Tema 5: Problemas de recorrido y búsqueda

- Duración: 10 horas
- Índice:
  1. Introducción: problemas de recorrido y búsqueda
  2. Esquemas de recorrido y búsqueda
  3. Ejemplos
  4. Combinación de técnicas de recorrido y búsqueda

Objetivos

* Introducir y definir los problemas de **Recorrido** y **Búsqueda** (sobre vectores)

* Introducir los **Esquemas Algorítmicos de Recorrido y Búsqueda** como una aplicación (instancia) de la estrategia de diseño iterativo a partir del Invariante y Función Limitadora

* Introducir la **Composición** de dichos esquemas mediante la presentación de algoritmos clásicos como el de **Selección Directa** (ordenación lenta)

Bibliografía

♦"Curso de Programación". Castro, J., Cucker, F., Messeguer, X., et al. McGraw-Hill, 2ª ed., 1995.

♦"Esquemas algorítmicos fundamentales. Secuencias e iteración". Scholl, P.C., Peyrin, J.P. Ed. Masson, 1991.

1. Introducción

Ejemplos de problemas del mismo tipo:

- **EJEMPLO 1:** Dado un vector de enteros **v**, calcular la **suma** de sus componentes
- **EJEMPLO 2:** Dado un vector de enteros **v**, calcular la **media aritmética** de sus componentes
- **EJEMPLO 3:** Calcular el **máximo/mínimo** de un vector de enteros **v**
- **EJEMPLO 4:** Dado un vector de enteros **v**, **duplicar** el valor de los elementos situados entre dos posiciones dadas, **izq** y **der**, del vector

EJEMPLO 1: Dado un vector **v: vector** [**1..Nmax**] **de entero**. Obtener en **s** la **suma** de sus elementos.

tipo t_vector = vector [1.. **Nmax**]de entero **ftipo**

algoritmo sumaV **(DATOS** v: **t_vector**; n:entero;
RESULTADOS s: entero**)**

P= {1≤**n**≤**Nmax}**

Q= {s= Σ i: 1≤i≤n: v[i] **}**

v[1] + v[2] + ...+ v[n]

+ v[i] **n** veces

falgoritmo

Estrategia iterativa

EJEMPLO 2: Dado un vector **v: t_vector**. Obtener la **media aritmética** de sus componentes.

algoritmo mediaV **(DATOS** v: **t_vector**; n:entero;
RESULTADOS m: real**)**

P= {1≤n≤**Nmax}**

Q= {m= (Σ i: 1≤i≤n: v[i])/n **}**

Estrategia iterativa

+ v[i] **n** veces

/ n

falgoritmo

EJEMPLO 3: Dado un vector **v: t_vector**. Obtener en la variable **min** su **mínimo**.

algoritmo mínimo **(DATOS** v: **t_vector;** n:entero;

RESULTADOS min:entero**)**

P= {$1 \leq n \leq Nmax$**}**

Q= { $(\forall i: 1 \leq i \leq n: v[i] \geq min) \wedge (\exists i: 1 \leq i \leq n: v[i] = min)$**}**

¿ v[i] ≤ min ? — **n** veces

Estrategia iterativa

falgoritmo

EJEMPLO 4: dado un vector **v: t_vector** con **n** elementos, **duplicar** los elementos situados entre las posiciones **izq** y **der** ($1 \leq izq \leq der \leq n$), ambas incluidas.

algoritmo duplicar **(DATOS** v: **t_vector;** n , izq, der :entero;

RESULTADOS v: **t_vector)**

P= {$1 \leq izq \leq der \leq n \leq Nmax \wedge (\forall i: 1 \leq i \leq n: v[i] = V_i)$ **}**

Q= { $(\forall i: 1 \leq i < izq: v[i] = V_i) \wedge (\forall i: der < i \leq n: v[i] = V_i)$

$\wedge \ (\forall i: izq \leq i \leq der: v[i] = 2 * V_i)$**}**

2 * v[i] — der-izq +1 veces

falgoritmo

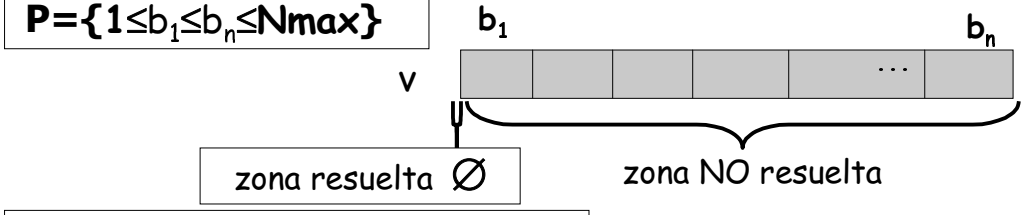
Estrategia iterativa

Estrategia iterativa RECORRIDO	i.e.	trata(v[i]) Qué hace	∀i: Rango(i) Cuántas veces
Q E1:	trata(v[i])=	+v[i]	∀i: 1≤i≤n
Q E2:	trata(v[i])=	+v[i]	∀i: 1≤i≤n
Q E3:	trata(v[i])=	¿ v[i] ≤ min ?	∀i: 1≤i≤n
Q E4:	trata(v[i])=	2 * v[i]	∀i: izq≤i≤der

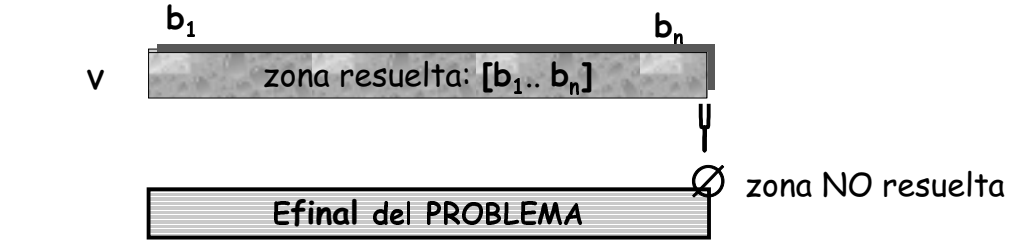
❑ Definición de Problemas de Recorrido

Sea v un vector [1..**Nmax**] de TipoD con b_n-b_1+1 elementos

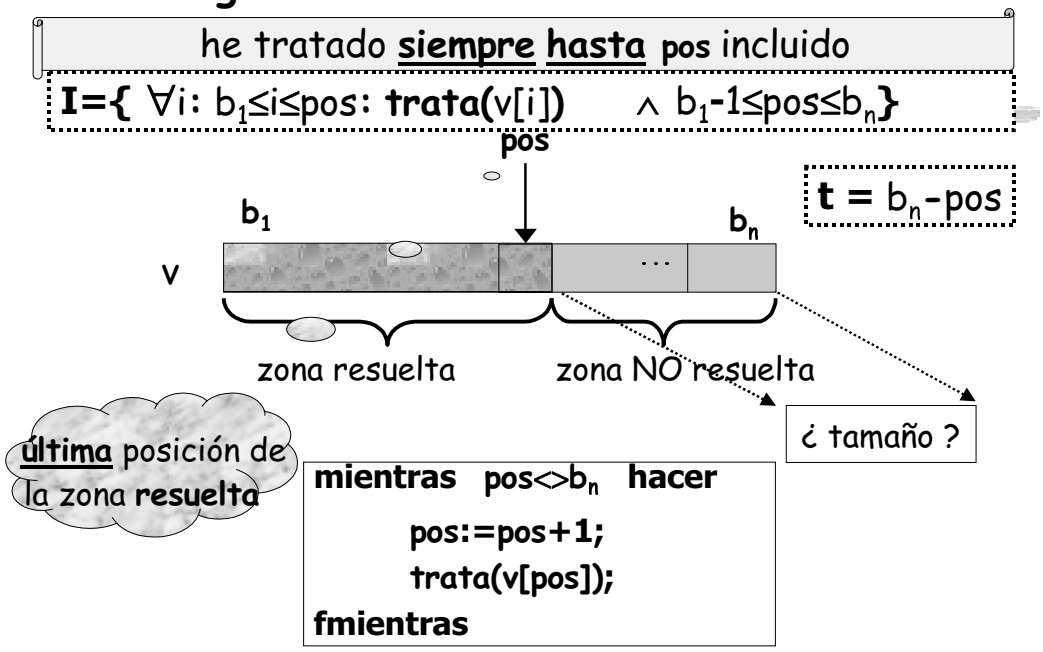
P={1≤b_1≤b_n≤Nmax}



Q={ ∀i: b_1≤i≤b_n: trata(v[i])}



❑ Estrategia 1 en Problemas de Recorrido



Esquema iterativo

inicialización;

mientras B **hacer**

cuerpo

fmientras

terminación

Esquema recorrido ascendente 1

pos:= b_1 - 1;
inicialización_tratamiento;

mientras pos <> b_n **hacer**

pos := pos + 1;
trata(v[pos]**)** ;

fmientras

tratamiento_final;

Esquema recorrido descendente 1

pos:= b_n + 1;
inicialización_tratamiento;

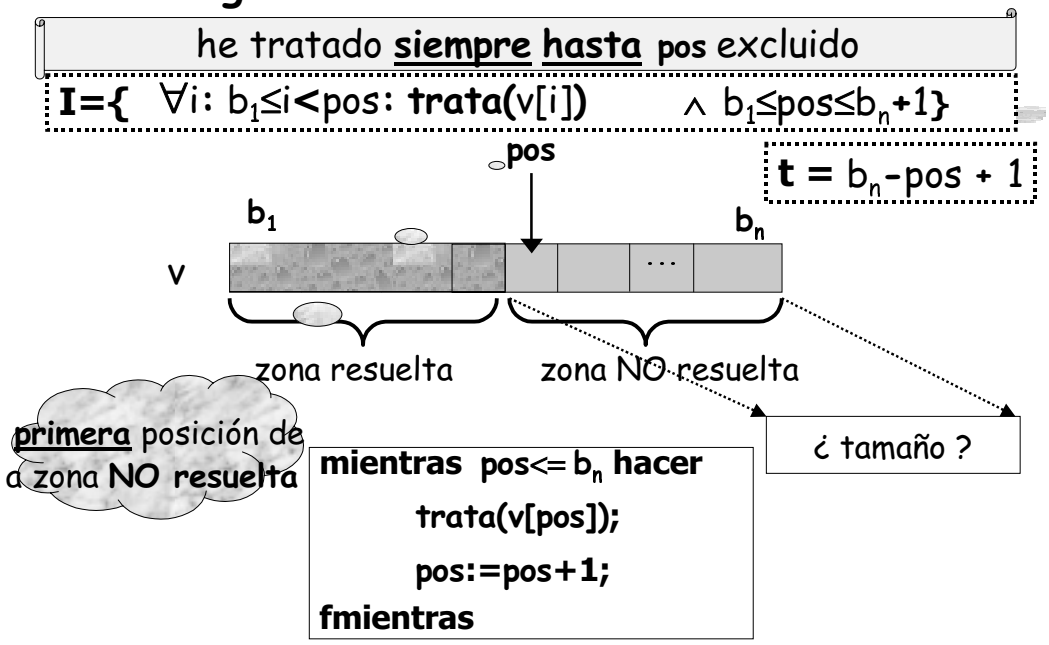
mientras pos <> b_1 **hacer**

pos := pos - 1;
trata(v[pos]**)** ;

fmientras

tratamiento_final;

❑ Estrategia 2 en Problemas de Recorrido



Esquema recorrido ascendente 2

```
pos:= b1;
inicialización_tratamiento;

mientras pos<> bn+1 hacer
    trata(v[pos]) ;
    pos := pos + 1;
fmientras

tratamiento_final;
```

Esquema recorrido descendente 2

```
pos:= b_n;
inicialización_tratamiento;

mientras pos<> b_1-1 hacer
    trata(v[pos]) ;
    pos := pos - 1;
fmientras

tratamiento_final;
```

Ejemplo: dado un vector de enteros **v**, calcular la **media aritmética** de sus componentes

algoritmo mediaV (**DATOS** v: t_vector; n:entero;
RESULTADOS m: **real)**

P= {$1 \leq n \leq Nmax$**}**

Q= {$m=(\sum i:1 \leq i \leq n:v[i])/n$**}**

RECORRIDO

he tratado **siempre hasta pos** incluido

I= { $\forall$ i : $b_1 \leq i \leq pos$: **trata(**v[i]**)**

$\wedge$ $b_1-1 \leq pos \leq b_n$ **}**

t = b_n **-pos**

Esquema recorrido ascendente 1

pos:= b_1 - 1;
inicialización_tratamiento;

mientras pos <> b_n **hacer**

pos := pos + 1;
trata(v[pos]**)** ;

fmientras

Efinal de la **iteración** | he **sumado siempre hasta pos** incluido

tratamiento_final;

Efinal del PROBLEMA **Q={**m=(Σi:1≤i≤n:v[i]) / n **}**

Ejemplo: dado un vector **v: t_vector** con **n** elementos, **duplicar** los elementos situados entre las posiciones **izq** y **der** (1≤**izq**≤**der**≤**n**), ambas incluidas

algoritmo duplicar **(DATOS** v: **t_vector**; n , izq, der :entero;
RESULTADOS v: **t_vector);**

P= {1≤izq≤der≤n≤Nmax ∧ (∀i:1≤i≤n:v[i]=V_i) **}**

Q= { (∀i: 1≤i< izq: v[i]=V_i) ∧ (∀i: der<i≤n: v[i]=V_i)
∧ (∀i: izq≤i≤der: v[i]=2*V_i)**}**

RECORRIDO

he tratado **siempre hasta pos** incluido

I= { ∀ i : b_1 ≤ i ≤ pos: **trata(**v[i]**)**
∧ b_1-1 ≤ pos ≤ b_n **}**

t = b_n - pos

Esquema recorrido ascendente 1

pos:= b_1 - 1;
inicialización_tratamiento;

mientras pos <> b_n **hacer**

pos := pos + 1;
trata(v[pos]**) ;**

fmientras

Efinal de la iteración he 2 * v[i] **siempre hasta** **pos** incluido

tratamiento_final;

Efinal del PROBLEMA **Q={** $(\forall i: izq \leq i \leq der: v[i]=2*V_i) \wedge ....$ **}**

Ejemplos de problemas del mismo tipo:

EJEMPLO 1: Dado un vector de enteros **v**, indicar si **e está** en el vector

EJEMPLO 2: Dado un vector de enteros **v**, indicar si **todos** sus elementos son **iguales** a uno dado **e**

EJEMPLO 3: Dado un vector **v** de **n** enteros, distintos entre sí, ordenado ascendentemente hasta su penúltima componente (**n**-1), indicar la **posición** de aquel elemento del vector tras el cual se situaría **v[n]** en una ordenación de todo el vector

EJEMPLO 4: Dado un vector de enteros **v**, indicar si es **capicúa**

EJEMPLO 1: dado un vector **v**: vector [**1..Nmax**] de entero. Indicar si **e** es uno de sus elementos.

tipo t_vector = vector [1.. **Nmax**]de entero **ftipo**

algoritmo buscaV **(DATOS** v: **t_vector**; n, e: entero;
RESULTADOS está: lógico**)**

P= {1≤n≤Nmax}

Q= {está=(∃i:1≤i≤n:v[i]=e) }

está

¿ v[i] = e ? tantas veces como (v[i]=e)=F

☞**Estrategia iterativa "rarita"**

falgoritmo

EJEMPLO 2: dado un vector v: t_vector. Indicar si **todos** sus elementos **son iguales** a uno dado e.

algoritmo soniguales **(DATOS** v: **t_vector**; n, e: entero;
RESULTADOS son: lógico**)**

P= {1≤n≤Nmax}

Q= {son=(∀i:1≤i≤n: (v[i]=e)) **}**

¬(∃i: 1≤i≤n: (v[i]≠e))

está

falgoritmo

¿ v[i] = e ?

tantas veces como (v[i]=e)=V

∀i: **Rango(i)**

¿ v[i] ≠ e ?

tantas veces como (v[i] ≠ e)=F

☞**Estrategia iterativa "rarita"**

EJEMPLO 3: dado un vector **v**: t_vector de **n** enteros, distintos entre sí, ordenado ascendentemente hasta su penúltima componente (**n**-1), indicar la **posición** de aquel elemento del vector tras el cual se situaría **v[n]** en una ordenación de todo el vector

algoritmo posV **(DATOS** v: **t_vector**; n:entero;
RESULTADOS posición: entero**)**

P= **{**$1 \le n \le Nmax \wedge (\forall k: 1 \le k < n-1: v[k] < v[k+1])$
$\wedge (\forall k: 1 \le k \le n: (\forall j: 1 \le j \le n: j \ne k \rightarrow v[k] \ne v[j]))$ **}**

Q= **{**$0 \le posición < n \wedge (\forall i: 1 \le i \le posición: v[i] < v[n])$
$\wedge (\forall i: posición+1 \le i \le n-1: v[i] > v[n])$**}**

falgoritmo

algoritmo posV **(DATOS** v: **t_vector**; n:entero;
RESULTADOS posición: entero**)**

P= **{**$1 \le n \le Nmax \wedge (\forall k: 1 \le k < n-1: v[k] < v[k+1])$
$\wedge (\forall k: 1 \le k \le n: (\forall j: 1 \le j \le n: j \ne k \rightarrow v[k] \ne v[j]))$ **}**

Q= **{**$0 \le posición < n \wedge (\forall i: 1 \le i \le posición: v[i] < v[n])$
$\wedge (\forall i: posición+1 \le i \le n-1: v[i] > v[n])$**}**

$\neg(\exists i: posición+1 \le i \le n-1 : (v[i] < v[n]))$

está

falgoritmo

¿ v[i] < v[n]?

Estrategia iterativa "rarita" ☜

tantas veces como (v[i]<v[n])=F

EJEMPLO 4: dado el vector v: t_vector, indicar si es **capicúa**

algoritmo capicua (**DATOS** v: **t_vector**; n:entero;
RESULTADOS es: lógico)

P= {1≤n≤Nmax}

está

¬(∃i: 1≤i≤n: (v[i]≠v[n-i+1]))

Q= {es=(∀i:1≤i≤n: (v[i]=v[n-i+1]))}

falgoritmo

¿ v[i] ≠ v[n-i+1] ?

tantas veces como (v[i] ≠ v[n-i+1])=F

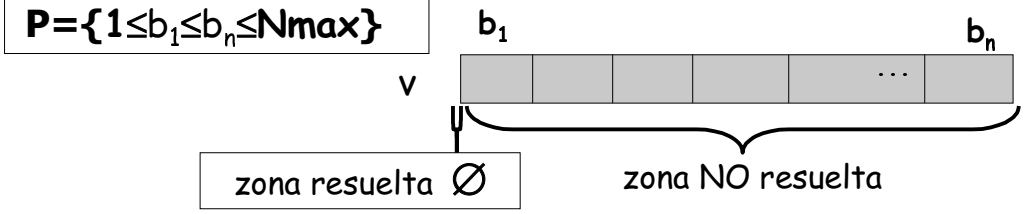
Estrategia iterativa "rarita"

Estrategia iterativa BÚSQUEDA i.e.	cumple_P(v[i]) Qué hace	∀i: Rango(i) ∨ está Cuántas veces
Q E1:	**cumple_P(v[i])=** ¿ v[i]=e ?	**∀i: 1≤i≤n ∨ (está=V)**
Q E2:	**cumple_P(v[i])=** ¿ v[i]≠e ?	**∀i: 1≤i≤n ∨ (está=V)**
Q E3:	**cumple_P(v[i])=** ¿ v[i]<v[n] ?	**∀i: 1≤i≤n-1 ∨ (está=V)**
Q E4:	**cumple_P(v[i])=** ¿ v[i]≠v[n-i+1] ?	**∀i: 1≤i≤n ∨ (está=V)**

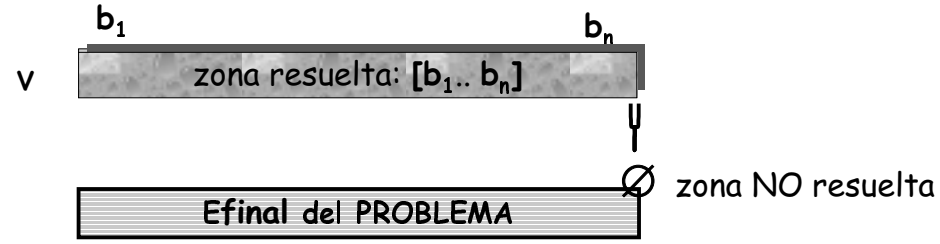
❑ Definición de Problemas de Búsqueda

Sea v un vector [1..**Nmax**] de TipoD con b_n-b_1+1 elementos

P={$1 \le b_1 \le b_n \le$ **Nmax}**



Q={ está=($\exists i: b_1 \le i \le b_n$**: cumple_P(** v[i] **)) }**

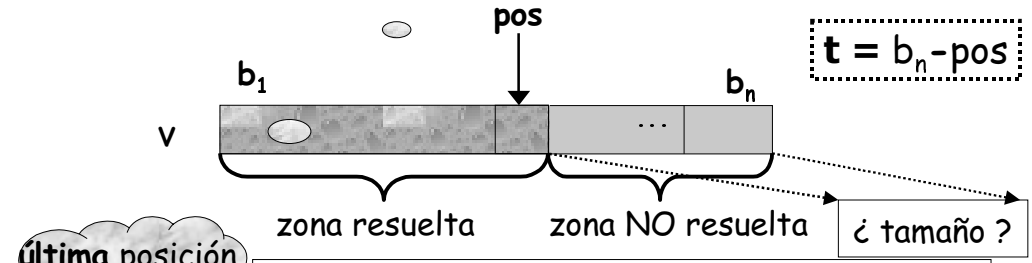


❑ Estrategia 1 en Problemas de Búsqueda

si está, **<u>está</u>** en pos

I={$\forall i: b_1 \le i \le$ **pos-1**: $\neg$**cumple_P(** v[i] **)** $\wedge$

está=($\exists i: b_1 \le i \le$ **pos**: **cumple_P(** v[i] **))** $\wedge$ $b_1-1 \le pos \le b_n$**}**



<u>última</u> posición de la zona **resuelta**

mientras pos<>b_n $\wedge$ $\neg$está **hacer**
  pos:=pos+1;
  si cumple_P(v[pos]**) entonces** está **:= V; fsi**
fmientras

Esquema iterativo

inicialización;

mientras B **hacer**

cuerpo

fmientras

terminación

Esquema búsqueda ascendente 1

pos**:=** b_1 **-** 1**;**
está**:= F;**
inicialización_tratamiento**;**

mientras pos **<>** $b_n \wedge \neg$**(**está**)** **hacer**

pos **:=** pos **+** 1**;**
si cumple_P(v[pos]**) entonces** está **:= V ; fsi**

fmientras

tratamiento_final**;**

terminación

Esquema búsqueda descendente 1

pos**:=** b_n **+ 1;**
está**:= F;**
inicialización_tratamiento**;**

mientras pos **<>** $b_1 \wedge \neg$**(**está**)** **hacer**

pos **:=** pos **- 1;**
si cumple_P(v[pos]**) entonces** está **:= V ; fsi**

fmientras

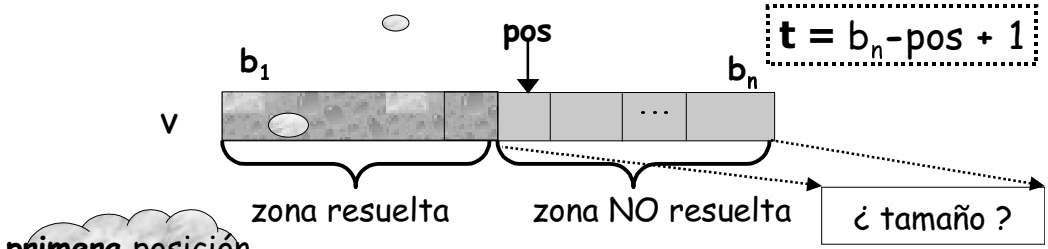
tratamiento_final**;**

terminación

❑ Estrategia 2 en Problemas de Búsqueda

si está, **<u>está</u>** en pos-1

I={$\forall$i:$b_1 \leq i <$ **pos-1**: $\neg$**cumple_P(** v[i] **)** $\wedge$
está=($\exists$i:$b_1 \leq i \leq$ **pos-1**: **cumple_P(** v[i] **))** $\wedge$ $b_1 \leq pos \leq b_n+1$**}**



<u>primera</u> posición dela zona **NO resuelta**

mientras pos<>b_n**+1** $\wedge \neg$**(**está**)** **hacer**
si cumple_P(v[pos]**) entonces** está **:= V ; fsi**
pos:=pos+1;
fmientras

Esquema búsqueda ascendente 2

pos**:=** b_1**;**
está**:= F;**
inicialización_tratamiento**;**

mientras pos **<>** b_n **+** 1 $\wedge$ $\neg$**(**está**)** **hacer**

si cumple_P(v[pos]**) entonces** está **:= V ; fsi**
pos **:=** pos **+** 1**;**

fmientras

tratamiento_final**;**

terminación

Esquema búsqueda descendente 2

pos**:=** b_n **;**
está**:= F;**
inicialización_tratamiento**;**

mientras pos **<>** b_1 **-** 1 $\wedge$ $\neg$**(**está**)** **hacer**

si cumple_P(v[pos]**) entonces** está **:= V ; fsi**
pos **:=** pos **-** 1**;**

fmientras

tratamiento_final**;**

terminación

EJEMPLO 2: dado un vector v: t_vector. Indicar si **todos** sus elementos **son iguales** a uno dado e.

algoritmo soniguales **(DATOS** v: **t_vector**; n, e: entero;
RESULTADOS son: lógico**)**

P= {$1 \le n \le Nmax$**}**

Q= {son=($\forall$i:$1 \le i \le n$: (v[i]=e)) **}**

está
$\neg(\exists i: 1 \le i \le n: (v[i] \neq e))$

BÚSQUEDA

si está, **está** en pos

I={ $\forall i: b_1 \le i \le$ **pos-1** : **¬cumple_P(** v[i] **)** $\wedge$

está=($\exists i: b_1 \le i \le$ **pos** : **cumple_P(** v[i] **))**

$\wedge \quad b_1 - 1 \le pos \le b_n$ **}**

Esquema búsqueda ascendente 1

pos**:=** b_1 **- 1;**
está**:= F;**
inicialización_tratamiento**;**

mientras pos **<>** b_n $\wedge$ **¬(**está**)** **hacer**
pos **:=** pos **+** 1**;**
si cumple_P(v[pos]**)** **entonces** está **:= V ; fsi**

Efinal de la **iteración**

si **está**, **está en pos**

fmientras

si está =**V**, **está** en pos (*$1 \le pos \le n$*)
si está =**F**, **no está** **Y** pos=n

tratamiento_final**;**

Efinal del PROBLEMA **Q={**son=($\forall$i:$1 \le i \le n$: (v[i]=e)) **}**

Esquema búsqueda ascendente 1

pos:=
está:= F;

mientras pos <> ∧ ¬(está) **hacer**
  pos := pos + 1;
 si **entonces** está := V ; **fsi**
fmientras

Efinal de la iteración

si está =F, **no está** Y pos=n

Efinal del PROBLEMA **Q={**son=(∀i:1≤i≤n: (v[i]=e)) **}**

Esquema búsqueda ascendente 1

pos:=
está:= F;

mientras pos <> ∧ ¬(está) **hacer**
  pos := pos + 1;
 si **entonces** está := V ; **fsi**
fmientras

Efinal de la iteración

si está =V, **está** en pos (*1≤pos≤n*)
(∃i: 1≤i≤n: (v[i]≠e))

Efinal del PROBLEMA **Q={**son=(∀i:1≤i≤n: (v[i]=e)) **}**

EJEMPLO 3: dado un vector **v** de **n** enteros, distintos entre sí, or… aquel eleme… una ordena…

BÚSQUEDA DESCENDENTE

BÚSQUEDA.. desde **v[n]**

algoritmo posV **(DATOS** …. **RESULTADOS** posición: entero**)**
P= {.....}
Q= {$0 \le posición < n \wedge (\forall i: 1 \le i \le posición: v[i] < v[n])$
$\wedge (\forall i: posición+1 \le i \le n-1: v[i] > v[n])$**}**
$\neg(\exists i: posición+1 \le i \le n-1 : v[i] < v[n])$ **está**

si está, **está** en pos+1

I={ $\forall i: \; pos+1 < i \le b_n \; : \; \neg$**cumple_P(**v[i]**)** $\wedge$

está=($\exists i: pos+1 \le i \le b_n$: **cumple_P(**v[i]**)** **)**

$\wedge \quad b_1 - 1 \le pos \le b_n$ **}**

Esquema búsqueda descendente 2

pos**:=** b_n**;**
está**:= F;**
inicialización_tratamiento**;**

mientras pos **<>** $b_1 - 1 \wedge \neg$**(**está**)** **hacer**
si cumple_P(v[pos]**)** **entonces** está **:= V ; fsi**
pos **:=** pos **-** 1**;**
fmientras

Efinal de la iteración

si **está**, **está en pos+1**

tratamiento_final**;**

si está **=V, está** en pos+1 (*1≤pos≤n-1 *)
si está **=F, no está Y** pos=0

Efinal del PROBLEMA Q={$0 \le posición < n \wedge (\forall i: 1 \le i \le posición: v[i] < v[n])$
$\wedge (\forall i: posición+1 \le i \le n-1: v[i] > v[n])$**}**

Esquema búsqueda descendente 2

pos:=
está:= F;

mientras pos <> ∧ ¬(está) **hacer**
si **entonces** está := V ; **fsi**
pos := pos - 1;
fmientras

Efinal de la iteración | si está =F, no está Y pos= 0

Efinal del PROBLEMA **Q={**0≤posición<n ∧ (∀i:1≤i≤posición:v[i]<v[n])
∧ (∀i:posición+1≤i≤n-1:v[i]>v[n])**}**

Esquema búsqueda descendente 2

pos:=
está:= F;

mientras pos <> ∧ ¬(está) **hacer**
si **entonces** está := V ; **fsi**
pos := pos - 1;
fmientras

∃i: 1≤i≤n-1 : v[i]<v[n]

Efinal de la iteración | si está =V, está en pos+1 (*1≤pos≤n-1 *)

Efinal del PROBLEMA **Q={**0≤posición<n ∧ (∀i:1≤i≤posición:v[i]<v[n])
∧ (∀i:posición+1≤i≤n-1:v[i]>v[n])**}**

3. Ejemplos

- Dado un vector v: **vector** [**1..Nmax**] **de entero**. Obtener en la variable min su **mínimo**.
- Dado un vector de enteros v, indicar si es **capicúa**.
- Dado un vector de enteros v, indicar si **está ordenado en sentido ascendente**.
- Dado un vector v: **vector** [**1..Nmax**] **de entero**, con n elementos, **desplazar** una posición hacia la derecha todos los elementos de v comprendidos entre las posiciones j e i, ambas inclusive, **1≤j≤i<n**.

Ejemplo: dado un vector de enteros v, obtener en una variable min el **mínimo** de dicho vector

algoritmo mínimo (**DATOS** v: t_vector; n: entero;
RESULTADOS min: entero)

P= {1≤n≤**Nmax}**

Q= {(∀i:1≤i≤n:v[i]≤min) ∧ (∃i:1≤i≤n:v[i]=min)**}**

I={

t =

Esquema

Einicial de la iteración

mientras B **hacer**

fmientras

Efinal de la iteración

Efinal del PROBLEMA **Q {**$(\forall i{:}1\leq i\leq n{:}v[i]\leq min) \wedge (\exists i{:}1\leq i\leq n{:}v[i]=min)$**}**

Ejemplo: Dado un vector de enteros v, indicar si **es capicúa.**

algoritmo capicua **(DATOS** v: t_vector; n: entero;
RESULTADOS es: lógico**)**
P= {$1\leq n\leq Nmax$**}**

Q= {es = $(\forall i{:}1\leq i\leq n{:}\ (v[i]=v[n-i+1]))$ **}**
falgoritmo

I={

t =

Esquema

Einicial de la iteración

mientras B **hacer**

fmientras

Efinal de la iteración

Efinal del PROBLEMA **Q= {**es = ($\forall$i:1≤i≤n: (v[i]=v[n-i+1])) **}**

Ejemplo: Dado un vector de enteros v, indicar en una única iteración si **está ordenado en sentido ascendente**

algoritmo ord_asc **(DATOS** v: t_vector; n: entero;
RESULTADOS asc: lógico**)**
P= {1≤n≤Nmax**}**

Q= {asc=($\forall$i:1≤i**<**n: (v[i]≤v[i+1])) **}**
falgoritmo

I={

t =

Esquema

Einicial de la iteración

mientras B **hacer**

fmientras

Efinal de la iteración

Efinal del PROBLEMA **Q= {**asc=($\forall$i:1≤i**<**n: (v[i]≤v[i+1])) **}**

Ejemplo: Dado un vector de enteros v, **desplazar** una posición hacia la derecha todos los elementos de v comprendidos entre las posiciones j e i, ambas inclusive, 1≤j≤i<n.

algoritmo desplazar **(DATOS** v: t_vector; n, j ,i: entero;
RESULTADOS v: t_vector**)**

P=

Q=

falgoritmo

I={

t =

Esquema

Einicial de la iteración

mientras B **hacer**

fmientras

Efinal de la iteración

Efinal del PROBLEMA Q=

4. Combinación de técnicas de recorrido y búsqueda

Ejemplo 1: Dado un vector de enteros v, diseñar un algoritmo que calcule la **suma** de los elementos del vector que aparecen **tras el primer número impar**.

Ejemplo 2: Dado un vector v de n elementos de tipo TipoBase, ordenar crecientemente sus elementos **por Selección Directa**.

Ejemplo 3: Dado un vector v1[1..n] de caracteres, en el que no aparecen elementos repetidos, y otro vector v2[1..m] de caracteres, (n<=m), se quiere probar si todos los elementos del vector v1 están también en el vector v2, es decir, si v1 es **subconjunto** de v2 .