UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

PRIMERA CONSULTA – BUSQUEDAS

Presentado a:

Julio Cesar Florez Baez

Presentado por:

Johan Esteban Castaño Martinez - 20191020029

Jhony Alejandro Caro Umbariba - 20191020055

Samuel Andrés Romero Bueno - 20191020127

Facultad de Ingeniería.

Ciencias de la Computación II.

14 de septiembre de 2022.

INDICE

1. Búsquedas:

1.1. Primera definición

La búsqueda es una operación importante en el procesamiento de información, que se utiliza para recuperar datos que se habían almacenado con anticipación, el resultado puede ser de éxito si se encuentra la información o de fracaso, en caso contrario. Las búsquedas se pueden llevar a cabo sobre elementos ordenados o desordenados, siendo más fácil esta operación cuando los datos se encuentran ordenados.

Los métodos de búsqueda se pueden clasificar en internos y externos, según la ubicación de los datos sobre los cuales se realizará la búsqueda. Se denomina búsqueda interna cuando todos los elementos se encuentran en la memoria interna de una computadora y externos si los elementos están en una memoria secundaria.[1]

1.2. Segunda definición:

Con mucha frecuencia los programadores trabajan con grandes cantidades de datos almacenados en arrays y registros y, por ello, será necesario determinar si un array contiene un valor que coincida con un valor clave. El proceso de encontrar un elemento específico de un array se denomina búsqueda.[2]

1.3. Tercera definición:

La búsqueda es una operación fundamental, intrínseca a una gran cantidad de tareas de las computadoras, que consiste en recuperar uno o varios elementos particulares de un gran volumen de información previamente almacenada. Normalmente se considera que la información está dividida en registros, cada uno de los cuales posee una clave para utilizar en la búsqueda. El objetivo de esta operación es encontrar todos los registros cuyas claves coincidan con una cierta clave de búsqueda, con el propósito de acceder a la información (y no solamente a la clave) para su procesamiento.

Las aplicaciones de la búsqueda están muy difundidas y abarcan una variada gama de operaciones diferentes. Por ejemplo, un banco necesita hacer un seguimiento de las cuentas de todos sus clientes y buscar en ellas para verificar diversos tipos de transacciones. De igual forma un sistema de reservas de unas líneas aéreas tiene necesidades similares, aunque la mayor parte de los datos sean de vida corta.[3]

2. Búsquedas internas:

---

[1] (Osvaldo Cairó, 2006)

[2] (Joyanes Aguilar, Zahonero Martíinez, & Sanchez Garcia, 2007)

[3] (Sedgewick, 1995)

2.1. Primera definición:

La búsqueda interna trabaja con elementos que se encuentran almacenados en la memoria principal de una computadora, estos pueden estar en estructuras estáticas, como, por ejemplo, almacenamiento en arreglos o dinámicas, listas ligadas y árboles. los métodos internos más importantes son: [4]

- Secuencia Lineal.
- Binaria.
- Por transformación de claves.
- Árboles de búsqueda.

2.2. Segunda definición:

En la búsqueda interna los elementos están en todo momento en memoria principal, salvo cuando termina el proceso que se vuelven a almacenar en el dispositivo externo. La memoria principal es limitada, por contra, el número de registros u objetos de los que puede constar un archivo externo es muy elevado. [5]

2.3. Tercera definición:

Es aquella que se realiza completamente en la memoria RAM.[6]

3. Métodos de búsqueda:

3.1. Método Secuencial:

3.1.1. Primera definición:

La búsqueda secuencial consiste en revisar elemento tras elemento hasta encontrar el dato buscado, o llegar al final del conjunto de datos disponible.

Primero se tratará sobre la búsqueda secuencial en arreglos, y luego en listas enlazadas. En el primer caso, se debe distinguir entre arreglos ordenados y desordenados. Esta última consiste, básicamente, en recorrer el arreglo de izquierda a derecha hasta que se encuentre el elemento buscado o se termine el arreglo.

Algoritmo secuencial desordenado: Este algoritmo busca secuencialmente el elemento X en un arreglo unidimensional desordenado V, de N componentes. I es una variable de tipo entero.

---

[4] (Osvaldo Cairó, 2006)
[5] (Joyanes Aguilar, Zahonero Martíinez, & Sanchez Garcia, 2007)
[6] (Ruano, 2019)

Son dos los posibles resultados que se pueden obtener al aplicar el algoritmo: la posición en la que encontró el elemento o un mensaje de error de fracaso si no se encuentra. Si hubiera dos o más ocurrencias del mismo valor, se encuentra la primera de ellas. También es posible modificar el algoritmo para obtener todas las ocurrencias del dato buscado.

Algoritmo secuencial desordenado recursivo: Este algoritmo busca secuencialmente y de forma recursiva el elemento X en el arreglo unidimensional desordenado V de N componentes. I es un parámetro de tipo entero que inicialmente se encuentra en 1.

Algoritmo secuencial ordenado: Este algoritmo busca secuencialmente al elemento X en un arreglo unidimensional ordenado V, de N componentes. V se encuentra ordenado crecientemente: V[1] ≤ V[2] ≤ … V[N]. I es una variable de tipo entero.

Como el arreglo está ordenado, se establece una nueva condición: el elemento buscado tiene que ser mayor que el del arreglo. Cuando el ciclo se interrumpe, se evalúa cuál de las condiciones es falsa.

Algoritmo secuencial ordenado recursivo: Este algoritmo busca en forma secuencial y recursiva al elemento X en un arreglo unidimensional ordenado V de N componentes. V se encuentra de manera creciente: V[1] ≤ V[2] ≤ … ≤ V[N]. I inicialmente tiene el valor de 1.

El método de búsqueda secuencial también se puede aplicar a listas ligadas. Consiste en recorrer la lista nodo tras nodo, hasta encontrar el elemento buscado o hasta llegar al final de la lista.

Algoritmo secuencial lista desordenada: Este algoritmo busca en forma secuencial al elemento X en una lista simplemente ligada, que almacena información que está desordenada. P es un apuntador al primer nodo de la lista. INFO y LIGA son los campos de cada nodo. Q es una variable de tipo apuntador.

Algoritmo secuencial lista desordenada recursivo: Este algoritmo busca de manera secuencial y en forma recursiva al elemento X en una lista simplemente ligada, que almacena información que esta desordenada. P es un apuntador al primer nodo de la lista. INFO y LIGA son los campos de cada nodo.

El número de comparaciones es uno de los factores más importantes que se utilizan para determinar la complejidad de los métodos de búsqueda. Para

analizar la complejidad de la búsqueda secuencial, se deben establecer los casos más favorables o desfavorables que se presenten. [7]

3.1.2. Segunda definición:

La búsqueda secuencial busca un elemento de una lista utilizando un valor destino llamado clave. En una búsqueda secuencial (a veces llamada búsqueda lineal), los elementos de una lista se exploran (se examinan) en secuencia, uno después de otro.

El algoritmo de búsqueda secuencial compara cada elemento del array con la clave de búsqueda. Dado que el array no está en un orden prefijado, es probable que el elemento a buscar pueda ser el primer elemento, el último elemento o cualquier otro. De promedio, al menos el programa tendrá que comparar la clave de búsqueda con la mitad de los elementos del array. El método de búsqueda lineal funcionará bien con arrays pequeños o no ordenados. [8]

3.1.3. Tercera definición:

El método de búsqueda más simple consiste en almacenar todos los registros en un array. Cuando se inserta un nuevo registro, se pone al final del array; cuando se lleva a cabo una búsqueda, se recorre secuencialmente el array. El siguiente programa muestra una implementación de las funciones básicas que utiliza esta sencilla organización e ilustra a su vez algunos de los convenios que se utilizarán en la implementación de los métodos de búsqueda.[9]

Ejemplo:

```
Procedimiento BuscarSecuencial2( )
…
Inicio
   Leer(A) // arreglo de elementos
   Leer(t) // elemento a buscar
   i = 1
   A[n + 1] = t // centinela
   mientras (A[i] <> t) hacer
        i = i + 1
   fin-mientras
   si i >= n + 1 entonces
        Escribir("No se ha encontrado el elemento")
   sino
        Escribir("Se ha encontrado el elemento en la posición", i)
   fin-si
Fin-procedimiento
```

Imagen 1. Ejemplo del método secuencial sacado de https://www.slideshare.net/AlvaroRuano1/bsqueda-secuencial-y-binaria.

[7] (Osvaldo Cairó, 2006)
[8] (Joyanes Aguilar, Zahonero Martíinez, & Sanchez Garcia, 2007)
[9] (Sedgewick, 1995)

3.2. Método Binario:

3.2.1. Primera definición:

La búsqueda binaria consiste en dividir el intervalo de búsqueda en dos partes, comparando el elemento buscado con el que ocupa la posición central en el arreglo. En caso de que no fueran iguales, se redefinen los extremos del intervalo, según el elemento central sea mayor o menor que el elemento buscado, disminuyendo de esta forma el espacio buscado. El proceso concluye cuando el elemento es encontrado o cuando el intervalo de búsqueda se anula.

El método de búsqueda binaria funciona exclusivamente con arreglos ordenados. No se pueden utilizar con listas simplemente ligadas, no se puede retroceder para establecer intervalos de búsqueda, ni con arreglos desordenados. Con cada iteración del método el espacio de búsqueda se reduce a la mitad, por lo tanto, el número de comparaciones a realizar disminuye notablemente.

Algoritmo binario: Este algoritmo busca al elemento *X* en un arreglo unidimensional ordenado *V* de *N* componentes. *IZQ, CEN, DER*, son variables de tipo entero. *BAN* es una variable de tipo booleano.

Algoritmo binario sin bandera: Este algoritmo busca al elemento *X* en el arreglo unidimensional ordenado *V* de *N* componentes. *IZQ, DER, CEN* son variables de tipo entero.

Algoritmo binario recursivo: Este algoritmo busca al elemento *X* en el arreglo unidimensional ordenado *V* de *N* componente. *IZQ* ingresa inicialmente al algoritmo con el valor de 1. *DER*, por otra parte, ingresa con el valor de N. *CEN* es una variable de tipo entero.

Para analizar la complejidad del método de búsqueda binaria es necesario establecer los casos más favorables y desfavorables que se pudieran presentar en el proceso de búsqueda. El primero sucede cuando el elemento buscado es el central, en dicho caso se hará una sola comparación; el segundo sucede cuando el elemento no se encuentra en el arreglo; entonces se harán $Log_2(n)$ comparaciones, ya que con cada comparación el número de elementos en los cuales se debe buscar se reduce en un factor de 2. De esta forma se determinan los números mínimo, mediano y máximo de comparaciones que se deben realizar cuando se utiliza este tipo de búsqueda.[10]

[10] (Osvaldo Cairó, 2006)

$$C_{\text{mín}} = 1 \qquad C_{\text{med}} = \frac{\left(1 + \log_2(N)\right)}{2} \qquad C_{\text{máx}} = \log_2(N)$$

3.2.2. Segunda definición:

Si la lista está ordenada, la búsqueda binaria proporciona una técnica de búsqueda mejorada. Localizar una palabra en un diccionario es un ejemplo típico de búsqueda binaria. Dada la palabra, se abre el libro cerca del principio, del centro o del final dependiendo de la primera letra de la palabra que busca. Se puede tener suerte y acertar con la página correcta; pero, normalmente, no será así y se mueve el lector a la página anterior o posterior del libro. Por ejemplo, si la palabra comienza con "J" y se está en la "L" se mueve uno hacia atrás. El proceso continúa hasta que se encuentra la página buscada o hasta que se descubre que la palabra no está en la lista.[11]

3.2.3. Tercera definición:

Si el conjunto de registros es grande, entonces el tiempo total de búsqueda se puede reducir significativamente utilizando un procedimiento de búsqueda basado en la aplicación del paradigma de «divide y vencerás»: se divide el conjunto de registros en dos partes, se determina a cuál de las dos partes debe pertenecer la clave buscada, y a continuación se repite el proceso en esa parte. Una forma razonable de dividir en partes el conjunto de registros consiste en mantener los registros ordenados y después utilizar los índices del array ordenado para delimitar la parte del array sobre la que se va a trabajar.[12]

Ejemplo:

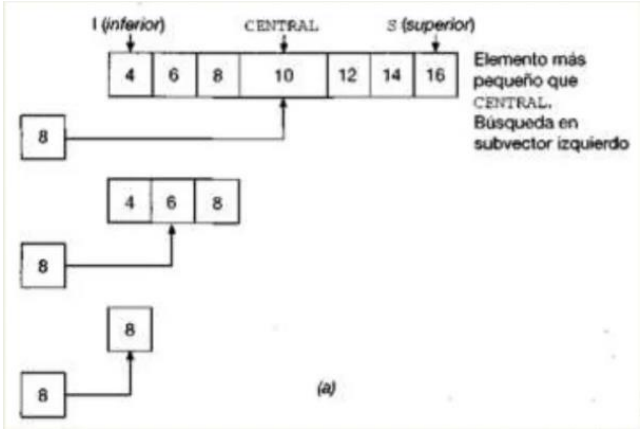

Imagen 2. Ejemplo del método binario sacado de: https://www.slideshare.net/AlvaroRuano1/bsqueda-secuencial-y-binaria

[11] (Joyanes Aguilar, Zahonero Martíinez, & Sanchez Garcia, 2007)
[12] (Sedgewick, 1995)

3.3. Funciones Hash:

3.3.1. Definición Función Hash

3.3.1.1. Primera definición:

Método de búsqueda externa por transformación de claves (HASH). Los archivos se encuentran normalmente organizados en áreas llamadas cubetas. Estas se encuentran formadas por cero, uno o mas 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.

La elección adecuada de una función hash y de un método para resolver colisiones es fundamental para lograr mayor eficiencia en la búsqueda.[13]

3.3.1.2. Segunda definición:

Las funciones hash o dispersión convierten el dato considerado campo clave (tipo entero o cadena de caracteres) en un índice dentro del rango de definición del array o vector que almacena los elementos, de tal forma que sea adecuado para indexar el array. La idea que subyace es utilizar la clave de un elemento para determinar su dirección o posición en un almacenamiento secuencial, pero sin desperdiciar mucho espacio, para lo que se realiza una transformación, mediante una función hash.[14]

3.3.1.3. Tercera definición:

También llamado búsqueda mediante transformación de claves, no requiere que los datos estén ordenados, consiste en convertir una clave dada en una dirección dentro del listado donde se almacenan los valores, está misma conversión se realizara al momento de realizar la búsqueda, obteniendo de forma inmediata la posición dentro del listado.[15]

3.3.2. Hash Mod:

3.3.2.1. 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

[13] (Osvaldo Cairó, 2006)
[14] (Joyanes Aguilar, Zahonero Martíinez, & Sanchez Garcia, 2007)
[15] (Sedgewick, 1995)

clave del dato a buscar. La función hash queda definida por la siguiente fórmula:

$$H(K) = (K \; mod \; N) + 1$$

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.

$$H(K1) = (7259 \; mod \; 100) + 1 = 60$$
$$H(K2) = (9359 \; mod \; 100) + 1 = 60$$

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:

$$H(K1) = (7259 \; mod \; 97) + 1 = 82$$
$$H(K2) = (9359 \; mod \; 97) + 1 = 48$$

Con N = 97 se ha eliminado la colisión.[16]

3.3.2.2. Segunda definición:

Una función de dispersión que utiliza la aritmética modular genera valores dispersos calculando el resto de la división entera entre la clave x y el tamaño de la tabla.[17]

$$\mathrm{h(x) = \; x \; modulo \; m}$$

Normalmente, la clave asociada con un elemento es de tipo entero. A veces, los valores de la clave no son enteros, entonces, previamente, hay que

[16] (Osvaldo Cairó, 2006)
[17] (Joyanes Aguilar, Zahonero Martíinez, & Sanchez Garcia, 2007)

transformar la clave a un valor entero. Por ejemplo, si la clave es una cadena de caracteres, se puede transformar considerando el valor ASCII de cada carácter como si fuera un dígito entero de base 128.

La operación resto(módulo) genera un número entre 0 y m-1 cuando el segundo operando es m. Por tanto, esta función de dispersión proporciona valores enteros dentro del rango 0 ... m-1.

Ejemplo:
Considerar una aplicación en la que se debe almacenar n = 900 registros. El campo clave elegido es el número de identificación. Elegir el tamaño de la tabla de dispersión y calcular la posición que ocupa los elementos cuyo número de identificación es:

$$245643 \qquad 245981 \qquad 257135$$

Una buena elección de m, en este supuesto, es 997 al ser un número primo próximo y tener como factor de carga (n/m) aproximadamente 0.8 cuando se hayan guardado todos los elementos.
Teniendo en cuenta el valor de m, se aplica la función hash de aritmética modular y se obtienen estas direcciones:

$$h(245643) = 245643 \; modulo \; 997 = 381$$
$$h(245981) = 245981 \; modulo \; 997 = 719$$
$$h(257135) = 257135 \; modulo \; 997 = 906$$

3.3.2.3. Tercera definición:

Se utiliza la función MOD para dividir la entrada dentro de del rango de índice (tamaño del array o lista). Se mejora la efectividad utilizando números primos cercanos al rango.[18]

Ejemplo:

$$7490312$$
$$Tamaño \; 100, el \; menor \; primo \; es \; 97 \; .$$
$$7490312 \; MOD \; 97 = 69$$

3.3.3. Hash Método Cuadrado:

3.3.3.1. Primera definición:

[18] (Ruano, 2019)

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:

$$H(K) = dígitos\_centrales\,(K^2) + 1$$

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.

$$K1^2 = 52\,693\,081$$
$$K2^2 = 87\,590\,881$$

$$H(K1) = dígitos\_centrales\,(52\,693\,081) + 1 = 94$$
$$H(K2) = dígitos\_centrales\,(87\,590\,881) + 1 = 91$$

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.[19]

3.3.3.2. Segunda definición:

Esta técnica de obtener direcciones dispersas consiste, en primer lugar, en calcular el cuadrado de la clave x y, a continuación, extraer del resultado, x^2, los dígitos que se encuentran en ciertas posiciones. El número de dígitos a extraer depende del rango de dispersión que se quiere obtener. Así, si el rango es de 0 ... 999 se extraen tres dígitos, siempre, los que están en las mismas posiciones. [20]

Un problema potencial, al calcular x^2, es que sea demasiado grande y exceda el máximo entero. Es importante, al aplicar este método de dispersión, utilizar siempre las mismas posiciones de extracción para todas las claves.

Ejemplo:
Aplicar el método de Mitad del Cuadrado a los registros:

[19] (Osvaldo Cairó, 2006)
[20] (Joyanes Aguilar, Zahonero Martíinez, & Sanchez Garcia, 2007)

$$245643 \qquad 245981 \qquad 257135$$

Una vez elevado al cuadrado el valor de la clave, se eligen los dígitos que se encuentran en las posiciones 4, 5 y 6 por la derecha. El valor de esa secuencia es la dirección obtenida al aplicar este método de dispersión:

$$245643, h(245643) = 483;$$
$$245981, h(245981) = 652;$$
$$257135, h(257135) = 408;$$

paso a paso:

$$245643\ l\ 2456432\ l\ 60340483449\ l\ (dígitos\ 4,5\ y\ 6\ por\ la\ derecha)\ 483$$
$$245981\ l\ 2459812\ l\ 60506652361\ l\ (dígitos\ 4,5\ y\ 6\ por\ la\ derecha)\ 652$$
$$257135l\ 2571352\ l\ 66118408225\ l\ (dígitos\ 4,5\ y\ 6\ por\ la\ derecha)\ 408$$

3.3.3.3. Tercera definición:

Se calcula el cuadrado de la clave y luego se seleccionan números de la parte intermedia. Siempre se deben seleccionar las mismas posiciones dentro del cuadrado. [21]

Ejemplo:

$$7490312$$
$$7490312^{\,2} = 56{,}104{,}773{,}857{,}344$$
$$Se\ toman\ las\ posiciones\ 6\ y\ 7$$
$$Salida: 77$$

3.3.4. Hash por truncamiento:

3.3.4.1. 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:

$$H(K) = elegirdígitos\ (d1, d2, \ldots, dn) + 1$$

[21] (Ruano, 2019)

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.

$$H(K) = elegirdígitos\,(7259) + 1 = 75 + 1 = 76$$
$$H(K) = elegirdígitos\,(9359) + 1 = 95 + 1 = 96$$

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.

A	B	C	D	...	Z
01	02	03	04	...	27

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. [22]

3.3.4.2. Segunda definición:

El truncamiento consiste en tomar algunos dígitos de la clave y con estos formar una posición en un array. Se ignora parte de la clave para con el resto formar un índice de almacenamiento. La idea consiste en tener un listado de números, seleccionar por ejemplo la posición 2, 4 y 6; para así tener una posición en donde poder almacenar la clave. Llevando esto a un ejemplo práctico: [23]

[22] (Osvaldo Cairó, 2006)
[23] (Bustamante & Guzmán, 2014)

$$5700931 -> 703$$
$$3498610 -> 481$$
$$0056241 -> 064$$
$$9134720 -> 142$$
$$5174829 -> 142$$

Lo positivo del truncamiento y una de las ventajas por sobre otros métodos de búsqueda y ordenamiento es que no solo funciona con valores numéricos, también funciona con caracteres alfabéticos, esto se puede aplicar de dos formas:

- Transformar cada carácter en un número asociado a su posición en el abecedario.
- Transformar cada carácter en un número asociado a su valor según el código ASCII.

Una vez obtenida la clave en forma numérica, se puede utilizar normalmente. La elección de los dígitos a considerar es arbitraria, pueden tomarse posiciones pares o impares. Luego se les puede unir de izquierda a derecha o de derecha a izquierda. La función queda definida por:

$$H(K) = dígitos(t1\ t2\ t3 \dots tn) + 1$$

Existen casos en los que a la clave generada se le agrega una unidad, esto es para los casos en el que el vector de almacenamiento tenga valores entre el 1 y el 100, así se evita la obtención de un valor 0.

$$H(K1) = dígitos(7\ 2\ 5\ 9) + 1 = 76$$
$$H(K2) = dígitos(9\ 3\ 5\ 9) + 1 = 96$$

Una de las principales desventajas de utilizar truncamiento y en sí el método de hashing es que dos o más claves puede tomar una misma posición dentro de la tabla de hash, a esto es a lo que se le llama Colisión. Cuando hay colisiones se requiere de un proceso adicional para encontrar la posición disponible para la clave, lo cual disminuye la eficiencia del método.[24]

3.3.4.3. Tercera definición:

Ignora una parte de la clave y se utiliza la parte restante directamente como índice. Es un método muy rápido, pero falla para distribuir las claves de forma uniforme.[25]

Ejemplo:

[24] (Joyanes Aguilar, Zahonero Martíinez, & Sanchez Garcia, 2007)
[25] (Ruano, 2019)

$$7490312$$
$$Se\ trunca\ 90312$$
$$Salida: 74$$

3.3.5. Hash por plegamiento:

3.3.5.1. 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:

$$H(K) = dígmensig\ ((d1\ \ldots\ di) + (di+1\ \ldots\ dj) + \ldots + (d1\ \ldots\ dn)) + 1$$

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.

$$H(K1) = dígmensig\ (72 + 59) + 1 = dígmensig\ (131) + 1 = 32$$
$$H(K2) = dígmensig\ (93 + 59) + 1 = dígmensig\ (152) + 1 = 53$$

De la suma de las partes se toman solamente dos dígitos porque los índices del arreglo varían de 1 a 100.[26]

3.3.5.2. Segunda definición:

La técnica del plegamiento se utiliza cuando el valor entero del campo clave elegido es demasiado grande, pudiendo ocurrir que no pueda ser almacenado en memoria. Consiste en partir la clave x en varias partes x_1, x_2, x_3 ... x_n, la combinación de las partes de un modo conveniente (a menudo sumando las partes) da como resultado la dirección del registro. Cada parte xi, con a lo

[26] (Osvaldo Cairó, 2006)

sumo la excepción de la última, tiene el mismo número de dígitos que la dirección especificada. La función hash se define:

$$h(x) = x1 + x2 + x3 + \ldots + xr$$

La operación que se realiza para el cálculo de la función hash desprecia los dígitos más significativos obtenidos del acarreo.[27]

Ejemplo:
Los registros de pasajeros de un tren de largo recorrido se identifican por un campo de seis dígitos, que se va a utilizar como clave para crear una tabla dispersa de m = 1.000 posiciones (rango de 0 a 999). La función de dispersión utiliza la técnica de plegamiento de tal forma que parte a la clave, 6 dígitos, en dos grupos de tres y tres dígitos y, a continuación, se suman los valores de cada grupo.

Aplicar esta función a los registros:

$$245643 \qquad 245981 \qquad 257135$$

La primera clave, 245643, al dividirla en dos grupos de 3 dígitos se obtiene: 245 y 643. La dirección obtenida:

$$h(245643) = 245 + 643 = 888$$

Para las otras claves:

$$h(245981) = 245 + 981 = 1226\ l\ 226\ (se\ ignora\ el\ acarreo\ 1)$$
$$h(257135) = 257 + 135 = 392$$

A veces, se determina la inversa de las partes pares, x2, x4 ... antes de sumarlas, con el fin de conseguir mayor dispersión. La inversa en el sentido de invertir el orden de los dígitos. Así, con las mismas claves se obtienen las direcciones:

$$h(245643) = 245 + 346 = 591$$
$$h(245981) = 245 + 189 = 434$$
$$h(257135) = 257 + 531 = 788$$

3.3.5.3. Tercera definición:

Se divide la clave en diferentes partes. Las distintas partes se combinan de un modo conveniente, a menudo suma o multiplicación. Es recomendable que

[27] (Joyanes Aguilar, Zahonero Martíinez, & Sanchez Garcia, 2007)

las partes tengan la misma cantidad de dígitos que se espera tenga la salida. Se truncan los dígitos “sobrantes” más significativos de la salida.[28]

Ejemplo:

$$7490312$$

$$74 + 90 + 31 + 2 \ = \ 197$$

$$Se\ trunca\ a\ 97$$

$$Salida: 97$$

[28] (Ruano, 2019)

Bibliografía

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

[2.] 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.

[3]. Osvaldo Cairó, S. G. (2006). *Estructuras de Datos.* McGraw-Hill. Tercera Edición.

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

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