Slide 1

TEMA 13: Algoritmos de búsqueda y ordenación

Índice

1. Búsqueda **1**

1.1. Búsqueda en vectores 2

1.1.1. Secuencial 3

1.1.2. Búsqueda binaria 6

2. Ordenación **9**

2.1. Método de inserción 11

Slide 2

Búsqueda

Entrada: Una colección de datos estructurada
Un dato del mismo tipo

Salida: La "posición" del dato en la estructura si está (primera aparición)
Alguna indicación o valor si no está

Algorimos dependientes de la estructura:

- Conjunto
- Vector, vector ordenado
- Matriz
- Cadena de caracteres
- Fichero ...

Slide 3

Búsqueda en vector

Entrada:

- `v : array [1..N] of Tregistro`, donde uno de los campos, `clave`, es el de búsqueda
- Un rango de búsqueda, dado por dos valores o expresiones, `inicial` y `final` con $1 \leq$ `inicial` $\leq$ `final` $\leq$ `n`
- Una clave que se busca: `buscada`

Salida:

- Índice de la primera aparición del dato en el vector, si está: `pos`
- Variable lógica que determina si se ha encontrado: `encontrado`

Slide 4

Búsqueda secuencial

```
i := inicial;
while (i<final) and (v[i].clave <> buscada) do
      i := succ(i);
(* i=final ó v[i].clave=buscada *)
if v[i].clave = buscada then
   encontrado := TRUE
else
   encontrado := FALSE;
(* encontrado := v[i].clave=buscada *)
pos := i
```

Slide 5

Búsqueda secuencial. Otra versión

```
i := inicial;
encontrado := FALSE;
while (i<=final) and not encontrado do
      if buscado = v[i].clave then
         encontrado := TRUE
      else
         i := i+1 ;
(* i>final ó encontrado en i<=final  *)
if encontrado then pos := i
```

Slide 6

Búsqueda secuencial con centinela

Se busca en el vector de `inicial` a `n`. Se utiliza una posición más

```
v[succ(n)].clave := buscada;
i := inicial;
while (v[i].clave <> buscada) do
      i := succ(i);
(* v[i].clave=buscada  i está en [1,succ(n)] *)
if i <> succ(n) then
   encontrado := TRUE
else
   encontrado := FALSE;
(* encontrado := i<>succ(n) *)
pos := i
```

Slide 7

Búsqueda secuencial sobre vector ordenado

```
pos := inicial; encontrado := FALSE;
hecho := FALSE; (* encontrado un elemento >= buscado *)
while (pos<=final) and not hecho do
      if buscado <= v[i].clave then
         hecho := TRUE
      else
         pos := pos + 1 ;
(* hecho  o pos>final *)
if hecho then
   begin
      if v[pos].clave = buscado then
         encontrado := TRUE
   end
else
      encontrado := FALSE
```

Slide 8

Búsqueda binaria o dicotómica

Entrada: vector ordenado por la clave.

Salida: *una* aparición (no necesariamente la primera)

```
izq := inicial; der := final; encontrado := FALSE;
while (not encontrado) and (izq <= der) do
      begin medio := (izq + der) div 2;
            if v[medio].clave = buscada then
                encontrado := TRUE
            else
                if v[medio].clave > buscada then
                    der := medio - 1
                else
                    izq := medio + 1
      end;
if encontrado then pos := medio
```

Slide 9

Búsqueda binaria sin comparación de igualdad
ERRÓNEA

```
izq := inicial; der := final;
while (izq < der) do
      begin
          medio := (izq + der) div 2;
          if v[medio].clave > buscada then
              der := medio - 1
          else
              izq := medio
      end;
pos := izq;
encontrado := v[pos].clave = buscada
```

Slide 10

Búsqueda binaria sin comparación de igualdad
CORRECTA

```
izq := inicial; der := final;
while (izq < der) do
      begin
          medio := (izq + der) div 2;
          if v[medio].clave < buscada then
              izq := medio + 1
          else
              der := medio
      end;
pos := izq; (* o medio*)
encontrado := v[pos].clave = buscada
```

Slide 11

Entrada: Una colección de datos estructurada

Salida: La misma colección de datos, ordenada (relación de orden: tipo ordenado)

Consideraciones

- Estructura de los datos (vector, fichero . . .)
- Tipo de ordenación (estable, interna . . .)
- Eficiencia del algoritmo (peor caso, mejor caso, caso medio).
- Estrategia básica

Slide 12

Ordenación: ESTRATEGIAS

- Inserción
  - simple
  - con centinela
  - binaria
  - . . . (Shell)
- Selección (. . .)
- Intercambio (. . . , QuickSort)
- Mezcla
- . . .

Slide 13

INSERCIÓN DIRECTA (SIMPLE)

```
for i := 2 to n do
    begin
        aux := v[i]; k := i-1;
        while (v[k] > aux) and (k>1) do begin
             v[k+1] := v[k];
             k := k-1
        end;
        if v[k] <= aux then v[k+1] := aux
        else begin
                  v[2] := v[1];
                  v[1] := aux
             end
    end
```

Slide 14

INSERCIÓN DIRECTA CON CENTINELA

Se usa una posición más `v[0]`

```
for i := 2 to n do
    begin
        v[0] := v[i]; k := i-1;
        while (v[k] > v[0]) do begin
             v[k+1] := v[k];
             k := k-1
        end;
        (* v[k] <= v[0] y k>=0 *)
        v[k+1] := v[0]
    end
```

Slide 15

INSERCIÓN BINARIA

```
for i := 2 to n do
    begin
        aux := v[i]; izq := 1; der := i-1;
        while izq <= der do begin
              medio := (izq + der) div 2;
              if aux < v[medio]
                 then der := medio -1
                 else izq := medio +1
        end;
        for j := i-1 downto izq do
            v[j+1] := v[j];
        v [izq] := aux
    end
```

Slide 16

Algunas páginas web de simulaciones (java): `http://`

- `www.cs.hope.edu/ alganim/animator/Animator.html`
  Simulación paso a paso de los métodos de
  - Básicos de Intercambio (burbuja), Selección, Inserción,
  - Quick Sort, Shell y Mezcla
- `www.inf.ethz.ch/ personal/staerk/algorithms/SortAnimation.html`
  Simulación de los métodos de
  - Básicos de Intercambio (burbuja), Selección, Inserción,
  - Quick Sort, Heap Sort y Mezcla
- `www.cs.ubc.ca/spider/harrison/Java/sorting-demo.html`
  Como el anterior, más métodos.