ESTRUCTURA DE DATOS 2
Código ST0247

Laboratorio Nro. 1
Implementación de Grafos

Juan Sebastián Sanín Villarreal
Universidad Eafit
Medellín, Colombia
jssaninv@eafit.edu.co

Juan Pablo Peña Franco
Universidad Eafit
Medellín, Colombia
jppenaf@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1 Implementamos en nuestro algoritmo las tablas de hash y dentro de este mismo proceso implementamos también las listas de parejas cadena a decimal. Utilizamos los Strings ya que en el proceso de lectura de los datos al utilizar el split, quedaban como Strings, entonces para evitarnos el pasar la variable a entero y se presentarán posibles fallos en el futuro, decidimos dejarlos así.

El mapa está representado de la forma que en la key de la hashMap hay una lista de parejas donde podemos encontrar el value con un String que es el la otro ID y el double que es la distancia de las dos vías.

3.2 Se presenta un alto consumo de memoria ya que sería un Orden de O(n^2) ya que toca recorrer cada casilla para poder analizar si esta su respectivo enlace, también genera más gasto de memoria ya que toca separar un espacio específico de memoria sin saber si necesitaremos más o menos de lo que planteamos dependiendo si hay o no intersección con una coordenada dada se generan 90000000 con el O(n^2) y es una cantidad de memoria de 670GB.

3.3 Solo implementamos los Strings guardando las primeras coordenadas en la variable y así guardarlo en un espacio del HashMap. Seguido a ello evaluamos con qué coordenada tenía su intersección y así guardaremos en el value de esa key la distancia y el ID, por ende, no se nos presentó ningún problema que no empezara desde 0 ya que solo era hacer una comparativa de Strings.

3.4

Para resolver el problema usamos como estructura de datos una matriz, la cual nos ayudó a ver la solución de una manera más sencilla, ya que gracias a su estructura logramos identificar fácilmente cuando había una conexión entre dos nodos y poder trabajar a partir de eso.

Los algoritmos que se realizaron son muy sencillos. Hay 3 ciclos, el primero sirve para asignarle a los nodos un tercer color que va a servir como auxiliar, en el segundo ciclo se le asigna el número 1 en la matriz a la posición donde el usuario quiera el arco, ej: matriz[arco1][arco2] = 1.

En el último ciclo (en realidad son 2 ciclos) se recorre la matriz y se buscan las posiciones donde están los número 1 anteriormente guardados para asignarles un color y verificar que sea bi-coloreable el grafo. Finalmente se llama un método "respuesta" al cual se le indica el resultado anterior y escribe la respuesta por consola y además indica al usuario que debe hacer para seguir con otro grafo o terminar el programa y por último se utiliza el método del main para que el programa pueda ser ejecutado.

3.5

La complejidad del algoritmo es O(max($mat^2.length$, numArcos, nodos)), es decir O($mat^2.length$), puesto a que hay tres ciclos y en el último en realidad hay dos porque con ese se recorre la matriz.

3.6

mat.length: es el tamaño de la matriz y se usa dos veces la misma variable para los dos ciclos.

numArcos: es el número de arcos que el usuario asigna para el grafo.

nodos.length: es el tamaño del arreglo donde se van a guardar los colores de los nodos.

4) Simulacro de Parcial

4.1

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

4.2

0 → [3,4]
1 → [0,2,5]
2 → [1,4.6]
3 → [7]
4 → [2]
5 → []
6 → [2]
7 → []

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4.3 *b)* $O(n^2)$

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473