Búsqueda

Eva Lucrecia Gibaja Galindo
Dpto. Informática y Análisis Numérico

ojo! para búsqueda dicotómica necesitamos el vector ordenado

Búsqueda en un vector

- **Recorrido con tratamiento selectivo**
  - La información puede aparecer más de una ocasión
  - Hay que recorrer completamente el vector
- **Búsqueda secuencial o lineal**
  - La información aparece como máximo una vez
  - La búsqueda debe detenerse al encontrarla
  - Devolvemos la posición del elemento si ha sido encontrado en el vector, -1 en caso contrario
- **Búsqueda dicotómica o binaria**
  - La información está ordenada respecto a algún criterio
  - Podemos evitar el recorrido secuencial

Tratamiento selectivo

- Recorrido secuencial y completo del vector para realizar una operación con los elementos que cumplen una determinada condición

```
int busquedaSelectivaSumaPares(int* Vector,
int tope)
{
  int i, suma=0;
  for(i=0; i<tope; i++)
  {
    if(Vector[i]%2==0)
    {
      suma = suma+Vector[i];
    }
  }
  return(suma);
}
```

tratamiento selectivo

Búsqueda secuencial o lineal

- Recorrido secuencial de un vector para encontrar la posición de un elemento

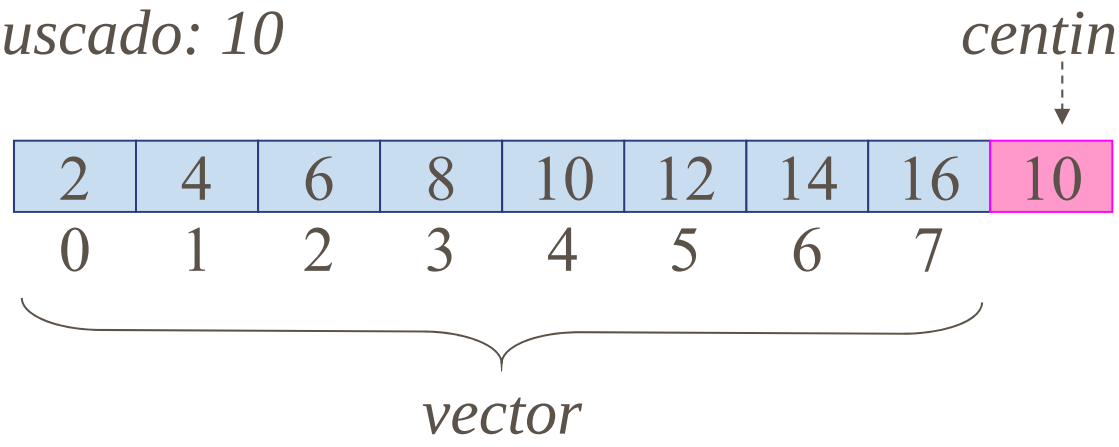
```
int busquedaSecuencialFor(int* Vector, int tope, int elemento)
{
  int i, encontrado=0, posicion=-1;

  for(i=0; (i<tope)&&(!encontrado); i++)
  {
     if(Vector[i]==elemento)
     {
       posicion = i;
       encontrado = 1;
     }
  }
  return(posicion);
}
```

La variable "*encontrado*" indica si el elemento esta en el vector

Búsqueda lineal con centinela

- Reserva una posición adicional al final e introduce en ella el elemento a buscar (centinela) ⇨ la condición encontrado siempre será cierta. Si el valor devuelto es la posición del centinela el elemento no esta en el vector
- Evita comparaciones y gana en rapidez
- La restricción que impone es el deber de reservar siempre una posición adicional en el vector donde se realiza la búsqueda

Elemento buscado: 10

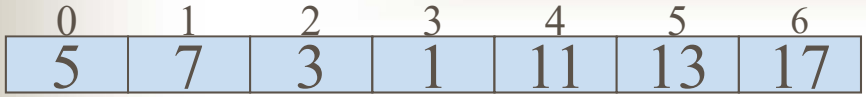


Búsqueda lineal con centinela

```
int busquedaSecuencialCentinela(int* Vector, int
tope, int elemento)
{
  int i;

  for(i=0; Vector[i]!=elemento; i++);
  return(i);
}
```

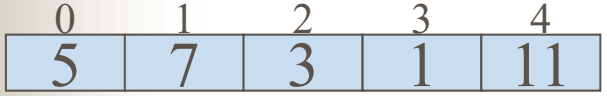
Búsqueda lineal recursiva

```
int busquedaLinealRecursiva(int* V, int posicion, int elemento)
{
  if(posicion<0)
  {
    return(-1); //Primer caso base
  }
  else
  {
    if(V[posicion]==elemento)
    {
      return(posicion); //Segundo caso base
    }
    else
    {
      return(busquedaLinealRecursiva(V, posicion-1, elemento));
    }
  }
}
```

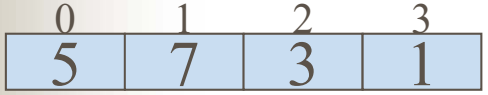


busqueda(V, 6, 7) =>Encontrado=>return(1)

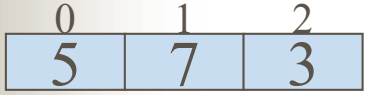
busqueda(V, 5, 7) =>Encontrado=>return(1)



busqueda(V, 4, 7)=>Encontrado=>return(1)



busqueda(V, 3, 7)=>Encontrado=>return(1)

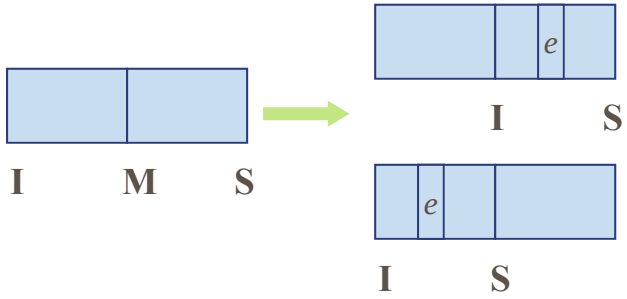


busqueda(V, 2, 7)=>Encontrado=>return(1)

busqueda(V, 1, 7)=>Encontrado=>return(1)

Búsqueda dicotómica o binaria

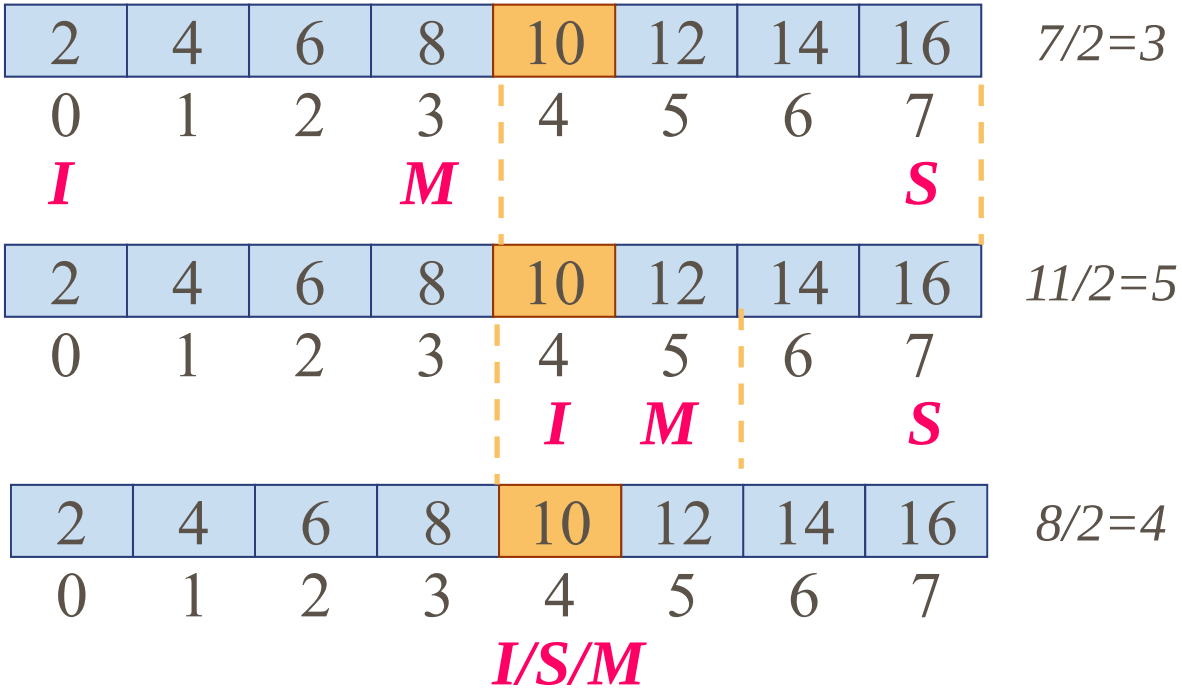
```
int busquedaDicotomica(int* V, int posicion, int elemento){
int encontrado=0, inf=0, sup=posicion, medio;
while((inf<=sup)&&(!encontrado))
{    medio= (inf+sup)/2;
     if(V[medio]==elemento)
     {encontrado = 1;}
     else
     {
       if(V[medio]>elemento)
       {sup = medio-1;}
       else
       {inf = medio+1;}
     }
}
 if(encontrado)
   return(medio);
 else
   return(-1);
}
```



Requiere del vector ordenado!!!

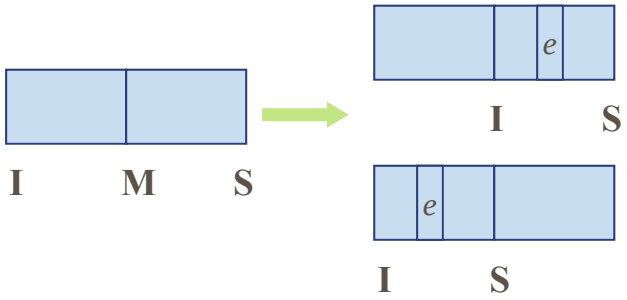
Búsqueda dicotómica o binaria

elemento = 10



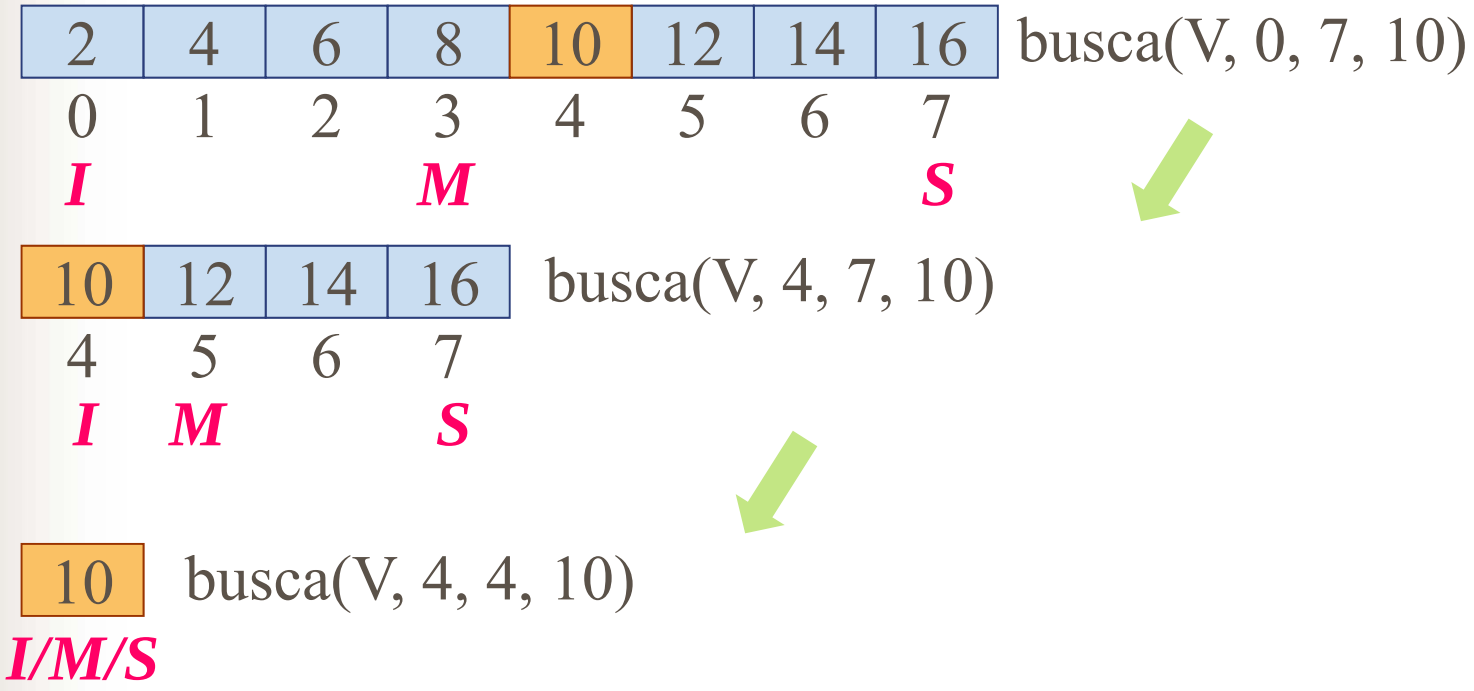
Búsqueda dicotómica recursiva

```
int busquedaDicotomicaRecursiva(int* V, int inf, int sup, int elemento)
{  int medio;

   if(inf>sup)
     {return(-1);}  //Caso base
   else
   { medio= (inf+sup)/2;
     if(V[medio]==elemento)
          {return medio;}
     else
     {    if(V[medio]<elemento)
          {return(busquedaDicotomicaRecursiva(V, medio+1, sup, elemento));}
          else
          {return(busquedaDicotomicaRecursiva(V, inf, medio-1, elemento));}
     }
   }
}
```



Búsqueda dicotómica recursiva

elemento buscado = 10



Búsqueda dicotómica recursiva

- Si el tamaño del vector es pequeño, será más eficiente la búsqueda lineal
- En otro caso, haremos una búsqueda dicotómica

```
int peque(int n)
{return(n<10);}
int busquedaDicotomicaRecursiva2(int* V, int inf, int sup, int elemento)
{  int medio, i, posicion=-1;
   if(peque(sup-inf))
   { for(i=inf; i<=sup; i++)
     { if(V[i]==elemento)
       {posicion = i;}
     }
     return(posicion);
   }
```

Búsqueda dicotómica recursiva

```
else
   {  medio= (inf+sup)/2;
      if(inf>sup)
      {return(-1);}
      else
      { if(V[medio]==elemento)
        {return medio;}
        else
        {  if(elemento<V[medio])
           {return(busquedaDicotomicaRecursiva(V, inf, medio-1, elemento));}
           else
           {return(busquedaDicotomicaRecursiva(V, medio+1, sup, elemento));}
        }
      }
  }
}
```

Búsqueda por interpolación

- La búsqueda binaria no busca como lo hace el ser humano
  - Por ejemplo, al buscar en la guía telefónica, una persona cuyo apellido empieza con *W* tenemos idea de una cierta proporcionalidad, respecto del tamaño del archivo, y vamos directamente a la parte alta de la guía telefónica
- La idea es modificar la búsqueda binaria 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:

```
pos = inf+ ((sup-inf)*(elemento-V[inf]))/(V[sup]-V[inf]);
```

Búsqueda por interpolación

```
int busquedaInterpolacion(int* V, int posicion, int elemento){
  int encontrado=0, pos=-1, inf = 0, sup=posicion;
  while((inf<=sup)&&(!encontrado))
  {
     pos = inf+ ((sup-inf)*(elemento-V[inf]))/(V[sup]-V[inf]);
     if(V[pos]==elemento)
     { encontrado = 1;}
     else
     {
       if(V[pos]>elemento)
       { sup = pos-1;}
       else
       { inf = pos+1;}
     }
  }
  if(encontrado)
  { return(pos);}
  else
  { return(-1);}}
```

elemento = 14

2	4	6	8	10	12	14	16
0	1	2	3	4	5	6	7
I						***P***	***S***

P=0+7(14-2)/14=6*

Búsqueda. Resumen

- La búsqueda es una de las operaciones de tratamiento de vectores más habituales
- Hemos estudiado:
  - **Búsqueda secuencial**. Es el método más simple, recorre completamente el vector y no precisa que el vector es ordenado
  - **Búsqueda secuencial con centinela**. Recorre el vector hasta que se encuentra el valor buscado o se determina que éste no existe
  - **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*)
  - **Búsqueda por interpolación**. Es una modificación del algoritmo de búsqueda binaria, también de complejidad O(*log n*)
- Es preferible buscar en vectores ordenados

Tema de ordenación

Otras estrategias de búsqueda

- **Búsqueda por Fibonacci sobre un vector ordenado**

  Es un método mejorado al de búsqueda binaria. Se diferencia de ella en que la partición de los subvectores se realiza usando los números de Fibonacci
- ***Hashing*.** En este método se requiere que los elementos estén ordenados. El método consiste en asignar el índice a cada elemento mediante una función de conversión llamada función *hash*
- **Arboles de búsqueda**