajas de una matriz dispersa:

1. Mínimo consumo de memoria
2. Mínimo uso de procesador
3. Localidad de datos

Sus formas de representación son:

1. Formato simple o elemental.
2. Formato CSR(Compressed Storage Row)
3. DIA (Diagonal Storage Format)
4. Ellpack-itpack
5. Estrategias dinámicas

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 & 3 & 4 & 0 \\ 0 & 5 & 0 & 6 & 0 & 0 & 0 \\ 0 & 7 & 8 & 9 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 4 & 5 & 0 \\ 0 & 0 & 6 & 0 & 7 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 9 \end{pmatrix}$$

$$d = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{pmatrix}$$
$$f = \begin{pmatrix} 1 & 5 & 7 & 10 & 12 & 15 & 18 & 19 \end{pmatrix}$$
$$c = \begin{pmatrix} 1 & 2 & 5 & 6 & 2 & 4 & 2 & 3 & 4 & 1 & 4 & 2 & 5 & 6 & 3 & 5 & 6 & 7 \end{pmatrix}$$

Figura 7. Formato CRS

Almacenamiento del formato CRS:

1. Un vector de un punto flotante de tamaño k en el que se almacenan los coeficientes.
2. Existe otro vector de tamaño k en el que se almacenan los números de columna de los elementos distintos de cero.
3. Un vector de tamaño n+1 siendo n la cantidad de filas de la matriz, en el cual se almacena la posición de la primera ocurrencia de cada fila.

El formato CCS(Compressed Column Storage) es igual al CRS pero en lugar de almacenar por filas se almacena por columnas y también se le hace llamar como el comprimido por columnas.

Ventajas y Desventajas de los formatos estáticos:

Ventajas:

1. Intuitivo
2. Se utilizan estructuras del mismo tamaño
3. Es igual para cuando se quiere acceder por fila que por columna.

Desventajas:

1. Utiliza más memoria que otras implementaciones.
2. Acceder por fila o por columna es más difícil que otros formatos.

Conclusiones

La implementación de listas ortogonales para formar matrices con cuatro apuntadores o de matrices dispersas es de gran ayuda para el ahorro de memoria.

Al usar cualquiera de las mencionadas nos facilita los cálculos sobre matrices de grandes dimensiones.

Con todo esto se concluye que el uso de los TDA es de gran importancia en la computación para facilitar cálculos grandes, ahorrar memoria, facilitar al computador su labor y llevar una secuencia sin pérdida de datos sobre los archivos de entrada ya que cada dato será debidamente ingresado en su respectivo lugar.

Referencias bibliográficas

1.Antonakos, James L. and Mansfield, Kenneth C., Jr. Practical Data Structures Using C/C++ (1999). Prentice-Hall. ISBN 0-13-280843-9, pp. 165–190

2.Collins, William J. Data Structures and the Java Collections Framework (2002,2005) New York, NY: McGraw Hill. ISBN 0-07-282379-8, pp. 239–303

3.Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford Introductions to Algorithms (2003). MIT Press. ISBN 0-262-03293-7, pp. 205–213, 501–505

4.Donald Knuth. Fundamental Algorithms, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89683-4. Sections 2.2.3–2.2.5, pp.254–298.

5.National Institute of Standards and Technology (August 16, 2004). Definition of a linked list. Retrieved December 14, 2004.