UNIVERSIDAD ADOLFO IBÁÑEZ
FACULTAD DE INGENIERÍA Y CIENCIAS

Ayudantía 7: Árboles en C

Profesores: Sebastián Sáez, Diego Ramos

Ayudantes: Diego Duhalde, Benjamín Wiedmaier, Fernando Zamora

PREGUNTAS

1. Defina qué es un árbol binario y explique cómo se representa en memoria con nodos y punteros. Mencione la definición de altura de un árbol y de nodo hoja.

2. Escriba en C una función que permita insertar valores en un árbol binario de búsqueda (BST). La función debe tener la siguiente forma: `nodo* insert(nodo* node, int data)`.

3. Implemente en C la función `void inorder(nodo* root);` que recorra un BST en orden y muestre los datos con `printf`. ¿Cómo cambiaría para preorden y postorden?

4. Dado el siguiente fragmento de código de inserción en BST:

```
if (node == NULL) {
    return newNode(data);
}

if (data < node->data) {
    node->left = insert(node->left, data);
} else {
    node->right = insert(node->right, data);
    return node;
}
```

   ¿Qué hace cada línea? Explique paso a paso.

5. Escriba la función `nodo* minValueNode(nodo* node);` que devuelva el nodo con el valor mínimo de un subárbol.

6. Analice el siguiente árbol y determine la salida de un recorrido inorden:

$$\{10, 5, 15, 3, 7, 12, 18\}$$

7. Considere el árbol resultante de insertar en orden los valores `[8, 3, 10, 1, 6, 14, 4, 7, 13]`.

   a) Dibuje la estructura del BST.

   b) ¿Cuál es el resultado de su recorrido postorden?

RESPUESTAS

1. Un árbol binario es una estructura de datos jerárquica en la que cada nodo tiene como máximo dos hijos, denominados hijo izquierdo e hijo derecho. Se representa en memoria mediante nodos que contienen un valor, un puntero al hijo izquierdo y un puntero al hijo derecho. La altura de un árbol es la longitud del camino más largo desde la raíz hasta una hoja. Un nodo hoja es un nodo que no tiene hijos.

2. La implementación del código la puede encontrar en la rama de ayudantías del repositorio del curso. Link.

3. Sea R la raíz, I el subárbol izquierdo y D el subárbol derecho, entonces, en inorder (I, R, D) se recorre primero el subárbol izquierdo, luego el nodo actual y finalmente el subárbol derecho. Luego, para preorder (R, I, D) se visita primero el nodo actual, luego el subárbol izquierdo y finalmente el subárbol derecho. Por último, en postorder (I, D, R) se recorre primero el subárbol izquierdo, luego el derecho y finalmente el nodo actual. La implementación del código la puede encontrar aquí.

4. El paso a paso es el siguiente:

   a) La primera línea verifica si el nodo actual es `NULL`. Si es así, significa que hemos encontrado la posición donde debe insertarse el nuevo nodo, por lo que se llama a la función `newNode(data)` para crear un nuevo nodo con el valor dado y se retorna.

   b) La segunda línea compara el valor `data` con el valor del nodo actual (`node->data`). Si `data` es menor, se procede a insertar el valor en el subárbol izquierdo llamando recursivamente a la función `insert` con `node->left`.

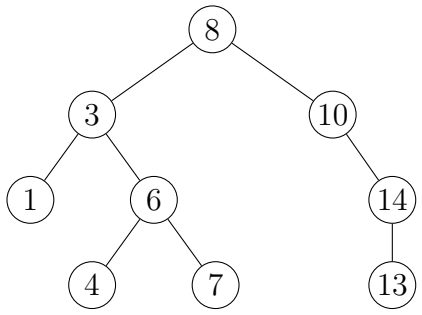
   c) La tercera línea maneja el caso en que `data` es mayor o igual al valor del nodo actual. En este caso, se procede a insertar el valor en el subárbol derecho llamando recursivamente a la función `insert` con `node->right`.

   d) Finalmente, se retorna el nodo actual (`node`) para mantener la estructura del árbol.

5. La implementación del código la puede encontrar en la rama de ayudantías del repositorio del curso. Link.

6. La salida de un recorrido inorden para el árbol dado es: `3, 5, 7, 10, 12, 15, 18`. Esto se debe a que en un recorrido inorden, se visitan los nodos en el siguiente orden: subárbol izquierdo, nodo raíz, subárbol derecho.

7. a) La estructura del BST resultante es la siguiente:



   b) El resultado del recorrido postorden es: `1, 4, 7, 6, 3, 13, 14, 10, 8`.