Algoritmos: Seminario 3

Aplicación del Teorema de resolución de recurrencias Divide y Vencerás
Suma Subsecuencia Máxima

José María Casanova, Elena Hernández, Alberto Valderruten, Oscar Fontenla

Dept. Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

jcasanova@udc.es, elena.hernandez@udc.es, alberto.valderruten@udc.es oscar.fontenla@udc.es

UNIVERSIDADE DA CORUÑA

- **Teorema de resolución de recurrencias Divide y Vencerás**

$$T(n) = \ell T(n/b) + cn^k, n > n_0 \tag{1}$$

Con $\ell \geq 1, b \geq 2, k \geq 0, n_0 \geq 1 \in \mathbb{N}$ y $c > 0 \in \mathbb{R}$,
cuando n/n_0 es potencia exacta de b ($n \in \{bn_0, b^2 n_0, b^3 n_0 \ldots\}$).

- **Teorema Divide y Vencerás**:
Si una recurrencia es de la forma (1), se aplica

$$T(n) = \begin{cases} \theta(n^k) & si \quad \ell < b^k \\ \theta(n^k logn) & si \quad \ell = b^k \\ \theta(n^{log_b \ell}) & si \quad \ell > b^k \end{cases} \tag{2}$$

En análisis de algoritmos, se suelen manejar desigualdades:
$T(n) \leq \ell T(n/b) + cn^k, n > n_0$ con n/n_0 potencia exacta de b

$$\Rightarrow T(n) = \begin{cases} O(n^k) & si \quad \ell < b^k \\ O(n^k logn) & si \quad \ell = b^k \\ O(n^{log_b \ell}) & si \quad \ell > b^k \end{cases}$$

Reglas para calcular O

1. operación elemental = 1 $\leftrightarrow$ Modelo de Computación
2. **secuencia**: $S_1 = O(f_1(n)) \wedge S_2 = O(f_2(n))$
   $\Rightarrow \boxed{S_1; S_2} = O(f_1(n) + f_2(n)) = O(max(f_1(n), f_2(n)))$
   - También con Θ
3. **condición**: $B = O(f_B(n)) \wedge S_1 = O(f_1(n)) \wedge S_2 = O(f_2(n))$
   $\Rightarrow \boxed{\textbf{si } B \textbf{ entonces } S_1 \textbf{ sino } S_2} = O(max(f_B(n), f_1(n), f_2(n)))$
   - Si $f_1(n) \neq f_2(n)$ y $max(f_1(n), f_2(n)) > f_B(n) \leftrightarrow$ **Peor caso**
   - ¿Caso medio? $\rightarrow f(n)$: promedio de f_1 y f_2 ponderado con las frecuencias de cada rama $\rightarrow O(max(f_B(n), f(n)))$
4. **iteración**: $B; S = O(f_{B,S}(n)) \wedge$ nº iter$= O(f_{iter}(n))$
   $\Rightarrow \boxed{\textbf{mientras } B \textbf{ hacer } S} = O(f_{B,S}(n) * f_{iter}(n))$
   ssi el coste de las iteraciones no varía, sino: $\sum$ costes indiv.
   $\Rightarrow \boxed{\textbf{para } i \leftarrow x \textbf{ hasta } y \textbf{ hacer } S} = O(f_S(n) *$nº iter$)$
   ssi el coste de las iteraciones no varía, sino: $\sum$ costes indiv.
   - B es comparar 2 enteros $= O(1)$; nº iter $= y - x + 1$

Suma de la Subsecuencia Máxima (1)

- **Problema de la Suma de la Subsecuencia Máxima**
  $a_1..a_n \rightarrow \sum_{k=i}^{j} a_k$ máxima?
  Ejemplo: $SSM(-2, 11, -4, 13, -5, -2) = 20$ [2..4]
- *SSM* **recursiva**: estrategia $\boxed{\text{Divide y Vencerás}}$
  Divide la entrada en 2 *mitades* $\rightarrow$ 2 soluciones recursivas
  Vence usando las 2 soluciones $\rightarrow$ solución para entrada original

  La *SSM* puede estar: $\begin{cases} \text{- en la 1ª mitad} \\ \text{- en la 2ª mitad} \\ \text{- entre las 2 mitades} \end{cases}$

  Las dos primeras soluciones son las obtenidas recursivamente.
  La 3ª solución se obtiene sumando:
  - la *SSM* de la 1ª mitad *que incluye el extremo derecho*, y
  - la *SSM* de la 2ª mitad *que incluye el extremo izquierdo*.

Suma de la Subsecuencia Máxima (2) - *SSM* recursiva

```
    función SSM ( a[1..n] ) : valor          función interfaz
      devolver SSM recursiva (a, 1, n)
    fin función

    función SSM recursiva (var a[1..n], izq, der) : valor
{1}   si izq = der entonces
{2}     si a[izq] > 0 entonces
{3}       devolver a[izq]                    caso base: si >0, es SSM
        sino
{4}       devolver 0
        fin si
      sino
{5}     Centro := (izq + der) div 2 ;
{6}     Primera solución := SSM recursiva (a, izq, Centro) ;
{7}     Segunda solución := SSM recursiva (a, Centro + 1, der) ;
```

```
{8}        Suma máxima izquierda := 0 ; Suma izquierda := 0 ;
{9}        para i := Centro hasta izq paso -1 hacer
{10}         Suma izquierda := Suma izquierda + a[i] ;
{11}         si Suma izquierda > Suma máxima izquierda entonces
{12}            Suma máxima izquierda := Suma izquierda
           fin para ;

{13}       Suma máxima derecha := 0 ; Suma derecha := 0 ;
{14}       para i := Centro + 1 hasta der hacer
{15}         Suma derecha := Suma derecha + a[i] ;
{16}         si Suma derecha > Suma máxima derecha entonces
{17}            Suma máxima derecha := Suma derecha
           fin para ;

{18}       devolver max (Primera solución, Segunda solución,
                            Suma máxima izquierda + Suma máxima derecha)
        fin si
      fin función
```

Suma de la Subsecuencia Máxima - Ejercicio

- Entender y ejecutar el algoritmo SSM recursivo con un ejemplo y dibujar el arbol de recursividad.
- Analizar el algoritmo SSM planteando la relación de recurrencia y aplicando teorema de resolución de recurrencias divide y vencerás.

- **Análisis**:
  Caso base: $\{1\text{-}4\} \Rightarrow T(1) = \Theta(1)$
  Ciclos $\{9\text{-}12\}$ y $\{14\text{-}17\}$: $\Theta(n)$ *en conjunto*: $a_1..a_n$
  Llamadas recursivas $\{6\}$ y $\{7\}$: $T(n/2)$ cada una (aprox.)
  Resto = $\Theta(1)$: se puede ignorar frente a $\Theta(n)$
  Relación de recurrencia: $\begin{cases} T(1) = 1 \\ T(n) = 2T(n/2) + n, n > 1 (*) \end{cases}$
- **Aplicando teoremas:**

  $\boxed{\textit{Teorema de resolución de recurrencias Divide y Vencerás}}$

  $T(n) = lT(n/b) + cn^k, n > n_0,$
  con $l \geq 1$, $b \geq 2$, $c > 0 \in \mathbb{R}$, $k \geq 0 \in \mathbb{N}$, $n_0 \geq 1 \in \mathbb{N}$
  $\{l = 2, b = 2, c = 1, k = 1, n_0 = 1\}$: caso $l = b^k$
  $\Rightarrow T(n) = \Theta(n^k logn) \Rightarrow T(n) = \Theta(nlogn)$

Suma de la Subsecuencia Máxima (5)

- **Observación**: pasar el vector *a* **por referencia** (var), sino:

  Sea $R(n)$: nº de copias de *a*: $\begin{cases} R(1) = 0 \\ R(n) = 2R(n/2) + 2, n > 1 \end{cases}$

  $\Rightarrow R(n) = 2n - 2$ copias $*\Theta(n)$ cada una $\Rightarrow T(n) = \Theta(n^2)$

  También la complejidad espacial sería cuadrática!

Bibliografía

- G. Brassard y P. Bratley,
  Fundamentals of Algorithmics, Prentice Hall 1996.
- G. Brassard y P. Bratley,
  Fundamentos de Algoritmia, Prentice Hall 1997.