Búsqueda

La búsqueda es una tarea muy habitual, en los cápitulos anteriores usamos el esquema B, pero existen mejoras que podemos aplicar en secuencias ordenadas para no recorrer toda la secuencia cada vez que queremos encontrar un elemento.

Podemos usar dos alternativas para buscar sobre secuencias ordenadas: **Búsqueda secuencial** o **Binaria**.

Búsqueda Secuencial

Es la búsqueda mas sencilla y es muy similar al esquema B

```
Algoritmo BúsquedaSecuencial
Léxico
  s ∈ arreglo [limInf...limSup] de Telem
  e ∈ Telem
  i ∈ (limInf...limSup+1)
Inicio
  i ← limInf
  mientras i < limSup+1 y s[i] < e hacer
    i ← i + 1

  fmientras
  según
    i > limSup: {e0 no esta en s}
    i <= limSup:  según
                    s[i] > e: {e0 no esta en s}
                    s[i] = e: {e0 esta en s[i]}
                  fsegún
  fsegún
Fin
```

Búsqueda Binaria

Es uno de los metodos mas sencillos para resolver ecuaciones en una variable.

Determina si una secuencia ordenada con acceso directo contiene un elemento **e** dado.

Principio General:

- Se compara **e** con el elemento medio de la secuencia.
- Si **e** es igual al elemento medio, es **hallado**.
- Si **e** es mayor que el elemento medio: Se busca en la mitad derecha de la secuencia.
- Si **e** es menor que el elemento medio se busca en la mitad derecha de la secuencia.

```
Algoritmo BúsquedaBinaria
Léxico
  s ∈ arreglo [limInf...limSup] de Telem
  e ∈ Telem
  inf,sup,medio ∈ (limInf...limSup+1)
Inicio
  según
    e < s[limInf] o e > s[limSup] : {e0 no está en s}
    s[limInf] <= e <= s[limSup]:    inf ← limInf
                                    sup ← limSup
                                    mientras inf < sup hacer
                                      medio ← (inf+sup) div 2
                                      segun
                                        e > s[medio]: inf ← medio + 1
                                        e <= s[medio]: sup ← medio
                                      fsegun
                                    fmientras
                                    segun
                                      e = s[inf]: {e0 está en la lista}

                                      otros: {e0 no está enla lista}
                                    fsegun
  fsegun
Fin
```

Ordenamiento

Ordenar una secuencia es organizar sus elementos de modo que formen una secuencia no decreciente.

Ordernar una secuencia facilita los procesos de busqueda permitiendonos elegir un tratamiento de busqueda mas adecuado.

Nociones Importantes

Estabilidad

Este concepto solo tiene sentido solo para secuencias cuyos elementos sean compuestos, es decir tienen mas de un campo.

Un metodo de ordenamiento estable mantiene el orden relativo que tenian originalmente los elementos con claves iguales.

Algoritmos de Ordenamiento

los 3 mas basicos son:

- BubbleSort(Ordenamiento por metodo de la burbuja)
- InsertionSort(Ordenamiento por inserción)
- SelectionSort(Ordenamiento por selección)

BubbleSort (Burbuja)

Se recorre la secuencia S llevando el mayor el elemento encontrado hacia la ultima posición. Luego se repite para la subsecuencia resultante de no considerar el ultimo elemento.

siendo i el limite hasta el cual j se mueve j el indice del primer elemento del par y s la secuencia

```
i ← n
mientras i > 1 hacer {mientras no sea el primer elemento}
  j ← 1
  mientras j < i hacer
    si s[j] > s[j+1] entonces
      intercambiar(s[j],s[j+1])
    fsi
```

```
    fsi
    j ← j + 1
  fmientras
  i ← i - 1
fmientras
```

InsertionSort (inserción)

Sigue la logica de un jugador de cartas. Cada valor de s es insertado de forma ordenada entre los valores de la subsecuencia ordenada S[1..i-1]

```
i ← 2
mientras i <= n hacer
  aux ← s[i]
  j ← i - 1
  mientras j > 0 y s[j] > aux hacer
    s[j+1] ← s[j]
    j ← j-1
  fmientras
  s[j+1] ← aux
  i ← i+1
 fmientras
```

SelectionSort (selección)

Busca el menor elemento de la subsecuencia no ordenada S[i..n] y lo intercambia con s[i].

```
i ← 1
mientras i < n hacer
  j ← i
  min ← i
  mientras j <= n hacer
    si s[j] > s[min] entonces
      min ← j
    fsi
    j ← j+1
  fmientras
  intercambiar(s[i],s[min])
```

```
  i ← i + 1
fmientras
```