Ciencia & Desarrollo, Vol. 17, 23 (2) 69 - 73 (2018)
ISSN 2304-8891 Versión impresa
ISSN 2617-6033 Versión electrónica
Depósito legal N° 2017-16148

EVALUACIÓN DEL RENDIMIENTO EN OPERACIONES BÁSICAS DE TABLAS HASH FRENTE A LOS ARREGLOS EN ESTRUCTURA DE DATOS

PERFORMANCE EVALUATION IN BASIC OPERATIONS OF HASH TABLES VERSUS ARRAYS IN DATA STRUCTURES

[1]**Victor Hugo Vega Cueva**

RESUMEN

Los datos almacenados en estructuras de tipo tabla hash alcanzan un mejor rendimiento en operaciones básicas como inserción, actualización, eliminación y búsqueda de datos; puesto que su nivel de complejidad para realizar operaciones simples es de O(1) en la "notación O grande". Esto hace entender que el tiempo que esta estructura de datos utiliza para dar un resultado es constante; por otro lado, mientras el desempeño de los arreglos de datos tiene un valor de O(n), significa que dicho desempeño está fuertemente ligado a la cantidad de datos que almacene. En consecuencia, se puede afirmar que el uso de tablas hash en casos específicos presenta una amplia mejora en el rendimiento frente a los arreglos de datos.

Palabras clave: Arreglos de datos, estructura de datos, notación O grande, tablas hash.

ABSTRACT

The data stored in hash table structures reach a better performance in basic operations such as insertion, update, deletion and data search; since its level of complexity to perform simple operations is O (1) in the "O large notation". This makes us understand that the time that this data structure uses to give a result is constant; on the other hand, while the performance of the data fixes has a value of O (n), it means that said performance is strongly linked to the amount of data that it stores. Consequently, it can be said that the use of hash tables in specific cases presents a broad improvement in performance over data arrangements.

Keywords: Arrays, data structure, big O notation, hash tables.

[1]Bachiller en Ciencias con mención en Informática y Sistemas. Tacna- Perú. E-mail: victor.hugo.vega1@gmail.com

Presentado: 20/04/18 Aprobado: 15/12/2018

INTRODUCCIÓN

En el desarrollo de software, especialmente en la etapa de codificación, es muy importante saber elegir las estructuras de datos que se utilizarán en la codificación de un algoritmo, debido a que estas inciden directamente en el tiempo que el software emplea para dar una respuesta al usuario. Existen muchas maneras de resolver un problema en el desarrollo de software, una de ellas es el uso de los algoritmos. Algunos de estos algoritmos resuelven el problema de manera más fácil y rápida que otros, por cuanto es sumamente relevante el modo como estos se codifican, he allí la importancia de saber escoger una buena estructura de datos para resolver un problema específico. Hoy en día todos pueden resolver problemas, sin embargo, lo importante es determinar cómo estos se resuelven de la manera más óptima posible. En las ciencias de la computación se presentan muchos factores que influyen en el rendimiento del software, en este caso en particular, nos centraremos en el uso de estructuras de datos.

El presente trabajo de investigación busca analizar la diferencia en el rendimiento de las tablas hash frente a los arreglos de datos durante el desarrollo de operaciones básicas. Para ello se estudió el tiempo de respuesta a fin de determinar si existe alguna diferencia entre el uso de estas dos estructuras de datos. Por lo cual, se implementó un programa que permitirá evaluar el tiempo de respuesta en las operaciones ya antes mencionadas. El lenguaje de programación utilizado es c# y la librería que ayudó a medir el tiempo de respuesta en cada caso fue System.Diagnostics de Visual Studio.

JUSTIFICACIÓN

Esta investigación aporta lineamientos generales de alta importancia para medir la complejidad de algoritmos. Además, ofrece evidencias prácticas y comprobadas que demuestran las diferencias en el rendimiento entre las tablas hash y los arreglos de datos.

OBJETIVO

Objetivo general

El objetivo de la investigación es comparar el rendimiento de las tablas hash frente a los arreglos de datos, mediante la cuantificación del tiempo de respuesta en operaciones básicas en estas dos estructuras de datos.

Objetivo específico

Determinar el tiempo que se tarda en insertar, actualizar, eliminar y buscar un registro en una tabla hash, así como en un arreglo de datos.

BASES TEÓRICAS

Notación O grande

La notación O grande ayuda a determinar tanto el tiempo como la complejidad de los algoritmos. Usando la notación O grande, el tiempo tomado y el espacio requerido para correr un algoritmo pueden ser evaluados. Esta información resulta útil para establecer prerrequisitos de algoritmos, tanto como para desarrollar y diseñar aquellos que serían eficientes en términos de tiempo y de espacio.

La notación O grande ha sido extremadamente valiosa para clasificar algoritmos por su rendimiento. Desarrolladores usan esta notación con la finalidad de alcanzar la mejor solución para un problema dado. Por ejemplo, para el algoritmo de ordenamiento quick sort, en el peor escenario, la complejidad es de $O(n^2)$; mientras que para el algoritmo de ordenamiento burbuja, en un escenario normal, la complejidad sería O(n2). Entonces el algoritmo quick sort puede ser evaluado como un mejor algoritmo de ordenamiento para cualquier desarrollador que ha escogido entre estos dos. (Instructional Software Research and Developmente Group, 2006, p. 17).

Complejidades más comunes en los algoritmos

Un algoritmo con complejidad O(1) —también conocido como de tiempo constante— es aquel que toma una cantidad de tiempo que no crece con relación a N(el número de datos a procesar). Un algoritmo con complejidad O(N) —llamado asimismo de tiempo lineal— es donde, para una cantidad grande de N, la complejidad crecerá de forma lineal con respecto al valor de N. Un algoritmo $O(N^2)$ —denominado también de tiempo cuadrático— es uno donde, para una cantidad relativamente grande de N, la complejidad crecerá de manera cuadrática con respecto a la cantidad de N. En cuanto a los algoritmos de O(log n) y O(n log n), estos siguen un rendimiento similar en su función. Cabe señalar que los algoritmos con una complejidad exponencial no son adecuados para un uso práctico, como lo muestra la Figura 1 (Martelli, 2006, p. 476).

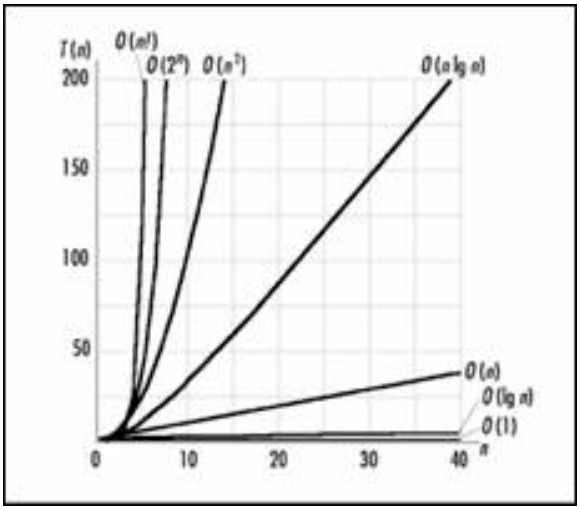

Figura 1. Descripción gráfica del crecimiento en los diferentes tipos de notación O grande
Fuente: Loudon (1999)

Tablas Hash

La tabla hash es una efectiva estructura de datos para implementar diccionarios. Aunque buscar un elemento en una tabla hash puede tomar el mismo tiempo que buscarlo en una lista enlazada O(n), esto último solo se haría en el peor de los casos. El tiempo promedio de búsqueda en las tablas hash es de O(1).

Una tabla hash involucra la generalización de un arreglo de datos ordinario. Direccionando de modo directo estos datos dentro de un arreglo, se consigue examinar de manera arbitraria la posición de dicho arreglo en un tiempo O(1). Este direccionamiento directo es aplicable cuando en un arreglo se puede alojar una posición para cada posible clave de la tabla hash.

Direccionamiento directo

El direccionamiento directo es una simple técnica que trabaja bien cuando el universo de claves es razonablemente pequeño. Supongamos que una aplicación necesita una cantidad de datos, se usan las claves dadas por un universo de U= {0, 1, 2 ,3 ,4… m-1}, donde m no es muy grande. Asumiendo que las claves no se repetirán, podemos representar nuestra tabla hash como T[0, 1, 2, 3, 4… m-1], en donde cada posición le corresponde a una clave en el universo U.
Las operaciones básicas son fáciles de implementar:

Tabla-hash-buscar(T,k)
Return T[k]
Tabla-hash-insertar(T,x)
T[key[x]] ← X
Tabla-hash-eliminar(T,x)
T[key[x]] ← nulo

Para cada una de estas operaciones el tiempo requerido es corto O(1) (Cormen, Leiserson, Rivest & Stein, 2003, p. 222).

Arreglo de Datos

Un array o arreglo de datos es un conjunto finito y ordenado de elementos. La propiedad "ordenado" significa que el elemento primero, segundo, tercero…. enésimo de un arreglo consigue ser identificado. Los elementos de un array son homogéneos, es decir, se trata del mismo tipo de datos. Un array puede tener todos sus elementos de tipo cadena, como el ejemplo de la Figura 2, otro puede tenerlos de tipo entero, etc. Los arreglos de datos se conocen también como matrices en matemáticas, y tablas en cálculos financieros. (Aguilar, 2008, p. 248).

	ALUMNOS
1	Luis Francisco
2	Jose
3	Victoria
	.
i	Martin
	.
30	Graciela

Figura 2. Representación gráfica de un arreglo de datos de nombres
Fuente: Aguilar (2008)

MATERIALES Y MÉTODOS

Caracterización o tipo del diseño de investigación

El diseño de la presente investigación es de tipo descriptivo comparativo.

Población y Muestra

La población está conformada por un total de 9 millones de datos generados, que podría representar los códigos de los ciudadanos de una ciudad relativamente grande como, por ejemplo, Lima. De esta población, se escogió una muestra que se representa con la letra (n); la cual fue calculada de la siguiente manera:

Fórmula
Los datos a considerar son los siguientes:

$$n=\frac{NxZ^2xpxq}{e^2x(N-1)+Z^2xpxq}$$

N= tamaño de la población.
p= probabilidad a favor.
q= probabilidad en contra.
Z= nivel de confianza de 95%.
e= 5%

$$n=\frac{9000000x1.96^2x0.5x0.5}{0.05^2x(9000000-1)+1.96^2x0.5x0.5}$$

$$n=385$$

El resultado del cálculo de la muestra es 385 registros; sin embargo, por tratarse de una población teóricamente grande lo más recomendable es elegir una muestra de 10000 registros, debido a que, al tratarse de un estudio descriptivo para poblaciones grandes, resulta mejor usar una muestra mayor a fin de incrementar el nivel de confianza de los resultados.

Tratamiento de datos

Para el tratamiento de los datos se utilizó la estadística descriptiva. Se empleó como herramienta el software Visual Studio 2017 y Microsoft Excel para graficar los resultados.

Asimismo, fue desarrollada una aplicación de consola que permitió generar la simulación de los códigos y los nombres de los ciudadanos, y medir de esta forma el tiempo de respuesta, dato al cual finalmente queríamos llegar.

Limitaciones

No se presentaron mayores limitaciones en el presente trabajo. Esto en razón de que el desarrollo e implantación de la aplicación que nos permitió generar los datos y evaluar el rendimiento de las tablas hash, así como los arreglos de datos, fueron realizados en una computadora personal con características básicas.

RESULTADOS

Muestra 1

Con respecto a la primera tarea de inserción de datos, en la Figura 3 se observa que el tiempo al ingresar los 10000 registros en un arreglo de datos es de entre 0 y 1 milisegundo por dato; en tanto que el programa demoró en ingresar la misma cantidad de registros en un tiempo de 0 milisegundos por registro para una tabla hash.

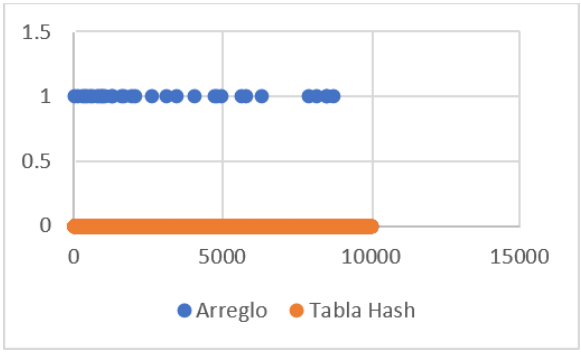

Figura 3. Representación gráfica de los resultados en la inserción de 10000 registros

Para la segunda tarea de búsqueda de datos, en la Figura 4 puede observarse que el tiempo empleado en buscar los 10000 registros en un arreglo de datos varía entre 0 y 300 milisegundos por dato; mientras que el programa demoró en buscar la misma cantidad de registros un tiempo de 0 milisegundos por registro para una tabla hash.

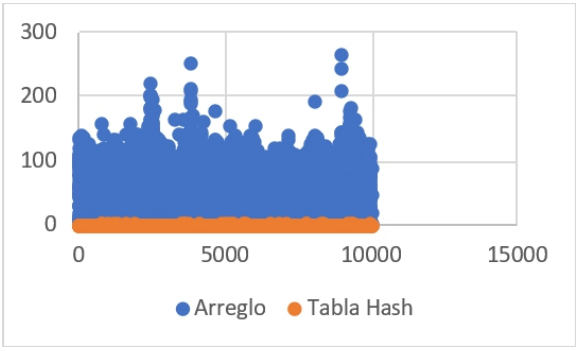

Figura 4. Representación gráfica de los resultados en la búsqueda de 10000 registros

En cuanto a la tercera tarea de actualización de datos, se observa en la Figura 5 que el tiempo en actualizar los 10000 registros en un arreglo de datos varía entre 0 y 10000 milisegundos por dato; por otro lado, el programa demoró en actualizar la misma cantidad de registros 0 milisegundos por registro para una tabla hash.

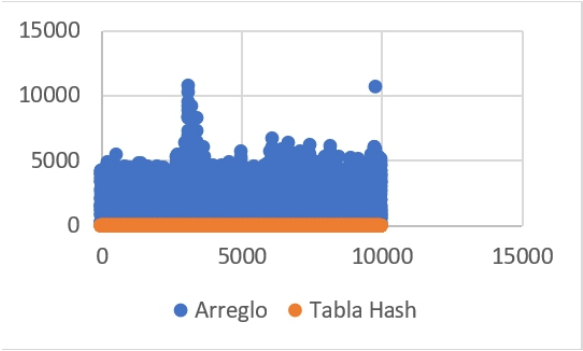

Figura 5. Representación gráfica de los resultados en la actualización de 10000 registros.

En relación con la cuarta tarea de eliminación de datos, en la Figura 6 se muestra que el tiempo en eliminar los 10000 registros en un arreglo de datos varía entre 0 y 10000 milisegundos por dato; mientras que el programa demoró en eliminar la misma cantidad de registros 0 milisegundos por registro para una tabla hash.

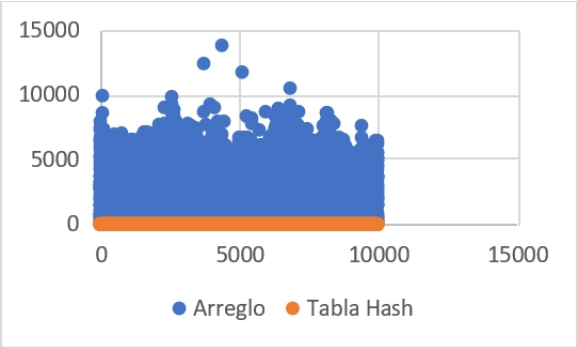

Figura 6. Representación gráfica de los resultados en la eliminación de 10000 registros

En la siguiente tabla se grafica los valores de las medias de los diez mil registros en cada operación. En ella se ve que los tiempos de inserción, actualización, modificación y eliminación en el caso de las tablas hash es constante e igual a cero; en tanto que, en el caso de los arreglos de datos, el valor de búsqueda, actualización y eliminación tiene una media de 54 milisegundos.

Tabla 1. *Resumen de las medias calculadas para cada operación de las 10000 operaciones en los arreglos de datos y en las tablas hash.*

	Arreglo de Datos	Tabla Hash
Inserción	0	0
Búsqueda	54	0
Actualización	54	0
Eliminación	55	0

DISCUSIÓN DE RESULTADOS

A fin de obtener los datos, se elaboró un software en el lenguaje de programación c#; el cual realizaba las operaciones de inserción, búsqueda, actualización y eliminación; y, simultáneamente, obtenía el tiempo que cada registro demoraba en culminar su operación. Dichos datos fueron transcritos luego en una hoja de texto. Posteriormente, estas hojas de texto se tabularon en hojas de cálculo de Microsoft Excel con la finalidad de generar después los gráficos que nos ayudaron a ver los resultados.

Como pudimos observar en las operaciones de inserción y eliminación, no hubo mayor diferencia entre el uso de arreglos de datos y tablas hash; lo que no sucedió del mismo modo en las operaciones de búsqueda y actualización, donde el tiempo de respuesta que utilizaron las tablas hash fueron ampliamente mejores a los tiempos que dieron los arreglos en estas dos operaciones.

CONCLUSIONES

Los resultados de esta investigación demuestran que el rendimiento de las tablas hash es mucho más favorable que el de los arreglos de datos, puesto que las primeras tuvieron un tiempo de respuesta O(1) en la notación O grande para la realización de sus operaciones básicas; en cambio, los arreglos de datos tuvieron un tiempo de respuesta mayor de O(n) en las operaciones de búsqueda y actualización, así como O(1) en inserción y eliminación.

En consecuencia, se puede asegurar que las tablas hash tienen un mejor desempeño en operaciones con ciertos tipos de datos; mas no se tiene igual seguridad de que estas tengan el mismo nivel de rendimiento en todos los aspectos frente a los arreglos de datos.

REFERENCIAS BIBLIOGRÁFICAS

Aguilar, L. J. (2008). Fundamentos de Programación. Madrid: McGRAW-HILL.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2003). Introduction to algorithms. London: McGraw-Hill.

Instructional Software Research and Development Group. (2006). Data Structures Using C. Dew Delhi: Tata McGraw-Hill.

Loudon, K. (1999). Algorithms With C. California: O'REILLY.

Martelli, A. (2006). Python in a Nutshell. California: O'REILLY.