Laboratorio 3: Estructuras de Datos no Lineales

Luis Adrian Aguilar Casacante - B00092
Robin González Ricz - B43011
Giancarlo Marín Henández - B54099

1 de marzo de 2017

Índice general

Índice de figuras

0.1. Enunciado

Implementar las estructuras no lineales de datos utilizando plantillas, herencia y POO en C++. Las estructuras que debe implementar recibirán en los argumentos de la plantilla el tipo de dato que almacenará Debe implementar:

0.1.1. Arbol de Búsqueda Binario(BST)

1. Esta clase debe utilizar dos clases auxiliares: Node, que representan los nodos del árbol y Data, que representa la información que se almacena. El BST debe implementar los siguientes métodos:

   a) void insert(Data* d)
   b) void removing(Node<Data>* n)
   c) void remove(Data* d)
   d) Data* get(Node<Data>* n)
   e) void set(Node<Data>* n, Data * d)
   f) Node<Data>* finding(Data* d, Node<Data>* n)
   g) Node<Data>* find(Data* d)
   h) void balance()
   i) void printTree()
   j) void inOrder(Node<Data>* n, string s)
   k) postOrder(Node<Data>* n, string s)
   l) preOrder(Node<Data>* n, string s)
   m) Node<Data>* replacementFor(Node<Data>* n)
   n) Node<Data>* minGreater(Node<Data>* n)
   ñ) Node<Data>* maxLesser(Node<Data>* n)

0.1.2. Grafos

- Grafo con arreglos
  - Esta clase representa un grafo por medio de una matriz de adyacencia. Utilice una clase Data para almacenar los datos en el grafo.
- Grafo con punteros
  - Esta clase representa un grafo por medio de punteros y nodos. Utilice una clase Data para almacenar los datos en el grafo, una clase Node para representar los nodos y una clase Edge para representar las aristas.

Los grafos deben implementar los siguientes métodos:

1. void addEdge(double w, Node<Data>* s, Node<Data>* d)
2. void addNode(Data d)
3. void delEdge()
4. void delNode()
5. Node<Data>* firstNode()
6. Node<Data>* nextNode(Node<Data>)
7. Node<Data>* Node(int i)
8. Data* getData(Node<Data>*)
9. Node<Data>* dfs(Data*)
10. Node<Data>* bfs(Data*)
11. double* dijkstra()
12. double** floyd()

Haga un programa de prueba para sus estructuras. Asegúrese de hacer inserciones, borrados y búsquedas.

Por último realice una revisión de:

1. Arboles rojo y negro
2. Cliques

Capítulo 1

Introducción

El presente laboratorio realiza la implementacion de diferentes estructuras de datos en C++. Además presentar una revisión teórica de Arboles Rojo-Negro y Clíques los cuales son un tipo de Arbol y de Grafo respectivamente, para una mayor comprensión de las distintas estructuras.

Capítulo 2

Clases en c++

2.1. Arbol de Búsqueda Binario(BST)

2.2. Grafos

2.2.1. Grafo con Arreglos

2.2.2. Grafo con Punteros

Capítulo 3

Revisión

3.1. Arboles Rojo-Negro

Los árboles rojo-negro son estructuras de datos al igual que los demás tipos de árboles. Estos son balanceados de tal forma que el tiempo de ejecución se represente como O(log(n)) en el peor de los casos. Dichos árboles se caracterizan porque *cada nodo almacena un bit adicional de información llamado color, el cual puede ser rojo o negro. Sobre este atributo de color se aplican restricciones que resultan en un árbol en el que ningún camino de la raíz a una hoja es más de dos veces más largo que cualquier otro, lo cual significa que el árbol es balanceado. Cada nodo de un árbol rojo negro contiene la siguiente información: color, clave, hijo izquierdo, hijo derecho y padre. Si un hijo o el padre de un nodo no existe, el apuntador correspondiente contiene el valor NULO, el cual consideraremos como un nodo cuyo color es negro.*[1]

Los árboles Rojo-Negro den de satisfacer las siguientes 6 propiedades:

1. Todo nodo es rojo o negro.
2. La raíz negra.
3. Toda hoja (NULO) es negra.
4. Un nodo es rojo cuando sus dos hijos son negros.
5. En un camino, no puede haber más de dos nodos rojos consecutivos, pero sí pueden haber n nodos negros consecutivos. Esto significa que un nodo rojo no puede tener ningún hijo rojo.
6. Para cada nodo, todas las rutas de un nodo a las hojas (NULOs) contienen el mismo número de nodos negros. El número de nodos negros en esta ruta se conoce como ALTURA-NEGRA del nodo.

3.2. Clíques

Se definen los cliques como grafos en los cuales cada uno de sus nodos se conecta directamente con cada uno de los demás nodos del grafo. Es decir, cada nodo debe interconectarse con todos y cada uno de los otros nodos que componen el grafo.

Los Clíques pueden ser también subgrafos que se encuetren dentro de otro grafo más grande. Incluso, un mismo grafo puede tener varios subgrafos que a su vez sean Clíques.

Existen Clíques máximos *Sin embargo, es necesario tener cuidado ya que los Cliques Máximos son muchas veces llamados simplemente Cliques. Un Clíque Máximo es uno que no puede ser extendido incluyendo más vértices adyacentes* [2]

Como dato curioso, el Clique recibe su nombre de una palabra de origen Francés y adaptada por los ingleses, la cual hace referencia a un pequeñ grupo de personas que comparten los mismo intereses.

Capítulo 4

Conclusiones.

-
-
-

Capítulo 5

Anexos

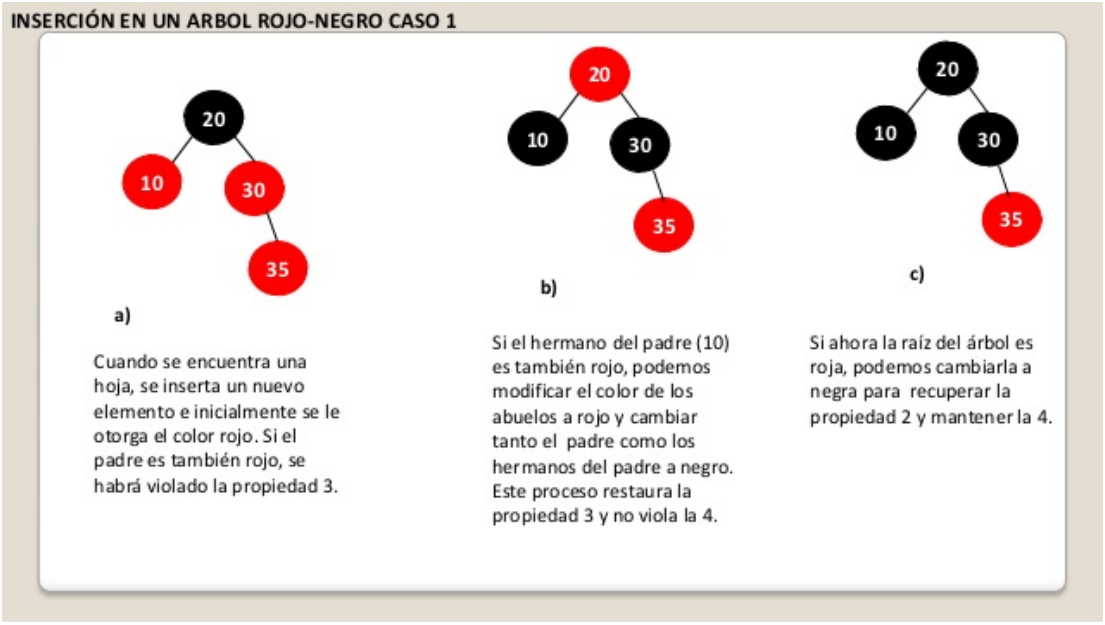

Figura 5.1: *Arbol Rojo-Negro*

Bibliografía

[1] Universidad Francisco de Paula Santander. Arbol RojiNegro. http://ingsistemas.ufps.edu.co/SEED/arbolrojinegro.html, 2016. [Online; accessed 26-Febrero-2017].

[2] F Harry. Graph Theory. Reading, MA: Addison-Wesley. http://mathworld.wolfram.com/Clique.html, 1994. [Online; accessed 27-Febrero-2017].

[3] Castillo Martinez Jessica Aracely and Magaly Martinez Juárez. Análisis, Programación y Diseño de Algoritmos. https://www.slideshare.net/whaleejaa/arbol-rojo-negro, 2016. [Online; accessed 26-Febrero-2017].