Taller 2

Árbol roji Negro

Para este árbol se utilizo la clase set de las librerías estándares de c++, que tiene implementado un árbol de este tipo por debajo, esta clase tiene una gran cantidad de funciones de las cuales se utilizaron la de insertar , eliminar y la utilización de los iteradores para poder recorrer el árbol y compararlo con los otros, este árbol se tomo como el correcto y se comparaba con los otros para ver si los otros estaban correctos.

Árbol Binario Ordenado

Para este árbol se utilizo la clase ArbolBinarioOrd la cual fue una implementación de nuestro equipo, en donde se realizaron las funciones para poder insertar , eliminar y llenar una lista para poder luego compara con el árbol roji negra, y verificar que la respuesta es correcta.

Heap

Para esta estructura se utilizo una clase de la librería estándar de c++, la cual es priority_queue con un comparador para que el heap estuviera de menor a mayor, también se creo una función especial para crear la lista para poder compara los datos, y otra función para poder eliminar un dato especifico de este heap.

Medición de tiempo

Plates0.txt:
Tiempo de llenado arbol RN = 0.000117segs.
Tiempo de llenado arbol ORD = 7.6e-05segs.
Tiempo de llenado HEAP = 9.7e-05segs.

Plates1.txt:
Tiempo de llenado arbol RN = 0.000786segs.
Tiempo de llenado arbol ORD = 0.000749segs.
Tiempo de llenado HEAP = 0.000763segs.

Plates3.txt:
Tiempo de llenado arbol RN = 0.004072segs.
Tiempo de llenado arbol ORD = 0.004186segs.
Tiempo de llenado HEAP = 0.004215segs.

Como se puede notar la diferencia de tiempo entre las tres estructuras, no es muy grande ya que casi todas funcionan con complejidades parecidas, aunque sus utilizaciones pueden diferir, la recomendación para este tipo de programas seria la estructura de árbol roji negra, ya que esta es la estructura que se mantiene mas va lanceada y con una gran cantidad de datos va a funcionar mejor que las otras estructuras de datos, además de que se hacer mas fácil buscar un dato por las funciones implantadas que tiene.