

# Algoritmos de tipo Divide y Vencerás

1. Planteamiento general
2. Determinación del caso base
3. Ejemplos clásicos
  - 3.1. Búsqueda binaria
  - 3.2. Algoritmos de ordenación
  - 3.3. Problema de la selección
  - 3.4. Multiplicación de matrices de Strassen

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- **Idea general**
  - Descomponer un problema en subproblemas (más pequeños)
  - Resolver cada subproblema
  - Combinar las soluciones

Algorítmica - José María Gómez Hidalgo

# 1. Planteamiento general

```
Función DV(X: Datos del problema)  
si pequeño(X)  
entonces  
  S = sencillo(X)  
sino (X1, ..., XI) = descomponer(X)  
  para i = 1 hasta I  
    hacer Si = DV(Xi)  
  S = combinar(S1, ..., SI)  
devolver S
```

Algorítmica - José María Gómez Hidalgo

# 1. Planteamiento general

- Asumiendo la existencia de
  - pequeño(X) - función booleana que determina si es práctico aplicar sencillo(X)
  - sencillo(X) - algoritmo práctico para el problema si su tamaño es suficientemente pequeño
  - descomponer(X) - función que descompone P en I subproblemas P1, ..., PI con entradas X1, ..., XI
  - combinar(S1, ..., SI) - función que combina I soluciones Si para Pi, en una única para P

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- Efectividad del enfoque
  - Debe elegirse correctamente el umbral a partir del cual aplicar sencillo(X)
  - Las funciones de descomposición y combinación deben ser eficientes
  - Los subproblemas deben ser aproximadamente del mismo tamaño

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- Complejidad
  - pequeño(X) tarda un tiempo asimilable a otras funciones del esquema
  - sencillo(X) tarda  $f(n)$
  - descomponer(X) y combinar(S1, ... S1) tardan entre los dos  $g(n)$
  - el tamaño de cada subproblema es de  $n/b$

Algorítmica - José María Gómez Hidalgo

# 1. Planteamiento general

- Complejidad

$t(n) = f(n)$  si pequeño(X)

$t(n) = l.t(n/b) + g(n)$  en otro caso

y si  $g(n) \in \theta(n^k)$

$t(n) \in \theta(n^k)$

si  $l < b^k$

$t(n) \in \theta(n^k \cdot \log n)$

si  $l = b^k$

$t(n) \in \theta(n^{l \log l})$

si  $l > b^k$

Algorítmica - José María Gómez Hidalgo

# 1. Planteamiento general

- Observación

– Si  $l =$  número de subproblemas

– Y  $n/b =$  tamaño de los subproblemas

– No necesariamente  $l = b$  !!

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- Ejemplo (Multiplicación de grandes enteros)
  - Multiplicar dos números  $a$  y  $b$  enteros muy grandes
  - Algoritmo clásico  $\in \theta(n^2)$  con  $n$  = número de cifras del mayor de los números

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- Ejemplo (Multiplicación de grandes enteros)
  - Enfoque  $DyV1$
  - Dividir cada número en  $n/2$  cifras
  - Ejemplo  $a = 981$ ,  $b = 1234$ 
    - $w = 09$ ,  $x = 81$
    - $y = 12$ ,  $z = 34$
    - $a \cdot b = (10^2w + x)(10^2y + z) = 10^4wy + 10^2(wz + xy) + xz = 1080000 + 127800 + 2754 = 1210554$

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- Ejemplo (Multiplicación de grandes enteros)
  - Complejidad  $DyV1$ 
$$t(n) = 1 \text{ si } n = 1$$
$$t(n) = 4 \cdot t(n/2) + n \text{ en otro caso}$$
$$\Rightarrow t(n) \in \theta(n^2)$$

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- Ejemplo (Multiplicación de grandes enteros)
  - Enfoque  $DyV2$
  - No calcular  $(wz + xy) = (w + x)(y + z) - wy - xz$
  - $p = wy$ ,  $q = xz$ ,  $r = (w + x)(y + z)$ 
    - $a \cdot b = 10^4p + 10^2(r - p - q) + q$

Algorítmica - José María Gómez Hidalgo

## 1. Planteamiento general

- Ejemplo (Multiplicación de grandes enteros)
    - Complejidad  $DyV^2$
    - Usando el algoritmo clásico para p, q, r, ahorramos 25%
    - Además
      - $t(n) = 1$  si  $n = 1$
      - $t(n) = 3.t(n/2) + n$  en otro caso
- $\Rightarrow t(n) \in \theta(n^{\log_3}) = \theta(n^{1.585})$

Algorítmica - José María Gómez Hidalgo

## 2. Determinación del caso base

- Ejemplo (Multiplicación de grandes enteros)
    - Supongamos que usamos el algoritmo  $DyV^2$  con
      - $t(n) = h(n)$  si  $n \leq n_0$
      - $t(n) = 3.t(n/2) + g(n)$  en otro caso
- siendo  $h(n) = n^2$   $\mu s$  el tiempo de la multiplicación clásica y  $g(n) = 16n$   $\mu s$  el tiempo de las sumas

Algorítmica - José María Gómez Hidalgo

## 2. Determinación del caso base

- La elección del umbral de aplicación de sencillo(X) (es decir, pequeño(X)) es fundamental
- Si no se elige adecuadamente, el enfoque D y V es muy ineficiente

Algorítmica - José María Gómez Hidalgo

## 2. Determinación del caso base

- Ejemplo (Multiplicación de grandes enteros)
  - Si  $n_0 = 1$ 
    - DyV2 tarda 41 segundos para 5000 cifras
    - M. Clásica tarda 25 segundos para 5000 cifras
  - Si  $n_0 = 64$ 
    - DyV2 tarda 6 segundos para 5000 cifras

Algorítmica - José María Gómez Hidalgo

## 2. Determinación del caso base

- El umbral depende de
  - los algoritmos
  - las implementaciones
- Cálculo del umbral
  - Calcular la complejidad de los algoritmos
  - Calcular las constantes de la implementación
  - Calcular el  $n_0$  tal que los tiempos del algoritmo DV y del clásico coincidan en él

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.1. Búsqueda binaria

- Problema
  - Dada una matriz  $T$  de  $n$  componentes ordenada crecientemente ( $T[i] \leq T[i+1]$ ), y dado un elemento  $x$ , encontrar el  $i$  tal que
    - $1 \leq i \leq n+1$
    - $T[i-1] < x \leq T[i]$
  - Observación
    - Si  $x$  no está en  $T$ , se pide dónde debería estar

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.1. Búsqueda binaria

- Solución clásica (secuencial)

```
Función bsecuencial(T : matriz [1..n] de entero, x: entero) :  
entero  
  para i = 1 hasta n  
    hacer  
      si (x ≤ T[i])  
        entonces devolver i  
      devolver n+1
```

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.1. Búsqueda binaria

- Complejidad (secuencial)
  - $O(n)$  en el caso peor
  - $\Omega(1)$  en el caso mejor
  - $\theta((n+1)/2) = \theta(n)$  en media

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.1. Búsqueda binaria

- Enfoque Divide y Vencerás (binaria)
  - Dividir la matriz en dos partes de igual tamaño
  - Buscar en una sola de las dos partes
  - (Para reducir el número de llamadas, mirar primero si x puede estar en la matriz)

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.1. Búsqueda binaria

Función bbinaria(T: matriz [1..n] de entero, x: entero) :  
entero  
si ((n = 0) o (x > T[n]))  
entonces devolver n+1  
sino devolver brecursiva(T, 1, n, x)

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.1. Búsqueda binaria

Función brecursiva(T : matriz [1..n] de entero, i, j, x : entero) :

entero

si (i = j)

entonces devolver i

k = (i+j) div 2

si x ≤ T[k]

entonces devolver brecursivo(T, i, k, x)

sino devolver brecursivo(T, k+1, j, x)

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.1. Búsqueda binaria

- Complejidad

$$t(n) = 1 \text{ si } n = 1$$

$$t(n) = t(n/2) + g(n) \text{ si } n = 2^i, n > 1$$

$$\text{con } g(n) \in \theta(1)$$

$$\Rightarrow t(n) \in \theta(\log_2 n) = \theta(\log n)$$

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.1. Búsqueda binaria

- Observaciones
  - Perdemos eficiencia en el caso mejor
    - El caso mejor de bsecuencial es más general
  - Ganamos eficiencia en todos los demás casos
  - La versión iterativa de bbinaria es equivalente en tiempo

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Problema
  - Dada una matriz  $T$  de  $n$  componentes, ordenarla ascendentemente
    - $T[i] \leq T[i+1]$  para  $1 \leq i \leq n-1$
  - Disponemos de dos algoritmos que ordenan matrices en tiempo  $O(n^2)$ 
    - Ordenación por selección
    - Ordenación por inserción

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Idea de un algoritmo de tipo D y V
  - Descomponer la matriz original en dos partes
  - Ordenar cada parte por separado
  - Combinar las partes ordenadas
- Dos enfoques
  - Descomposición simple, combinación compleja
  - Descomposición compleja, combinación simple

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Ordenación por mezcla
  - Dividir la matriz por la mitad
  - Ordenar cada mitad
  - Mezclar las mitades ordenadas
- Precisa espacio auxiliar (no es *in situ*)

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

Procedimiento ordmezcla(T: matriz [1..n+1] de entero, i,j: entero)

U, V: matriz [1..n+1] de entero

si (i < j)

entonces

k = (i+j) div 2

para l = i hasta k hacer U[l] = T[l]

para l = k+1 hasta j hacer V[l] = T[l]

ordmezcla(U,i,k)

ordmezcla(V,k+1,j)

mezclar(U,V,T,k)

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

Procedimiento mezclar(U,V,T: matriz [1..n+1] de entero, i,j,k: entero)

U[k+1] = ∞; V[j+1] = ∞

{marcadores para evitar comparaciones}

i1 = i; j1 = k+1 {índices}

para k2 = i hasta j

hacer si (U[i1] < V[j1])

entonces T[k2] = U[i1]

i2 = i2 +1

sino T[k2] = V[j1]

j2 = j2 +1

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Complejidad
  - La descomposición y la mezcla se hacen en tiempo lineal
    - $t(n) = 2t(n \operatorname{div} 2) + g(n)$  si  $n$  es par
    - $t(n) = t(n \operatorname{div} 2) + t(n \operatorname{div} 2 + 1) + g(n)$  si  $n$  es impar
  - $t(n) = 2t(n/2) + g(n)$  si  $n = 2^i$
  - con  $g(n) \in \theta(n)$

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Complejidad
  - $t(n) \in \theta(n \cdot \log_2 n)$
  - Precisa mucho espacio auxiliar
  - Caso base:  $j$  -i suficientemente pequeño => ordenar por inserción

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Ordenación rápida (Hoare)
  - Se elige un elemento “pivote”
  - Se desplazan a un lado los elementos menores que él y a otro los mayores
  - Se ordenan las submatrices
  - No hace falta combinar los resultados

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Observaciones
  - Para que los subcasos queden equilibrados, la elección ideal del pivote es la mediana
- La descomposición
  - no debe necesitar espacio auxiliar (*in situ*)
  - debe hacerse en una sola pasada

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

Procedimiento pivote(T: matriz [1..n] de entero, i,j, var l:  
entero)  
  p = T[i]; k = i; l = j+1  
  repetir k = k+1 hasta (T[k] > p ó k ≥ j)  
  repetir l = l - 1 hasta (T[l] ≤ p)  
  mientras (k < l)  
    hacer intercambiar(T,k,l)  
    repetir k = k+1 hasta (T[k] > p)  
    repetir l = l - 1 hasta (T[l] ≤ p)  
  intercambiar(T,i,l)

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

Procedimiento ordrapida(T: matriz [1..n] de entero, i,j:  
entero)  
  si (i < j)  
    entonces  
      pivote(T,i,j,l)  
      ordrapida(T,i,l-1)  
      ordrapida(T,l+1,j)

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Complejidad
  - El tamaño de los subproblemas depende de la posición donde va el elemento pivote
  - La posición del pivote depende de los datos del problema
  - Estudio del mejor y peor casos, y en media

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Complejidad (peor caso)
  - El peor caso se da cuando los subproblemas están más desequilibrados (por ejemplo, T ordenada)

$$t(n) = t(n-1) + t(1) + g(n)$$

con  $g(n) \in \theta(n)$  (cálculo del pivote)

$$\Rightarrow t(n) \in \theta(n^2)$$

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Complejidad (mejor caso)
  - El mejor caso se da cuando los subproblemas están perfectamente equilibrados (por ejemplo, cuando el elemento pivote es siempre la mediana de la submatriz correspondiente)

$$t(n) = 2t(n/2) + g(n) \text{ si } n = 2^i$$

$$\text{con } g(n) \in \theta(n) \Rightarrow t(n) \in \theta(n \cdot \log n)$$

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Complejidad (en media)
  - Suposiciones
    - Los n elementos de T son distintos
    - pivote(T, l, n, l) asigna a l una posición cualquiera entre l y n => todas las posiciones equiprobables
  - Se estudia el número medio de llamadas y operaciones adicionales
    - =>  $t(n) \in \theta(n \cdot \log n)$

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.2. Algoritmos de ordenación

- Observaciones
  - Una versión práctica
    - Tomar la mediana de  $T[1]$ ,  $T[n/2]$ ,  $T[n]$
  - Situación general
    - Los elementos se repiten => modificar pivote( $T, i, j, l$ ) para obtener pivote( $T, i, j, k, l$ ) de modo que entre  $k$  y  $l$  estén los elementos repetidos
  - Se puede lograr  $\theta(n \cdot \log n)$  pero con una constante oculta alta

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.3. Problema de la selección

- Problema
  - Dada una matriz  $T$  de  $n$  componentes con elementos desordenados, encontrar el  $k$ -ésimo elemento más pequeño
    - Si  $k = 1$ , se busca el mínimo
    - Si  $k = n$ , se busca el máximo
    - Si  $k = n/2$ , se busca la mediana (elemento que deja a su izda. el 50% de los datos, y a su dcha. el otro 50%)
    - Si  $k = 1/4, 1/2$  y  $3/4$ , se buscan los *cuartiles*
    - Si  $k = 1/10, 1/5, 3/10, \dots, 9/10$ , se buscan los *deciles*

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.3. Problema de la selección

- Algoritmo sencillo
  - Ordenar la matriz
  - Seleccionar el elemento en la posición  $k$
- Complejidad
  - Usando la ordenación rápida
    - $O(n^2)$  en el caso peor
    - $O(n \cdot \log n)$  en el caso mejor y en media

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.3. Problema de la selección

- Opción
  - Usar el procedimiento pivote( $T, i, j, l$ )
  - El pivote queda en la posición  $l$ 
    - Si  $l = k$ , hemos acertado
    - Si  $l > k$ , hay que buscar el elemento en las casillas de  $i$  a  $l-1$
    - Si  $l < k$ , hay que buscar el elemento en las casillas de  $l+1$  a  $k$

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.3. Problema de la selección

Función seleccion(T: matriz [1..n] de entero, k: entero):

```
entero  
i = 1; j = n  
pivote(T,i,j,l)  
mientras (k ≠ l)  
  hacer si (l > k)  
    entonces j = l-1  
    sino i = l+1  
  pivote(T,i,j,l)  
devolver T[l]
```

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.3. Problema de la selección

- **Complejidad**
    - Depende de los datos de entrada, porque se basa en el procedimiento pivote
  - **Caso mejor**
    - Acertamos a la primera
- $t(n) \in \theta(n)$

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.3. Problema de la selección

- Complejidad
  - Caso peor
    - Por ejemplo,  $k = 1$  y  $T = [9, 1, 2, 3, 4, 5, 6, 7, 8]$   
 $t(n) \in \theta(n^2)$
  - En media
    - Se hace un número logarítmico de llamadas a `pivote()`, cada una con coste lineal  
 $t(n) \in \theta(n \cdot \log n)$

Algorítmica - José María Gómez Hidalgo

## 3. Ejemplos clásicos

### 3.4. Multiplicación de matrices de Strassen

- Problema
  - Multiplicar dos matrices de  $n \times n$  componentes
- Solución elemental
  - La que viene dada por la definición clásica

Algorítmica - José María Gómez Hidalgo

### 3. Ejemplos clásicos

#### 3.4. Multiplicación de matrices de Strassen

```
Función multiplicar(A,B: matriz [1..n][1..n] de entero):  
matriz [1..n][1..n] de entero  
R : matriz [1..n][1..n] de entero  
para i = 1 hasta n  
hacer para j = 1 hasta n  
hacer R[i][j] = 0  
para k = 1 hasta n  
hacer R[i][j] = R[i][j] + A[i][k] . B[k][j]  
devolver R
```

Algorítmica - José María Gómez Hidalgo

### 3. Ejemplos clásicos

#### 3.4. Multiplicación de matrices de Strassen

- Complejidad
  - $t(n) \in \theta(n^3)$
- Enfoque Divide y vencerás
  - Descomponer las matrices en menores de tamaño inferior

Algorítmica - José María Gómez Hidalgo

### 3. Ejemplos clásicos

#### 3.4. Multiplicación de matrices de Strassen

- Descomposición

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \quad R = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix}$$

$$\begin{aligned} R_{11} &= A_{11}XB_{11} + A_{12}XB_{21} \\ R_{12} &= A_{11}XB_{12} + A_{12}XB_{22} \\ R_{21} &= A_{21}XB_{11} + A_{22}XB_{21} \\ R_{22} &= A_{21}XB_{12} + A_{22}XB_{22} \end{aligned}$$

Algorítmica - José María Gómez Hidalgo

### 3. Ejemplos clásicos

#### 3.4. Multiplicación de matrices de Strassen

- Complejidad

$$t(n) = 1 \text{ si } n \leq 2$$

$$t(n) = 8t(n/2) + n^2 \text{ si } n > 2, n = 2^i$$

$$\Rightarrow t(n) \in \theta(n^3)$$

Algorítmica - José María Gómez Hidalgo

### 3. Ejemplos clásicos

#### 3.4. Multiplicación de matrices de Strassen

- Reducción del número de multiplicaciones (Strassen, 1970)

$$\begin{aligned} P &= (A_{11}+A_{22}) (B_{11}+B_{22}) \\ Q &= (A_{21}+A_{22})B_{11} \\ R' &= A_{21}(B_{12}-B_{22}) \\ S &= A_{22}(B_{21}-B_{11}) \\ T &= (A_{11}+A_{12})B_{22} \\ U &= (A_{21}-A_{11}) (B_{11}+B_{12}) \\ V &= (A_{12}-A_{22}) (B_{21}+B_{22}) \end{aligned} \quad \Rightarrow \quad \begin{aligned} R_{11} &= P+S-T+V \\ R_{12} &= R'+T \\ R_{21} &= Q+S \\ R_{22} &= P+R'-Q+U \end{aligned}$$

Algorítmica - José María Gómez Hidalgo

### 3. Ejemplos clásicos

#### 3.4. Multiplicación de matrices de Strassen

- Complejidad

$$t(n) = 1 \text{ si } n \leq 2$$

$$t(n) = 7t(n/2) + n^2 \text{ si } n > 2, n = 2^i$$

$$\Rightarrow t(n) \in \theta(n^{\log_2 7}) = \theta(n^{2,81})$$

Algorítmica - José María Gómez Hidalgo