Fully Retroactive Arbol Binario Busqueda*

Ingrid Sally Espinel Quispe
Universidad Nacional San Agustin
Peru
saingrid@gmail.com

1 IMPLEMENTACIÓN

Las estructuras de datos totalmente retroactivas son estructuras que permiten consultas y actualizaciones para ser realizado no solo en el presente sino también en el pasado. Proporcionaron una implementación de $O(\sqrt{m}\ \log m)$ cola de prioridad totalmente retroactiva.
La estructura del árbol se puede crear usando dos metodologías:

- Preservando la estructura:

  El árbol de búsqueda binaria se puede crear conservando la estructura. Al preservar la estructura, queremos decir que si un elemento 'x' se inserta en cualquier momento 'T' usando insertar_insertar (Insertar (x), t), la estructura es creado como si en realidad se creó el árbol de búsqueda binaria usando la operación de inserción (x) en el tiempo 't'. Entonces, 'x' debería ser presente en la posición exacta como si hubiera sido insertado en el momento 't' usando la operación de inserción (x) en lugar de operación retroactiva insertar_insertar (Insertar (x), t).

- Sin preservar la estructura:

  Otro método para crear el árbol de búsqueda binaria retroactivamente es sin preservar su estructura, es decir, si elemento se inserta en cualquier momento 't' utilizando retroactivo operación insertar_insertar (Insertar (x), t), puede ser insertado como si estuviera insertado en este momento, sin embargo el campo de tiempo del nodo insertado mantendrá el valor igual al tiempo 't', ya que en el árbol de búsqueda binaria no hay diferencia donde insertamos un valor si solo somos preocupado con su operación de "búsqueda".

La Implementación de la estructura se realize en c++ se utilize dos struct (nodoq , nodo), la clase principal cola y la function insert tree.
Dentro de la clase de principal tenemos:

Normal Operaciones Soporta:

Update operación:
1. insertar(x): insertar un elemento en árbol binario búsqueda

Operaciones retroactivas soporta:
Update operación:
1. insert_operacion (insert(x), t): insertar la insert(x) operación en tiempo ‘t’, i.e. inserta en un elemento x de un árbol búsqueda binaria en un tiempo ‘t’.
2. delete_operacion (t): elimina la insert(x) operación en un tiempo ‘t’, i.e. Elimina en elemento x de haver insertado en un tiempo t de un árbol búsqueda binaria.

Query operación
1. buscar_root (t): retorna el root de árbol búsqueda binario en un tiempo.
2. estructura (t): retorna mientras que la estructura árbol búsqueda binaria en un tiempo ‘t’.
3. Busqueda (x,t): buscar cualquier elemento ‘x’ es presente in the árbol binario búsqueda en un tiempo t.

2 Experimentos

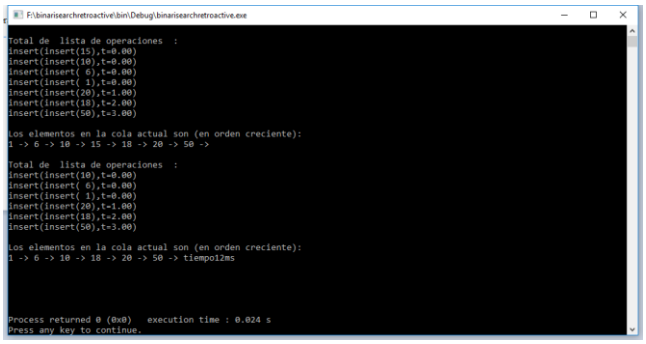
2.1 Hardware

Procesador: Intel® Core™ i7-6700 CPU @ 3.40 GHZ 3.41 GHZ

RAM 16 GB

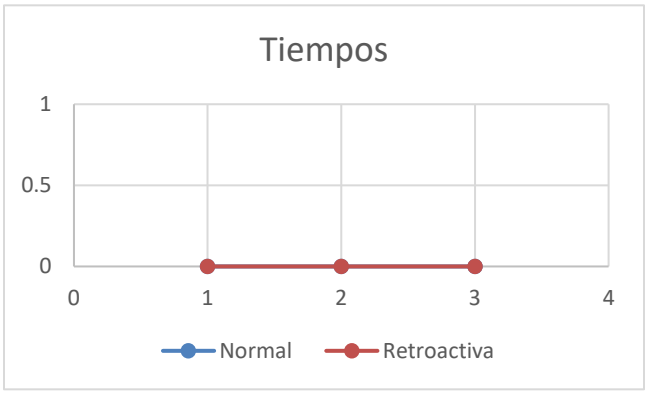
Sistema Operativo de 64 bits procesador

2.2 Pruebas



En la tabla siguiente se muestra se muestra una comparacion de estructura Normal y una retroactiva

	Normal	Retroactiva
Numero	Tiempo	Tiempo
1000	0 ms	0 ms
10000	0 ms	1257 ms
100000	2807 ms	146489 ms



Operaciones	Normal	retrocativa
insert_operation (Insert(x), t)	O(log n)	O(log n)
delete_operation (t)	O(log n)	O(log n)
find_root (t)	O(1)	O(1)
search (x,t)	O(log n)	O(log n)

3 CONCLUSIONES

- Se usa para evaluar el estado real de la estructura de datos si se realizó una operación incorrecta en el pasado.
- Esta estructura nos ayuda con la operación de 'retroceso' en el estado actual de la estructura de datos, para podernos respaldar todas las operaciones realizadas en la estructura de datos solo antes de esa instancia particular y luego de nuevo realizar el operaciones posteriores, sin esa falla operación. Pero, esto se considera como un tiempo ineficiente y solución de consumo.

REFERENCIAS

[1] E. D. Demaine, J. Iacono, and S. Langerman. Retroactive data structures. ACM Transactions on Algorithms (TALG), 3(2):13, 2007

[2] Fleischer, R. “A simple balanced search tree with O(1) worst-case update time”. Int. J. Found. Comput. Sci. 7, 2, 137–149, 1996.