Algoritmos sobre secuencias y conjuntos de datos

Suma de la Subsecuencia Máxima
Búsqueda Binaria

Alberto Valderruten

Dept. de Computación, Universidade da Coruña

alberto.valderruten@udc.es

Índice

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: $\begin{cases} \text{- en la 1}^{\text{a}} \text{ mitad} \\ \text{- en la 2}^{\text{a}} \text{ mitad} \\ \text{- entre las 2 mitades} \end{cases}$

  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) ;
```

Suma de la Subsecuencia Máxima (3) - *SSM* recursiva

```
      Suma máxima izquierda := 0 ; Suma izquierda := 0 ;
      para i := Centro hasta izq paso -1 hacer
        Suma izquierda := Suma izquierda + a[i] ;
        si Suma izquierda > Suma máxima izquierda entonces
           Suma máxima izquierda := Suma izquierda
      fin para ;

      Suma máxima derecha := 0 ; Suma derecha := 0 ;
      para i := Centro + 1 hasta der hacer
        Suma derecha := Suma derecha + a[i] ;
        si Suma derecha > Suma máxima derecha entonces
           Suma máxima derecha := Suma derecha
      fin para ;

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

Suma de la Subsecuencia Máxima (4) - *SSM* recursiva

- **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: $\begin{cases} T(1) = 1 \\ T(n) = 2T(n/2) + n, n > 1 (*) \end{cases}$

1. Razonando con inducción:

$$\begin{array}{cccccccc} T(& 2 &) = & 4 & = & 2 & * & 2 \\ & 4 & & 12 & & 4 & * & 3 \\ & 8 & & 32 & & 8 & * & 4 \\ & 16 & & 80 & & 16 & * & 5 \\ & ... & & & & & & \end{array}$$

$\Rightarrow T(n) = n(k+1)$ para $n = 2^k$: *demostrar la hipótesis*
$\Rightarrow T(n) = n(log_2 n + 1) = \Theta(nlogn)$

Suma de la Subsecuencia Máxima (5) - *SSM* recursiva

2. Manejando proyecciones:

a) dividir (*) por n: $\frac{T(n)}{n} = \frac{T(n/2)}{n/2} + 1$

b) proyectar ($n = 2^k$): $\frac{T(n/2)}{n/2} = \frac{T(n/4)}{n/4} + 1$

$\frac{T(n/4)}{n/4} = \frac{T(n/8)}{n/8} + 1$

...

$\frac{T(2)}{2} = \frac{T(1)}{1} + 1$

c) sumar: $\frac{T(n)}{n} = T(1) + log_2 n \Rightarrow T(n) = \Theta(nlogn)$

3. Aplicando teoremas:

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

$T(n) = lT(n/b) + cn^k, n > n_0,$

con $l \geq 1$, $b \geq 2$, $c > 0 \in R$, $k \geq 0 \in N$, $n_0 \geq 1 \in N$

$\{l = 2, b = 2, c = 1, k = 1, n_0 = 1\}$: caso $l = b^k$

$\Rightarrow T(n) = \Theta(n^k logn) \Rightarrow T(n) = \Theta(nlogn)$

$\rightarrow$ **Usar teoremas!**

Suma de la Subsecuencia Máxima (6)

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

  Sea $R(n)$: n^{o} 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!

- *SSM* **en línea**:

  - Acceso secuencial

  - Respuesta para la subsecuencia parcial

  $\Rightarrow$ $\boxed{\text{Algoritmo en línea}}$

  Análisis: *espacio constante* (no se necesita memorizar la entrada) y tiempo lineal

  $\rightarrow$ *mejor imposible*

  Ejercicio: modificar el algoritmo para asegurar espacio constante

Suma de la Subsecuencia Máxima (7) - *SSM* en línea

```
     función SSM en línea ( a[1..n] ) : <i, j, valor>
{1}    i:=1 ; EstaSuma:=0 ; SumaMax:=0 ; MejorI:=0 ; MejorJ:=0 ;
{2}    para j := 1 hasta n hacer
{3}       EstaSuma := EstaSuma + a[j] ;
{4}       si EstaSuma > SumaMax entonces
{5}          SumaMax := EstaSuma ;
{6}          MejorI := i ;
{7}          MejorJ := j
{8}       sino si EstaSuma < 0 entonces
{9}          i := j+1 ;
{10}         EstaSuma := 0
          fin si
       fin para ;
{11}   devolver < MejorI, MejorJ, SumaMax>
     fin función
```

Índice

Búsqueda Binaria (1)

- Ejemplo de *algoritmo logarítmico*
- Dados x y un vector *ordenado* a_1, a_2, ... a_n de enteros,

$$\text{devolver: } \begin{cases} i \text{ si } \exists a_i = x \\ \text{“elemento no encontrado”} \end{cases}$$

$\rightarrow$ Comparar x y a_{medio}, con $medio = (i+j)div2$, siendo $a_i..a_j$ el *espacio de búsqueda*:

1. $x = a_{medio}$: terminar (interrupción)
2. $x > a_{medio}$: seguir buscando en $a_{medio+1}..a_j$
3. $x < a_{medio}$: seguir buscando en $a_i..a_{medio-1}$

- ¿nº iter? $\leftrightarrow$ evolución del tamaño d del espacio de búsqueda

Invariante: $d = j - i + 1$

$$\text{¿Cómo decrece d? } \begin{cases} i \leftarrow medio + 1 \\ j \leftarrow medio - 1 \end{cases}$$

- *Peor caso*: se alcanza la terminación “normal” del bucle $\equiv i > j$

Búsqueda Binaria (2)

```
     función Búsqueda Binaria (x, a[1..n]) : posición
     {a: vector ordenado de modo no decreciente}

{1}     i := 1 ; j := n ;                    {espacio de búsqueda: i..j}
{2}     mientras i <= j hacer
{3}        medio := (i + j) div 2 ;
{4}        si a[medio] < x entonces
{5}           i := medio + 1
{6}        sino si a[medio] > x entonces
{7}           j := medio - 1
{8}        sino devolver medio               {se interrumpe el bucle}
        fin mientras;
{9}     devolver "elemento no encontrado"    {fin normal del bucle}
     fin función
```

Búsqueda Binaria (3) - Análisis del peor caso

- Sea $< d, i, j >$ iteración $< d', i', j' >$:
  1. $i \leftarrow medio + 1$:

$$
\begin{aligned}
i' &= (i+j)div2 + 1 \\
j' &= j \\
d' &= j' - i' + 1 && = j - (i+j)div2 - 1 + 1 \\
&&& \leq j - (i+j-1)/2 \\
&&& = (j-i+1)/2 \\
&&& = d/2
\end{aligned}
$$

$$\boxed{\rightarrow d' \leq d/2}$$

  2. $j \leftarrow medio - 1$:

$$
\begin{aligned}
i' &= i \\
j' &= (i+j)div2 - 1 \\
d' &= j' - i' + 1 && = (i+j)div2 - i - 1 + 1 \\
&&& \leq (i+j)/2 - i \\
&&& < (j-i+1)/2 \\
&&& = d/2
\end{aligned}
$$

$\boxed{\rightarrow d' < d/2}$ *(decrece más rápido)*

Búsqueda Binaria (4) - Análisis del peor caso

- ¿$T(n)$? Sea d_l: d *después* de la l-ésima iteración

  $$\begin{cases} d_0 = n \\ d_l \leq d_{l-1}/2 \ \forall l \geq 1 \end{cases} \quad \text{(inducción)} \rightarrow d_l \leq n/2^l$$

  hasta $d < 1 \rightarrow l = \lceil log_2 n \rceil + 1 = O(logn)$ iteraciones

  Cada iteración es $\Theta(1)$ (reglas) $\Rightarrow T(n) = O(logn)$
- **Razonamiento alternativo**: *pensar en versión recursiva*

  $$T(n) = \begin{cases} 1 & \text{si } n = 0, 1 \\ T(n/2) + 1 & \text{si } n > 1 \end{cases}$$

  Teorema Divide y Vencerás: $l = 1, b = 2, c = 1, k = 0, n_0 = 1$

  Caso $l = b^k \Rightarrow T(n) = \Theta(n^k logn) \rightarrow T(n) = \Theta(logn)$
- **Observaciones**:
  - Pensar en versión recursiva puede ser otro recurso útil
  - es Divide y Vencerás? $\rightarrow$ $\boxed{\text{Algoritmos de reducción}}$ $(l = 1)$
  - $T(n) = \Theta(logn)$

    $\leftrightarrow$ los datos ya están en memoria (Modelo de Computación)