

**TERCERA CONSULTA – BÚSQUEDAS EXTERNAS**

**Presentado a:**

Julio Cesar Florez Baez

**Presentado por:**

Johan Esteban Castaño Martinez - 20191020029

Jhony Alejandro Caro Umbariba - 20191020055

Samuel Andrés Romero Bueno - 20191020127

**Equipo Número 1**

Facultad de Ingeniería.

Ciencias de la Computación II.

25 de septiembre de 2022.

**INDICE**

[1. Búsquedas externas: 3](#_Toc115011955)

[2. Metodos de búsqueda: 3](#_Toc115011955)

[2.1. Método secuencial: 3](#_Toc115011955)

[2.2. Método binario: 5](#_Toc115011955)

[2.3. Funciones Hash: 7](#_Toc115011955)

[2.3.1. Definición Función Hash: 7](#_Toc115011955)

2.3.2. [Hash Mod: 9](#_Toc115011955)

2.3.3. [Hash Método Cuadrado: 11](#_Toc115011955)

2.3.4. [Hash por Truncamiento: 12](#_Toc115011955)

[2.3.5. Hash por Plegamiento: 14](#_Toc115011955)

2.3.6. [Hash por Conversión de Bases: 15](#_Toc115011955)

[3. Búsquedas dinámicas: 16](#_Toc115011955)

[4. Tipos de Búsquedas Dinámicas: 17](#_Toc115011955)

4.[1. Expanción total: 17](#_Toc115011955)

4.[2. Reducción total: 19](#_Toc115011955)

4.[3. Expanción parcial: 20](#_Toc115011955)

4.[4. Reducción parcial: 21](#_Toc115011955)

1. **Búsquedas Externas:**
   1. Primera definición

Existen casos en los cuales no se pude manejar toda la información en la memoria principal, sino que es necesario trabajar con información almacenada en archivos. Este tipo de búsqueda se denomina búsqueda externa.

Los archivos se usan normalmente cuando el volumen de datos es significativo o cuando la aplicación exige permanencia de los datos, aun después de que esta se termine de ejecutar. Como los archivos se encuentran almacenados en dispositivos periféricos -cintas, discos, etc.-, las operaciones de escritura y lectura de datos tienen un alto costo en cuanto a tiempo, por los accesos a estos periféricos. Para disminuir tiempo de acceso es muy importante optimizar las operaciones de búsqueda, inserción y eliminación en archivos. Una forma de hacerlo es trabajar con archivos ordenados.[[1]](#footnote-1)

* 1. Segunda definición:

Para ordenar secuencias grandes de elementos, que posiblemente no pueden almacenarse en memoria interna, se aplican los algoritmos de ordenación externa. La ordenación externa está ligada con los archivos y los dispositivos en que se encuentran, al leer el archivo para realizar la ordenación el tiempo de lectura de los registros es notablemente mayor que el tiempo que se tarda en realizar las operaciones de ordenación.[[2]](#footnote-2)

* 1. Tercera definición:

1. **Métodos de búsqueda:**
   1. **Método Secuencial:**
      1. Primera definición:

Los archivos secuenciales son aquellos cuyos componentes o registros ocupan posiciones relativas consecutivas. Todo componente o registro de un archivo tiene generalmente un campo que lo identifica, llamado campo clave. Este se encuentra formado por un conjunto de caracteres de dígitos. Además, ocupa la misma posición relativa en todos los registros de un mismo archivo. Algunos ejemplos de campo clave son el número de cliente -archivo de clientes-, el numero de contribuyente -archivo de hacienda- la matricula de un alumno -archivo de alumnos-, el numero de empleado -archivo de empleados-, etc. Puede suceder que la clave de un registro este formada por mas de un campo. Por ejemplo, en un sistema de inventarios cada pieza se podría identificar por un campo que haga referencia al departamento al cual pertenece, y otro campo para la pieza en sí.

El método de búsqueda secuencial consiste en recorrer el archivo comparando la clave buscada con la clave del registro en curso. El recorrido lineal del archivo termina cuando se encuentra el elemento, o cuando se alcanza el final del archivo. Se pueden presentar algunas variantes dentro de este método, dependiendo sobre todo de si el archivo esta ordenado o desordenado.[[3]](#footnote-3)

Ejemplo:

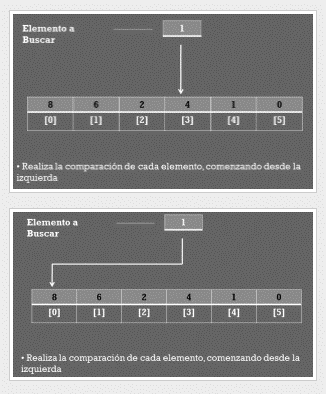
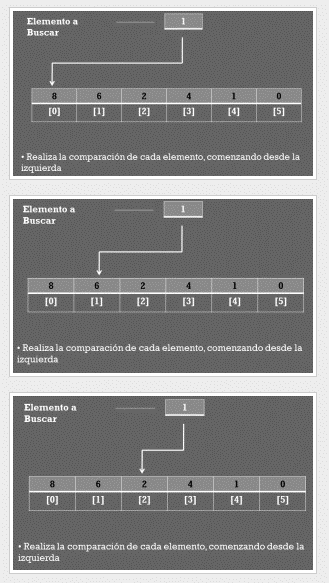


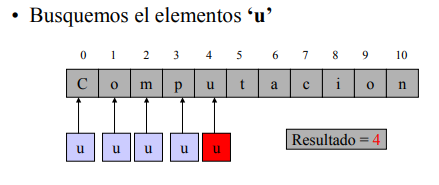
Imagen 1. Ejemplo del método secuencial tomado de: “https://joomat.wordpress.com/2012/06/20/algoritmos-de-busqueda/”

* + 1. Segunda definición:

El acceso secuencial o búsqueda secuencialmente implica el acceso a un archivo según el orden de almacenamiento de sus registros, uno tras otro. Es decir, en el orden en que fueron almacenados en el dispositivo.[[4]](#footnote-4)

Ejemplo:

Suponiendo que las letras observadas a continuación sean las claves de almacenamiento de archivos en el disco del computador, la búsqueda secuencial del archivo “u” estaría dada de la siguiente manera:



* + 1. Tercera definición:
  1. **Método Binario:**
     1. Primera definición:

El principio que rige el método de búsqueda binaria en la búsqueda externa es el mismo que en búsqueda binaria interna. El archivo debe estar ordenado y se debe conocer su numero de elementos (N) para aplicar este método.

Cabe destacar que un gran inconveniente de la búsqueda binaria externa es que requiere accesos a diferentes posiciones de dispositivo periférico en el cual esta almacenado el archivo; ello produce un alto costo en tiempo de acceso, que hace muy impráctica esta búsqueda.[[5]](#footnote-5)

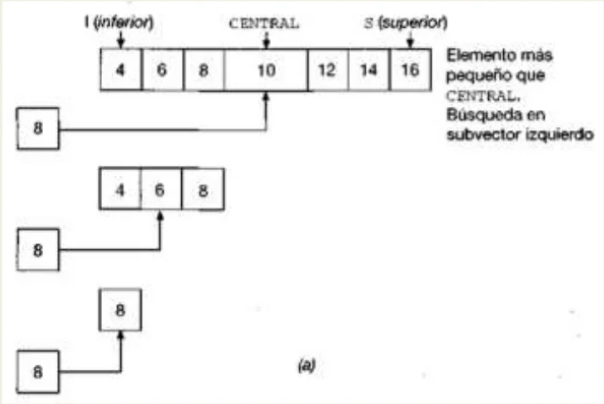


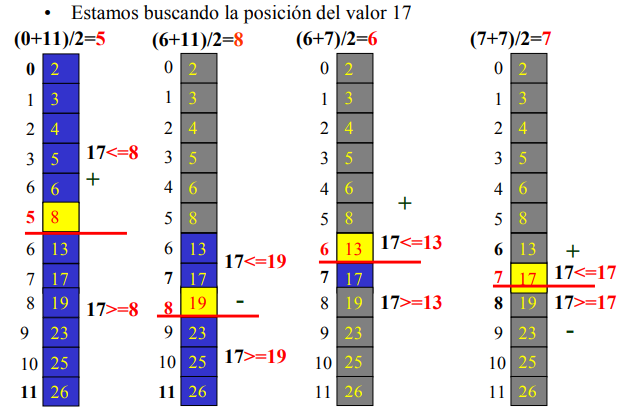
Imagen 2. Ejemplo del método binario sacado de:

<https://www.slideshare.net/AlvaroRuano1/bsqueda-secuencial-y-binaria>

* + 1. Segunda definición:

El acceso directo implica el acceso a un registro determinado, sin que ello implique la consulta de los registros precedentes. Este tipo de acceso sólo es posible con soportes direccionales.[[6]](#footnote-6)

Ejemplo:



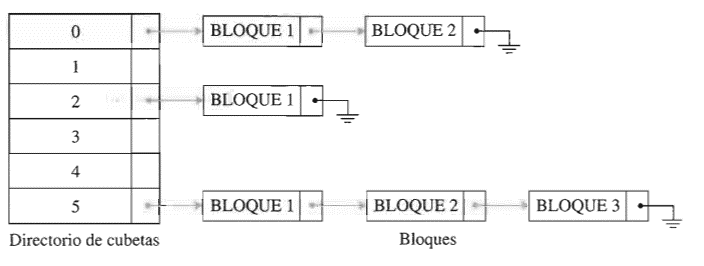
* + 1. Tercera definición:
  1. **Funciones Hash:**
     1. **Definición Función Hash**
        1. Primera definición:

Los archivos se encuentran normalmente organizados en áreas llamadas cubetas. Estas se encuentran formadas por cero, uno o más bloques de registros. Por lo tanto, la función hash aplicada a una clave, dará como resultado un valor que hace referencia a una cubeta en la cual se puede encontrar el registro buscado.

Tal como se mencionó en búsqueda interna, la elección adecuada de una función hash y de un método para resolver colisiones es fundamental para lograr mayor eficiencia en la búsqueda.

Los bloques contienen un número fijo de registros. Con respecto a las cubetas, no se establece un límite en cuanto al número de bloques que pueden almacenar. Esta característica permite solucionar, al menos parcialmente el problema de colisiones. Sin embargo, si el tamaño de las cubetas crece considerablemente, se perderán las ventajas propias de este método. Es decir, si el número de bloques que se debe recorrer en una cubeta es grande, el tiempo necesario para ello será significativo; por lo tanto, ya no se contara con la ventaja del acceso directo que caracteriza al método por transformación de claves.[[7]](#footnote-7)

Ejemplo:



Una función hash se puede definir como una transformación de clave a una dirección. Al aplicar una función hash a una clave se obtiene el número de cubeta en el cual se puede obtener el registro con dicha clave.

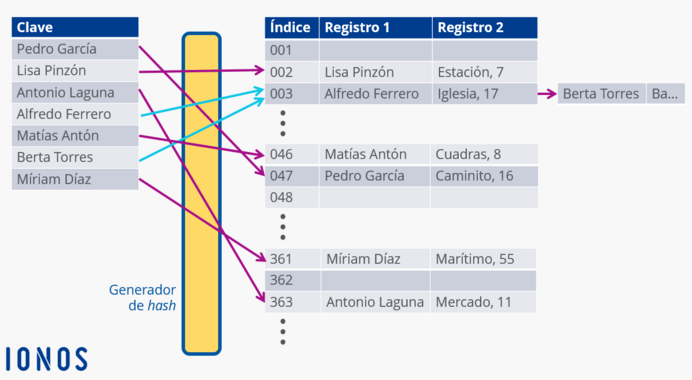
La función debe transformar las claves para que la dirección resultante sea un número comprendido entre los posibles valores de las cubetas. Por ejemplo, si se tienen 10 000 cubetas numeradas de 0 a 9 999, las direcciones producidas por la función deben ser valores comprendidos entre O y 9 999. Si las claves fueran alfabéticas o alfanuméricas, primero deberán convertirse en numéricas, tratando de no perder información, para luego ser transformadas en una dirección. Es importante que la función distribuya homogéneamente las claves entre los números de cubetas disponibles.

Las funciones módulo, cuadrado, plegamiento y truncamiento presentadas anteriormente para búsqueda interna son válidas también para búsqueda externa. Otra función que se puede utilizar para el cálculo de direcciones es la de conversión de bases, aunque no proporciona mayor homogeneidad en la distribución. De todas, la función módulo es, sin embargo, la que ofrece mayor uniformidad.[[8]](#footnote-8)

* + - 1. Segunda definición:

el método consiste en asignar el índice a cada elemento mediante una transformación del elemento, esto se hace mediante una función de conversión llamada función hash. hay diferentes funciones para transformar el elemento y el número obtenido es el índice del elemento. la función de hash ideal debería ser biyectiva, esto es, que a cada elemento le corresponda un índice, y que a cada índice le corresponda un elemento, pero no siempre es fácil encontrar esa función, e incluso a veces es inútil, ya que puedes no saber el número de elementos a almacenar. la función de hash depende de cada problema y de cada finalidad, y se pueden utilizar con números o cadenas, pero las más utilizadas son: restas sucesivas, aritmética modular, entre otras.[[9]](#footnote-9)

Ejemplo:



* + - 1. Tercera definición:
    1. **Hash Mod:**
       1. Primera definición:

La función hash por módulo o división consiste en tomar el residuo de la división de la clave entre el número de componentes del arreglo. Supongamos, por ejemplo, que se tiene un arreglo de N elementos, y K es la clave del dato a buscar. La función hash queda definida por la siguiente fórmula:

En la fórmula se observa que al residuo de la división se le suma 1, esto con el fin de obtener un valor comprendido entre 1 y N.

Para lograr mayor uniformidad en la distribución, es importante que N sea un número primo o divisible entre muy pocos números. Por lo tanto, si N no es un número primo se debe considerar el valor primo más cercano.

El siguiente ejemplo presenta un caso de función hash por módulo:

Supongamos que N = 100 es el tamaño del arreglo, y las direcciones que se deben asignar a los elementos (al guardarlos o recuperarlos) son los números comprendidos entre 1 y 100. Sean K1 = 7 259 y K2 = 9 359 son las dos claves a las que se deben asignar posiciones en el arreglo. Se aplica la fórmula para calcular las direcciones correspondientes a K1 y K2.

Como H(K1) es igual a H(K2) y K1 es distinto de K2, se está ante una colisión que se debe resolver porque a los dos elementos le corresponderá la misma dirección. Observemos, sin embargo, que, si aplicamos la fórmula con un número primo cercano a N, el resultado cambiario:

Con N = 97 se ha eliminado la colisión.[[10]](#footnote-10)

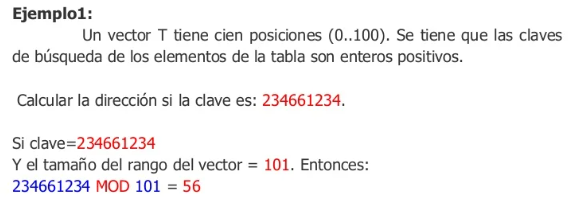
* + - 1. Segunda definición:

Convertir la clave a un entero, dividir por el tamaño del rango del índice y tomar el resto como resultado. La función de conversión utilizada es

MOD (Modulo o resta de división entera). Donde el mes el tamaño del arreglo con índices de 0 hasta n-1. Los valores de la función y direcciones van de 0 a n-1 ligeramente menor al tamaño del array. La mejor elección de los módulos son los números primos.[[11]](#footnote-11)

La clave de búsqueda en una cadena de caracteres tal como el nombre para obtener direcciones de conversión el método más simple es asignar a cada carácter de la cadena un valor entero (ejemplo A=1, B=2, C=3, etc.) y sumar los valores de los caracteres en la cadena al resultado se le aplica entonces el modulo.

Ejemplo:



* + - 1. Tercera definición:
    1. **Hash Método Cuadrado:**
       1. Primera definición:

La función hash cuadrado consiste en elevar al cuadrado la clave y tomar los dígitos centrales como dirección. El número de dígitos que se debe considerar se encuentra determinado por el rango del índice. Sea K la clave del dato a buscar, la función hash cuadrado queda definida, entonces, por la siguiente fórmula:

La suma de una unidad a los dígitos centrales es útil para obtener un valor comprendido entre 1 y N.

El siguiente ejemplo presenta un caso de función hash cuadrado:

Sea N = 100 el tamaño del arreglo, y sus direcciones que deben tomar sus elementos los números comprendidos entre 1 y 100. Sean K1 = 7 259 y K2 = 9 359 dos claves a las que se deben asignar posiciones en el arreglo. Se aplica la fórmula para calcular las direcciones correspondientes a K1 y K2.

Como el rango de índices en el ejemplo varía de 1 a 100, se toman solamente los dos dígitos centrales del cuadrado de las claves.[[12]](#footnote-12)

* + - 1. Segunda definición:

Este método consiste en calcular el cuadrado de la clave x. La función de conversión se define como F(x)=C donde C se obtiene eliminando dígitos de ambos extremos de x2; para todas las claves se deben usar las mismas posiciones de x2.[[13]](#footnote-13)

Ejemplo: Una empresa tiene 80 empleados y cada uno de ellos tiene un número de identificación de 4 dígitos y el conjunto de direcciones de memoria varia en un rango de 0 a 100 calcular las direcciones que se obtendrán al aplicar mitad del cuadrado.

* + - 1. Tercera definición:
    1. **Hash por truncamiento:**
       1. Primera definición:

La función hash por truncamiento consiste en tomar algunos dígitos de la clave y formar con ellos una dirección. Este método es de los más sencillos, pero es también de los que ofrecen menos uniformidad en la distribución de las claves.

Sea K la clave del dato a buscar. K está formada por los dígitos d1, d2, …, dn. La función hash por plegamiento queda definida por la siguiente fórmula:

La elección de los dígitos es arbitraria. Se podrían tomar los de las posiciones impares o de las pares. Luego se podrían unir de izquierda a derecha o de derecha a izquierda. La suma de una unidad a los dígitos seleccionados es útil para obtener un valor entre 1 y 100.

El siguiente ejemplo presenta un caso de función hash por truncamiento:

Sea N = 100 el tamaño del arreglo, y las direcciones que deben tomar sus elementos los números comprendidos entre 1 y 100. Sean K1 = 7 259 y K2 = 9 359 dos claves a las que se deben asignar posiciones en el arreglo. Se aplica la fórmula para calcular las direcciones correspondientes a K1 y K2.

En este ejemplo se toman el primer y tercer números de la clave y se unen de izquierda a derecha. Es importante destacar que en todos los casos anteriores se presentaron ejemplos de claves numéricas. Sin embargo, en la práctica las claves pueden ser alfabéticas o alfanuméricas. En general, cuando aparecen letras en las claves se suele asociar a cada una un entero con el propósito de convertirlas en numéricas.

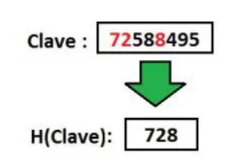
Si, por ejemplo, la clave fuera ADA, su equivalente numérico sería 010401. Si hubiera combinación de letras y números, se procederá de la misma manera. Por ejemplo. dada una clave Z4F21, su equivalente numérico sería 2740621. Otra alternativa sería tomar el valor decimal asociado para cada carácter según el código ASCII. Una vez obtenida la clave en su forma numérica, se puede utilizar normalmente cualesquiera de las funciones antes mencionadas.[[14]](#footnote-14)

* + - 1. Segunda definición:

Ignora parte de la clave y utiliza la parte restante directamente como índice (Considerando campos no numéricos y sus códigos numéricos). Si las claves, por ejemplo, son enteros de 8 dígitos y la tabla de transformación tiene 1000 posiciones entonces el primero, segundo y quinto dígitos desde la derecha pueden formar la función de conversión.[[15]](#footnote-15)

El truncamiento es un método muy rápido, pero falla para distribuir las claves de modo uniforme.

Ejemplo:



* + - 1. Tercera definición:
    1. **Hash por plegamiento:**
       1. Primera definición:

La función hash por plegamiento consiste en dividir la clave en partes, tomando igual número de dígitos, aunque la última puede tener menos, y operar con ellas, asignando como dirección los dígitos menos significativos. La operación entre las partes se puede realizar por medio de sumas o multiplicaciones. Sea K la clave del dato a buscar. K está formada por los dígitos d1, d2, …, dn. La función hash por plegamiento queda definida por la siguiente fórmula:

El operador que aparece en la fórmula operando las partes de la clave es el de suma, pero, puede ser el de la multiplicación. En este contexto, la suma de una unidad a los dígitos menos significativos –dígmensig– es para obtener un valor comprendido entre 1 y N.

El siguiente ejemplo presenta un caso de función hash por plegamiento:

Sea N = 100 el tamaño del arreglo, y las direcciones que deben tomar sus elementos los números comprendidos entre 1 y 100. Sean K1 = 7 259 y K2 = 9 359 dos claves a las que se deben asignar posiciones en el arreglo. Se aplica la fórmula para calcular las direcciones correspondientes a K1 y K2.

De la suma de las partes se toman solamente dos dígitos porque los índices del arreglo varían de 1 a 100.[[16]](#footnote-16)

* + - 1. Segunda definición:

Consiste en la partición de la clave en diferentes partes y la combinación de las partes en un modo conveniente (a menudo utilizando suma o multiplicación) para obtener el índice. La clave x se divide en varias partes donde cada parte tienen el mismo número de dígitos que la dirección especificada.[[17]](#footnote-17)

Esta técnica consiste en la partición de la clave en diferentes partes y la combinación de las partes en un modo conveniente (a menudo utilizando suma o multiplicación) para obtener el índice.

Ejemplo:

El número de identificación de los empleados es el campo clave de una empresa consta de cuatro dígitos y las direcciones reales son 100. Se desea calcular las direcciones correspondientes por el método de plegamiento.

Claves: 4205, 3355, 8148.

* + - 1. Tercera definición:
    1. **Hash por conversión de Bases:**
       1. Primera definición:

La conversión de bases consiste en modificar de manera arbitraria la base de la clave obteniendo un número que corresponda a una cubeta. Si el número de dígitos del valor resultante excede el orden de las direcciones, entonces se suprimirán los dígitos más significativos.

Ejemplo:

Supongamos que se tienen 100 cubetas, cada una de ellas referenciada por un número entero comprendido entre 1 y 100. Sea K = 7259 la clave del registro que se busca. Se elige el 9 como base a la cual se convierte la clave.

Se toman entonces como dirección el 19 y los dígitos mas significativos, 5 y 3, se desprecian.[[18]](#footnote-18)

* + - 1. Segunda definición:

Para generar la dispersión de las claves, primero se transforma la cadena, que es el campo clave, en un valor entero. Una vez hecha la transformación, se aplica el método de la multiplicación. La transformación de la cadena se realiza considerando que es una secuencia de valores numéricos en base 27.[[19]](#footnote-19)

Por ejemplo, la cadena 'RIO' se transforma en:

El valor entero de cada carácter es su ordinal en el código *ASCII*.

* + - 1. Tercera definición:

1. **Búsquedas Dinámicas:**
   1. Primera definición:

La principal característica del hashing dinámico es su dinamismo para variar el número de cubetas en función de su densidad de ocupación. Se comienza a trabajar con un número determinado de cubetas, y a medida que éstas se van llenando se asignan nuevas cubetas al archivo, Existen básicamente dos formas de trabajar con el hashing dinámico:[[20]](#footnote-20)

* Por medio de expansiones totales.
* Por medio de expansiones parciales.
  1. Segunda definición:

Las tablas hash se presentaron como una alternativa hacia las estructuras tipo árbol ya que permitían el almacenamiento de grandes volúmenes de información y algoritmos eficientes para la administración sobre estas estructuras (inserción, eliminación y búsqueda).

Sin embargo, presentan 2 grandes problemas:

1. No existen funciones hash perfectas que permitan asegurar que por cada transformación de un elemento habrá una única correspondencia en la clave que contiene este elemento.

2. Son estructuras estáticas que no pueden crecer ya que necesitan un tamaño fijo para el funcionamiento de la estructura.

Para solucionar el segundo problema se implementa la utilización de métodos totales y métodos parciales. Convirtiendo la tabla hash en una estructura dinámica capaz de almacenar un flujo de información y no una cantidad fija de datos.[[21]](#footnote-21)

* 1. Tercera definición:

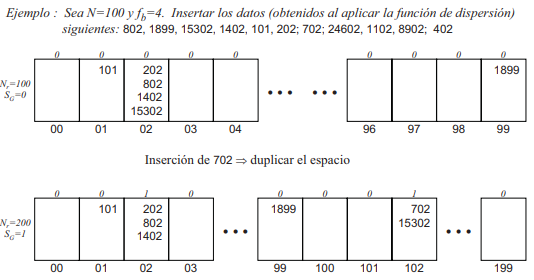
1. **Tipos de búsquedas Dinámicas:**
   1. **Expansión total:**
      1. Primera definición:

El método de expansiones totales es probablemente el más utilizado. Consiste en duplicar el número de cubetas en la medida en que éstas superan la densidad de ocupación previamente establecida.

El dinamismo de este método también se da en sentido contrario; es decir, que a medida que la densidad de ocupación de las cubetas disminuye, se reduce el número de éstas. Así, se gana flexibilidad en cuanto a que se pueden incrementar los espacios de almacenamiento, pero también se pueden reducir si la demanda de espacio así lo indica.[[22]](#footnote-22)

Así, por ejemplo, si el número inicial de cubetas es N y se hace una expansión total, el valor resultante —nuevo número de cubetas— será 2N. Si se hace una segunda expansión total, se tendrá 4N, y así sucesivamente.

Ejemplo:



* + 1. Segunda definición:

El método de las expansiones totales consiste en realizar una duplicación del tamaño del arreglo establecido para realizar la tabla hash, esta expansión se ejecuta cuando se supera la densidad de ocupación.2 Así si se tiene una tabla hash de tamaño N, al realizar la expansión total se obtendrá una tabla hash de 2N, al realizar una segunda expansión se obtendrá una tabla hash de 4N, al realizar una tercera expansión se obtendrá una tabla hash de 8N y en general el tamaño de la tabla para una i-ésima expansión se define como aparece a continuación:

Dónde:

: Tamaño de la Tabla.

: Número de expansiones que se quieren realizar.

: Nuevo tamaño de la Tabla.

La densidad de ocupación se define como el cociente entre el número de registros ocupados y el número de registros disponibles; así se tiene que:

Dónde:

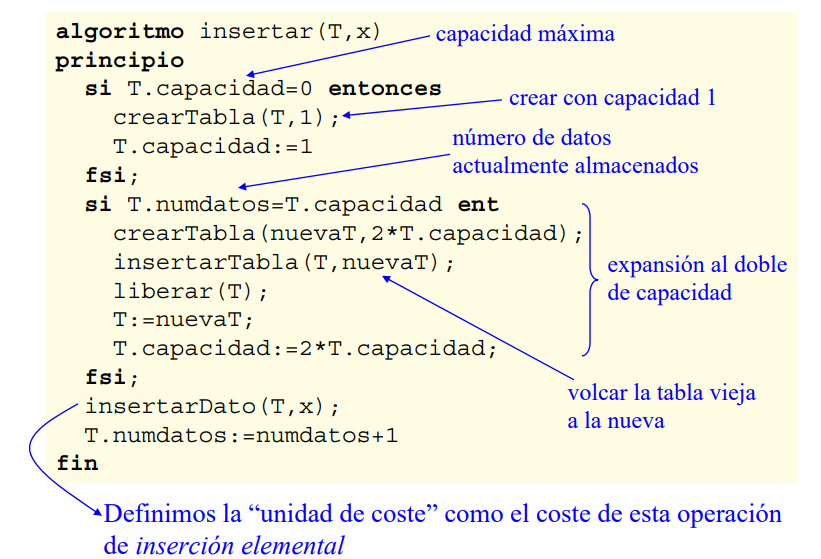
: Registros ocupados.

: Registros disponibles.

: Densidad de ocupación.

Cada vez que se pretende insertar un elemento es necesario calcular la densidad de ocupación, si se supera esta densidad se procede a implementar la expansión. Al realizar cada de una de las expansiones es necesario volver a implementar la función hash para cada uno de los registros almacenados en la tabla y volver a insertarlos de nuevo en la tabla.[[23]](#footnote-23)

Ejemplo:



* + 1. Tercera definición:
  1. **Reducción total:**
     1. Primera definición:

Este método surge como una consecuencia del método de expansiones totales presentado anteriormente. En este método la densidad de ocupación disminuye de tal manera que acepta una reducción del tamaño de la tabla hash a la mitad.

Para realizar una reducción la densidad de ocupación se debe disminuir a un valor menor al rango establecido y los registros se deben eliminar de tal manera que los registros resultantes se puedan ingresar en una tabla hash que posea la mitad del tamaño de la tabla original. Cada vez que se implementa una reducción es necesario volver a utilizar la función hash con cada uno de los registros almacenados.

Así si se tiene una tabla hash de N, la primera reducción dará como resultado la N/2, la segunda reducción dará como resultado N/4, la tercera reducción dará N/8 y la i-ésima reducción dará como resultado:

N: Tamaño de la Tabla. i: Número de expansiones que se quieren realizar. T: Nuevo tamaño de la Tabla.[[24]](#footnote-24)

* + 1. Segunda definición:

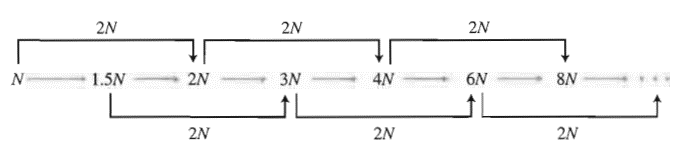
Las reducciones totales son la contraparte de las expansiones totales, es decir, reducen el tamaño de la tabla hash a la mitad.

Esta acción se ejecuta al momento de realizar una extracción o eliminación de una de las claves de la tabla. Es necesario después de aplicar la reducción volver a utilizar la función hash a cada uno de los registros almacenados.[[25]](#footnote-25)

* + 1. Tercera definición:
  1. **Expansión parcial:**
     1. Primera definición:

El método de las expansiones parciales consiste en incrementar en 50% el número de cubetas, haciendo de esta forma que dos expansiones parciales equivalgan a una total.

Así, por ejemplo, si el número inicial de cubetas es N, y se hace una expansión parcial, el valor resultante será 1.5N. Si se hacen otras expansiones parciales se tendrá 2N, luego 3N, y así sucesivamente.[[26]](#footnote-26)



* + 1. Segunda definición:

El método de las expansiones parciales consiste en incrementar en un 50% el tamaño del arreglo establecido para realizar la tabla hash, esta expansión se ejecuta cuando se supera la densidad de ocupación. Así si se tiene una tabla hash de tamaño N, al realizar la expansión parcial se obtendrá una tabla hash de 1.5 N, al realizar una segunda expansión se obtendrá una tabla hash de 2.25 N, al realizar una tercera expansión se obtendrá una tabla hash de 3.375 N y en general el tamaño de la tabla para una i-ésima expansión se define como:

Donde:

: Tamaño de la Tabla.

: Número de expansiones que se quieren realizar.

: Nuevo tamaño de la Tabla.

Cada vez que se pretende insertar un elemento es necesario calcular la densidad de ocupación, si se supera esta densidad se procede a implementar la expansión. Al realizar cada de una de las expansiones es necesario volver a implementar la función hash para cada uno de los registros almacenados en la tabla hash y volver a insertarlos de nuevo en la tabla.[[27]](#footnote-27)

* + 1. Tercera definición:
  1. **Reducción parcial:**
     1. Primera definición:

Este método surge como una consecuencia del método de expansiones parciales. En este método la densidad de ocupación disminuye de tal manera que acepta una reducción del tamaño de la tabla hash al 50%.

Para realizar una reducción la densidad de ocupación debe disminuir a un valor menor al rango establecido y los registros se deben eliminar de tal manera que los registros resultantes se puedan ingresar en una tabla hash que posea la mitad del tamaño de la tabla original. Cada vez que se implementa una reducción es necesario volver a utilizar la función hash con cada uno de los registros almacenados.

Así si se tiene una tabla hash de N, la primera reducción dará como resultado la 0.5 N, la segunda reducción dará como resultado 0.25 N, la tercera reducción dará 0.125 N y la i-ésima reducción dará como resultado:

N: Tamaño de la Tabla. i: Número de reducciones que se quieren realizar. T: Nuevo tamaño de la Tabla.[[28]](#footnote-28)

* + 1. Segunda definición:

Este método surge con la necesidad de optimizar el uso de memoria, puesto que al momento de las reducciones y expansiones totales se siguen presentando grandes espacios de memoria sin utilización. Esta manera de reducción permite disminuir el tamaño de la tabla hash en un 50%[[29]](#footnote-29). De esta manera, 3 interacciones de reducción con una tabla hash de N inicial seria de la siguiente manera:

Como en la reducción total después de implementar una reducción es necesario volver a utilizar la función hash con cada uno de los registros almacenados.

* + 1. Tercera definición:

**Bibliografía**

1. *Estructuras de Datos en Java*. (24 de Septiembre de 2022). Obtenido de https://sites.google.com/a/espe.edu.ec/programacion-ii/home/tablas-hash
2. García Díaz V. (2011). *Estructuras de datos.* Universidad de Oviedo.
3. Villagran Torres M. (29 de mayo de 2014). *Análisis algoritmos*. Comunicación de sistemas en P.D.I Chile. Obtenido de <https://es.slideshare.net/djmauriciovillagran/aritmetica-modular>
4. Joyanes Aguilar, L., Zahonero Martíinez, I., & Sanchez Garcia, L. (2007). *Estructuras de Datos en C++.* Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Sofware, Facultad de Informática, Escuela Universitaria de Informática, Universidad Pontificia de Salamanca campus Madrid. Primera edición.
5. Frittelli, V. Steffolani, F. Harach, J. Serrano, D. Fernández, J. Scarafia, D. Teicher, R. Bett, G. Tartabini, M. Strub, A. Archivos Hash: Implementación y Aplicaciones. Universidad Tecnológica Nacional, Facultad Regional Córdoba.
6. Ruano, A. (7 de Marzo de 2019). *Búsqueda secuencial y binaria*. Obtenido de Slideshare: https://es.slideshare.net/AlvaroRuano1/bsqueda-secuencial-y-binaria

1. (Estructuras de Datos, pág. 420) [↑](#footnote-ref-1)
2. (Joyanes Aguilar, Luis. Estructuras de Datos en C++, 2007) [↑](#footnote-ref-2)
3. (Estructuras de Datos, pág. 422) [↑](#footnote-ref-3)
4. (Joyanes Aguilar, Luis. Estructuras de Datos en C++, 2007) [↑](#footnote-ref-4)
5. (Estructuras de Datos, pág. 427) [↑](#footnote-ref-5)
6. (Joyanes Aguilar, Luis. Estructuras de Datos en C++, 2007) [↑](#footnote-ref-6)
7. (Estructuras de Datos, pág. 428) [↑](#footnote-ref-7)
8. (Estructuras de Datos, pág. 429) [↑](#footnote-ref-8)
9. (Frittelli, V. Archivos Hash: Implementación y Aplicaciones.) [↑](#footnote-ref-9)
10. (Estructuras de Datos, pág. 403) [↑](#footnote-ref-10)
11. (Frittelli, V. Archivos Hash: Implementación y Aplicaciones.) [↑](#footnote-ref-11)
12. (Estructuras de Datos, pág. 404) [↑](#footnote-ref-12)
13. (Frittelli, V. Archivos Hash: Implementación y Aplicaciones.) [↑](#footnote-ref-13)
14. (Estructuras de Datos, pág. 406) [↑](#footnote-ref-14)
15. (Frittelli, V. Archivos Hash: Implementación y Aplicaciones.) [↑](#footnote-ref-15)
16. (Estructuras de Datos, pág. 405) [↑](#footnote-ref-16)
17. (Frittelli, V. Archivos Hash: Implementación y Aplicaciones.) [↑](#footnote-ref-17)
18. (Estructuras de Datos, pág. 429) [↑](#footnote-ref-18)
19. (Frittelli, V. Archivos Hash: Implementación y Aplicaciones.) [↑](#footnote-ref-19)
20. (Estructuras de Datos, pág. 433) [↑](#footnote-ref-20)
21. (García Díaz V. Estructuras de datos., 2011) [↑](#footnote-ref-21)
22. (Estructuras de Datos, pág. 433) [↑](#footnote-ref-22)
23. (Villagran Torres M. Análisis algoritmos. , 2014) [↑](#footnote-ref-23)
24. (Estructuras de Datos en Java, 2022) [↑](#footnote-ref-24)
25. (Villagran Torres M. Análisis algoritmos. , 2014) [↑](#footnote-ref-25)
26. (Estructuras de Datos, pág. 437) [↑](#footnote-ref-26)
27. (Villagran Torres M. Análisis algoritmos. , 2014) [↑](#footnote-ref-27)
28. (Estructuras de Datos en Java, 2022) [↑](#footnote-ref-28)
29. (Villagran Torres M. Análisis algoritmos. , 2014) [↑](#footnote-ref-29)