UNR - FCEIA

Estructuras de Datos y Algoritmos I

Trabajo Práctico N^{o} 2 - Arboles de Intervalo AVL

Spoletini Bruno, Poma Lucas

Introducción

En este trabajo utilizamos un árbol de intervalos AVL, sobre el cuál implementamos un conjunto de funciones tales como insertar un intervalo, borrar un intervalo, e intersecar, la cuál nos permite saber si un intervalo dado interseca con otro intervalo perteneciente al árbol, en tiempo a lo sumo O(ln n).
Para interactuar con nuestro programa, creamos un intérprete, el cuál utilizaremos para realizar un conjunto de operaciones sobre el árbol.

Intérprete

El intérprete cuenta con seis comandos disponibles para manipular el árbol de intervalos desde la entrada estandar:

- i [a,b]: inserta el intervalo [a,b] en el arbol.
- e [a,b]: elimina el intervalo [a,b] del arbol.
- ? [a,b]: intersecta el intervalo [a,b] con los intervalos del arbol.
- dfs: imprime los intervalos del arbol con recorrido primero en profundidad.
- bfs: imprime los intervalos del arbol con recorrido primero a lo ancho.
- salir: destruye el arbol y termina el programa.

Compilación y Ejecución del programa

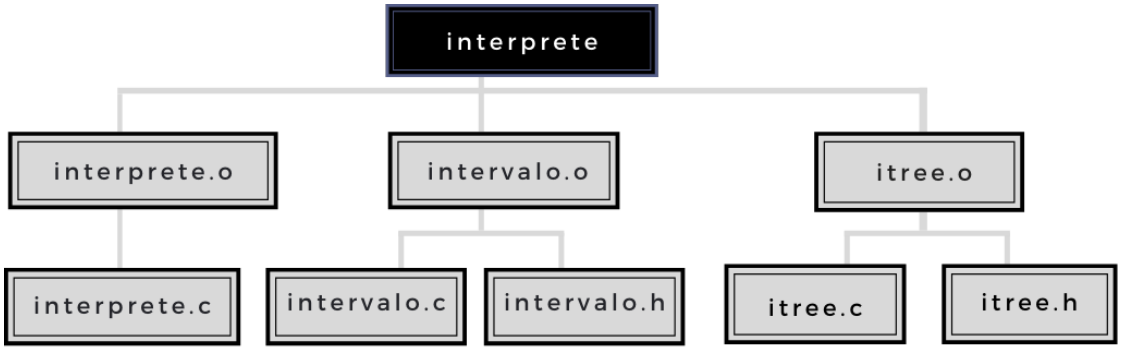
El programa cuenta con un archivo makefile, que se encarga de compilar los archivos correspondientes al funcionamiento del programa.
Para ejecutarlo, basta con utilizar el comando make en la terminal, estando situado en el directorio principal.
Esto generará un ejecutable "interprete".

Estructura del programa

- interprete.c: contiene las funciones de validacion de los datos de entrada y del intérprete.
- itree.c: contiene las funciones que manejan la estructura de árbol de intervalos AVL.
- itree.h: archivo cabecera de itree.h, y contiene las declaraciones de estructuras de árbol AVL.
- intervalo.c: contiene las funciones que manipulan la estructura de datos intervalo

- intervalo.h: archivo cabecera de intervalo.c, y contiene la declaracion de la estructura intervalo.

Árbol de dependencias



Estructuras de datos

Para este trabajo utilizamos una estructura de árbol de intervalos AVL, en la que cada nodo contiene su altura, un puntero a una estructura intervalo, el máximo extremo derecho de sus ramas y dos punteros a nodo. La misma está definida de la siguiente manera:

```
typedef struct _INodo{
    Intervalo *intervalo;
    double maximo;
    int altura;
    struct _INodo *izq;
    struct _INodo *der;
} INodo;
```

Utilizamos también la estructura "intervalo", la cuál contiene dos variables tipo double.

Problemas encontrados y resoluciones propuestas

El principal problema con el que nos encontramos se centra en las rotaciones necesarias para que el árbol mantenga su condición de AVL luego de hacer inserciones y eliminar nodos del mismo.
Pudimos solucionar este problema con la ayuda de la separación de los casos en papel, y la ayuda de algunos videos que explicaban cada caso. Otro problema que encontramos fue a la hora de pensar en todos los casos para aprovechar el valor maximo que contiene cada nodo a la hora de utilizar la funcion intersecar, para lograr que esta funcion sea lo mas eficiente posible.

Fuentes consultadas

https://es.wikipedia.org/wiki/Arbol_AVL
http://es.tldp.org/Tutoriales/doc-programacion-arboles-avl/avl-trees.pdf
https://es.qwe.wiki/wiki/AVL_tree
https://runestone.academy/runestone/static/pythoned/Trees/ImplementacionDeUnArbolAVL.html
https://www.youtube.com/watch?v=gx7NL3aQM5Qt=325s