

**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.

14 de septiembre de 2022.

**INDICE**

[1. Búsquedas: 3](#_Toc113805475)

[2. Búsquedas internas: 3](#_Toc113805476)

[3. Métodos de búsqueda: **¡Error! Marcador no definido.**](#_Toc113805477)

[3.1. Método Secuencial: **¡Error! Marcador no definido.**](#_Toc113805478)

[3.2. Método Binario: 7](#_Toc113805479)

[3.3. Funciones Hash: 9](#_Toc113805480)

[3.3.1. Definición Función Hash: 9](#_Toc113805481)

[3.3.2. Hash Mod: 9](#_Toc113805481)

[3.3.3. Hash Método Cuadrado: 11](#_Toc113805482)

[3.3.4. Hash por truncamiento: 13](#_Toc113805483)

[3.3.5. Hash por plegamiento: 16](#_Toc113805484)

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:
  2. 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 do 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 si.

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.[[2]](#footnote-2)

Ejemplo:

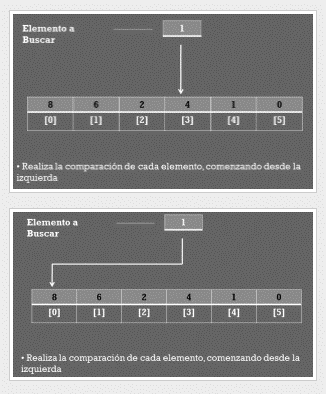
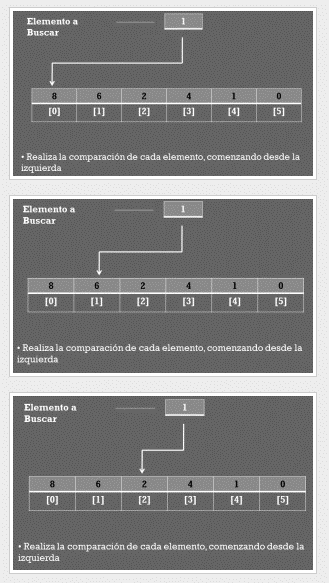


Imagen 1. Ejemplo del método secuencial tomado de: “https://joomat.wordpress.com/2012/06/20/algoritmos-de-busqueda/”

* + 1. Segunda definición:
    2. 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.[[3]](#footnote-3)

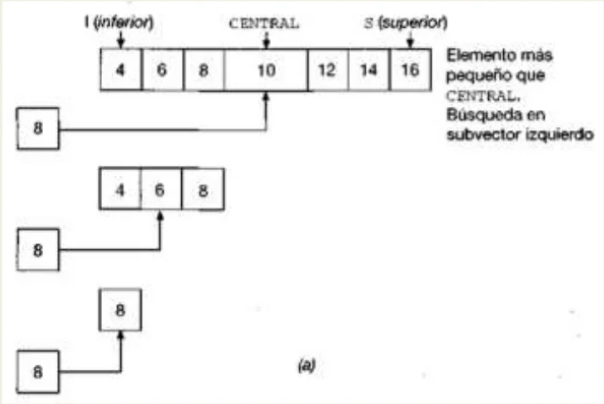


Imagen 2. Ejemplo del método binario sacado de:

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

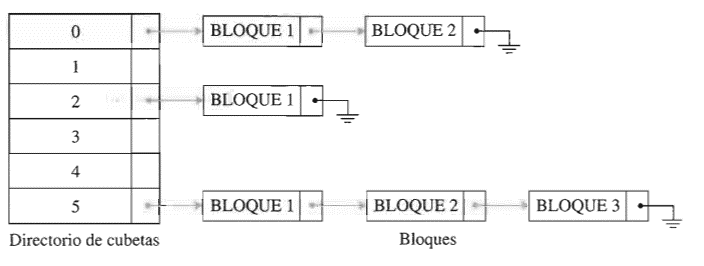
* + 1. Segunda definición:
    2. Tercera definición:
  1. **Funciones Hash:**
     1. **Definición Función Hash**
        1. Primera definición:

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.[[4]](#footnote-4)

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.[[5]](#footnote-5)

* + - 1. Segunda definición:
      2. 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.[[6]](#footnote-6)

* + - 1. Segunda definición:
      2. 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.[[7]](#footnote-7)

* + - 1. Segunda definición:
      2. 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.[[8]](#footnote-8)

* + - 1. Segunda definición:
      2. 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.[[9]](#footnote-9)

* + - 1. Segunda definición:
      2. 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 = 7 259 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.[[10]](#footnote-10)

* + - 1. Segunda definición:
      2. 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:[[11]](#footnote-11)

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

1. **Tipos de búsquedas Dinámicas:**
   * 1. **Expansiones totales:**
        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.[[12]](#footnote-12)

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.

* + - 1. Segunda definición:
      2. 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.

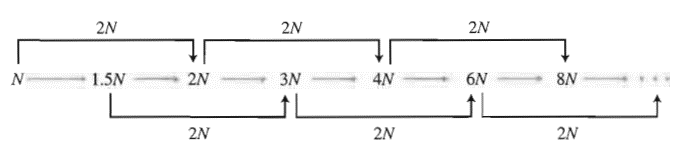
Así si se tiene una tabla hash de N, la primera reducción dará como resultado la N/2, la segunda reducción dará como resultado N/4, la tercera reducción dará N/8 y la i-ésima reducción dará como resultado:

N: Tamaño de la Tabla. i: Número de expansiones que se quieren realizar. T: Nuevo tamaño de la Tabla.[[13]](#footnote-13)

* + - 1. Segunda definición:
      2. Tercera definición:
    1. **Expansiones parciales:**
       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.[[14]](#footnote-14)



* + - 1. Segunda definición:
      2. 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.[[15]](#footnote-15)

* + - 1. Segunda definición:
      2. Tercera definición:

**Bibliografía**

Bustamante, M., & Guzmán, L. (2014). *Algoritmos de búsqueda - Truncamiento.* Universidad Tecnológica de Chile.

Cairó, O., & Guardati, S. (2006). *Estructuras de Datos.* Ciudad de Mexico: McGraw-Hill. Tercera Edición.

*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

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.

Ruano, A. (7 de Marzo de 2019). *Búsqueda secuencial y binaria*. Obtenido de Slideshare: https://es.slideshare.net/AlvaroRuano1/bsqueda-secuencial-y-binaria

Sedgewick, R. (1995). *Algoritmos en C++.* Ediciones Díaz de Santos.

1. (Estructuras de Datos, pág. 420) [↑](#footnote-ref-1)
2. (Estructuras de Datos, pág. 422) [↑](#footnote-ref-2)
3. (Estructuras de Datos, pág. 427) [↑](#footnote-ref-3)
4. (Estructuras de Datos, pág. 428) [↑](#footnote-ref-4)
5. (Estructuras de Datos, pág. 429) [↑](#footnote-ref-5)
6. (Estructuras de Datos, pág. 403) [↑](#footnote-ref-6)
7. (Estructuras de Datos, pág. 404) [↑](#footnote-ref-7)
8. (Estructuras de Datos, pág. 406) [↑](#footnote-ref-8)
9. (Estructuras de Datos, pág. 405) [↑](#footnote-ref-9)
10. (Estructuras de Datos, pág. 429) [↑](#footnote-ref-10)
11. (Estructuras de Datos, pág. 433) [↑](#footnote-ref-11)
12. (Estructuras de Datos, pág. 433) [↑](#footnote-ref-12)
13. (Estructuras de Datos en Java, 2022) [↑](#footnote-ref-13)
14. (Estructuras de Datos, pág. 437) [↑](#footnote-ref-14)
15. (Estructuras de Datos en Java, 2022) [↑](#footnote-ref-15)