Algoritmos de Búsqueda

Programación Lógica

UNIVERSIDAD SIGLO 21 | MIEMBRO DE LA RED ILUMNO

1. Búsqueda secuencial

En la algoritmia, la problemática de búsqueda hace referencia a la definición sobre la existencia o no de un elemento dentro de un conjunto. Existen distintos algoritmos que pueden ser utilizados para buscar un elemento y su elección dependerá de la forma en la que los elementos del conjunto se encuentren organizados.

El algoritmo de búsqueda secuencial es utilizado cuando los elementos del conjunto no se encuentran ordenados. Este algoritmo consiste en recorrer secuencialmente la estructura de datos para comparar cada uno de los elementos con el valor que se desea buscar. Joyanes Aguilar (2008), en *Fundamentos de programación*, propone distintas alternativas de algoritmos de búsqueda secuencial para buscar un elemento en un arreglo unidimensional (vector).

A continuación, expondremos una de las variantes de este algoritmo. Suponiendo un arreglo unidimensional *A* de *N* elementos, si se desea buscar el valor *K*, el pseudocódigo de búsqueda secuencial es el siguiente:

```
leer(K)
para i ← 1 hasta N hacer
        si A[i] == K entonces
                escribir("Elemento econtrado en la posición ", i)
        fin-si
fin-para
```

En el pseudocódigo anterior, para cada ocurrencia en la que se encuentre el valor *K* en el arreglo *A*, se mostrará un mensaje informando la posición en la que se ha encontrado el valor. El recorrido del arreglo se puede realizar con cualquiera de las estructuras repetitivas. Se recomienda revisar las distintas variantes de este método de búsqueda en la bibliografía de Joyanes Aguilar (2008), en *Fundamentos de programación*.

Si el valor que buscar se encuentra una sola vez en el arreglo, entonces el peor escenario es que dicho valor se encuentre en la última posición o no se encuentre. En este caso se deben realizar *N* comparaciones. En el mejor de los escenarios, si el valor se encuentra en la primera posición, entonces se realiza una sola comparación. Esto indica que la cantidad promedio de comparaciones que realiza este método de búsqueda es $(N + 1)/2$.

2. Búsqueda binaria

La búsqueda secuencial puede resultar muy lenta si el número de elementos del conjunto es grande. El algoritmo de búsqueda binaria reduce considerablemente los tiempos de búsqueda, pero es requisito para aplicar este algoritmo que los elementos del conjunto se encuentren ordenados. Este algoritmo utiliza la técnica de "divide y vencerás". Se compara si el elemento central del conjunto ordenado es el valor que se busca. En el caso de que lo sea, se finaliza la búsqueda; en caso contrario, se debe verificar si el valor que buscar es más chico que el valor central para continuar trabajando con la primera mitad de valores del conjunto. Si el valor que buscar es más grande que el elemento central, entonces se debe continuar trabajando con la segunda mitad de valores del conjunto. Con la sublista de elementos, se debe repetir el proceso de comparación del valor que buscar con el elemento central, pero en este caso de la sublista. Si el elemento central de la sublista no es el valor que se busca, se prosigue con la subdivisión del conjunto hasta determinar la existencia, o no, del valor.

Suponiendo un arreglo unidimensional *X* ordenado, de *N* elementos, si se desea buscar el valor *K*, el pseudocódigo de búsqueda binaria es el siguiente:

```
leer(K)
BAJO ← 1
ALTO ← N
CENTRAL ← (BAJO + ALTO) div 2
mientras (BAJO <= ALTO) Y ( X[CENTRAL] <> K ) hacer
    si K < X[CENTRAL] entonces
        ALTO ← CENTRAL - 1
    si-no
        BAJO ← CENTRAL + 1
    fin-si
    CENTRAL ← (BAJO + ALTO) div 2
fin-mientras
si K == X[CENTRAL] entonces
    escribir("Valor encontrado en la posición ", CENTRAL)
fin-si
```

Ejemplo de búsqueda binaria

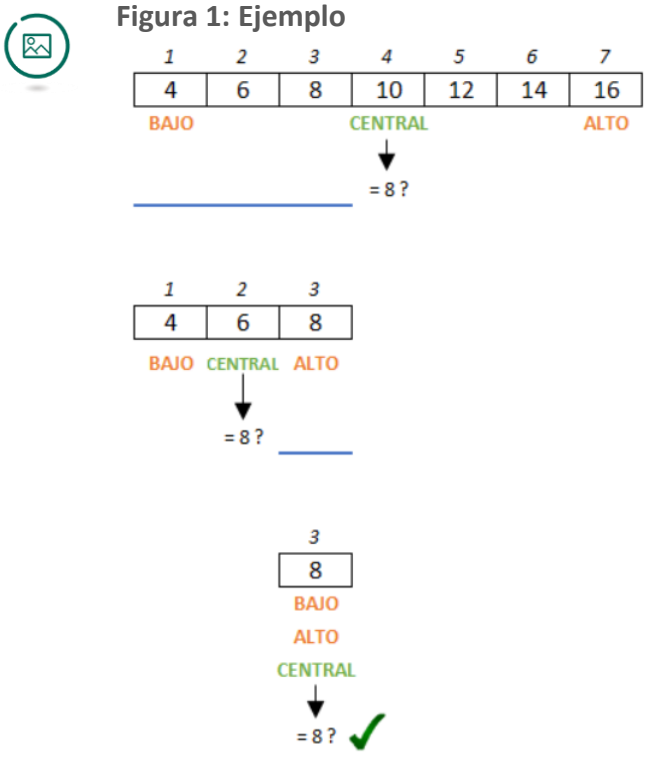
Arreglo inicial:

1	2	3	4	5	6	7
4	6	8	10	12	14	16

K = 8

En este escenario el valor que buscar es 8.

Figura 1: Ejemplo



K	BAJO	ALTO	CENTRAL
8	1	7	4
		3	2
	3		3

Fuente: elaboración propia.

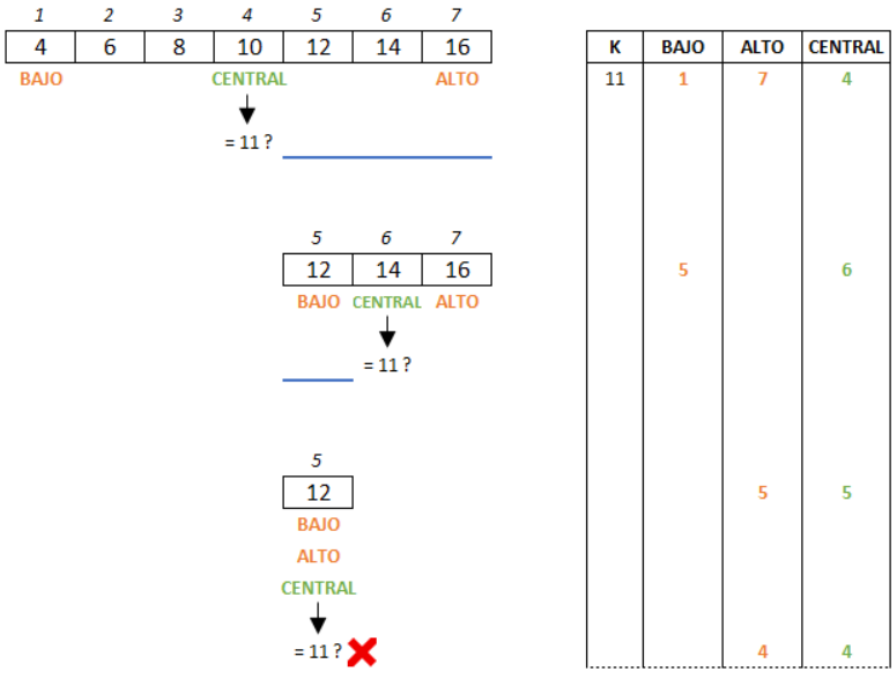
En el siguiente escenario, el valor que buscar es 11.

Arreglo inicial:

1	2	3	4	5	6	7
4	6	8	10	12	14	16

K = 11

Figura 2: Ejemplo



Fuente: elaboración propia.

El método de búsqueda binaria requiere que el conjunto de elementos se encuentre ordenado, pero, una vez cumplido este requisito, la cantidad de comparaciones necesarias para buscar un elemento se reduce considerablemente. Para un arreglo de N elementos, en el peor de los casos, se necesitarán $\log_2(N + 1)$ comparaciones y, en el mejor de los casos, se realizará una comparación. Por lo tanto, el método de búsqueda binaria necesita realizar, en promedio, la siguiente cantidad de comparaciones para encontrar un elemento:

$$\frac{1 + \log2(N + 1)}{2}$$

2. Ejemplo de algoritmo integrador de ordenamiento y búsqueda

A continuación, se expone un algoritmo utilizando el lenguaje de pseudocódigo del programa PSeInt. El algoritmo crea un vector y despliega por pantalla un menú de opciones para ordenar y buscar algún elemento del arreglo.

```
Algoritmo Ordenamiento
    Definir N, opcionIngresada, i, j, cantidadComparaciones, AUX, k, cantidadOperacionesBusqueda, ALTO, BAJO, CENTRAL
Como Entero
    Definir sw, encontrado Como Logico

    Escribir "Algoritmo de ordenamiento y búsqueda"

    Escribir "Ingrese la cantidad de elementos con los que va trabajar"
    Leer N

    Dimension X(N)

    Repetir
        Escribir ""
        Escribir "Menú de operaciones"
        Escribir "1. Generar vector aleatorio"
        Escribir "2. Mostrar vector"
        Escribir "3. Ordenar vector por método de burbuja"
        Escribir "4. Ordenar vector por método de burbuja mejorada"
        Escribir "5. Ordenar vector por método de inserción"
        Escribir "6. Ordenar vector por método de selección"
        Escribir "7. Buscar un elemento (búsqueda secuencial)"
        Escribir "8. Buscar un elemento (búsqueda binaria)"
        Escribir "0. Salir"
        Escribir Sin Saltar "-- Ingrese una opción"
        Leer opcionIngresada
        Escribir ""

        Segun opcionIngresada Hacer
            1:
                Escribir "<< Generación aleatoria del vector >>"
                Para i<-1 hasta 20 hacer
                    X[i]<-azar(100)
                FinPara
            2:
                Escribir "<< Mostrando elementos del vector >>"
                Para i<-1 hasta N hacer
                    Escribir Sin Saltar X[i],"  "
                FinPara
                Escribir ""
            3:
```

```
Escribir "<< Ordenamiento por método burbuja >>"
cantidadComparaciones <-0
Para i<-1 hasta N-1 hacer
    Para j<-1 hasta N-1 hacer
        Si X[j] > X[j+1] entonces
            AUX<-X[j]
            X[j]<-X[j+1]
            X[j+1]<-AUX
        FinSi
        cantidadComparaciones<-cantidadComparaciones+1
    FinPara
FinPara
Escribir "Cantidad de comparaciones realizadas: ",cantidadComparaciones
4:
Escribir "<< Ordenamiento por método burbuja mejorada >>"
cantidadComparaciones <-0
Para i<-1 hasta N-1 hacer
    Para j<-1 hasta N-i hacer
        Si X[j] > X[j+1] entonces
            AUX<-X[j]
            X[j]<-X[j+1]
            X[j+1]<-AUX
        FinSi
        cantidadComparaciones<-cantidadComparaciones+1
    FinPara
FinPara
Escribir "Cantidad de comparaciones realizadas: ",cantidadComparaciones
5:
Escribir "<< Ordenamiento por método de inserción >>"
Para i<-2 hasta N hacer
    AUX <- X[i]
    k <- i-1
    sw <- Falso
    Mientras NO(sw) y (k >= 1) hacer
        Si AUX < X[k] entonces
            X[k+1] <- X[k]
            k <- k-1
        SiNo
            sw <- Verdadero
        FinSi
    FinMientras
    X[k+1] <- AUX
FinPara
6:
Escribir "<< Ordenamiento por método de selección >>"
cantidadComparaciones <-0
Para i<-1 hasta N-1 hacer
    AUX <- X[i]
    k <- i
    Para j<-i+1 hasta N hacer
        Si X[j] < AUX entonces
            AUX<-X[j]
            k <- j
        FinSi
        cantidadComparaciones<-cantidadComparaciones+1
    FinPara
    X[k] <- X[i]
    X[i] <- AUX
FinPara
Escribir "Cantidad de comparaciones realizadas: ",cantidadComparaciones
```

```
        7:
            Escribir "<< Búsqueda secuencial de un elemento >>"
            Escribir Sin Saltar "Ingrese el elemento que desea buscar"
            Leer K
            encontrado <- Falso
            Para i<-1 hasta N hacer
                Si X[i] = K entonces
                    Escribir "Se ha encontrado el elemento en la posición ",i
                    encontrado <- Verdadero
                FinSi
            FinPara
            Si encontrado = Falso entonces
                Escribir "No se ha encontrado el elemento"
            FinSi
        8:
            Escribir "<< Búsqueda binaria de un elemento >>"
            Escribir Sin Saltar "Ingrese el elemento que desea buscar"
            Leer K

            cantidadOperacionesBusqueda <- 0

            BAJO <- 1
            ALTO <- N
            CENTRAL <- redon((BAJO + ALTO) / 2)
            Mientras (BAJO <= ALTO) Y (X[CENTRAL] <> K) hacer
                Si K < X[CENTRAL] entonces
                    ALTO <- CENTRAL - 1
                SiNo
                    BAJO <- CENTRAL + 1
                FinSi
                CENTRAL <- redon((BAJO + ALTO) / 2)
                cantidadOperacionesBusqueda <- cantidadOperacionesBusqueda + 1
            FinMientras
            Si K=X[CENTRAL] Entonces
                Escribir "Se ha encontrado el elemento en la posición ",CENTRAL
            SiNo
                Escribir "No se ha encontrado el elemento"
            FinSi
            Escribir "Cantidad de operaciones de búsqueda realizadas: ",cantidadOperacionesBusqueda
        0:
            Escribir "<< Gracias por utilizar el programa. Hasta luego! >>"
        De Otro Modo:
            Escribir "<< Opción ingresada incorrecta >>"
    Fin Segun

Hasta Que opcionIngresada = 0

FinAlgoritmo
```

Referencias

Joyanes Aguilar, L. (2008). *Fundamentos de programación. Algoritmos, estructura de datos y objetos*. Madrid, ES: McGraw-Hill.