

Reflexión Individual

Actividad 5.2

Santiago Álvarez Valdivia

A01640172

**Programación de Estructuras de Datos y
Algoritmos Fundamentales**

Tecnológico de Monterrey

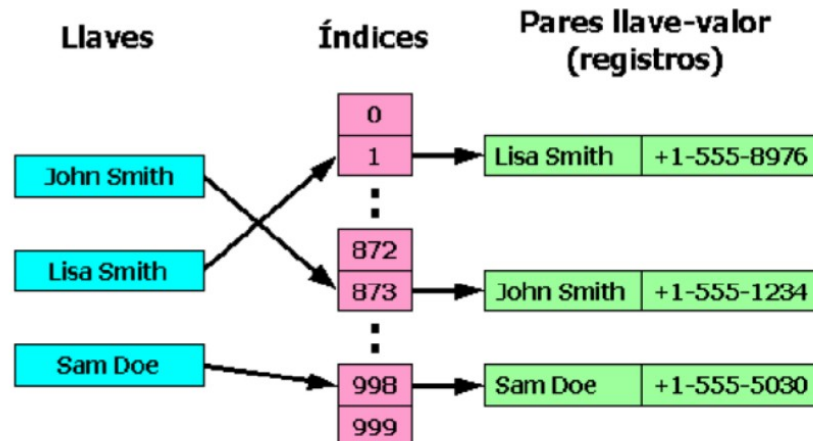
Campus Guadalajara

Escuela de Ingeniería y Ciencias

Profesor: Eduardo Arturo Rodríguez Tello

04 de diciembre de 2021

Introducción



Una tabla o mapa hash es una estructura de datos, que asocia llaves, o claves, con valores (o registros). Funciona al transformar la clave de un elemento en un *hash*, número que se utiliza para localizar el valor deseado. En el ejemplo superior, los registros son los recuadros verdes, que son los elementos como tal. Se están usando los nombres (parte de los registros) como llaves (recuadros azules). Las llaves se están usando para generar números (en rosa), que a su vez se usan como posición para guardar el elemento. Así para acceder a un registro completo, podemos usar la llave, convertirla en un hash de nuevo, y buscar en la posición especificada.

Al proceso que convierte una llave en un hash se le llama *hashing function* (función de hashing), y este proceso es específico del programa en cuestión.

La principal ventaja de una tabla hash es su velocidad. El tiempo de acceso promedio de un elemento es de $O(1)$ [1], y las tablas hash son especialmente eficientes cuando el número máximo de registros puede ser conocido de antemano.

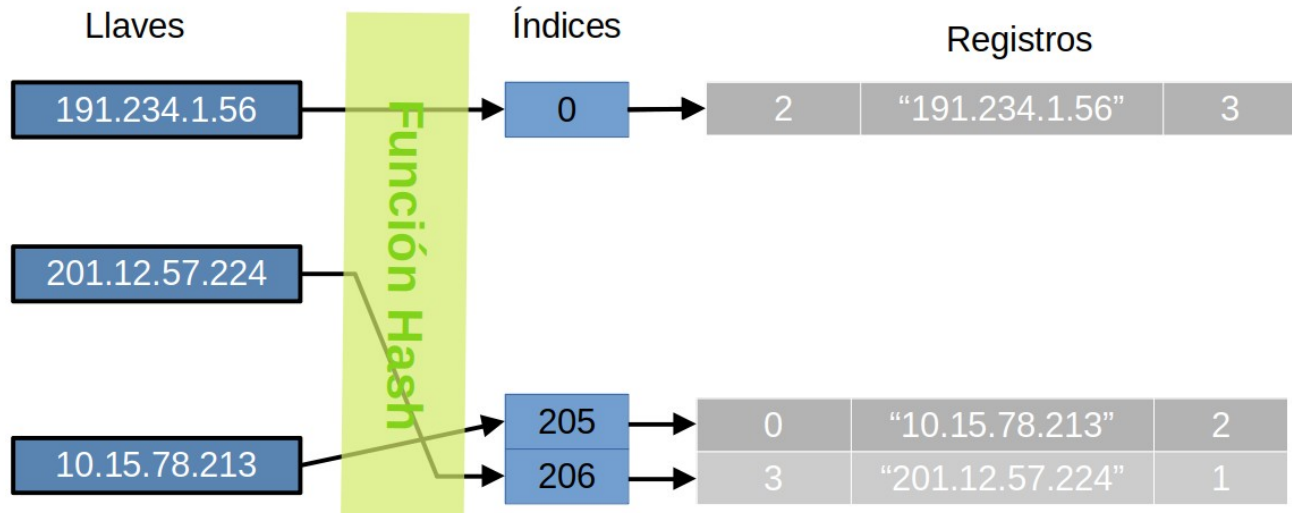
Descripción del programa adjunto

Para realizar este programa, se empleó la especificación dada por el profesor, que se resume en leer datos de un grafo dirigido, y almacenar el resumen de cada nodo como un registro en una tabla hash. Después se realiza una consulta, con un objeto de búsqueda definido por el usuario, para mostrar el resumen del nodo, y sus sucesores.

En este caso, los nodos corresponden a direcciones IP, y las relaciones están dadas por las interacciones entre distintas direcciones, especificadas también en el archivo que contiene los datos.

Tablas Hash en el programa adjunto

En este programa, la información que contiene la tabla hash es la siguiente



Donde los registros contienen el número de nodos predecesores, el número de nodos sucesores y la dirección IP.

En la introducción, se habló de que el tiempo de acceso medio de un elemento es de $O(1)$. Bien, esa es la complejidad al no haber colisiones. Se le llama colisión a lo que sucede cuando la dirección en la que va a ser guardado un elemento ya está ocupada. En este programa, la respuesta a esto es ir dando “saltos”, hasta encontrar una nueva posición. Una colisión, evidentemente aumenta el tiempo de ejecución del programa, al tener que dar pasos extra para almacenar y buscar valores.

Mientras mayor sea el número de elementos disponibles en la tabla en relación a los elementos a insertar, menos probables son las colisiones, y menor es el retraso debido a ellas.

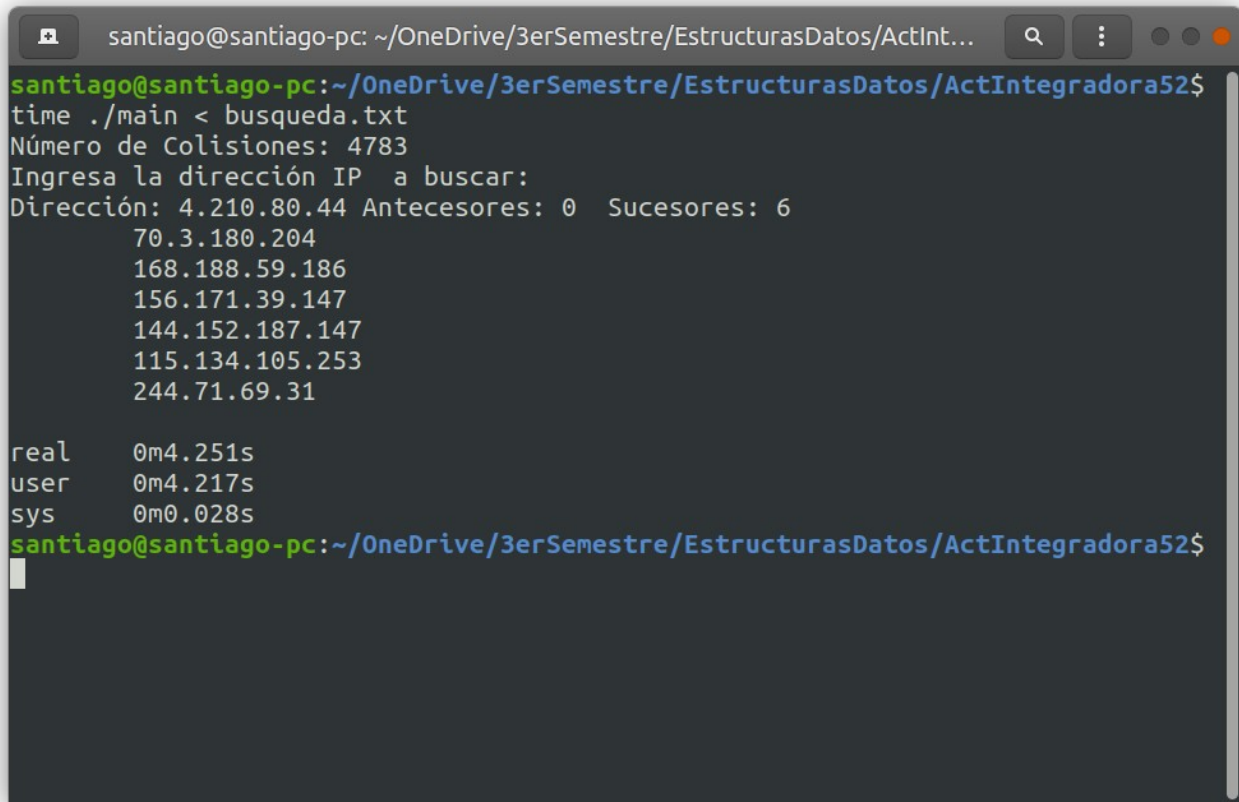
En este programa, si bien se emplearon tablas hash, se puede decir que no se aprovechó todo su potencial, debido a que sólo se realiza una búsqueda por cada ejecución, lo que no termina de mostrar las ventajas en tiempos de acceso de los datos.

Nota acerca de tiempo de ejecución

A pesar de lo deseado en el desarrollo del programa, el presente programa no es de ejecución instantánea, sino que tarda unos segundos (dependiendo del equipo) antes de preguntar al usuario por el elemento a buscar. Esto se debe, más que a problemas con alguna de las estructuras de datos, en el retraso que simboliza recorrer toda la lista de

adyacencias del grafo para encontrar a los antecesores de un nodo, para cada uno de los nodos.

Aquí mostramos una captura del tiempo de ejecución, para una ejecución completa.

A screenshot of a terminal window on a Linux system. The window title is 'santiago@santiago-pc: ~/OneDrive/3erSemestre/EstructurasDatos/ActInt...'. The prompt is 'santiago@santiago-pc:~/OneDrive/3erSemestre/EstructurasDatos/ActIntegradora52\$'. The user has run 'time ./main < busqueda.txt'. The output shows 'Número de Colisiones: 4783', a prompt 'Ingresa la dirección IP a buscar:', and a list of IP addresses: 'Dirección: 4.210.80.44 Antecesores: 0 Sucesores: 6', '70.3.180.204', '168.188.59.186', '156.171.39.147', '144.152.187.147', '115.134.105.253', and '244.71.69.31'. At the bottom, it shows timing statistics: 'real 0m4.251s', 'user 0m4.217s', and 'sys 0m0.028s'. The prompt is now 'santiago@santiago-pc:~/OneDrive/3erSemestre/EstructurasDatos/ActIntegradora52\$' with a cursor on a new line.

```
santiago@santiago-pc: ~/OneDrive/3erSemestre/EstructurasDatos/ActInt...
santiago@santiago-pc:~/OneDrive/3erSemestre/EstructurasDatos/ActIntegradora52$
time ./main < busqueda.txt
Número de Colisiones: 4783
Ingresa la dirección IP a buscar:
Dirección: 4.210.80.44 Antecesores: 0 Sucesores: 6
70.3.180.204
168.188.59.186
156.171.39.147
144.152.187.147
115.134.105.253
244.71.69.31

real    0m4.251s
user    0m4.217s
sys     0m0.028s
santiago@santiago-pc:~/OneDrive/3erSemestre/EstructurasDatos/ActIntegradora52$
```

Las condiciones en las que se realizó la prueba fueron las siguientes:

- Laptop personal
- Procesador intel Core i5-4300U
- 8 GB RAM
- Empleando compilación optimizada (O3), en el compilador g++ (versión 10.3.0)

Referencias

- [1] Data Structures Handbook. (2019, 17 septiembre). *Hash Tables*. Recuperado 30 de noviembre de 2021, de <https://www.thedshandbook.com/hash-tables/>
- [2] Universidad Don Bosco. (s. f.). *Tablas hash*. Recuperado 30 de noviembre de 2021, de https://www.udb.edu.sv/udb_files/recursos_guias/informatica-ingenieria/programacion-con-estructuras-de-datos/2020/i/guia-8.pdf