Laboratorio Nro. 1: Implementación de Grafos

Alejandro Cano Munera
Universidad Eafit
Medellín, Colombia
acanom@eafit.edu.co

Sebastián Giraldo Gómez
Universidad Eafit
Medellín, Colombia
sgiraldog@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. El ejercicio de dibujar el mapa de Medellín funciona con listas de adyacencia, primero se crea un método que lee toda la información contenida en el archivo de texto y posteriormente se crea un nodo y las aristas de este con la información dada.

   Matriz de adyacencia: Para la matriz de adyacencia se creó una matriz de tamaño n x n, donde n es el número de vértices en el grafo, dentro de cada posición de la matriz se almaceno el peso del arco entre dos nodos adyacentes. Lo que más se destaca de esta estructura es que se puede acceder en complejidad O(1) al peso del arco contenido entre dos nodos dados, lo cual hace muy eficientes las búsquedas de este tipo.

   Listas de adyacencia: Para la lista de adyacencia se creó una lista enlazada de listas enlazadas, donde cada nodo de la lista principal representa un vértice del grafo y la lista enlazada asociada a este representa los sucesores a dicho nodo. Lo que más se destaca de esta estructura es que se ahorra una significante cantidad de memoria, ya que, a diferencia de la matriz de adyacencia, no se guarda información acerca de los nodos que no están relacionados.

2. En grafos es más conveniente utilizar matrices de adyacencia cuando se trata de grafos completos, es decir, cuando todos los nodos están relacionados entre sí, ya que si se requiere la información entre dos nodos se pude acceder en el peor de los casos en un tiempo de O(1), sin embargo, cuando los nodos del grafo solo se relacionan con algunos otros nodos del mismo, es más conveniente usar listas de adyacencia, ya que se reduciría el espacio de memoria utilizada a la hora de generar el grafo; si utilizamos matrices de

adyacencia en un grafo que no es completo, se estaría reservando memoria para la información de dos nodos que no están conexos entre sí.

3. Para el mapa de Medellín es mejor usar listas de adyacencia, la razón principal es que ahorraríamos memoria a la hora de cargar el grafo. Este mapa no representa un grafo completo, es decir, si escogemos un nodo arbitrariamente del mapa, este no se relacionará con todos los nodos del mapa, sino que solo se relacionará con sus vecinos. Implementando listas de adyacencia reduciremos el espacio de memoria que ocupe el grafo, ya que no se reservaran espacios de la misma para almacenar información que no utilizaremos.

4. Para el caso de Facebook, es mejor usar listas de adyacencia, ya que una persona en su caso promedio no se relacionará con todos los usuarios de esta red social, de esta manera se ahorra memoria y se agiliza a la hora de listar o buscar los amigos de una persona. Haciendo un cálculo con la información dada donde Facebook tiene alrededor de 100 millones de usuarios y cada usuario tiene en promedio 200 amigos tendríamos que usando matrices de adyacencia estaríamos reservando alrededor de 10.000 billones de espacios de memoria para contener toda la información, y usando listas de adyacencia se reservarían alrededor de 20 mil millones, una diferencia de más del 100%.

5. Para el caso de los enrutadores es mejor usar matrices de adyacencia, la principal razón es que se puede saber en un tiempo con complejidad de O(1) la información requerida entre dos enrutadores. Esto se puede lograr debido a que cada enrutador guarda en la tabla la información que comparte con otro, y se puede encontrar de una manera eficaz donde se encuentra almacenada la información en el sistema, es decir, basta con ubicar la fila que representa un enrutador, y la columna que representa otro para halla la información entre ellos, debido a que se trata de matrices esto se logra en complejidad de O(1). Por otra parte, debido a que los enrutadores de cierto espacio se relacionan todos entre si sabiendo la distancia del uno con el otro, representarían un grafo completo y no se desperdiciaría memoria.

6. Coloración: Para resolver este problema se implementa un grafo creado a partir de matrices de adyacencia. Teniendo la información de los vértices y los arcos relacionados a este almacenados de esta manera se procede a verificar si es posible colorear el mapa con dos colores.

7. Coloración: $T(n) = n*n*C1 + C2$ es $O(n^2)$)

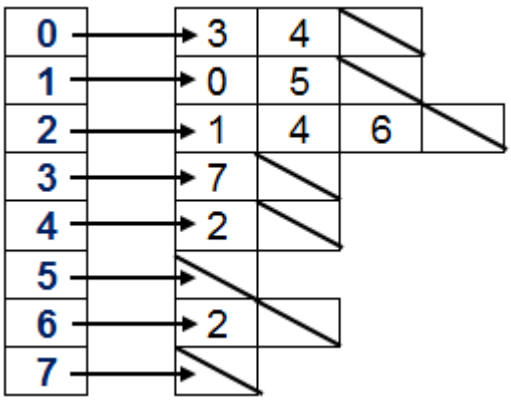
8. Coloración: En este ejercicio n significa el número de vértices en el mapa a colorear, es decir, el número de nodos que se recorren verificando como se debe hacer la coloración.

4) Simulacro de Parcial

1. Matriz de adyacencia:

	0	1	2	3	4	5	6	7
0				1	1			
1	1					1		
2	1				1		1	
3								1
4			1					
5								
6			1					
7								

2. Lista de adyacencia:



3. b)

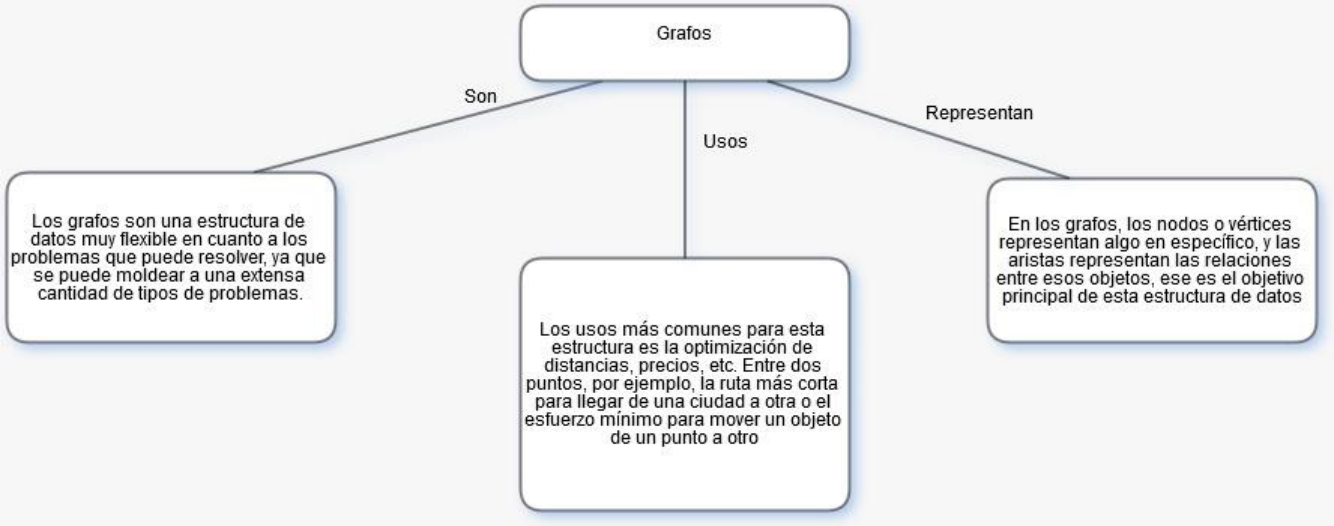
DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

5. Lectura recomendada:

a) Título: Data Structures and Algorithms. Chapter 13: Graphs

b) Resumen: En un sentido matemático, un árbol es un tipo de grafo. En programación, sin embargo, son usados de maneras diferentes. La forma de un grafo se puede definir por un problema físico, por ejemplo, los vértices de un grafo pueden representar ciudades, mientras las aristas representan vuelos. Mientras un árbol tiene una forma definida porque permite facilidad en la búsqueda e inserción de datos. Los grafos no intentan reflejar posiciones geométricas mostradas en un mapa, sólo muestra las relaciones de los vértices y las aristas. Por eso, uno de los problemas más comunes solucionados con grafos es buscar la manera más fácil de llegar a un puto final desde un punto de inicio

c) Mapa de Conceptos:



DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co