Zapopan, Jalisco a 24 de octubre de 2016

Análisis y Diseño de Algoritmos

Juan Antonio Vega Fernández



Directorio

Santiago Ruiz Angulo 3118030

Oscar Dávila Ramírez 311XXXX

Sumario

[Tabla Hash 3](#_Toc464561662)

[Algoritmo Hash 3](#_Toc464561663)

[Arboles Balanceados 4](#_Toc464561664)

[Arboles AVL 4](#_Toc464561665)

[Análisis de Complejidad 5](#_Toc464561666)

[Inserción 5](#_Toc464561667)

[Eliminación 5](#_Toc464561668)

[Búsqueda 5](#_Toc464561669)

[Anexo 5](#_Toc464561670)

[Referencias 6](#_Toc464561671)

# Tabla Hash

Las tablas hash son estructuras de datos que se utilizan para almacenar un número elevado de datos sobre los que se necesitan operaciones de búsqueda e inserción muy eficientes. Una tabla hash almacena un conjunto de pares “(clave, valor/datos)”. La clave es única para cada elemento de la tabla y es el dato que se utiliza para buscar un determinado valor/datos.

El tiempo medio de recuperación de información es constante, es decir, no depende del tamaño de la tabla ni del número de elementos almacenados en la misma.

Una tabla hash se construye con tres elementos básicos:

* Un vector capaz de almacenar “m” elementos.
* Función de dispersión que permita a partir de los datos (llamados clave) obtener el índice donde estará el dato en el arreglo.
* Una función de resolución de colisiones:
  + Encadenamiento separado: Las colisiones se resuelven insertándolas en una lista. De esa forma tendríamos como estructura un vector de listas. Al número medio de claves por lista se le llama factor de carga y habría que intentar que esté próximo a 1.
  + Direccionamiento abierto: Utilizamos un vector como representación y cuando se produzca una colisión la resolvemos reasignándole otro valor hash a la clave hasta que encontremos un hueco.

## Algoritmo Hash

**Función de Hashing**: Es una función h: X -> Z que transforma una cadena binaria o mensaje de longitud arbitraria (x) en otra cadena binaria (z) de longitud constante (n) llamada digesto.

Para la transformación del ID se utiliza el algoritmo de hash *FNV (Fowler/Noll/Vo)*, la cual surgió de una idea presentada al comité del IEEE POSIX P1003.2 por Glenn Fowler y Phong Vo en 1991, luego Landon Curt Noll la mejoró. Es rápida, produce pocas colisiones, realiza una buena dispersión de los hashes y funciona muy bien para hacer hashing en cadenas casi idénticas*.*

# Arboles Balanceados

Es un árbol binario de búsqueda en el cual se cumple: “Para todo nodo T del árbol, la altura de los subárboles izquierdo y derecho no debe diferir en más de una unidad”.

Esto es solamente posible cuando el número de hojas es 2*k*  para *k* ∈ Z+, en que caso el largo de todos los caminos desde la raíz hasta las hojas es exactamente *k*.

La idea central de éstos es la de realizar reacomodos o balanceos, después de inserciones o eliminaciones de elementos.

Los árboles balanceados se parecen mucho, en su mecanismo de formación, a los números de Fibonacci. El árbol de altura 0n es vacío, el árbol de altura 1 tiene un único nodo y en general el número de nodos del árbol con altura h>1 se calcula aplicando la siguiente fórmula recursiva:

K = Numero mínimo de nodos de árbol

h = Altura

## Arboles AVL

La denominación de árbol AVL viene dada por los creadores de tal estructura (Adelson-Velskii y Landis).

La propiedad de equilibrio que debe cumplir un árbol para ser AVL asegura que la profundidad del árbol sea por lo que las operaciones sobre estas estructuras no deberán recorrer mucho para hallar el elemento deseado. El tiempo de ejecución de las operaciones sobre estos árboles es, a lo sumo en el peor caso, donde n es la cantidad de elementos del árbol.

Sin embargo, esta misma propiedad de equilibrio de los árboles AVL implica una dificultad a la hora de insertar o eliminar elementos: estas operaciones pueden no conservar dicha propiedad.

### Operaciones

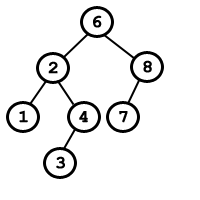


Figura 1. Árbol AVL de enteros

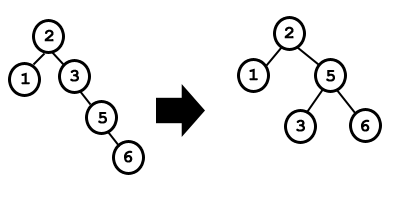


Figura 2. Rotación simple

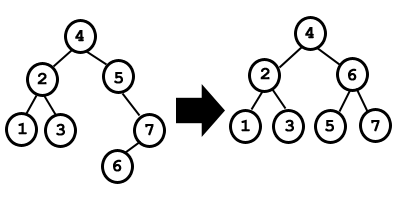


Figura 2. Rotación simple

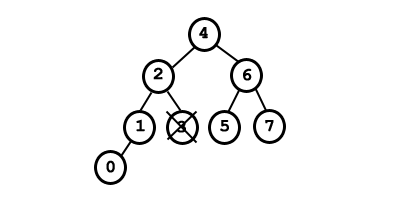
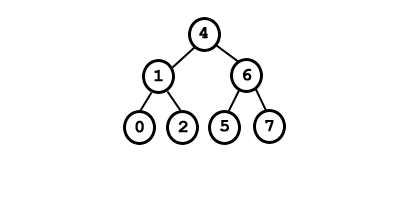
 

Figura 3. Borrado y Rotación

Las operaciones adicionales en un árbol AVL son análogas a las de árboles binarios de búsqueda.

# Análisis de Complejidad

## Inserción

## Eliminación

## Búsqueda

# Anexo

# Referencias

1. Pardo, A. (2011). *Tablas Hash*. Universidad Carlos III de Madrid. Recuperado el 16 de octubre de 2016, de <http://www.it.uc3m.es/abel/as/MMC/M2/HashTable_es.html>.
2. Benemérita Universidad Autónoma de Puebla. *Algoritmos de Búsqueda* [en línea]. [fecha de consulta: 16 de octubre de 2016]. Disponible en: <<http://www.cs.buap.mx/~iolmos/ada/TablasHashArbolesBinarios.pdf> >
3. Sama Villanueva, S. (s.f.). *Qué son las tablas hash.* Human Communication and Interaction. Recuperado el 16 de octubre de 2016, de <http://www.hci.uniovi.es/Products/DSTool/hash/hash-queSon.html>
4. CertiSur S.A. (s.f.). *Impacto De Recientes Ataques De Colisiones Contra Funciones De Hashing De Uso Corriente* [en línea]. [fecha de consulta: 16 de octubre de 2016]. Disponible en: <<https://www.certisur.com/sites/default/files/docs/ataques_funciones_hashing.pdf> >
5. Curt Noll, L. (s.f.). *FNV Hash*. Recuperado el 16 de octubre de 2016, de <http://www.isthe.com/chongo/tech/comp/fnv/index.html>
6. Á*rboles balanceados (AVL)* [en línea]. [fecha de consulta: 17 de octubre de 2016]. Disponible en: < <http://mapaches.itz.edu.mx/~mbarajas/edinf/avl.pdf> >
7. Universidad Autónoma de Nuevo León (s.f.). Á*rboles Balanceados* [en línea]. [fecha de consulta: 17 de octubre de 2016]. Disponible en: < <http://elisa.dyndns-web.com/teaching/aa/pdf/clase0210.pdf> >
8. Gurin, S. (2004). Á*rboles AVL* [en línea]. [fecha de consulta: 17 de octubre de 2016]. Disponible en: < <http://es.tldp.org/Tutoriales/doc-programacion-arboles-avl/avl-trees.pdf> >