Estructuras de Datos 2022-1
Práctica 5: Árboles Binarios de Búsqueda

Pedro Ulises Cervantes González
confundeme@ciencias.unam.mx

Emmanuel Cruz Hernández
emmanuel_cruzh@ciencias.unam.mx

Yessica Janeth Pablo Martínez
yessica_j_pablo@ciencias.unam.mx

América Montserrat García Coronado
ame_coronado@ciencias.unam.mx

Adrián Felipe Vélez Rivera
adrianf_velez@ciencias.unam.mx

Fecha límite de entrega: 24 de noviembre de 2021
Hora límite de entrega: 23:59:59 hrs

1. Objetivo

Implementar y conocer el funcionamiento interno de la estructura de datos conocida como Binary Search Tree Árbol (BST) o Árbol Binario de Búsqueda.

Esta estructura se basa en un orden jerárquico y necesita un conjunto de elementos comparables para darle un lugar específico a cada elemento dentro de un árbol binario.

2. Actividad

Implementa las operaciones de la interfaz *TDABinarySearchTree* en una clase llamada *BinarySearchTree*. El objetivo es programar el comportamiento de un árbol binario de búsqueda.

2.1. Retrieve *(1 puntos)*

Implementa el método `retrieve(K)`, que dada una clave k, regresa el elemento asociado a dicha clave o null si no existe.

2.2. Insert *2 puntos)*

Implementa el método `insert(T, K)`, que dada una clave k y un elemento t, inserta el elemento t con clave k en el árbol, considerando que en un nodo p con elemento *e(p)*, tiene en el subárbol izquierdo a todos los elementos cuya clave es menor a *e(p)* y como subárbol derecho a todos los elementos con clave mayor a *e(p)*.

2.3. Delete *(2 puntos)*

Implementa el método `delete(K)`, que dada una clave k elimina su elemento asociado en el árbol, además de regresar el elemento. En caso de no existir, se regresan null.

2.4. FindMin *(1 puntos)*

Implementa el método `findMin()` que regresa el elemento con clave mínima en el árbol o null si el árbol es vacío.

2.5. FindMax *(1 puntos)*

Implementa el método `findMin()` que regresa el elemento con clave máxima en el árbol o null si el árbol es vacío.

2.6. isEmpty *(1 puntos)*

Implementa el método `isEmpty()` que regresa *true* si el árbol es vacío y *false* en otro caso.

2.7. Recorridos *(1 puntos)*

Implementa los recorridos siguientes sobre el árbol binario de búsqueda.

- Preorden
- Inorden
- Postorden

Visitar en este caso se tomará como imprimir el elemento actual del árbol.

2.8. Main *(1 puntos)*

Crea un método *main* **en la clase BinarySearchTree** con un menú que permita hacer todas las operaciones definidas, así como salir del mismo.

3. Recursos de apoyo

- Presentación con animaciones:
  `https://docs.google.com/presentation/d/1n3zvmOyqUTIcNYxqnc7iIpcLZ-qqV_eBTdZas-o8Cr8/edit?usp=sharing`

4. Reglas Importantes

- No se recibirán prácticas en las que estén involucrados más de dos integrantes.
- Cumple con los lineamientos de entrega.
- Todos los archivos deberán contener nombre y número de cuenta.
- Tu código debe estar comentado. Esto abarca clases, atributos, métodos y comentarios extra.
- Utiliza correctamente las convenciones para nombrar variables, constantes, clases y métodos.
- No se pueden usar implementaciones de estructuras de datos ya hechas por Java. En caso de usarse, se evaluará con 0 la actividad.
- En caso de no cumplirse alguna de las reglas especificadas, se restará 0.5 puntos en tu calificación obtenida.