Universitat de Lleida

Divideix i venç

Tere Alsinet
Grau en Enginyeria Informàtica

Divideix i venç

- Descomposar el problema original en subproblemes més petits però de la mateixa naturalesa
- Resolució recursiva de cada subproblema
- Combinació, si és necessari, de les solucions del múltiples subproblemes

Divisió entera basada en restes

```
Funció div (entrada x, y: natural) retorna <c: nat, r: nat>
    casos:
        x<y      → c := 0    r := x
        x>=y     → <c',r'> := div (x-y, y)
                   c := c' +1
                   r := r'
    fcasos
    retorna < c, r >
fi_funció
```

Complexitat Temporal O(x/y)

Reduir l'ordre de complexitat → A cada crida dividir el cocient per la meitat → A cada crida 2*y

Divisió Eficient

Funció div_eficient (entrada x, y: natural) retorna <c: nat, r: nat>

casos:

$x<y$ → c := 0 r := x

$x>=y$ → <c',r'> := div_eficient (x, 2*y)

casos:

$r' < y$ → c := c' * 2 r := r'

$r' >= y$ → c := (c' * 2) +1 r := r' - y

fcasos

retorna < c, r >

fi_funció

Complexitat Temporal O($\log_2(x/y)$)

Funció recursiva NO final → Transformació a iteratiu → Dos bucles

Sèrie de Fibonacci

```
Funció fibonacci (entrada n: natural) retorna fn: nat
    casos:
        n=0          → fn :=0
        n=1          → fn := 1
        n >= 2       → fn := fibonacci (n-1) + fibonacci (n-2)
    fcasos
    retorna ( fn )
fi_funció
```

Complexitat Temporal $O(2^n)$

Reduir l'ordre de complexitat →

A cada crida calcular f_n i f_{n-1}

Sèrie de Fibonacci

Precondició: n >= 1

Funció fibonacci_eficient (entrada n: natural) retorna <f_n: nat, f_{n-1}: nat>

casos:

n=1 → $f_n := 1$ $f_{n-1} := 0$

n >= 2 → $< f_{n-1}, f_{n-2}> :=$ fibonacci_eficient (n-1)

$f_n := f_{n-1} + f_{n-2}$

fcasos

retorna < f_n, f_{n-1} >

fi_funció

Complexitat Temporal O(n)

Transformació a iteratiu →
Un únic bucle

Càlcul de la part entera de l'arrel quadrada de n

Precondició: $a^2 <= n$

Funció p_entera_arrel (entrada n: natural, a: natural) retorna r: nat

casos:

$(a+1)^2 > n$ → r := a

$(a+1)^2 <= n$ → r := p_entera_arrel (n, a+1)

fcasos

retorna (r)

fi_funció

Postcondició: $r^2 <= n \ \wedge \ (r+1)^2 > n$

Complexitat Temporal O(part entera de arrel n - a)

Si a=0, aleshores

Complexitat Temporal O(part entera de arrel n)

Reduir l'ordre de complexitat →

Acotar l'intèrval de cerca → 2 paràmetres: a, b

Càlcul eficient de la part entera de l'arrel quadrada de n

Precondició: $a^2 <= n \ \wedge \ b^2 > n$ // r està [a, b[

Funció p_entera_arrel_eficient (entrada n: natural; a, b : natural) retorna r: nat

casos:

a+1 = b → r := a

a+1 < b → m:= (a+b) div 2

casos

$m^2 <= n$ → r := p_entera_arrel_eficient(n, m, b)

$m^2 > n$ → r := p_entera_arrel_eficient(n, a, m)

fcasos

fcasos

retorna (r)

fi_funció

Postcondició: $r^2 <= n \ \wedge \ (r+1)^2 > n$

Si a=0 ^ b=n , aleshores

Complexitat Temporal O($\log_2$(part entera de arrel n))

Utilitzant divideix i venç dissenyar algorismes recursius per als problemes:

1- Càlcul del $\log_a b$

2- Càlcul del número combinatori n sobre m

3- Recubriment d'un tauler de n x n posicions amb triminos, sent $n = 2^k$ am $k >= 1$

4- Skyline d'una ciutat amb n edificis, sent $n = 2^k$ am $k >= 0$
Cada edifici es representa amb un triplet:

$(X_{origen}, X_{final}, Y)$