Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Laboratorio de Algoritmos y Estructuras II - CI2692

Implementación de tablas de Hash

Informe de laboratorio. Semana 8

Profesor: Guillermo Palma
Estudiantes:
Haydeé Castillo Borgo. Carnet: 16-10209
Jesús Prieto. Carnet: 19-10211

Trimestre Abril-Julio 2023

En el presente laboratorio se llevó a cabo un estudio experimental para comparar el tiempo de ejecución de dos implementaciones del TAD Diccionario (presentado en [1] y en el curso de teoría de Algoritmos y Estructuras II) generadas empleando dos clases de tablas de Hash: hash con encadenamiento y cuckoo hashing (desarrolladas en [2] y en el curso mencionado).

Un TAD Diccionario posee un conjunto de claves conocidas y una tabla que corresponde al conjunto de (*clave, valor*) que almacena, además de una serie de procedimientos. La implementación de dicha tabla es la que corresponde a las tablas de hash por encadenamiento y por cuckoo hashing, según sea el caso.

Asimismo, para el conjunto de claves conocidas se empleó un esquema similar al de las tablas de hash correspondientes: en el caso de la implementación por encadenamiento, se empleó un arreglo de listas circulares compuestas por nodos con claves enteras (dicha clase de lista fue presentada en el laboratorio anterior); y en el caso del cuckoo hashing se empleó un arreglo del doble del tamaño de las tablas de hashing, para así poder almacenar los datos de forma similar a los dos arreglos usados en dicho esquema de cuckoo hashing.

Para el presente estudio se empleó un computador Intel® Core™ i5-2450M CPU @ 2.50GHz × 4, con 8Gb de RAM y sistema operativo Ubuntu 20.04.6 LTS. Además, se empleó el lenguaje de programación Kotlin en su versión 1.8.21 y Java Virtual Machine JVM, versión 11.0.19.

En el estudio experimental se llevaron a cabo 5 pruebas en las cuales ambas implementaciones del TAD Diccionario debían evaluar el mismo arreglo de pares de tamaño n, de la siguiente manera:

1. Se generó un arreglo con números enteros aleatorios en el rango $[0, n/3]$
2. Con el arreglo anterior, se generó un arreglo de pares (*clave, valor*) donde la clave correspondía a un número entero y el valor al mismo número en forma de String
3. Para cada uno de los elementos del arreglo de pares se hizo lo siguiente: se buscó si el elemento pertenecía a la tabla de hash, en caso de que no entonces dicho elemento se agregaba, y en caso de que sí entonces se eliminaba.
4. Se tomó el tiempo de cada tabla de hash para procesar el arreglo de pares

Para dichas pruebas se consideró un tamaño $n = 5000000$ y los resultados promedios del tiempo de ejecución se presentan en la siguiente tabla y en el siguiente gráfico.

Tablas de Hash	**Tiempo promedio**
Hash con encadenamiento	$13{,}2552 \pm 1{,}0644$
Cuckoo hashing	$\mathbf{11{,}4334 \pm 0{,}6413}$

Tabla 1: Tiempos de ejecución (en segundos) de las tablas de Hash

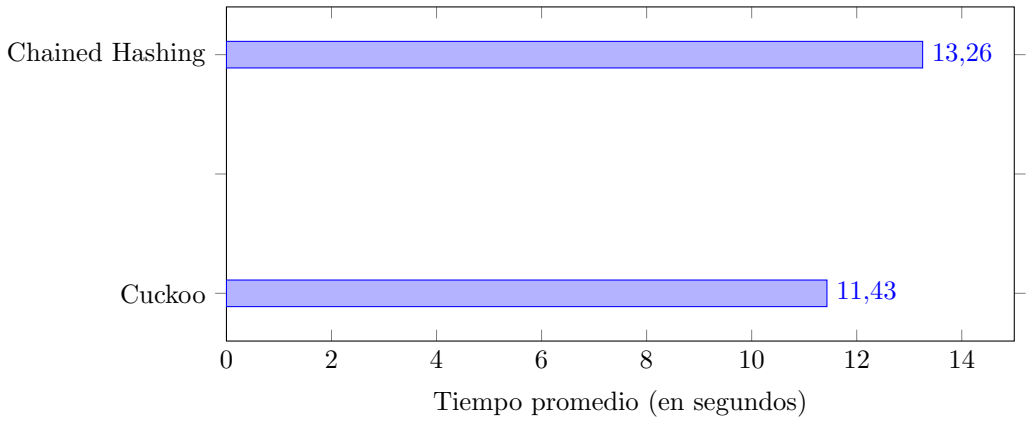

Tabla 2: Tiempos de ejecución de las tablas de Hash con 5 millones de elementos aleatorios

De los resultados anteriores se obtiene que la tabla de cuckoo hashing resultó ser más eficiente que la tabla de hash por encadenamiento.

En este caso, el cuckoo hashing produjó un mejor tiempo gracias a que en su diseño e implementación, la búsqueda de un elemento es más rápida que en el caso de una tabla de hash por encadenamiento, puesto que en el cuckoo hashing sólo es necesario verificar a lo mucho dos entradas de dos arreglos distintos, mientras que en el hash por encadenamiento es necesario buscar dentro de una lista doblemente enlazada en una posición determinada por la función de hash.

Además, en el caso del cuckoo hashing la eliminación también resulta más rápida que en la implementación del hash por encadenamiento, puesto que en este último se debe buscar el elemento en una lista doblemente enlazada en una posición determinada por la función de hash, mientras que en el cuckoo hasing sólo es necesario evaluar a lo mucho dos entradas en dos arreglos distintos.

Sin embargo, es importante mencionar que la diferencia en tiempo de ejecución entre ambas implementaciones es menor a los 5 segundos, por lo que no es muy considerable. Esto está relacionado con el hecho de que el procedimiento para agregar elementos de la tabla de hash por encadenamiento resulta más rápido que el del cuckoo hashing en el caso de presentarse colisiones, gracias al uso de listas doblemente enlazadas.

Referencias

[1] Ravelo, J. Especificación e implementación de tipos abstractos de datos. http: //ldc.usb.ve/ jravelo/docencia/algoritmos/material/tads.pdf, 2012.

[2] Cormen, T., Leirserson, C., Rivest, R., and Stein, C. *Introduction to Algorithms*, 3ra ed. McGraw Hill, 2009.