

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre

algoritmos

Cálculo de la eficiencia

híbrida

Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación

## Práctica 2

### El elemento en su posición

María Jesús López Salmerón

Nazaret Román Guerrero

Laura Hernández Muñoz

José Baena Cobos

Carlos Sánchez Páez

6 de abril de 2018

# Índice

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia híbrida

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

## 1 Presentación del problema

## 2 Diseño de algoritmos

- Algoritmo clásico
- Algoritmo Divide y Vencerás
- Medición de tiempos

## 3 Estudio de eficiencia

- Eficiencia empírica
  - Tamaños de problema
  - Comparación entre algoritmos
- Cálculo de la eficiencia híbrida
  - Errores en el cálculo de la constante oculta

## 4 Vectores con elementos repetidos

# Índice

## Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia híbrida

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

## 1 Presentación del problema

## 2 Diseño de algoritmos

- Algoritmo clásico
- Algoritmo Divide y Vencerás
- Medición de tiempos

## 3 Estudio de eficiencia

- Eficiencia empírica
  - Tamaños de problema
  - Comparación entre algoritmos
- Cálculo de la eficiencia híbrida
  - Errores en el cálculo de la constante oculta

## 4 Vectores con elementos repetidos

# El elemento en su posición

## Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia

híbrida

Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación

Sea  $v$  un vector **ordenado** y sin elementos repetidos,  
determinar si  $\exists i : v[i] = i$

# Ejemplos

## Presentación del problema

### Diseño de algoritmos

#### Algoritmo clásico

#### Algoritmo Divide y Vencerás

#### Medición de tiempos

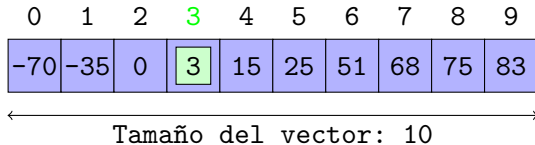
#### Estudio de eficiencia

#### Eficiencia empírica

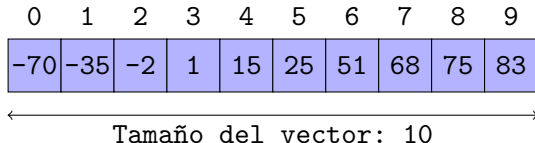
#### Tamaño de problema Comparación entre algoritmos Cálculo de la eficiencia búsqueda Errores en el cálculo de la constante oculta

#### Vectores con elementos repetidos

#### Fin de la presentación



Resultado del algoritmo : 3



Resultado del algoritmo : -1

# Índice

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia híbrida

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

## 1 Presentación del problema

## 2 Diseño de algoritmos

- Algoritmo clásico
- Algoritmo Divide y Vencerás
- Medición de tiempos

## 3 Estudio de eficiencia

- Eficiencia empírica
  - Tamaños de problema
  - Comparación entre algoritmos
- Cálculo de la eficiencia híbrida
  - Errores en el cálculo de la constante oculta

## 4 Vectores con elementos repetidos

# Índice

Presentación del problema

Diseño de algoritmos

**Algoritmo clásico**

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia híbrida

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

## 1 Presentación del problema

## 2 Diseño de algoritmos

- **Algoritmo clásico**
- Algoritmo Divide y Vencerás
- Medición de tiempos

## 3 Estudio de eficiencia

- Eficiencia empírica
  - Tamaños de problema
  - Comparación entre algoritmos
- Cálculo de la eficiencia híbrida
  - Errores en el cálculo de la constante oculta

## 4 Vectores con elementos repetidos

# Clásico

Presentación del problema

Diseño de algoritmos

**Algoritmo clásico**

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre

algoritmos

Cálculo de la eficiencia

híbrida

Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación

```
int elementoEnSuPosicion(const vector<int> &v) {  
    for (int i = 0; i < v.size() ; i++)  
        if (v[i] == i)  
            return i;  
    return -1;  
}
```



# Ejemplo

Presentación del problema

Diseño de algoritmos

**Algoritmo clásico**

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre

algoritmos

Cálculo de la eficiencia

híbrida

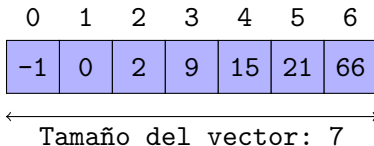
Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación



# Ejemplo. Paso 1

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre

algoritmos

Cálculo de la eficiencia

híbrida

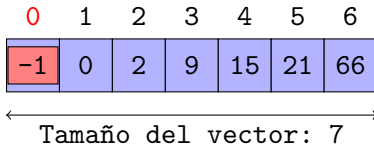
Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación



¿ $v[0] = 0$ ? **NO** → Continuamos.

# Ejemplo. Paso 2

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

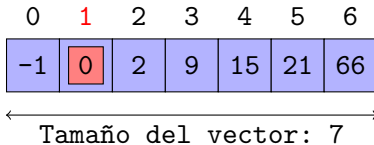
Estudio de eficiencia

Eficiencia empírica

Tamaños de problema  
Comparación entre  
algoritmos  
Cálculo de la eficiencia  
búsqueda  
Errores en el cálculo de la  
constante oculta

Vectores con elementos  
repetidos

Fin de la presentación



¿ $v[1] = 1$ ? **NO** → Continuamos.

# Ejemplo. Paso 3

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

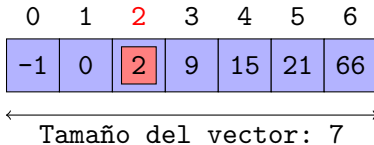
Estudio de eficiencia

Eficiencia empírica

Tamaños de problema  
Comparación entre  
algoritmos  
Cálculo de la eficiencia  
búsqueda  
Errores en el cálculo de la  
constante oculta

Vectores con elementos  
repetidos

Fin de la presentación



$v[2] = 2$ ? **SÍ**  $\rightarrow$  *return 2.*

# Índice

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

**Algoritmo Divide y Vencerás**

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia híbrida

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

## 1 Presentación del problema

## 2 Diseño de algoritmos

- Algoritmo clásico
- **Algoritmo Divide y Vencerás**
- Medición de tiempos

## 3 Estudio de eficiencia

- Eficiencia empírica
  - Tamaños de problema
  - Comparación entre algoritmos
- Cálculo de la eficiencia híbrida
  - Errores en el cálculo de la constante oculta

## 4 Vectores con elementos repetidos

# Divide y Vencerás

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema  
Comparación entre  
algoritmos  
Cálculo de la eficiencia  
búsqueda

Errores en el cálculo de la  
constante oculta

Vectores con elementos  
repetidos

Fin de la presentación

```
int elementoEnSuPosicion(const vector<int> &v, const int ini,
                        const int fin) {
    if (ini == fin) {           //Caso base
        if (v[ini] == ini)
            return ini;
        else
            return -1;
    }
    else { //Buscamos en la mitad adecuada.
        int mitad = (ini + fin) / 2;
        if (v[mitad] == mitad)
            return mitad;
        else if (v[mitad] > mitad)
            return elementoEnSuPosicion(v, ini, mitad-1);
        else
            return elementoEnSuPosicion(v, mitad + 1, fin);
    }
}
```

# Ejemplo

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

**Algoritmo Divide y Vencerás**

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre

algoritmos

Cálculo de la eficiencia

híbrida

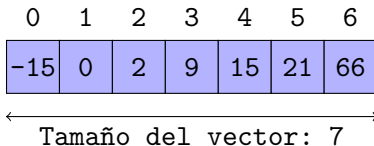
Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación



# Ejemplo. Paso 1

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

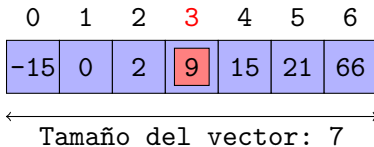
Estudio de eficiencia

Eficiencia empírica

Tamaños de problema  
Comparación entre  
algoritmos  
Cálculo de la eficiencia  
búsqueda  
Errores en el cálculo de la  
constante oculta

Vectores con elementos  
repetidos

Fin de la presentación



Calculamos el centro del vector.

$$mitad = \frac{ini+fin-1}{2} = 3$$



# Ejemplo. Paso 2

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaño de problema

Comparación entre algoritmos

Cálculo de la eficiencia

híbrida

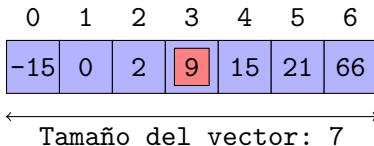
Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación



$$mitad = \frac{ini + fin = 6}{2} = 3$$

$$¿v[3] = 3? \text{ NO}$$

¿ $v[3] = 9 > 3$ ? SÍ → exploramos desde *inicio* = 0 a  
 $mitad - 1 = 2$

# Ejemplo. Paso 3

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia

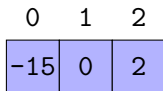
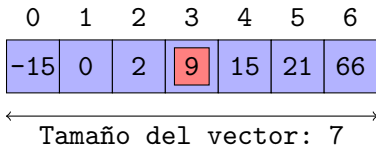
híbrida

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

repetidos

Fin de la presentación



← Tamaño del vector: 3 →

# Ejemplo. Paso 4

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia

hipótesis

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

0	1	2	3	4	5	6
-15	0	2	9	15	21	66

← Tamaño del vector: 7 →

0	1	2
-15	0	2

← Tamaño del vector: 3 →

$$mitad = \frac{ini+fin-2}{2} = 1$$

¿v[1] = 1? NO

¿v[1] = 0 < 1? SÍ → exploramos desde  $mitad + 1 = 2$  a  $fin = 2$

# Ejemplo. Paso 5

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia

hibrida

Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación

0	1	2	3	4	5	6
-15	0	2	9	15	21	66

← Tamaño del vector: 7 →

0	1	2
-15	0	2

← Tamaño del vector: 3 →

2
2

← Tamaño del vector: 1 →

Caso base: vector de un elemento. ¿ $v[2] = 2$ ? **SÍ** → *return 2*

# Índice

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

**Medición de tiempos**

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia híbrida

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

## 1 Presentación del problema

## 2 Diseño de algoritmos

- Algoritmo clásico
- Algoritmo Divide y Vencerás
- **Medición de tiempos**

## 3 Estudio de eficiencia

- Eficiencia empírica
  - Tamaños de problema
  - Comparación entre algoritmos
- Cálculo de la eficiencia híbrida
  - Errores en el cálculo de la constante oculta

## 4 Vectores con elementos repetidos

# Modificación de código fuente

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaño de problema  
Comparación entre  
algoritmos  
Cálculo de la eficiencia  
búsqueda  
Errores en el cálculo de la  
constante oculta

Vectores con elementos  
repetidos

Fin de la presentación

```
high_resolution_clock::time_point tantes;
high_resolution_clock::time_point tdespues;
duration<double> tiempo;
double acumulado = 0;
int pos;
for (int i = 0; i < 1000; i++) {
    tantes = high_resolution_clock::now();
    pos = elementoEnSuPosicion(myvector);
    tdespues = high_resolution_clock::now();
    tiempo = duration_cast<duration<double>>
                (tdespues - tantes);
    acumulado += tiempo.count();
}
acumulado /= 1000;
```

# Índice

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia híbrida

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

## 1 Presentación del problema

## 2 Diseño de algoritmos

- Algoritmo clásico
- Algoritmo Divide y Vencerás
- Medición de tiempos

## 3 Estudio de eficiencia

- Eficiencia empírica
  - Tamaños de problema
  - Comparación entre algoritmos
- Cálculo de la eficiencia híbrida
  - Errores en el cálculo de la constante oculta

## 4 Vectores con elementos repetidos

# Índice

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia híbrida

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

## 1 Presentación del problema

## 2 Diseño de algoritmos

- Algoritmo clásico
- Algoritmo Divide y Vencerás
- Medición de tiempos

## 3 Estudio de eficiencia

- Eficiencia empírica
  - Tamaños de problema
  - Comparación entre algoritmos
- Cálculo de la eficiencia híbrida
  - Errores en el cálculo de la constante oculta

## 4 Vectores con elementos repetidos



# Tamaños de problema

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

**Tamaños de problema**

Comparación entre

algoritmos

Cálculo de la eficiencia

híbrida

Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación

Algoritmo	Eficiencia	Tamaño inicial	Tamaño final	Incremento
Clásico	$O(n)$	1.000.000	1.480.000	20.000
Divide y Vencerás	$O(\log(n))$	1.000.000	13.000.000	500.000

# Algoritmo clásico

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

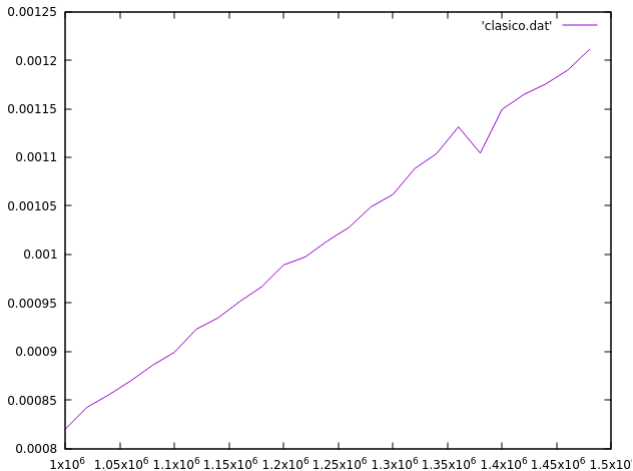
**Tamaños de problema**

Comparación entre  
algoritmos  
Cálculo de la eficiencia  
híbrida

Errores en el cálculo de la  
constante oculta

Vectores con elementos  
repetidos

Fin de la presentación



# Algoritmo Divide y Vencerás

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

**Tamaños de problema**

Comparación entre

algoritmos

Cálculo de la eficiencia

híbrida

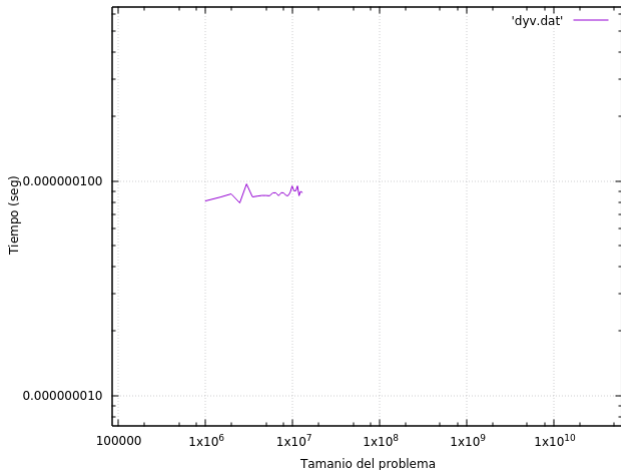
Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación



# Algoritmo Divide y Vencerás (zoom)

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

**Tamaños de problema**

Comparación entre  
algoritmos

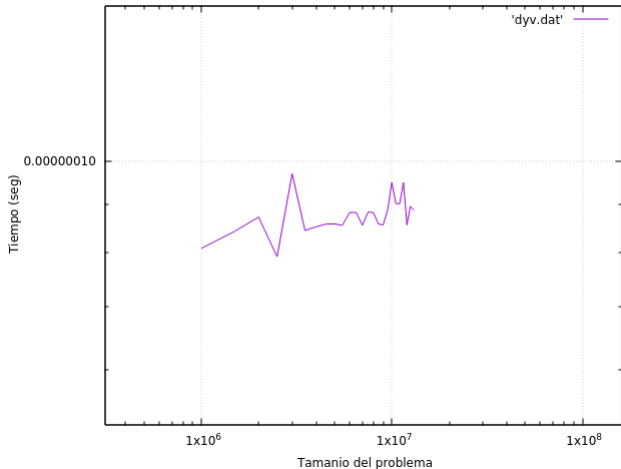
Cálculo de la eficiencia

híbrida

Errores en el cálculo de la  
constante oculta

Vectores con elementos  
repetidos

Fin de la presentación



# Comparación entre ambos algoritmos

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Venceras

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaño del problema

Comparación entre

algoritmos

Cálculo de la eficiencia

brinda

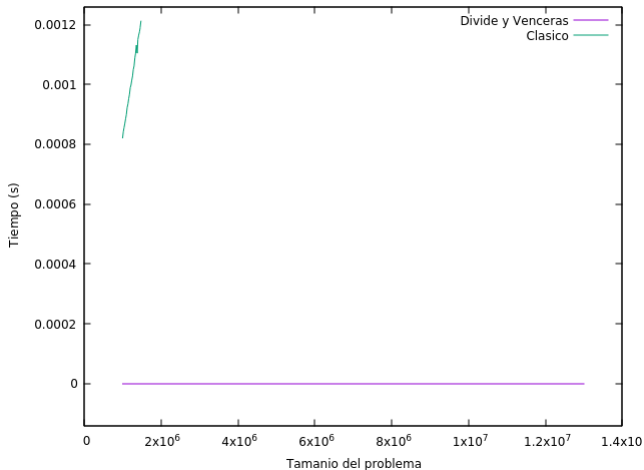
Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación



# Índice

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

**Cálculo de la eficiencia híbrida**

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

## 1 Presentación del problema

## 2 Diseño de algoritmos

- Algoritmo clásico
- Algoritmo Divide y Vencerás
- Medición de tiempos

## 3 Estudio de eficiencia

- Eficiencia empírica
  - Tamaños de problema
  - Comparación entre algoritmos
- **Cálculo de la eficiencia híbrida**
  - Errores en el cálculo de la constante oculta

## 4 Vectores con elementos repetidos

# Errores en el cálculo de la constante oculta

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia

híbrida

**Errores en el cálculo de la constante oculta**

Vectores con elementos repetidos

Fin de la presentación

Algoritmo	Eficiencia teórica	Valor de la constante oculta	Error
Clásico	$O(n)$	$8.19304 \cdot 10^{-10}$	0.1441 %
Divide y Vencerás	$O(\log(n))$	$5.61125 \cdot 10^{-9}$	0.9174 %

# Algoritmo clásico

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre

algoritmos

Cálculo de la eficiencia

híbrida

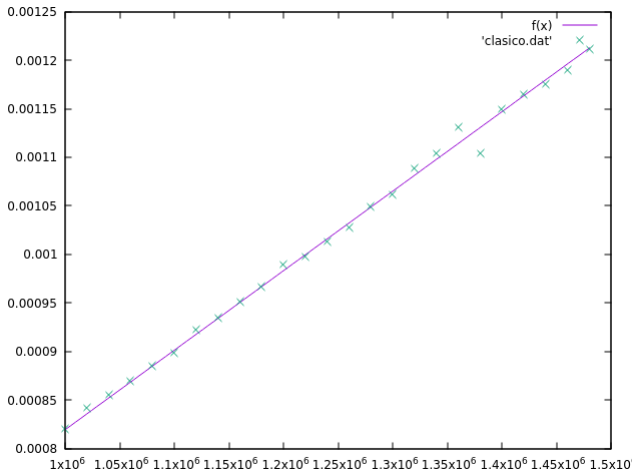
Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación





# Algoritmo Divide y Vencerás

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

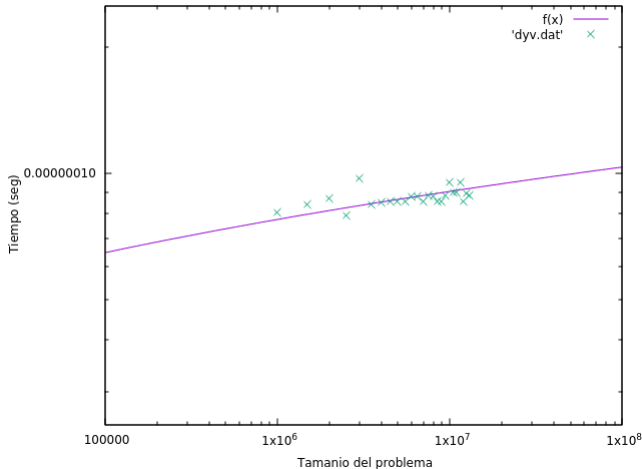
Estudio de eficiencia

Eficiencia empírica

Tamaños de problema  
Comparación entre  
algoritmos  
Cálculo de la eficiencia  
híbrida  
**Errores en el cálculo de la  
constante oculta**

Vectores con elementos  
repetidos

Fin de la presentación



# Índice

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia híbrida

Errores en el cálculo de la constante oculta

Vectores con elementos repetidos

Fin de la presentación

## 1 Presentación del problema

## 2 Diseño de algoritmos

- Algoritmo clásico
- Algoritmo Divide y Vencerás
- Medición de tiempos

## 3 Estudio de eficiencia

- Eficiencia empírica
  - Tamaños de problema
  - Comparación entre algoritmos
- Cálculo de la eficiencia híbrida
  - Errores en el cálculo de la constante oculta

## 4 Vectores con elementos repetidos

# Planteamiento

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia

híbrida

Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación

¿Seguirá funcionando el algoritmo Divide y Vencerás si se repiten elementos en el vector?

# Elementos repetidos (I)

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia

hibrida

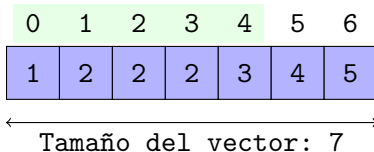
Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación



# Elementos repetidos (II)

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia

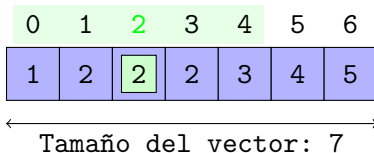
hibrida

Errores en el cálculo de la constante oculta

Vectores con elementos

repetidos

Fin de la presentación



# Elementos repetidos (III)

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema  
Comparación entre  
algoritmos  
Cálculo de la eficiencia  
híbrida  
Errores en el cálculo de la  
constante oculta

Vectores con elementos  
repetidos

Fin de la presentación



Calculamos el centro del vector.

$$mitad = \frac{ini+fin-6}{2} = 3$$

# Elementos repetidos (IV)

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre

algoritmos

Cálculo de la eficiencia

híbrida

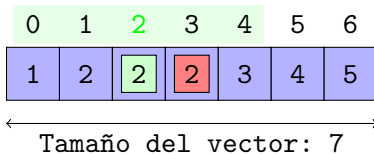
Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación



$$mitad = \frac{6}{2} = 3$$

$$¿v[3] = 3? \text{ NO}$$

$¿v[3] = 2 < 3? \text{ Sí} \rightarrow$  exploramos desde  $mitad + 1 = 4$  a  $fin = 6$

# Elementos repetidos (V)

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia

híbrida

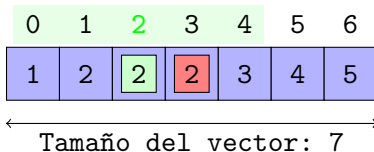
Errores en el cálculo de la

constante oculta

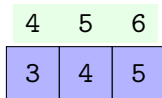
Vectores con elementos

repetidos

Fin de la presentación



$$mitad = \frac{6}{2} = 3$$



Tamaño del vector: 3

Al repetirse elementos, destruimos el axioma principal de la estrategia Divide y Vencerás → El algoritmo **no** es válido.



# Sin embargo...

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre

algoritmos

Cálculo de la eficiencia

híbrida

Errores en el cálculo de la

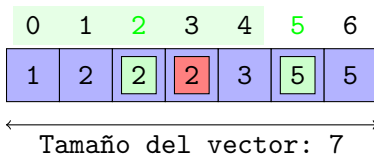
constante oculta

Vectores con elementos

repetidos

Fin de la presentación

Hay un caso en el que puede funcionar: si en alguna iteración  
 $v[\textit{mitad}] = \textit{mitad}$ .



# Fin

Presentación del problema

Diseño de algoritmos

Algoritmo clásico

Algoritmo Divide y Vencerás

Medición de tiempos

Estudio de eficiencia

Eficiencia empírica

Tamaños de problema

Comparación entre algoritmos

Cálculo de la eficiencia

híbrida

Errores en el cálculo de la

constante oculta

Vectores con elementos

repetidos

Fin de la presentación

# Fin de la presentación