

# Capítulo

# 9

## MÉTODOS DE BÚSQUEDA

### 9.1 INTRODUCCIÓN

Este capítulo se dedica al estudio de una de las operaciones más importantes en el procesamiento de información: la **búsqueda**. Esta operación se utiliza básicamente para recuperar datos que se habfan almacenado con anticipación. El resultado puede ser de éxito si se encuentra la información descada, o de fracaso, en caso contrario.

La búsqueda ocupa una parte importante de nuestra vida. Prácticamente todo el tiempo estamos *buscando* algo. El mundo en que se vive hoy día es desarrollado, automatizado, y la información representa un elemento de vital importancia. Es fundamental estar informados y, por lo tanto, buscar y recuperar información. Por ejemplo, se buscan números telefónicos en un directorio, ofertas laborales en un periódico, libros en una biblioteca, etcétera.

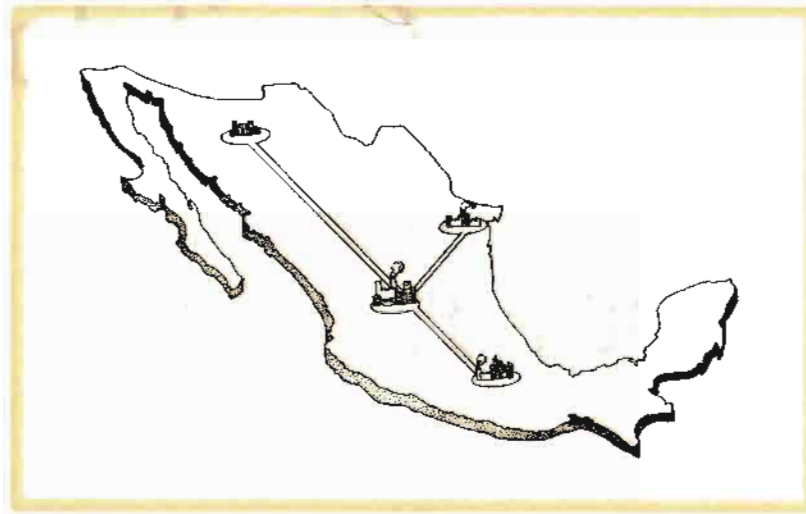
FIGURA 9.1

Ejemplo práctico de búsqueda.



**FIGURA 9.2**

Ejemplo práctico de búsqueda.



En los ejemplos mencionados, la búsqueda se realiza, generalmente, sobre elementos que están ordenados. Los directorios telefónicos están organizados alfabéticamente, las ofertas laborales están ordenadas por tipo de trabajo y los libros de una biblioteca están clasificados por tema. Sin embargo, puede suceder que la búsqueda se realice sobre una colección de elementos no ordenados. Por ejemplo, cuando se busca la localización de una ciudad dentro de un mapa.



Se concluye que la operación de búsqueda se puede llevar a cabo sobre elementos ordenados o desordenados. Cabe destacar que la búsqueda es más fácil y ocupa menos tiempo 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 principal de la computadora; por ejemplo, almacenados en arreglos, listas ligadas o árboles. Es **búsqueda externa** si los elementos están en memoria secundaria; es decir, si hubiera archivos en dispositivos como cintas y discos magnéticos.

En la siguiente sección se estudiarán los métodos internos, posteriormente en la sección 9.3 se hará lo propio con los externos.

## 9.2 BÚSQUEDA INTERNA

La **búsqueda interna** trabaja con elementos que se encuentran almacenados en la memoria principal de la máquina. Éstos pueden estar en estructuras estáticas —arreglos— o dinámicas —listas ligadas y árboles—. Los métodos de búsqueda interna más importantes son:

-  Secuencial o lineal
-  Binaria

- ▶ Por transformación de claves
- ▶ Árboles de búsqueda

### 9.2.1 Búsqueda secuencial

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, lo que ocurra primero. Normalmente cuando una función de búsqueda concluye con éxito, interesa conocer en qué posición fue hallado el elemento que se estaba buscando. Esta idea se puede generalizar para todos los métodos de búsqueda.

A continuación se presenta el algoritmo de búsqueda secuencial en arreglos desordenados.

**Algoritmo 9.1** Secuencial\_desordenado

#### **Secuencial\_desordenado** ( $V, N, X$ )

{Este algoritmo busca secuencialmente el elemento  $X$  en un arreglo unidimensional desordenado  $V$ , de  $N$  componentes}

{ $I$  es una variable de tipo entero}

1. Hacer  $I \leftarrow 1$
2. Mientras  $((I \leq N) \text{ y } (V[I] \neq X))$  Repetir  
    Hacer  $I \leftarrow I + 1$
3. {Fin del ciclo del paso 2}
4. Si  $(I > N)$   
    entonces  
        Escribir "La información no está en el arreglo"  
    sí no  
        Escribir "La información se encuentra en la posición",  $I$
5. {Fin del condicional del paso 4}

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

A continuación se presenta una variante de este algoritmo, pero utilizando recursividad, en lugar de iteratividad.

**Algoritmo 9.2** Secuencial\_desordenado\_recursivo**Secuencial\_desordenado\_recursivo ( $V, N, X, I$ )**

{Este algoritmo busca secuencialmente, y de forma recursiva, al elemento  $X$  en el arreglo unidimensional desordenado  $V$ , de  $N$  componentes}

{ $I$  es un parámetro de tipo entero que inicialmente se encuentra en 1}

1. Si ( $I > N$ )  
     entonces  
         Escribir "La información no se encuentra en el arreglo"  
     si no  
         1.1 Si ( $V[I] = X$ )  
             entonces  
                 Escribir "La información se encuentra en la posición",  $I$   
             si no  
                 Regresar a Secuencial\_desordenado\_recursivo con  $V, N, X$  e  $I + 1$   
         1.2 {Fin del condicional del paso 1.1}
2. {Fin del condicional del paso 1}

La búsqueda secuencial en arreglos ordenados es similar al caso anterior. Sin embargo, el orden entre los elementos del arreglo permite incluir una nueva condición que hace más eficiente al proceso. A continuación analizemos el algoritmo de búsqueda secuencial en arreglos ordenados.

**Algoritmo 9.3** Secuencial\_ordenado**Secuencial\_ordenado ( $V, N, X$ )**

{Este algoritmo busca secuencialmente al elemento  $X$  en un arreglo unidimensional ordenado  $V$ , de  $N$  componentes.  $V$  se encuentra ordenado crecientemente:  $V[1] \leq V[2] \leq \dots \leq V[N]$ }

{ $I$  es una variable de tipo entero}

1. Hacer  $I \leftarrow 1$
2. Mientras ( $(I \leq N)$  y  $(X > V[I])$ ) Repetir  
     Hacer  $I \leftarrow I + 1$
3. {Fin del ciclo del paso 2}
4. Si ( $(I > N)$  o  $(X < V[I])$ )  
     entonces  
         Escribir "La información no se encuentra en el arreglo"  
     si no  
         Escribir "La información se encuentra en la posición",  $I$
5. {Fin del condicional del paso 4}

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. Si  $(I > N)$  o si se comparó el elemento con un valor mayor a sí mismo ( $X < V[I]$ ), se está ante un caso de fracaso: el elemento no está en el arreglo. Si  $X = V[I]$  entonces se encontró al elemento en el arreglo.

A continuación se presenta el algoritmo de búsqueda en arreglos ordenados, pero en forma recursiva.

#### Algoritmo 9.4 Secuencial\_ordenado\_recursivo

##### Secuencial\_ordenado\_recursivo ( $V, N, X, I$ )

{Este algoritmo busca en forma secuencial y recursiva al elemento  $X$  en un arreglo unidimensional ordenado  $V$ , de  $N$  componentes.  $V$  se encuentra ordenado de manera creciente:  $V[1] \leq V[2] \leq \dots \leq V[N]$ .  $I$  inicialmente tiene el valor de 1}

1. Si  $((I \leq N) \text{ y } (X > V[I]))$   
     entonces  
         Llamar a Secuencial\_ordenado\_recursivo con  $V, N, X$  e  $I + 1$   
     si no
  - 1.1 Si  $((I > N) \text{ o } (X < V[I]))$   
     entonces  
         Escribir "La información no se encuentra en el arreglo"  
     si no  
         Escribir "La información se encuentra en la posición",  $I$
  - 1.2 {Fin del condicional del paso 1.1}
2. {Fin del condicional del paso 1}

El método de búsqueda secuencial también se puede aplicar a listas ligadas. Consiste en recorrer la lista nodo tras nodo, hasta encontrar al elemento buscado —éxito—, o hasta que lleguemos al final de la lista —fracaso—. La lista, como en el caso de arreglos, se puede encontrar ordenada o desordenada. El orden en el cual se puede recorrer la lista depende de sus características; puede ser simplemente ligada, circular o doblemente ligada. En este capítulo se presentará el caso de búsqueda secuencial en listas simplemente ligadas desordenadas. El lector, con los conocimientos que tiene sobre listas y búsqueda, puede implementar fácilmente los otros algoritmos.

#### Algoritmo 9.5 Secuencial\_lista\_desordenada

##### Secuencial\_lista\_desordenada ( $P, X$ )

{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}



1. Hacer  $Q \leftarrow P$
2. Mientras  $((Q \neq \text{NIL}) \text{ y } (Q^{\wedge}.\text{INFO} \neq X))$  Repetir  
    Hacer  $Q \leftarrow Q^{\wedge}.\text{LIGA}$
3. {Fin del ciclo del paso 2}
4. Si  $(Q = \text{NIL})$   
    entonces  
        Escribir "La información no se encuentra en la lista"  
    si no  
        Escribir "La información se encuentra en la lista"
5. {Fin del condicional del paso 4}

Si la lista estuviera ordenada se modificaría este algoritmo, incluyendo una condición similar a la que se escribió en el algoritmo 9.3. Esto último con el objetivo de disminuir el número de comparaciones.

A continuación se presenta la variante recursiva de este algoritmo de búsqueda secuencial en listas simplemente ligadas desordenadas.

#### Algoritmo 9.6 Secuencial\_lista\_desordenada\_recursivo

##### Secuencial\_lista\_desordenada\_recursivo ( $P, X$ )

{Este algoritmo busca de manera secuencial y en forma recursiva 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}

1. Si  $((P \neq \text{NIL}) \text{ y } (P^{\wedge}.\text{INFO} \neq X))$   
    entonces  
        Regresar a Secuencial\_lista\_desordenada\_recursivo con  $P^{\wedge}.\text{LIGA}$  y  $X$   
    si no
  - 1.1 Si  $(P = \text{NIL})$   
            entonces  
                Escribir "La información no se encuentra en la lista"  
            si no  
                Escribir "La información se encuentra en la lista"
  - 1.2 {Fin del condicional del paso 1.1}
2. {Fin del condicional del paso 1}

### Análisis de la búsqueda secuencial

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 favorable o desfavorable que se presenten.

Al buscar, por ejemplo, un elemento en un arreglo unidimensional desordenado de  $N$  componentes, puede suceder que ese valor no se encuentre; por lo tanto, se harán  $N$

comparaciones al recorrer todo el arreglo. Por otra parte, si el elemento se encuentra en el arreglo, éste puede estar en la primera posición, en la última o en alguna intermedia. Si es el primero, se hará una comparación; si se trata del último, se harán  $N$  comparaciones; y si se encuentra en la posición  $i$  ( $1 < i < N$ ), entonces se realizarán  $i$  comparaciones.

Ahora bien, el número de comparaciones que se llevan a cabo si trabajamos con arreglos ordenados será el mismo que para desordenados, siempre y cuando el elemento se encuentre en el arreglo. Si no fuera éste el caso, entonces el número de comparaciones disminuirá sensiblemente en arreglos ordenados, siempre que el valor buscado esté comprendido entre el primero y el último elementos del arreglo.

Por otra parte, el número de comparaciones en la búsqueda secuencial en listas simplemente ligadas es el mismo que para arreglos. En la fórmula 9.1 se presentan los números mínimo, mediano y máximo de comparaciones que se ejecutan cuando se trabaja con la búsqueda secuencial.

$$C_{\min} = 1 \quad C_{\text{med}} = \frac{(1+n)}{2} \quad C_{\max} = N \quad \text{Fórmula 9.1}$$

La tabla 9.1 presenta, para distintos valores de  $N$ , los números mínimo, mediano y máximo de comparaciones que se requieren para buscar secuencialmente un elemento en un arreglo o lista ligada.

## 9.2.2 Búsqueda binaria

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. Para el 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 de búsqueda. El proceso concluye cuando el elemento es encontrado, o cuando el intervalo de búsqueda se anula, es vacío.

El método de búsqueda binaria funciona exclusivamente con arreglos ordenados. No se puede utilizar con listas simplemente ligadas —no podríamos 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 com-

**TABLA 9.1**  
Complejidad del método  
de búsqueda secuencial

$N$	$C_{\min}$	$C_{\text{med}}$	$C_{\max}$
10	1	5.5	10
100	1	50.5	100
500	1	250.5	500
1 000	1	500.5	1 000
10 000	1	5 000.5	10 000

paraciones a realizar disminuye notablemente. Esta disminución resulta significativa cuanto más grande sea el tamaño del arreglo. A continuación se presenta el algoritmo de búsqueda binaria.

#### Algoritmo 9.7 Binaria

##### Binaria ( $V, N, X$ )

{Este algoritmo busca al elemento  $X$  en un arreglo unidimensional ordenado  $V$  de  $N$  componentes}

{IZQ, CEN y DER son variables de tipo entero. BAN es una variable de tipo booleano}

1. Hacer  $IZQ \leftarrow 1$ ,  $DER \leftarrow N$  y  $BAN \leftarrow \text{FALSO}$
2. Mientras  $((IZQ \leq DER) \text{ y } (BAN = \text{FALSO}))$  Repetir
  - 2.1 Si  $(X = V[CEN])$ 

entonces

Hacer  $BAN \leftarrow \text{VERDADERO}$

si no {Se redefine el intervalo de búsqueda}

    - 2.1.1 Si  $(X > V[CEN])$ 

entonces

Hacer  $IZQ \leftarrow CEN + 1$

si no

Hacer  $DER \leftarrow CEN - 1$
    - 2.1.2 {Fin del condicional del paso 2.1.1}
  - 2.2 {Fin del condicional del paso 2.1}
3. {Fin del ciclo del paso 2}
4. Si  $(BAN = \text{VERDADERO})$ 

entonces

Escribir "La información está en la posición", CEN

si no

Escribir "La información no se encuentra en el arreglo"
5. {Fin del condicional del paso 4}

Analicemos ahora un ejemplo para ilustrar el funcionamiento de este algoritmo.

#### Ejemplo 9.1

Sea  $V$  un arreglo unidimensional de números enteros, ordenado de manera creciente, como se muestra en la figura 9.3.

En la tabla 9.2 se presenta el seguimiento del algoritmo 9.7 cuando  $X$  es igual a 325 ( $X = 325$ ).

En la figura 9.4 se observa gráficamente, para este caso en particular, cómo se va reduciendo el intervalo de búsqueda.

FIGURA 9.3

V									
101	215	325	410	502	507	600	610	612	670
1	2	3	4	5	6	7	8	9	10



TABLA 9.2

Búsqueda binaria

Paso	BAN	IZQ	DER	CEN	$X = V[CEN]$	$X > V[CEN]$
1	Falso	1	10	5	$325 = 502$ ? No	$325 > 502$ ? No
2	Falso	1	4	2	$325 = 215$ ? No	$325 > 215$ ? Sí
3	Falso	3	4	3	$325 = 325$ ? Sí	
4	Verdadero					

La tabla 9.3, por otra parte, muestra nuevamente el seguimiento del algoritmo 9.7 para  $X = 615$ , valor que no se encuentra en el arreglo.

La figura 9.5 representa gráficamente cómo se va reduciendo el intervalo de búsqueda hasta anularse ( $DER < IZQ$ ).

A continuación se presenta una variante del algoritmo de búsqueda binaria que no utiliza bandera —BAN—.

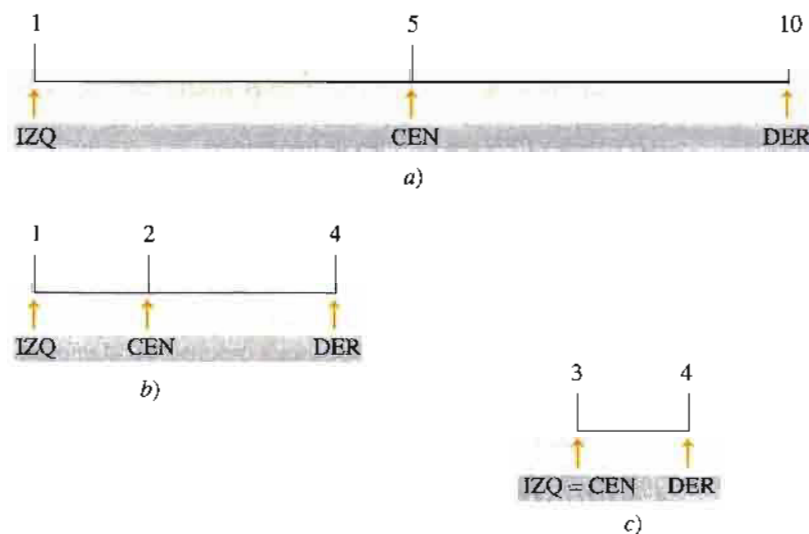
TABLA 9.3

Búsqueda binaria

Paso	BAN	IZQ	DER	CEN	$X = V[CEN]$	$X > V[CEN]$
1	Falso	1	10	5	$615 = 502$ ? No	$615 > 502$ ? Sí
2	Falso	6	10	8	$615 = 610$ ? No	$615 > 610$ ? Sí
3	Falso	9	10	9	$615 = 612$ ? No	$615 > 612$ ? Sí
4	Falso	10	10	10	$615 = 670$ ? No	$615 > 670$ ? No
5	Falso	10	9			

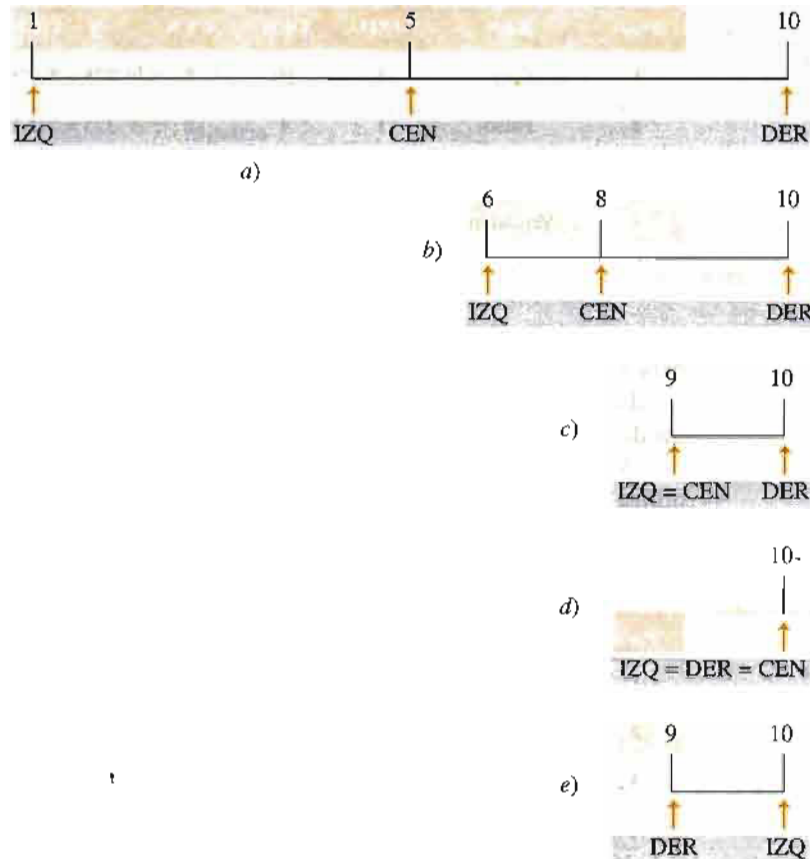
FIGURA 9.4

Reducción del intervalo de búsqueda. a) Paso 1. b) Paso 2. c) Paso 3 de la tabla 9.2.



**FIGURA 9.5**

Reducción del intervalo de búsqueda. a) Paso 1. b) Paso 2. c) Paso 3. d) Paso 4. e) Paso 5 de la tabla 9.3.

**Algoritmo 9.8** Binaria\_sin\_bandera**Binaria\_sin\_bandera** ( $V, N, X$ )

{Este algoritmo busca al elemento  $X$  en el arreglo unidimensional ordenado  $V$  de  $N$  componentes}

{IZQ, DER y CEN son variables de tipo entero}

1. Hacer  $IZQ \leftarrow 1$ ,  $DER \leftarrow N$  y  $CEN \leftarrow \text{PARTE ENTERA } ((IZQ + DER)/2)$

2. Mientras  $((IZQ \leq DER) \text{ y } (X \neq V[CEN]))$  Repetir

2.1 Si  $X > V[CEN]$

entonces

Hacer  $IZQ \leftarrow CEN + 1$

si no

Hacer  $DER \leftarrow CEN - 1$

2.2 {Fin del condicional del paso 2.1}

Hacer  $CEN \leftarrow \text{PARTE ENTERA } ((IZQ + DER)/2)$

3. {Fin del ciclo del paso 2}

4. Si  $(IZQ > DER)$

```

    entonces
        Escribir "La información no se encuentra en el arreglo"
    si no
        Escribir "La información se encuentra en la posición", CEN
5. {Fin del condicional del paso 4}

```

Finalmente, se presenta una versión recursiva de este algoritmo de búsqueda binaria.

#### Algoritmo 9.9 Binaria\_recursivo

##### Binaria\_recursivo (V, IZQ, DER, X)

{Este algoritmo busca al elemento  $X$  en el arreglo unidimensional ordenado  $V$  de  $N$  componentes. 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}

```

1. Si (IZQ ≥ DER)
    entonces
        Escribir X, "No se encuentra en el arreglo"
    si no
        Hacer CEN ← PARTE ENTERA ((DER + IZQ)/2)
1.1 Si (X = V[CEN])
    entonces
        Escribir "El dato buscado se encuentra en la posición", CEN
    si no
1.1.1 Si (X > V[CEN])
        entonces
            Regresar a Binaria_recursivo con V, CEN + 1, DER, X
        si no
            Regresar a Binaria_recursivo con V, IZQ, CEN - 1, X
1.1.2 {Fin del condicional del paso 1.1.1}
1.2 {Fin del condicional del paso 1.1}
2. {Fin del condicional del paso 1}

```

### Análisis de la búsqueda binaria

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 aproximadamente  $\log_2(n)$  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.

$$C_{\min} = 1 \quad C_{\text{med}} = \frac{(1 + \log_2(N))}{2} \quad C_{\max} = \log_2(N) \quad \text{Fórmula 9.2}$$

En la tabla 9.4 se presentan, para distintos valores de  $N$ , los números mínimo, mediano y máximo de comparaciones requeridas para buscar un elemento en un arreglo, aplicando el método de búsqueda binaria.

Si se comparan los valores de la tabla 9.1 con los de la tabla 9.4 resulta claro que el método de búsqueda binaria es más eficiente que el método de búsqueda secuencial. Además, la diferencia se hace más significativa conforme más grande sea el tamaño del arreglo. Sin embargo, no hay que olvidar que el método de búsqueda binaria trabaja solamente con arreglos ordenados; por lo tanto, si el arreglo estuviera desordenado antes de emplear este método, aquél debería ordenarse.

Cabe destacar, sin embargo, que la ordenación de un arreglo también implica comparaciones y movimientos de elementos, así que si se va a realizar sólo una búsqueda sobre un arreglo desordenado conviene utilizar el método secuencial. En cambio, si se realizan búsquedas en forma continua, convendría ordenarlo para poder aplicar el método de búsqueda binaria.

### 9.2.3 Búsqueda por transformación de claves

Los dos métodos analizados anteriormente permiten encontrar un elemento en un arreglo. En ambos casos el tiempo de búsqueda es proporcional a su número de componentes. Es decir, a mayor número de elementos se debe realizar mayor número de comparaciones. Se mencionó además que si bien el método de búsqueda binaria es más eficiente que el secuencial, existe la restricción de que el arreglo se debe encontrar ordenado.

Esta sección se dedica a un nuevo método de búsqueda. Este método, conocido como **transformación de claves** o *hash*, permite aumentar la velocidad de búsqueda sin necesidad de tener los elementos ordenados. Cuenta con la ventaja de que el tiempo de búsqueda es independiente del número de componentes del arreglo.

Supongamos que se tiene una colección de datos, cada uno de ellos identificado por una clave. Es claro que resulta atractivo tener acceso a ellos de manera directa; es decir, sin recorrer algunos datos antes de localizar al buscado. El método por transformación de claves permite realizar justamente esta actividad; es decir, localizar el dato en forma

**TABLA 9.4**  
Complejidad del método de  
búsqueda binaria

$N$	$C_{\min}$	$C_{\text{med}}$	$C_{\max}$
10	1	2.5	4
100	1	4	7
500	1	5	9
1 000	1	5.5	10
10 000	1	7.5	14