Laboratorio Nro. 1: Implementación de grafos

Isaias Labrador Sanchez
Universidad Eafit
Medellín, Colombia
ilabradors@eafit.edu.co

Santiago Hincapié Murillo
Universidad Eafit
Medellín, Colombia
shincapiem@eafit.edu.co

Andrés Almanzar Restrepo
Universidad Eafit
Medellín, Colombia
aalmanzarr@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1 Para la clase abstracta "digraph" únicamente almacena el tamaño del grafo, es decir el número de vértices que este tiene. Además, Como ya se sabe, contiene la firma de los métodos "addArc" que añade una arista entre dos vértices, "getSuccessors" que devuelve los vértices que se relacionen con el vértice parámetro, y el "getWeight" que devuelve el peso de la arista entre dos vértices. Para el grafo con listas únicamente se añade un arraylist de arreglos donde cada posición de la lista significa el vértice y la posición en el arreglo donde exista un peso diferente de 0 significa el vértice con el que tiene relación. Para el grafo con matriz de adyacencia simplemente se hace una matriz atributo que se inicializa al momento de crear el grafo con su respectivo tamaño.

3.2 Generalmente es mejor utilizar el grafo implementado con listas pues este ahorra espacio cuando el grafo no está completamente conectado (todos con todos). De Igual manera el grafo con listas no tiene una representación tan fácil como la matriz. La matriz es más eficiente cuando es un grafo dirigido y hay dos vértices que tienen aristas con diferente peso.

3.3 Listas Pues es un numero grande de datos, además se manejan dos datos que son X y Y del lugar, las listas facilitan tanto el acceso como ahorro de energía, aunque las matrices de adyacencia no se quedan atrás en cuanto al acceso ya que tiene número de posición al igual que las listas. Sin embargo por temas de ahorro de memoria son mejor las listas ya que las matrices de adyacencia requiere un almacenamiento |V| * |V|. Es decir O(n^2).

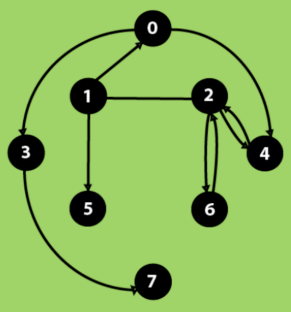
3.4 Son mejores las listas de adyacencia pues estas, primero, no utilizan un almacenamiento |V| * |V| = O(n^2), si no que requieren un almacenamiento O(n) y en cuanto acceso es lo mismo que con matrices de adyacencia, hablando en cuanto a complejidad.

3.5 Al igual que como respondimos en el numeral 3.3 las listas de adyacencia son mucho mejore para almacenar datos más grandes como es el caso al trabajar con tablas de enrutamiento con digamos todos los dispositivos del mundo para así garantizar una conexión, definitivamente son mejores las listas de adyacencia.

3.8 'n' o 'm' son las variables que se pueden dar para realizar el cálculo de complejidad.Como por ejemplo: Si hay un ciclo, el código se haría n veces hasta su condición de parada.

4) Simulacro de Parcial

1. Considere el siguiente grafo y complete la representación de **matrices de adyacencia**. Si no hay arco, por simplicidad, deje el espacio en blanco.



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

2. Lista de adyacencia
0 ->[3,4]
1 ->[0,2,5]
2 ->[1,4,6]
3 ->[7]
4 ->[2]
5 ->[]
6 ->[2]
7 ->[]

3. La canditad de memoria que utilizaría una representación usando listas de adyacencia para el peor grafo dirigido con n vértices sería: $o(n^2)$

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