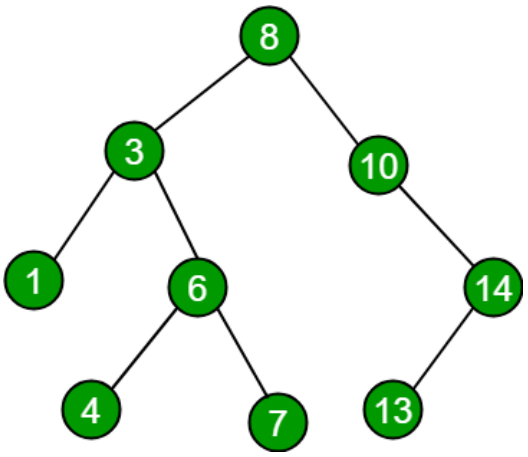
Carlos César Rodríguez García | A00344666
TC1031 Grupo 13

La importancia de los Binary Search Trees

Un Binary Search Tree o BST es un tipo de dato derivado de un Binary Tree, una estructura de datos no lineal donde cada nodo tiene, como máximo, dos hijos. La particularidad de los BST es que para cada nodo el hijo de la izquierda tendrá un valor (*key value*) menor al suyo; y el hijo derecho, uno mayor. De esta forma, los nodos en un BST se componen de un valor *key*, con el cual se compararán con otros nodos, y quizá algún dato (*dato*) asociado.

Gracias a lo anterior, se cumplen las siguientes reglas:

- Todas las *keys* en el subárbol izquierdo de *n* son menores a la *key* en *n*.
- todas las *keys* en el subárbol derecho de n son mayores a la *key* de *n*.



Ejemplo de *Binary Search Tree*. Recuperado de:
https://www.geeksforgeeks.org/check-if-given-sorted-sub-sequence-exists-in-binary-search-tree/

La importancia de los BST

Los BSTs pueden tener varias aplicaciones para resolver problemas reales, sin embargo, se reconoce que su mayor virtud se encuentra en el ordenamiento de información.

Como ya se ha visto en actividades anteriores del curso, el ordenamiento es una herramienta fundamental para la resolución de problemas en distintos contextos: ciencia de datos, hospitales; cuando ordenamos artículos por precio, cuando vemos la lista de contactos en nuestro celular, etc. Especialmente en estos tiempos, en los cuales se genera información

con prácticamente cualquier acción del ser humano, es difícil ordenar y encontrar la data correcta entre miles de data points; no obstante, un BST puede ayudar a simplificar estos procesos.

Con un BST es fácil realizar la búsqueda de un elemento pues un lookup se puede llevar a cabo en un tiempo O(log N), esto debido a que un árbol agrega una dimensión extra al caso común de un array (unidimensional) con lo cual es más fácil distribuir la información. De igual forma, un BST puede fácilmente imprimir todos los valores *key* de manera ordenada con un *Inorder Traversal.*

Operaciones comunes en un BST

Las operaciones más comunes en los BSTs son: *search*, *insert*, *delete* y *traversal*.

En las primeras 3 se recorre un único camino desde la raíz del árbol hasta llegar a una hoja, por lo cual sus complejidades son, en promedio, O(h), donde *h* representa la altura del árbol.

Así como el árbol mismo, todas estas funciones tienen una naturaleza recursiva, con lo cual se puede fácilmente acceder a nodos consecutivos. A continuación se muestra un ejemplo de inserción en un BST.

```
Node* insert(Node* currentNode, int key, int data){

    // Si no existe el nodo (se llega a la posición deseada) se crea.
    if (currentNode == NULL) {
        currentNode = new Node(key, data);
        return currentNode;
    }

    if (key < currentNode->key) {
        // Inserción a la izquierda, si el valor de la key
        // es menor que la key del nodo actual

        // Procesamiento de nodo izquierdo
        currentNode->left = insert(currentNode->left, key, data);
    } else {

        // Inserción a la derecha, si el valor de la key
        // es mayor que la key del nodo actual

        // Procesamiento de nodo derecho
```

```
            currentNode->right = insert(currentNode->right, key, data);
        }

        // Retorno de nodo (en la llamada principal, regresa "root")
        return currentNode;
    }
```

Por su parte, los *traversals* visitan todos los nodos del array, por lo que se puede asumir que su complejidad será de O(n). Los *traversals* pueden ser de distintos tipos:

- **Post-order:** se visita el subárbol a la izquierda, el subárbol a la derecha y hasta al final la raíz.
- **Pre-order:** se visita primero la raíz, después el subárbol izquierdo y por último, el derecho.
- **In-order:** se visita el subárbol a la izquierda, la raíz y el subárbol a la derecha.

Bibliografía

- Skotar, M. (Septiembre de 2019). Importance of Binary Search Tress. *Medium*. Recuperado de https://medium.com/@michaelskotar/importance-of-binary-search-trees-d354afc6e347
- University of Wisconsis. *Binary Search Trees.* Recuperado de http://pages.cs.wisc.edu/~vernon/cs367/notes/9.BST.html