

**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.

10 de septiembre de 2022.

**INDICE**

[1. Búsquedas: 2](#_heading=h.30j0zll)

[2. Búsquedas internas: 2](#_heading=h.1fob9te)

[3. Métodos de búsqueda: 2](#_heading=h.3znysh7)

[3.1. Método Secuencial: 2](#_heading=h.2et92p0)

[3.2. Método Binario: 2](#_heading=h.tyjcwt)

[3.3. Funciones Hash: 3](#_heading=h.3dy6vkm)

[3.3.1. Hash Mod: 3](#_heading=h.1t3h5sf)

[3.3.2. Hash Método Cuadrado: 3](#_heading=h.4d34og8)

[3.3.3. Hash por truncamiento: 3](#_heading=h.2s8eyo1)

[3.3.4. Hash por plegamiento: 3](#_heading=h.17dp8vu)

1. **Búsquedas:**
   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. Segunda definición:
  2. Tercera definición:

1. **Búsquedas internas:**
   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:

* Secuencia Lineal
* Binaria
* Por transformación de claves
* Árboles de búsqueda
  1. Segunda definición:
  2. Tercera definición:

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

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 un 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 queesta 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.

* + 1. Segunda definición:
    2. Tercera definición:
  1. **Método Binario:**
     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 binaria:**

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 binaria 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 binaria 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 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.



* + 1. Segunda definición:
    2. Tercera definición:
  1. **Funciones Hash:**
     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:

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) = (7 259 mod 100) + 1 = 60

H(K2) = (9 359 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 cambiaria:

H(K1) = (7 259 mod 97) + 1 = 82

H(K2) = (9 359 mod 97) + 1 = 48

Con N = 97 se ha eliminado la colisión.

* + - 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:

H(K) = dígitos\_centrales (K²) + 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.

K1² = 52 693 081

K2² = 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.

* + - 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:

H(K) = elegirdígitos (d1, d2, …, dn) + 1

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 formula para calcular las direcciones correspondientes a K1 y K2.

H(K) = elegirdígitos (7 259) + 1 =75 + 1 = 76

H(K) = elegirdígitos (9 359) + 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 numericas.

A B C D …Z

01 02 03 04 …27

Si, por ejemplo, la clave fuera **ADA**, su equivalente numérica sería **010401**. Si hubiera combinación de letras y números, se procederá de la misma manera. Por ejemplo. dada una clave **Z4F21**, su equivalente numérica 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.

* + - 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:

H(K) = dígmensig ((d1 … di) + (di+1 … dj) + … + (d1 … 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 formula 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 digitos porque los indices del arreglo varían de 1 a 100.

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

**Bibliografía**