SPLAY TREES: IMPLEMENTACIÓN EN C++

1. INTRODUCCIÓN

El objetivo de esta práctica es la implementación de la estructura de datos de conjuntos mediante un árbol biselado (Splay) y el estudio de los costes para las operaciones de inserción, búsqueda y borrado.

La implementación logra unos costes amortizados en O(log N) al utilizar unas operaciones de autoajuste de la altura del árbol, llamadas flotaciones (Splay). Estas flotaciones producen que el ultimo nodo accedido en las operaciones (llamado nodo origen de la flotación) ascienda hasta convertirse en la raíz del árbol principal, a través de una serie de operaciones descritas en la sección de diseño.

Estos arboles tienen la ventaja frente a los demás arboles autoajustables de que prioriza el tiempo de acceso a los elementos más accedidos.

2. DISEÑO

Lo especial de estos arboles es la flotación de los nodos. Esta se realiza a través de la selección del nodo origen y posteriormente de la aplicación de una rotación, elegida en función de la posición del nodo origen. Se aplica este mismo proceso un numero arbitrario de veces hasta que se posiciona el nodo origen en la raíz del árbol principal.

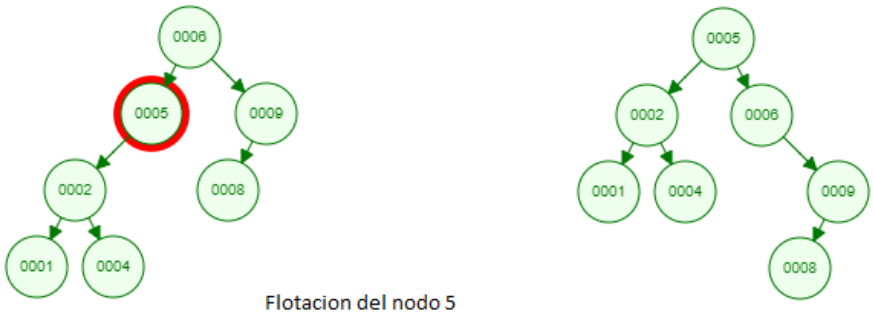
La elección del nodo origen cambia en función de la operación:

- En la inserción es el nodo insertado o el ultimo accedido
- En el borrado es el padre del borrado o el ultimo accedido
- En la búsqueda es el buscado o el ultimo accedido

Las rotaciones, distintas para los distintos casos en la posición de los nodos, se pueden descomponer en la composición de dos más simples: una rotación a la derecha y una a la izquierda iguales a las aplicadas en los casos de desequilibrio RR o LL en los arboles AVL.

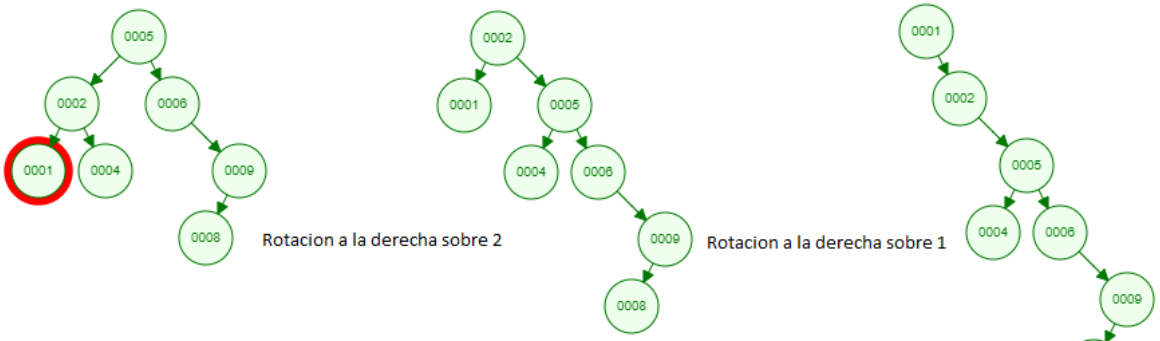
La elección de la rotación se elige en función de las siguientes consideraciones:

- Si el padre del nodo origen es la raíz del árbol, se realiza una rotación como las de los arboles AVL en los casos LL o RR. Si el nodo es el hijo izquierdo, rotación hacia la derecha (LL) y si es el derecho, rotación hacia la izquierda (RR), las dos sobre el padre del origen.



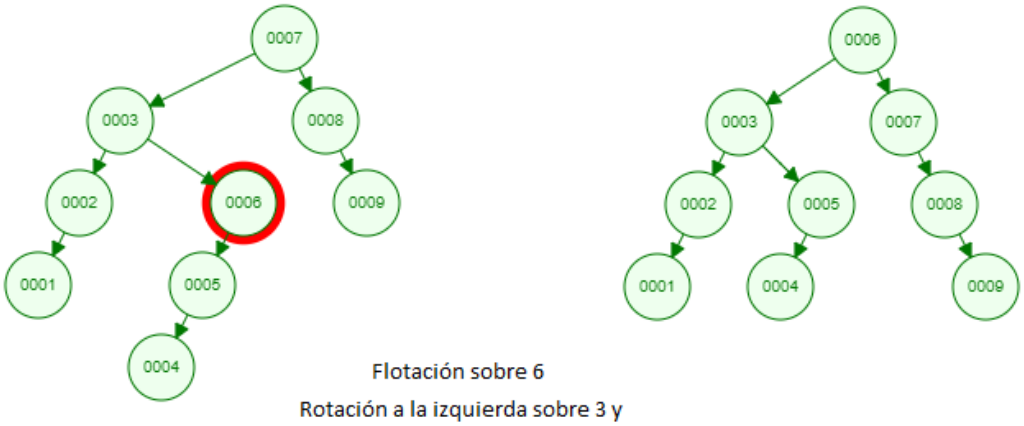
Flotacion del nodo 5
LL de AVL (rotacion a la derecha)

- Si el abuelo del nodo origen es el padre por la izquierda del padre del nodo, y este, tiene como hijo izquierdo al nodo origen, se aplican dos rotaciones a la derecha sobre el abuelo del origen.



Flotación del nodo 1

- Si el abuelo del nodo origen es el padre por la izquierda del padre del nodo, y este, tiene como hijo derecho al nodo origen, se aplica una rotación a la izquierda sobre el padre del origen y una a la derecha sobre el abuelo del origen.



Flotación sobre 6

Rotación a la izquierda sobre 3 y rotación a la derecha sobre 7

- Los demás casos son simetrías de las dos anteriores.

3. IMPLEMENTACIÓN

Para representar el árbol en C++ se ha utilizado un struct ***arbol***, que contiene la raíz y los subárboles izquierdo y derecho como punteros a otro árbol. La estructura del conjunto tiene un puntero al árbol principal.

Para las flotaciones es necesario conocer la posición del nodo origen a flotar, para ello se implementa el struct ***posicionOrigenFlotacion*** que almacena la distancia relativa entre el nodo en recorrido y el origen, y dos booleanos para indicar si el padre del origen es su padre por la izquierda y si el abuelo del origen es su abuelo por la izquierda. Esta estructura se utiliza en las operaciones que causan flotación al retornar las llamadas.

Una función ***flotarNodo*** recibe la estructura anterior y un puntero al árbol y aplica la rotación en cada caso.

4. GRÁFICAS Y GENERACIÓN DE PRUEBAS

Para generar casos de prueba voluminosos y probar los tiempos de las funciones se proporciona un programa que inserta números aleatorios en el árbol y, en cada intervalo de magnitud de elementos, se realiza una prueba de tiempos de cada instrucción. Esta prueba consiste en un numero de medidas de tiempos de ejecución y en una media de estas.

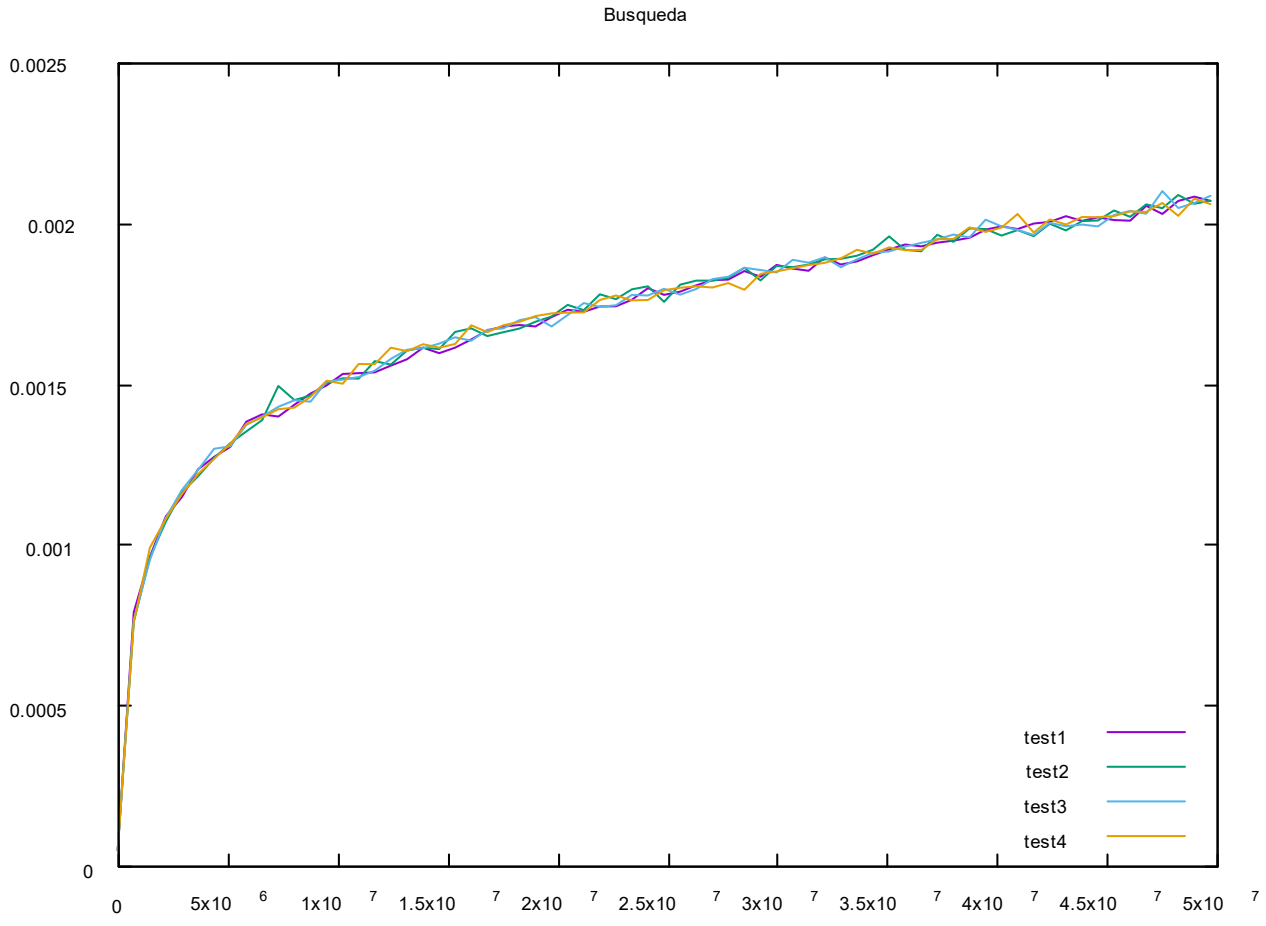
Al ejecutar el programa se le pueden pasar las siguientes opciones para alterar su funcionamiento:

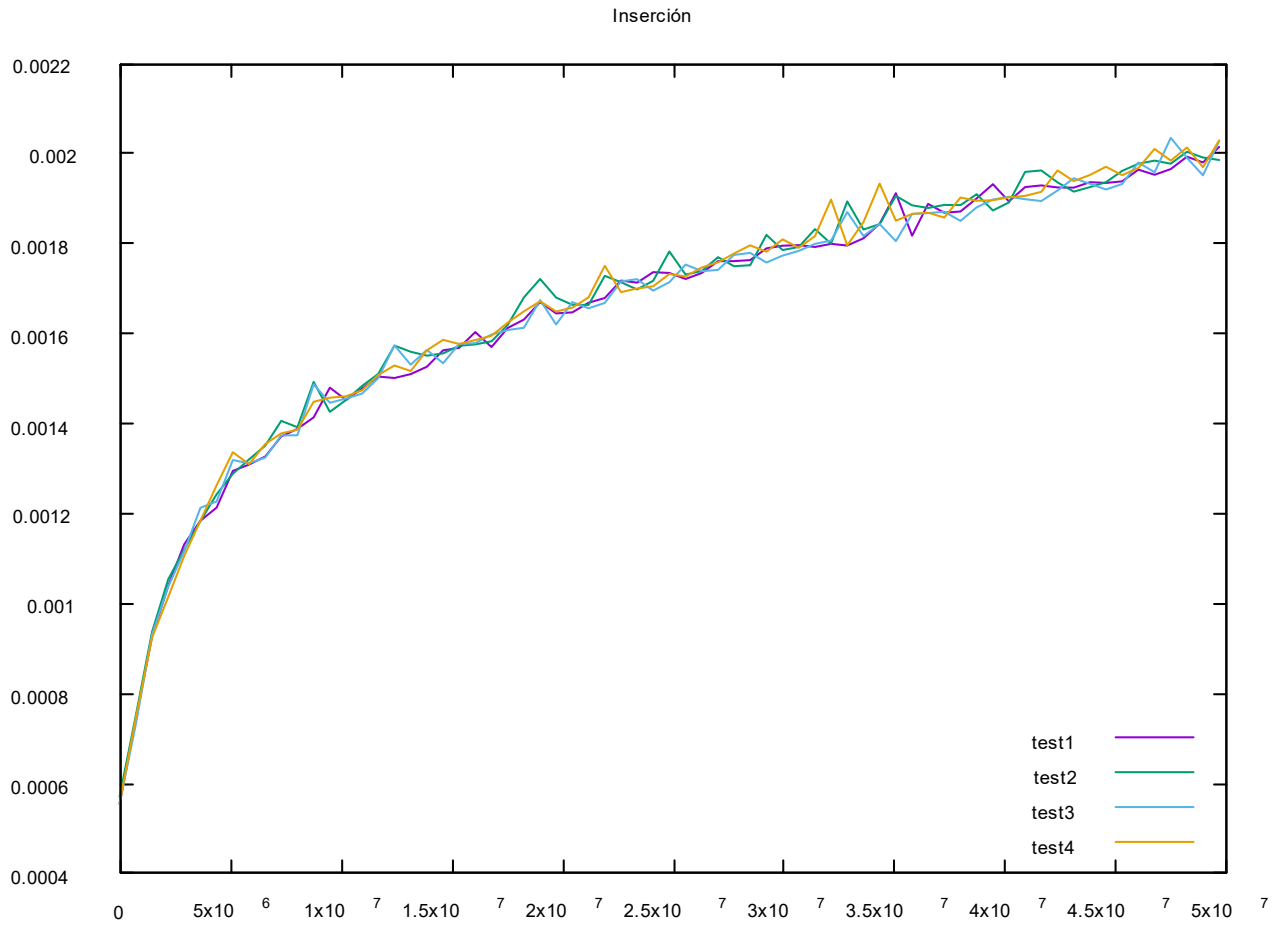
- -f : Nombre de archivo destino (por defecto default Test)
- -NE : Total de elementos insertados en el árbol, como condición de terminación del programa. (por defecto 20000000).
- -MIN y -MAX : Números para definir el rango en la generación de los números aleatorios (por defecto [0, 50000000]).

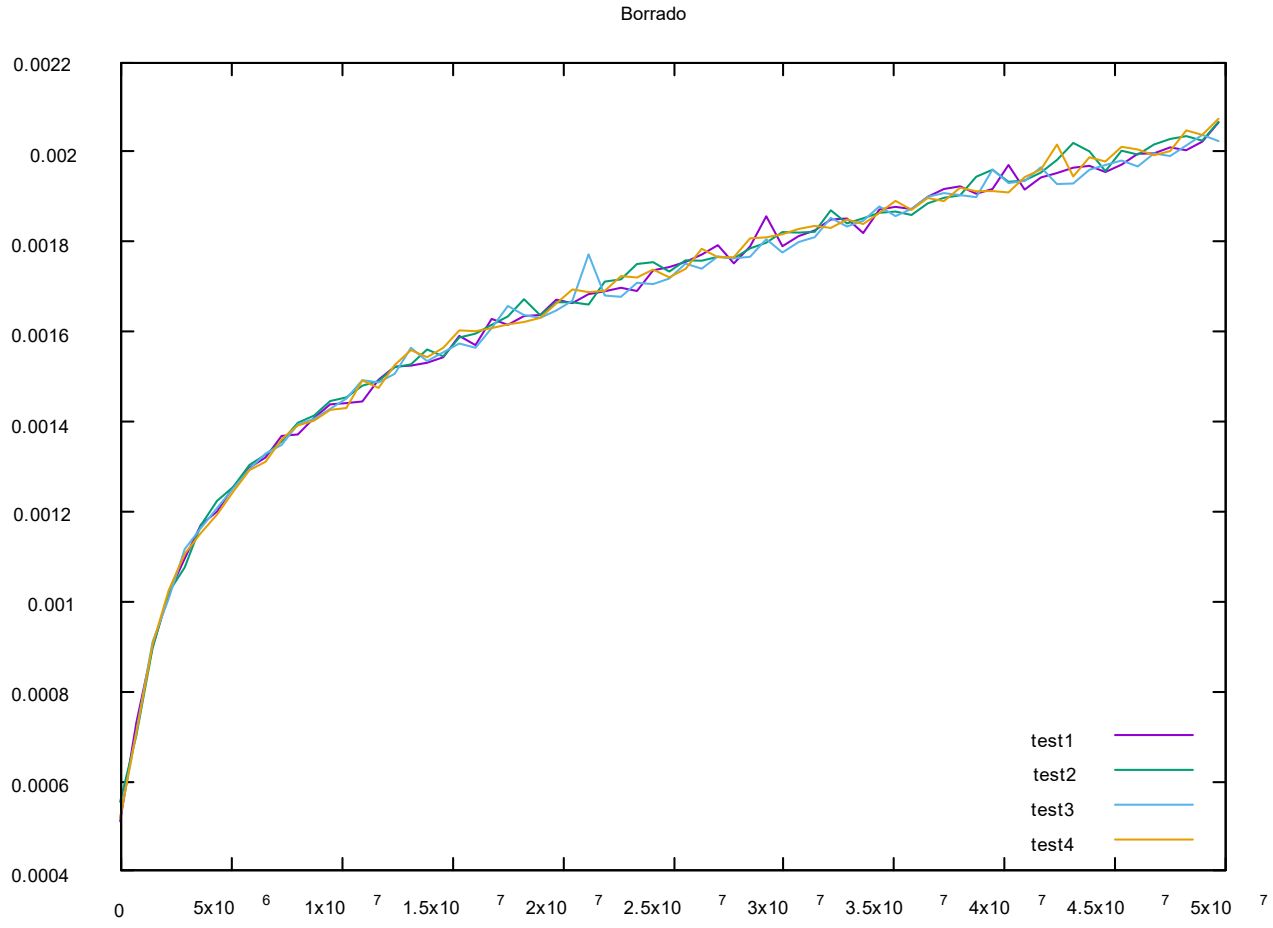
Graficas de tiempos:

Las graficas muestran un comportamiento logarítmico en cuanto al tiempo de ejecución, el coste es amortizado O(log N) (Aumenta 500 microsegundos al aumentar el tamaño en 25 millones de elementos).

Hay casos particulares en los que el coste no se amortiza, como por ejemplo al insertar una secuencia ordenada de números. Los elementos a pesar de las flotaciones quedan en forma de lista y los costes pasan a ser lineales.




Inserción
0.0022
0.002
0.0018
0.0016
0.0014
0.0012
0.001
0.0008
0.0006
0.0004
0
5x10 6
1x10 7
1.5x10 7
2x10 7
2.5x10 7
3x10 7
3.5x10 7
4x10 7
4.5x10 7
5x10 7
test1
test2
test3
test4


Borrado
0.0022
0.002
0.0018
0.0016
0.0014
0.0012
0.001
0.0008
0.0006
0.0004
0
5x10 6
1x10 7
1.5x10 7
2x10 7
2.5x10 7
3x10 7
3.5x10 7
4x10 7
4.5x10 7
5x10 7
test1
test2
test3
test4