




Algorítmica y Lenguajes de Programación

Búsqueda



Búsqueda. Introducción

- Hace dos lecciones se dijo que había tres tratamientos básicos sobre vectores:
 - Recorrido.
 - Ordenación.
 - Búsqueda.
- Durante las últimas clases hemos estudiado distintos métodos de ordenación de vectores; pudiendo adaptarse dichos métodos a otras estructuras de datos.
- **En la lección de hoy se estudiarán distintos algoritmos de búsqueda.**
- Los algoritmos de búsqueda tienen como finalidad localizar un elemento dado dentro de una estructura de datos o determinar su inexistencia.
- Naturalmente, la búsqueda será mucho más eficiente si la estructura de datos (en nuestro caso el vector) está ordenada.
- Así, los algoritmos que estudiaremos servirán para realizar **búsquedas en vectores ordenados**.



Búsqueda. Algoritmos de búsqueda

- A lo largo de esta lección estudiaremos los siguientes algoritmos de búsqueda:
 - Búsqueda secuencial.
 - Búsqueda secuencial con centinela.
 - Búsqueda binaria (o dicotómica)
 - Búsqueda por interpolación.
- Como en el caso de los métodos de ordenación, se presentará pseudocódigo, código FORTRAN y análisis de la complejidad para cada uno de los algoritmos.

3



Búsqueda. Búsqueda secuencial (i)

- El algoritmo de búsqueda más sencillo recorre el vector desde el primer elemento hasta el último y si encuentra el valor buscado retorna su posición.
- Obviamente, se trata del **único algoritmo posible si el vector no está ordenado**.
- Sin embargo, **si el vector está ordenado resulta muy ineficiente**.

4



Búsqueda. Búsqueda secuencial (ii)

```
entero función busquedaSecuencial (v ∈ vector(MAX) de elemento, valor
    ∈ elemento)
variables
    i, posicion ∈ entero

    posicion ← -1

    desde i ← 1 hasta MAX
        si v(i)=valor entonces
            posicion ← i
        fin si
    fin desde

    busquedaSecuencial ← posicion
fin función
```

5



Búsqueda. Búsqueda secuencial (iii)

```
integer function busquedaSecuencial (vector,valor)
implicit none
    integer vector(max)
    integer valor
    integer i,posicion

    posicion=-1

    do i=1,max
        if (vector(i)==valor) then
            posicion=i
        end if
    end do

    busquedaSecuencial=posicion
end function
```


6



Búsqueda. Búsqueda secuencial (iv)

- Como se puede apreciar el algoritmo **realiza siempre el mismo número de iteraciones independientemente de la posición en que se encuentre el valor buscado**.
- El número de iteraciones siempre es igual al tamaño del vector y, puesto que todas las operaciones del interior del bucle tienen coste unitario, **la complejidad del algoritmo de búsqueda secuencial es $O(n)$** .
- El algoritmo de búsqueda secuencial resulta muy ineficiente puesto que no se aprovecha del hecho de que el vector esté ordenado.
- Es posible modificarlo para que, teniendo en cuenta esa circunstancia, **finalice en cuanto localice el elemento o se determine que éste no existe**.
- **Esta modificación se conoce como búsqueda secuencial con centinela**.

7



Búsqueda. Búsqueda secuencial con centinela (i)

- Si el vector está **ordenado** no es necesario **recorrerlo completamente**.
- En cuanto el valor buscado es encontrado se puede finalizar el recorrido y en el momento en que aparece un valor **mayor que el buscado** también se puede detener la ejecución.
- Al aplicar esto al algoritmo de búsqueda secuencial se obtiene el **algoritmo de búsqueda con centinela**.

8

Búsqueda. Búsqueda secuencial con centinela (ii)

```
entero funcion busquedaSecuencialCentinela (v  $\hat{I}$  vector(MAX) de elemento, valor
 $\hat{I}$  elemento)
variables
  i  $\hat{I}$  entero

  i $\leftarrow$ 1

  mientras v(i)<valor y i<MAX hacer
    i $\leftarrow$ i+1
  fin mientras

  si v(i)=valor entonces
    busquedaSecuencialCentinela $\leftarrow$ i
  si no
    busquedaSecuencialCentinela $\leftarrow$ -1
  fin si
fin funcion
```

9

Búsqueda. Búsqueda secuencial con centinela (iii)

```
integer function busquedaSecuencialCentinela (vector,valor)
implicit none
  integer vector(max)
  integer valor
  integer i

  i=1

  do while ((vector(i)<valor).and.(i<max))
    i=i+1
  end do

  if (vector(i)==valor) then
    busquedaSecuencialCentinela=i
  else
    busquedaSecuencialCentinela=-1
  end if
end function
```

10

Búsqueda. Búsqueda secuencial con centinela (iv)

- Como se puede apreciar, la búsqueda secuencial con centinela no precisa recorrer el vector por completo.
- El **caso mejor** se produce cuando el elemento a buscar es el primero del vector o menor que todos los elementos del vector. Entonces la complejidad es **$O(1)$** .
- El **caso peor** se da cuando el elemento a buscar es el último del vector o mayor que todos los elementos del vector. Entonces la complejidad es **$O(n)$** .
- Por término medio, el algoritmo emplea del orden de $n/2$ iteraciones, por lo cual la complejidad del **caso medio** también sería **$O(n)$** .

11

Búsqueda. Búsqueda binaria (i)

- La idea básica de la búsqueda binaria o dicotómica es **reducir el tamaño del problema a la mitad en cada iteración**.
- Cuando el tamaño del problema, esto es del vector, sea 1 se puede resolver la búsqueda de forma directa.
- La estrategia empleada por este algoritmo es muy similar a la del juego "patata caliente"...

-¿Altura del Everest?

-7000

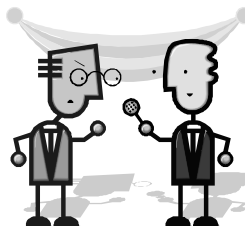
-Mayor

-9000

-Menor

-8000

...



12

Búsqueda. Búsqueda binaria (ii)

```
entero funcion busquedaBinaria (v  $\hat{I}$  vector(MAX) de elemento, valor  $\hat{I}$  elemento)
variables
  izq,der,medio,posicion  $\hat{I}$  entero
  encontrado  $\hat{I}$  logico

  encontrado $\leftarrow$ falso
  posicion $\leftarrow$ -1
  izq $\leftarrow$ 1
  der $\leftarrow$ MAX
  mientras no encontrado y izq  $\neq$  der hacer

    medio $\leftarrow$   $\frac{izq+der}{2}$ 

    si v(medio)=valor entonces
      posicion $\leftarrow$ medio
      encontrado $\leftarrow$ verdadero
    si no
      si v(medio)>valor entonces
        der $\leftarrow$ medio-1
      si no
        izq $\leftarrow$ medio+1
      fin si
    fin si
  fin mientras
  busquedaBinaria $\leftarrow$ posicion
fin funcion
```

13

Búsqueda. Búsqueda binaria (iii)

```
integer function busquedaBinaria (vector,valor)
implicit none
  integer vector(max), valor
  integer izq,der,medio,posicion
  logical encontrado

  encontrado=.false.
  posicion=-1
  izq=1
  der=max

  do while ((.not.encontrado).and.(izq<=der))
    medio=(izq+der)/2
    if (vector(medio)==valor) then
      posicion=medio
      encontrado=.true.
    else
      if (vector(medio)>valor) then
        der=medio-1
      else
        izq=medio+1
      end if
    end if
  end do

  busquedaBinaria=posicion
end function
```

14

Búsqueda. Búsqueda binaria (iv)

- En el vector (1,1,2,3,4,5,5,5,6,9) se va a buscar, de forma binaria, el número 7:

V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	V ₉	V ₁₀
1	1	2	3	4	5	5	5	6	9

V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	V ₉	V ₁₀
1	1	2	3	4	5	5	5	6	9

V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	V ₉	V ₁₀
1	1	2	3	4	5	5	5	6	9

V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	V ₉	V ₁₀
1	1	2	3	4	5	5	5	6	9

V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	V ₉	V ₁₀
1	1	2	3	4	5	5	5	6	9

V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	V ₉	V ₁₀
1	1	2	3	4	5	5	5	6	9

V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	V ₉	V ₁₀
1	1	2	3	4	5	5	5	6	9

- Obsérvese, que bastan 4 iteraciones para determinar que el número 7 no se encuentra en el vector.

15

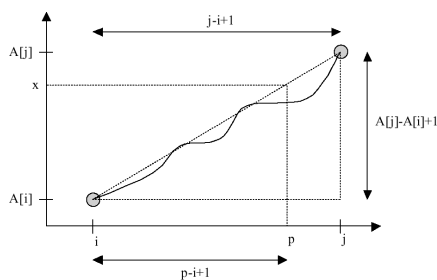
Búsqueda. Búsqueda binaria (v)

- El algoritmo de búsqueda binaria divide el tamaño del vector por 2 hasta quedarse con un vector de tamaño 1.
- Al llegar al tamaño unitario se puede determinar directamente si el elemento buscado existe o no.
- Así, el algoritmo de búsqueda binaria realiza $\log_2(n)$ iteraciones y, por tanto, su complejidad es **$O(\log n)$** .

16

Búsqueda. Búsqueda por interpolación (i)

- El algoritmo de búsqueda binaria siempre selecciona el elemento central del vector para compararlo con el elemento a buscar y dividir el vector.
- Es posible realizar una modificación a este algoritmo de tal forma que el elemento seleccionado no sea el central sino aquel que se "correspondería" con el elemento buscado si la distribución de valores en el vector fuera uniforme.**



Obtención de la posición siguiente a comprobar en el algoritmo de búsqueda por interpolación. La posición, p , se obtiene como resultado de suponer una distribución uniforme de los elementos del subvector $T[i..j]$

La complejidad de este algoritmo es también de $O(\log n)$ aunque la demostración está más allá del ámbito de este curso.

17

Búsqueda. Búsqueda por interpolación (ii)

```

entero funcion busquedaInterpolacion (v P vector(max) de elemento, valor P elemento)
variables
  izq, der, presunto P entero

  izq ← 1
  der ← max

  mientras (v(der) ≥ valor) y (v(izq) < valor)

    presunto ←  $\left\lceil \frac{\text{valor} - v(\text{izq})}{v(\text{der}) - v(\text{izq})} \cdot (\text{der} - \text{izq}) \right\rceil + \text{izq}$ 

    si valor > v(presunto) entonces
      izq ← presunto + 1
    si no
      si valor < v(presunto) entonces
        der ← presunto - 1
      si no
        izq ← presunto
    fin si
  fin si
fin mientras

si v(izq) = valor entonces
  busquedaInterpolacion ← izq
si no
  busquedaInterpolacion ← -1
fin si
fin funcion
  
```

18

Búsqueda. Búsqueda por interpolación (iii)

```
integer function busquedaInterpolacion (vector,valor)
implicit none
integer vector(max), valor
integer izq,der,presunto

izq=1
der=max

do while ((vector(der)>=valor).and.(vector(izq)<valor))
  presunto=((valor-vector(izq))/(vector(der)-vector(izq))*(der-izq))+izq

  if (valor>vector(presunto)) then
    izq=presunto+1
  else
    if (valor<vector(presunto)) then
      der=presunto-1
    else
      izq=presunto
    end if
  end if
end do

if (vector(izq)==valor) then
  busquedaInterpolacion=izq
else
  busquedaInterpolacion=-1
end if
end function
```

19

Búsqueda. Resumen

- La búsqueda es una de las operaciones de tratamiento de vectores más habituales.
- Una búsqueda en un vector no ordenado tendría una complejidad $O(n)$; por esa razón es preferible implementar algoritmos de búsqueda en vectores ordenados.
- Hemos estudiado 4 algoritmos de búsqueda:
 - Búsqueda secuencial.
 - Búsqueda secuencial con centinela.
 - Búsqueda binaria.
 - Búsqueda por interpolación.
- El primer algoritmo es el más simple de los cuatro, recorre completamente el vector y no precisa que el vector es ordenado; su complejidad es $O(n)$.
- El segundo algoritmo recorre el vector hasta que se encuentra el valor buscado o se determina que éste no existe. Su complejidad es $O(n)$.
- El algoritmo de búsqueda binaria divide en cada iteración el vector en dos realizando la búsqueda en una sola de las mitades. Su complejidad es $O(\log n)$.
- La búsqueda por interpolación es una modificación del algoritmo de búsqueda binaria, también de complejidad $O(\log n)$.

20