**Laboratorio Nro. 1  
Implementación de Grafos**

|  |  |
| --- | --- |
| **Miguel Ángel Martínez Florez**  Universidad Eafit  Medellín, Colombia  mamartinef@eafit.edu.co | **Pablo Micolta López**  Universidad Eafit  Medellín, Colombia  pmicoltal@eafit.edu.co |
| **Daniel Ricardo Palacios Diego**  Universidad Eafit  Medellín, Colombia  drpalaciod@eafit.edu.co |  |

**3) Simulacro de preguntas de sustentación de Proyectos**

**3.1**

**R/** Este problema nos pedía cargar un mapa de la ciudad, nuestra solución implemento dos tablas de hash, ya que, al darnos la id de los lugares a los que nos vamos a dirigir, esta estructura nos permitía un fácil acceso a estos. En el primero, se guardaban los nombres del lugar y en el segundo se guardaban todas las conexiones y su peso que tenía en forma de una lista de adyacencia porque esta nos permite un fácil procesamiento y menor peso.

**3.2**

**R/** Como cada nodo debe tener una posición tanto en eje vertical como en el horizontal**,** entonces cada una de esas posiciones va a consumir un espacio en la memoria, de ahí que las matrices tengan una complejidad de memoria de n2 y por lo tanto ocuparía 9x1010 bits de memoria.

**3.3**

**R/** nosotros guardamos cada línea del archivo en una lista como un arreglo, usando la función de los string.split() y con el conocimiento del formato en el que se nos está dando la información relacionamos la primera posición de la línea (donde estaba la ID del nodo) la cual la pusimos en los diccionarios mencionados previamente siendo esto la representación del nodo obviando la necesidad de que esas ID’s empiecen desde la ID 0 pues se están asociando a una clave que nos da acceso a su información sin necesidad de que estas ID’s cumplan un criterio fuera de estar en la primera posición de la linea.

**3.4**

**R/** En este problema, nos pedía separar los nodos de un grafo en dos conjuntos, en este caso lo tratamos como una matriz de adyacencia, ya que nos parecía que de esta manera podíamos acceder más fácilmente a la información. además de esto, se hizo otro arreglo donde se mantenía cuenta de en qué subconjunto estaba cada nodo, entonces asumimos que el nodo 0 estaría en el subconjunto 1 y por lo tanto todas sus conexiones estarían en el subconjunto 2; con esto en el arreglo recorrimos la matriz comparándola con los valores del arreglo si se encontraba con un nodo que no tenia valor se le asignaba un valor a base del valor del que se estaba evaluando y si en base al nodo que se esta evaluando hay una contradicción con lo ya puesto en el arreglo se denominaba como no divisible.

**3.5**

**R/** La complejidad es n2, porque la única complejidad variable es el recorrer el arreglo cuya complejidad es equivalente a nxn.

**3.6**

**R/** n es la cantidad de nodos presentes en el grafo

***4) Simulacro de Parcial***

R/

Un dibujo de una persona

Descripción generada automáticamente con confianza mediaImagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

* 1. .

R/

0-> [3,4]

1->[0,2,5]

2->[1,4,6]

3->[7]

4->[2]

5-> None

6->[2]

7->None

R/ O(n2 )

***5) Lectura recomendada (opcional)***

Mapa conceptual

**6)** **Trabajo en Equipo y Progreso Gradual (Opcional)**

***6.1*** *Actas de reunión*

***6.2*** *El reporte de cambios en el código*

***6.3*** *El reporte de cambios del informe de laboratorio*