DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Algoritmos "Divide y Vencerás"
Análisis y Diseño de Algoritmos

Algoritmos "Divide y Vencerás"

- Ejemplo: Multiplicación de enteros grandes
- La técnica "divide y vencerás"
  - Características
  - Método general "divide y vencerás"
  - Eficiencia de los algoritmos "divide y vencerás"
- Aspectos de diseño
  - Determinación del umbral
- Aplicaciones

Multiplicación de enteros

Multiplicación de enteros de n cifras:

Algoritmo clásico

1234*5678 = 1234* (5*1000 + 6*100+7*10+8)
= 1234*5*1000 + 1234*6*100 + 1234*7*10 + 1234*8

Operaciones básicas:

- Multiplicaciones de dígitos: O(1)
- Sumas de dígitos: O(1)
- Desplazamientos: O(1)

Eficiencia algoritmo: **$O(n^2)$**

Multiplicación de enteros

Multiplicación de enteros de n cifras:

Algoritmo "divide y vencerás" simple

1234 = 12*100 + 34
5678 = 56*100 + 78

1234*5678 = (12*100 + 34)*(56*100 + 78)
= 12*56*10000 + (12*78+34*56)*100 + (34*78)

Idea: Se reduce una multiplicación de 4 cifras
a cuatro multiplicaciones de 2 cifras,
más tres sumas y varios desplazamientos.

Multiplicación de enteros

Multiplicación de enteros de n cifras:
Algoritmo "divide y vencerás" simple

1. Dividir

   $X = 12345678 = xi*10^4 + xd$ $\quad xi=1234 \quad xd=5678$
   $Y = 24680135 = yi*10^4 + yd$ $\quad yi=2468 \quad yd=0135$

2. Combinar

$$X*Y = (xi*10^4 + xd) * (yi*10^4 + yd)$$
$$= xi*yi*10^8 + (xi*yd+xd*yi)*10^4 + xd*yd$$

Multiplicación de enteros

Multiplicación de enteros de n cifras:
Algoritmo "divide y vencerás" simple

En general:

$$X = xi*10^{n/2} + xd$$
$$Y = yi*10^{n/2} + yd$$

$$X*Y = (xi*10^{n/2} + xd) * (yi*10^{n/2} + yd)$$
$$= xi*yi*10^n + (xi*yd+xd*yi)*10^{n/2} + xd*yd$$

Multiplicación de enteros

```
función multiplica (X,Y,n)
{
    if (P es pequeño) {
        return X*Y;
    } else {
        Obtener xi, xd, yi, yd;                 // DIVIDIR
        z1 = multiplica (xi, yi, n/2);
        z2 = multiplica (xi, yd, n/2);
        z3 = multiplica (xd, yi, n/2);
        z4 = multiplica (xd, yd, n/2);
        aux = suma(z2,z3);                      // COMBINAR
        z1  = desplaza_izq(z1,n);
        aux = desplaza_izq(aux,n/2);
        z   = suma(z1,aux);
        z   = suma(z,z4);
        return z;
    }
}
```


Multiplicación de enteros

	Eficiencia
`función multiplica (X,Y,n)`	**Eficiencia**
`{`	
`if (P es pequeño) {`	O(1)
`return X*Y;`	O(1)
`} else {`	
`Obtener xi, xd, yi, yd;`	O(n)
`z1 = multiplica (xi, yi, n/2);`	T(n/2)
`z2 = multiplica (xi, yd, n/2);`	T(n/2)
`z3 = multiplica (xd, yi, n/2);`	T(n/2)
`z4 = multiplica (xd, yd, n/2);`	T(n/2)
`aux = suma(z2,z3);`	O(n)
`z1 = desplaza_izq(z1,n);`	O(n)
`aux = desplaza_izq(aux,n/2);`	O(n)
`z = suma(z1,aux);`	O(n)
`z = suma(z,z4);`	O(n)
`return z;`	O(1)
`}`	
`}`	

Multiplicación de enteros

Multiplicación de enteros de n cifras:

Algoritmo "divide y vencerás" simple

$T(n) = 4T(n/2) + n \in O(n^2)$

- El cuello de botella está en el número de multiplicaciones de tamaño n/2.
- Para mejorar la eficiencia debemos reducir el número de multiplicaciones necesario…

Multiplicación de enteros

Multiplicación de enteros de n cifras:

Algoritmo "divide y vencerás"

$r = (xi+xd)*(yi+yd) = xi*yi + (xi*yd+xd*yi) + xd*yd$

$p = xi*yi$

$q = xd*yd$

$X*Y = p*10^n + (r-p-q)*10^{n/2} + q$

- Luego podemos realizar una multiplicación de tamaño n a partir de 3 multiplicaciones de tamaño n/2.

Multiplicación de enteros

```
función multiplicaDV (X,Y,n)
{
    if (P es pequeño) {
        return X*Y;
    } else {
        Obtener xi, xd, yi, yd;                 // DIVIDIR
        s1 = suma(xi,xd);
        s2 = suma(yi,yd);
        p = multiplicaDV (xi, yi, n/2);
        q = multiplicaDV (xd, yd, n/2);
        r = multiplicaDV (s1, s2, n/2);
        aux = suma(r,-p,-q);                    // COMBINAR
        aux = desplaza_izq(aux,n/2);
        p = desplaza_izq(p,n);
        z = suma(p,aux,q);
        return z;
    }
}
```


Multiplicación de enteros

```
función multiplicaDV (X,Y,n)                    Eficiencia
{
    if (P es pequeño) {                         O(1)
        return X*Y;                             O(1)
    } else {
        Obtener xi, xd, yi, yd;                 O(n)
        s1 = suma(xi,xd);                       O(n)
        s2 = suma(yi,yd);                       O(n)
        p = multiplicaDV (xi, yi, n/2);         T(n/2)
        q = multiplicaDV (xd, yd, n/2);         T(n/2)
        r = multiplicaDV (s1, s2, n/2);         T(n/2)
        aux = suma(r,-p,-q);                    O(n)
        aux = desplaza_izq(aux,n/2);            O(n)
        p = desplaza_izq(p,n);                  O(n)
        z = suma(p,aux,q);                      O(n)
        return z;                               O(1)
    }
}
```

Multiplicación de enteros

Multiplicación de enteros de n cifras:

Algoritmo "divide y vencerás"

$$T(n) = 3\mathrm{T}(\mathrm{n}/2) + \mathrm{n} \in \mathrm{O}(n^{\log_2 3}) = O(n^{1.585})$$

	Implementación básica	Implementación eficiente
Operaciones	$\mathbf{n^2}$	$\mathbf{n^{1.585}}$
n = 10	0.1 ms	0.04 ms
n = 100	10 ms	1.48 ms
n = 1000	1 segundo	56.9 ms
n = 10000	100 segundos	2.19 segundos

La técnica "divide y vencerás"

La técnica "divide y vencerás" (DV) consiste en:

- Descomponer el problema que hay que resolver en cierto número de subproblemas más pequeños del mismo tipo.
- Resolver de forma sucesiva e independiente todos estos subproblemas.
- Combinar las soluciones obtenidas para obtener la solución del problema original.

La técnica "divide y vencerás"

Características de los problemas resolubles utilizando "divide y vencerás"

- El problema se puede descomponer en otros del mismo tipo que el original y de tamaño más pequeño (*formulación recursiva*).
- Los subproblemas pueden resolverse de manera independiente.
- Los subproblemas son disjuntos, sin solapamiento.
- La solución final se puede expresar como combinación de las soluciones de los subproblemas.

La técnica "divide y vencerás"

Método general "divide y vencerás"

```
DV(x)
if (x es suficientemente pequeño) {
    return algoritmo_específico(x);
} else {
    descomponer x en {x1,…, xk}
    for i = 1 to k
        yi ← DV(xi)
    y ← recombinar (y1, …, yk)
    return y;
}
```

La técnica "divide y vencerás"

Eficiencia de los algoritmos "divide y vencerás"

$$T(n) = aT\left(\frac{n}{b}\right) + g(n)$$

$$\text{con } g(n) \in O(n^k), a \geq 1, b \geq 2, k \geq 0$$

$$T(n) = \begin{cases} \Theta(n^k), & a < b^k \\ \Theta(n^k \log_b(n)), & a = b^k \\ \Theta(n^{\log_b(a)}) & a > b^k \end{cases}$$

Aspectos de diseño

- **Algoritmo recursivo**
  División del problema en subproblemas
  y combinación eficiente de las soluciones parciales.
  - Los subproblemas deben tener,
    aproximadamente, el mismo tamaño.

- **Algoritmo específico**
  para resolver problemas de tamaño pequeño.

- **Determinación del umbral**
  para decidir cuándo finalizar la descomposición recursiva
  del problema y aplicar el algoritmo específico.

Determinación del umbral

- Umbral óptimo **dependiente de la implementación**: Es difícil hablar del valor adecuado para el umbral n_0 si no tratamos con implementaciones concretas, ya que gracias a ellas conocemos las constantes ocultas que nos permitirán afinar el cálculo de la eficiencia del algoritmo.

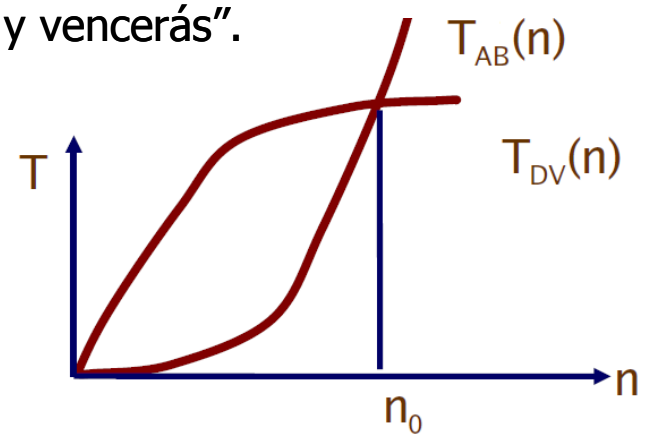
- De partida, no hay restricciones sobre el valor que puede tomar el umbral.

  p.ej. Un umbral óptimo infinito supondría no aplicar nunca DV de forma efectiva (siempre usaríamos el algoritmo específico).

Determinación del umbral

Método experimental

- Implementamos el algoritmo básico (AB) y el algoritmo "divide y vencerás" (DV).
- Resolvemos para distintos valores de n con ambos algoritmos: Conforme n aumente, el tiempo requerido por el algoritmo básico irá aumentando más que el del algoritmo "divide y vencerás".



Determinación del umbral

Método teórico

- La idea del método experimental se traduce en

$$T(n) = h(n) \quad \text{si } n \leq n_0 \quad \text{(algoritmo básico)}$$
$$T(n) = a\,T(n/b) + g(n) \quad \text{si } n > n_0 \quad \text{(algoritmo DV)}$$

- Teóricamente, el umbral óptimo será cuando coinciden los tiempos de ejecución de los dos algoritmos:

$$h(n) = T(n) = a\,h(n/b) + g(n), \quad n = n_0$$

Determinación del umbral

Método híbrido

- Calculamos las constantes ocultas utilizando un enfoque empírico (método experimental).
- Calculamos el umbral utilizando el criterio empleado para calcular el umbral teórico (método teórico)
- Probamos valores alrededor del umbral teórico (umbrales de tanteo) para determinar el umbral óptimo.

Determinación del umbral

Ejemplo: Multiplicación de enteros grandes

$T(n) = 3T(n/2) + g(n)$

$g(n) = 16n$ (ms)

$h(n) = n^2$ (ms)

p.ej. $n=1024$ $\quad n_0 = 1 \Rightarrow t(n) = 31m\ 56s$

$n_0 = \infty \Rightarrow t(n) = 17m\ 29s$

$h(n) = 3\ h(n/2) + g(n)$

$n^2 = \frac{3}{4}\ n^2 + 16\ n \Rightarrow n = \frac{3}{4}\ n + 16 \Rightarrow \mathbf{n_0 = 64}$

$n_0 = 64 \Rightarrow t(n) = 7m\ 44s$

Aplicaciones

- Algoritmo de búsqueda binaria.
- Algoritmos de ordenación (Mergesort, Quicksort).
- Problema de la selección (p.ej. mediana)
- Exponenciación rápida.
- Multiplicación de matrices: Algoritmo de Strassen.
- Subsecuencia de suma máxima.
- Par de puntos más cercano.
- Eliminación de superficies ocultas.
- Número de inversiones (rankings).
- FFT: Transformada Rápida de Fourier (convoluciones).
- Interacciones entre n partículas.
- Calendario de una liga…