FACULTAD DE CIENCIAS EXACTAS, INGENIERÍA Y AGRIMENSURA
ESCUELA DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN
ESTRUCTURA DE DATOS Y ALGORITMOS I

---

# Trabajo práctico II

# INFORME

**Integrantes del grupo:**

- Coltre, Andrés. C-5867/1
- Tramontini, Francisco. T-2877/1

2020

**DETALLES PROGRAMA, COMPILACIÓN y EJECUCIÓN**

Se trabajó con los siguientes archivos:

- *cola.c(.h)*: se encuentran las funciones que utilizaremos para manipular colas (una adaptación para árboles de intervalos), en el trabajo utilizaremos las colas para recorrer primero a lo ancho un árbol de intervalos.
- *itree.c(.h)*: se encuentran las funciones que utilizaremos para manipular árboles de intervalos, entre ellas se encontrarán funciones que permiten mantener un árbol de intervalos en AVL.
- *interprete.c*: manipula árboles de intervalos desde la entrada estándar.

COMPILACIÓN:

Utilizamos gcc. En una terminal de comandos, ingresemos la siguiente instrucción:

...:~$ *make*

EJECUCIÓN:

Desde una terminal de comandos:

...:~$ *./interprete*

Nota:

Para ingresar, eliminar o intersecar tener en cuenta:

La forma aceptada es <*comando*> <*espacio*> <*intervalo*> donde
<*comando*> = i *(insertar),* e *(eliminar),* ? (*intersecar)*
<*espacio*> = barra espaciadora del teclado.
<*intervalo*> = solo se acepta el intervalo [a,b] sin espacios donde a y b son valores (double).

Para los demás <*comandos*> que no necesitan un intervalo, tienen la forma
<*comando*> = dfs *(recorrido DFS)*, bfs *(recorrido BFS)*, salir *(termina el intérprete)*

Si se ingresa valores no correspondientes a <*comando*> y <*intervalo*> se informara que hubo un error.

**DESCRIPCIÓN DEL PROGRAMA Y DIFICULTADES ENCONTRADAS.**

En primer lugar empezamos pensado la estructura a utilizar.

A la hora de construir la estructura del árbol de intervalos no tuvimos inconvenientes, pensamos primero como base a una estructura de árboles de búsqueda binaria y le agregamos los intervalos como una estructura que maneja dos valores (**double**) como dato del árbol y un valor extra (**double**) donde maneja el máximo extremo derecho entre los intervalos contenidos en ese subárbol.

Para mantener los árboles balanceados en altura (AVL) decidimos aumentar la estructura con una variable (**int**) donde lleve la altura actual del nodo, esto nos facilitó no necesitar calcular la altura cada vez que queríamos balancear.

```
typedef struct _Intervalo {
    double a, b;
} Intervalo;
```

*(estructura para el intervalo)*

```
typedef struct _ITNodo {
    Intervalo interval;
    double extra;
    struct _ITNodo *left;
    struct _ITNodo *right;
    int altura;
} ITNodo;

typedef ITNodo *ITree;
```

*(estructura para el árbol de intervalos)*

Uno de los primeros problemas que encontramos al empezar a pensar el trabajo fue encontrar una función donde vaya actualizando el máximo extremo derecho entre los intervalos contenidos en ese subárbol. Para esto decidimos utilizar una función *(se encuentra en la bibliografía (1))*

$$double\ fmax\ (double\ x,\ double\ y)$$

está contenida en la librería ***math.h*** donde devuelve el máximo entre dos valores double, asi pudimos ir comparando los intervalos del nodo que le pasamos como argumento y sus hijos (si es que tiene) e ir actualizando.

A partir de esta función también nos inspiró una función análoga que actualiza las alturas, ya que cada nodo cumple la siguiente propiedad:

$$altura\ (nodo)\ =\ max\ (\ altura\ (hijo\ izquierdo),\ altura\ (hijo\ derecho))\ +1$$

En la función insertar tanto como en la eliminar tuvimos problemas con el tema de actualizar el máximo de los nodos cuando insertamos y cambiar el máximo en los nodos superiores si insertamos en un hijo de algún nodo y análogamente cuando se elimina un nodo con un extremo derecho máximo.
Para resolver este problema decidimos implementarlos de forma recursiva, y en cada paso llamar a las funciones que actualicen altura y máximo.
Esto es posible sin sacrificar eficiencia gracias a que las funciones que actualizan altura y máximo en si son eficientes, solamente teniendo que hacer comparaciones y asignaciones de valores.
Si la inserción provoca un desbalance del árbol, las funciones de balance y rotacion/es se encargan de actualizar los máximos y alturas.

Para el recorrido primero por profundidad *(DFS)* decidimos ir por el recorrido IN ORDEN, ya que con este recorrido reflejamos el árbol con intervalos ordenados, por ser binario de búsqueda.

En el recorrido primero a lo ancho *(BFS)*, la práctica de árboles sugeria hacerlo utilizando colas, entonces decidimos ir por ese camino.
Se tuvo que adaptar la implementación de colas para que maneje árboles de intervalos y a partir de eso pudimos implementar el recorrido.

Otra dificultad fue la implementación del interprete, tuvimos que investigar alguna función que nos facilita extraer los datos del intervalo cuando insertamos, eliminamos o intersecamos.
Utilizamos la función *sscanf (se encuentra en la bibliografía (2))*, esta separa el string del intervalo en dos doubles que representan los extremos del intervalo.
También tuvimos dificultades en detectar y verificar cuando se ingresa correctamente un intervalo o un comando. *sscanf* también nos ayudó en este sentido porque la misma función devuelve la cantidad de información que se pudo extraer del string, en el caso de los intervalos, nosotros queremos extraer los dos doubles y si *sscanf* retornaba un valor diferente a 2, sabemos que el intervalo es incorrecto. Además definimos una función auxiliar que compare la forma del intervalo, es decir [a,b] y que no acepte lo que no corresponde a un intervalo, entonces utilizamos dentro de la función auxiliar una función que está contenida en la librería ***ctype.h*** llamada *isdigit (se encuentra en la bibliografía (3))* la cual nos verifica que en el string pasado como intervalo haya dígitos y no otros elementos.
De manera análoga se usa *sscanf* para separar el string inicial donde contiene un substring con el comando, y otro substring con el intervalo.

**BIBLIOGRAFÍA**


1) https://en.cppreference.com/w/c/numeric/math/fmax
2) https://en.cppreference.com/w/c/io/fscanf
3) https://www.programiz.com/c-programming/library-function/ctype.h/isdigit