Divide-and-Conquer II

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y T.I.

CI-2612: Algoritmos y Estructuras II

Plan

1. El problema del subarreglo máximo
2. Multiplicación de matrices
   - Método clásico
   - Solución Divide and Conquer
   - Solución con método de Strassen

El problema del subarreglo máximo

Planteamiento del problema

Dado un arreglo $A[1..n]$, se quiere determinar un subarreglo $S[i,j]$ contínuo de A, tal que la la suma de los elementos en el subrreglo $S[i,j] = A[i] + A[i+1] + \cdots + A[j]$ sea la máxima posible.

Ejemplo:

$A[1..8]$:

-2	-3	4	-1	-2	1	5	-3

La región resaltada es el subarreglo máximo
$S[3,7] = A[3] + A[4] + A[5] + A[6] + A[7] = 4 - 1 - 2 + 1 + 5 = 7.$

Una primera solución del subarreglo máximo

- **Por fuerza bruta:** Se pueban todas las combinaciones $\binom{n}{2}$ de pares $S[i,j]$. Si cada $S[i,j]$ se computa en $O(1)$, entonces esto es $O(n^2)$
- **Por transformación:**
  - Se computa los valores $S[i,j+1]$ de la los valores computados de $S[i,j]$. Esto es $O(1)$
  - Se computa todos los pares tal que $S[i,i] = A[i]$ y $S[i,j+1] = S[i,j] + A[j+1]$. Esto es $O(n^2)$

i	1	2	3	4	5	6	7	8
A[i]	-2	-3	4	-1	-2	1	5	-3
S[1, i]	-2	-5	-1	-2	-4	-3	2	-1
S[2, i]	-	-3	1	0	-2	-1	4	1
S[3, i]	-	-	4	3	1	2	7	4
S[4, i]	-	-	-	-1	-3	-2	3	0
S[5, i]	-	-	-	-	-2	-1	4	1
S[6, i]	-	-	-	-	-	1	6	3
S[7, i]	-	-	-	-	-	-	5	2
S[8, i]	-	-	-	-	-	-	-	-3

Una solución basada en Divide-and-conquer

Sea el arreglo $A[low..high]$ y $mid = \lceil (high + low)/2 \rceil$. Hay tres posibles lugares para el subarreglo máximo $A[i..j]$ del

- Totalmente en $A[low..mid]$, esto es $low \leq i \leq j \leq mid$.
- Totalmente en $A[mid+1..high]$, esto es $mid + 1 \leq i \leq j \leq high$.
- Cruzando el punto medio, esto es $low \leq i \leq mid < j \leq high$

Una solución basada en Divide-and-conquer

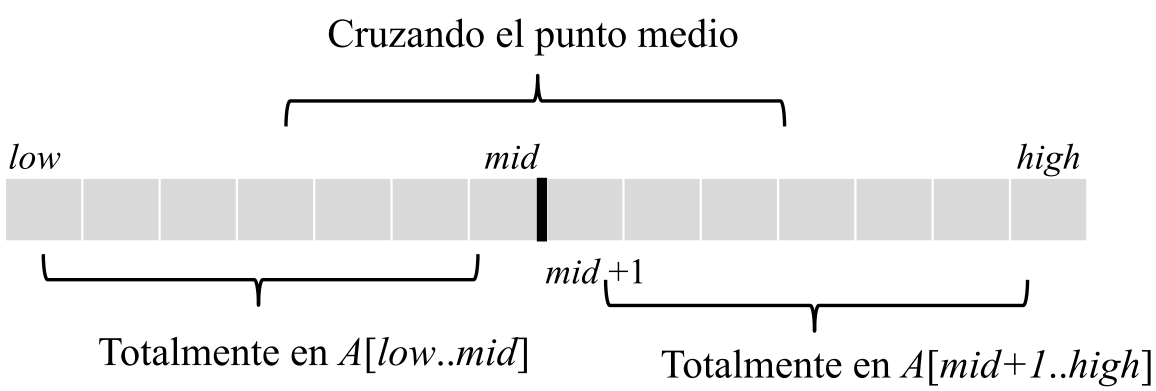

Figura: Posibles lugares del subarreglo máximo $A[i..j]$. Figura tomada de [1]

Una solución basada en Divide-and-conquer

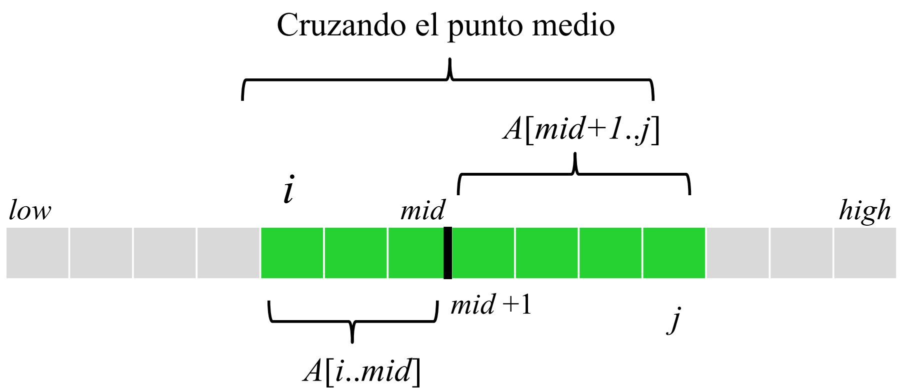

Figura: El subarreglo máximo $A[i..j]$ contiene dos subarreglos $A[i..mid]$ y $A[mid + 1..j]$. Figura tomada de [1]

Función FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)

inicio

  /* Encontrando un subarreglo máximo en $A[i..mid]$ */
  left-sum $= -\infty$;
  $sum = 0$;
  para $i = mid$ **a** low **hacer**
    $sum = sum + A[i]$;
    si $sum >$ *left-sum* **entonces**
      left-sum $= sum$;
      max-left $= i$;

  /* Encontrando un subarreglo máximo en $A[mid + 1..high]$ */
  right-sum $= -\infty$;
  $sum = 0$;
  para $j = mid + 1$ **a** $high$ **hacer**
    $sum = sum + A[j]$;
    si $sum >$ *right-sum* **entonces**
      right-sum $= sum$;
      max-right $= j$;

  /* Retorna los indices y la suma de los subarreglos máximos izq. y der. */
  retornar (max-left, max-right, left-sum + right-sum) ;

Ejemplo de FIND-MAX-CROSSING-SUBARRAY

- Subarreglo máximo en $A[i\,..\,mid]$, donde $mid = 4$

i	1	2	3	4	5	6	7	8
A[i]	-2	-3	4	-1	-2	1	5	-3

  - $S[4,4] = -1$
  - $S[3,4] = 3$. Mejor valor encotrado, left-sum $= 3$ y max-left$= 3$
  - $S[2,4] = 0$
  - $S[1,4] = -2$
- Subarreglo máximo en $A[mid + 1\,..\,high]$, donde $mid + 1 = 5$

i	1	2	3	4	5	6	7	8
A[i]	-2	-3	4	-1	-2	1	5	-3

  - $S[5,5] = -2$
  - $S[5,6] = -1$
  - $S[5,7] = 4$. Mejor valor encotrado, right-sum$= 4$ y max-right$= 7$
  - $S[5,8] = 1$
- El subarreglo máximo S[max-left,max-right] $=$ left-sum + right-sum, cruzando el punto medio es $S[3,7] = 7$

Función FIND-MAXIMUM-SUBARRAY(A, low, high)

inicio
  si *high == low* **entonces**
    retornar (low, high, A[low])
  en otro caso
    mid = $\lceil (low + high)/2 \rceil$;
    (left-low, left-high, left-sum) = FIND-MAXIMUM-SUBARRAY(A, low, mid) ;
    (right-low, right-high, right-sum) = FIND-MAXIMUM-SUBARRAY(A, mid + 1, high) ;
    (cross-low, cross-high, cross-sum) = FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high) ;
    si *left-sum* $\geq$ *right-sum and left-sum* $\geq$ *cross-sum* **entonces**
      retornar (left-low, left-high, left-sum)
    si no, si *right-sum* $\geq$ *left-sum and right-sum* $\geq$ *cross-sum* **entonces**
      retornar (right-low, right-high, right-sum)
    en otro caso
      retornar return (cross-low, cross-high, cross-sum);

Análisis del peor caso de encontrar un subarreglo máximo

- FIND-MAX-CROSSING-SUBARRAY es $\Theta(n)$
- Recurrencia FIND-MAXIMUM-SUBARRAY: $t(n) = 2t(n/2) + g(n)$
- Se tiene que $g(n) = \Theta(n)$
- Se tiene que $l = 2$, $b = 2$ y $k = 1$
- Como $l = b^k$ se aplica el segundo caso, esto es $t(n) = \Theta(n \log n)$
- Por lo tanto, $t(n) = \Theta(n \log n)$ en el peor caso.

Multiplicación de matrices

Planteamiento del problema

Sean A, B dos matrices cuadradas de tamaño $n \times n$, se quiere computar la matriz cuadrada C, tal que $C = A \times B$ de tamaño $n \times n$.

Una solución a la multiplicación de matrices

Método clásico

Dadas dos matrices A y B cuadradas tamaño $n \times n$, tal que los elementos de la matriz A son a_{ij}, y los elementos de la matriz B son b_{ij}. Si se multiplica las matrices A y B, se produce una matriz C, tal que $C = A \times B$ con tamaño $n \times n$ y con elementos $c_{ij} = \sum_{k=1}^{n} a_{ik} * b_{kj}$

Ejemplo del Método clásico

$$\begin{bmatrix} 8 & 3 & 0 & 1 \\ 1 & 2 & 3 & 4 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \times \begin{bmatrix} 5 & \cdot & 4 & \cdot \\ 4 & \cdot & 3 & \cdot \\ 3 & \cdot & 2 & \cdot \\ 1 & \cdot & 1 & \cdot \end{bmatrix} = \begin{bmatrix} 53 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 20 & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

- $c_{11} = \sum_{k=1}^{4} a_{1k} * b_{k1} = 8 * 5 + 3 * 4 + 0 * 3 + 1 * 1 = 53$
- $c_{23} = \sum_{k=1}^{4} a_{2k} * b_{k3} = 1 * 4 + 2 * 3 + 3 * 2 + 4 * 1 = 20$

Método clásico multiplicación de matrices

Algoritmo: Multiplicación de matrices cuadradas

Entrada: Matrices A$[1,\ldots,n][1,\dot{;}sn]$, B$[1,\ldots,n][1,\ldots,n]$ y el tamaño n de las matrices.

Salida : Matriz C$[1,\ldots,n][1,\ldots,n]$ con el resultado de $C = A \times B$.

inicio

```
C ← crear una matriz de ceros de tamaño n × n
para i ← 1 a n hacer
    para j ← 1 a n hacer
        para k ← 1 a n hacer
            C[i][j] ← C[i][j] + A[i][k] ∗ B[k][j]
retornar C
```


Análisis del Método clásico

- Tiene un tiempo de $\Theta(n^3)$.
- Aparte de la matriz C, no necesita almacenamiento extra.

Una solución basada en Divide and Conquer

Divide

Se divide la matrices de tamaño $n \times n$ en cuatro matrices de tamaño $n/2 \times n/2$. Se asume que n es potencia de 2, por lo que si $n \geq 2$, entonces $n/2$ es un entero.

Ejemplo: Partimos las matrices A y B en 4 $n/2 \times n/2$ matrices:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}.$$

Una solución basada en Divide and Conquer

Conquer

Se hacen recursivamente 8 multiplicaciones de 2 matrices de tamaño $n/2 \times n/2$, para obtener los resultados asociados a las 4 matrices de tamaño $n/2$ de la multiplicación $C = A \times B$.

Ejemplo: Se reescribe $C = A \times B$ como

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Se deben resolver las siguientes cuatro ecuaciones:

$$\begin{aligned} C_{11} &= A_{11} \times B_{11} + A_{12} \times B_{21} \\ C_{12} &= A_{11} \times B_{12} + A_{12} \times B_{22} \\ C_{21} &= A_{21} \times B_{11} + A_{22} \times B_{21} \\ C_{22} &= A_{21} \times B_{12} + A_{22} \times B_{22} \end{aligned}$$

Una solución basada en Divide and Conquer

Combine

Se obtiene la matriz resultante realizando 4 sumas de matrices de tamaño $n/2$.

Ejemplo: Una vez resueltas las cuatro ecuaciones:

$$\begin{aligned} C_{11} &= A_{11} \times B_{11} + A_{12} \times B_{21} \\ C_{12} &= A_{11} \times B_{12} + A_{12} \times B_{22} \\ C_{21} &= A_{21} \times B_{11} + A_{22} \times B_{21} \\ C_{22} &= A_{21} \times B_{12} + A_{22} \times B_{22} \end{aligned}$$

Se obtiene $C = A \times B$ sustituyendo los resultados de las ecuaciones:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} \times B_{11} + A_{12} \times B_{21} & A_{11} \times B_{12} + A_{12} \times B_{22} \\ A_{21} \times B_{11} + A_{22} \times B_{21} & A_{21} \times B_{12} + A_{22} \times B_{22} \end{bmatrix}$$

Función multMatrices(A, B, n)

si $n = 1$ **entonces** **retornar** C $\leftarrow$ A $\times$ B ;
Computar $A_{11}, A_{12}, A_{21}, A_{22}, B_{11}, B_{12}, B_{21}$ y B_{22} ;
$X_1 \leftarrow$ multMatrices(A_{11}, B_{11}, $n/2$) ;
$X_2 \leftarrow$ multMatrices(A_{12} , B_{21}, $n/2$) ;
$X_3 \leftarrow$ multMatrices(A_{11} , B_{12}, $n/2$) ;
$X_4 \leftarrow$ multMatrices(A_{12} , B_{22}, $n/2$) ;
$X_5 \leftarrow$ multMatrices(A_{21} , B_{11}, $n/2$) ;
$X_6 \leftarrow$ multMatrices(A_{22} , B_{21}, $n/2$) ;
$X_7 \leftarrow$ multMatrices(A_{21} , B_{12}, $n/2$) ;
$X_8 \leftarrow$ multMatrices(A_{22} , B_{22}, $n/2$) ;
$C_{11} \leftarrow X_1 + X_2$;
$C_{12} \leftarrow X_3 + X_4$;
$C_{21} \leftarrow X_5 + X_6$;
$C_{22} \leftarrow X_7 + X_8$;
$C \leftarrow \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$;
retornar C // Combinación de C_{11}, C_{12}, C_{21} y C_{22} ;

Análisis de la solución Divide and Conquer

- El paso Divide de las matrices A y B en 8 matrices $n/2 \times n/2$ toma $O(1)$.
- El paso Conquer lo componen las 8 llamadas recursivas.
- El paso Combine es $O(n^2)$. Esto es, la suma de dos matrices de tamaño $n/2 \times n/2$ (ej, $X_1 + X_2$) toma un tiempo $O(n^2)$.
- Sea $T(n)$ el número de operaciones de la función de `multMatrices`, tenemos que:

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 8T(n/2) + O(n^2) & \text{en caso contrario.} \end{cases}$$

- En consecuencia `multMatrices` es $O(n^3)$.

Algoritmo de Strassen

Divide

Se divide la matrices de tamaño $n \times n$ en cuatro matrices de tamaño $n/2 \times n/2$. Se asume que n es potencia de 2, por lo que si $n \geq 2$, entonces $n/2$ es un entero.

Ejemplo: Partimos las matrices A y B en 4 $n/2 \times n/2$ matrices:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}.$$

Algoritmo de Strassen

Conquer

Primero se computa 14 matrices de tamaño $n/2 \times n/2$ haciendo 10 operaciones de suma y resta. Luego se hacen recursivamente 7 multiplicaciones de 2 matrices de tamaño $n/2 \times n/2$, para obtener las 4 matrices de tamaño $n/2$ de la multiplicación $C = A \times B$.

$$\begin{aligned}\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} &= \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \\ &= \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_5 + P_1 - P_3 - P_7 \end{bmatrix}\end{aligned}$$

Se deben resolver las siguientes siete ecuaciones:

$$\begin{aligned} P_1 &= A_{11} \times (B_{12} - B_{22}) \\ P_2 &= (A_{11} + A_{12}) \times B_{22} \\ P_3 &= (A_{21} + A_{22}) \times B_{11} \\ P_4 &= A_{22} \times (B_{21} - B_{11}) \\ P_5 &= (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\ P_6 &= (A_{12} - A_{22}) \times (B_{21} + B_{22}) \\ P_7 &= (A_{11} - A_{21}) \times (B_{11} + B_{12}) \end{aligned}$$

- Las siete ecuaciones tienen 7 multiplicaciones, y 10 sumas y restas

Algoritmo de Strassen

Combine

Se obtiene los 4 términos de la matriz resultado con 8 sumas y restas de matrices $n/2 \times n/2$, con los 7 productos de matrices $P_1, P_2, \ldots, P_7$

Se computan los cuatro términos de la matriz resultante:

$$\begin{aligned} C_{11} &= P_5 + P_4 - P_2 + P_6 \\ C_{12} &= P_1 + P_2 \\ C_{21} &= P_3 + P_4 \\ C_{22} &= P_5 + P_1 - P_3 - P_7 \end{aligned}$$

Se obtiene $C = A \times B$ sustituyendo los resultados de las ecuaciones:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$
$$= \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_5 + P_1 - P_3 - P_7 \end{bmatrix}$$

Función strassen(A, B, n)

si $n = 1$ **entonces** **retornar** C $\leftarrow$ A $\times$ B ;
en otro caso
  Computar $A_{11}, A_{12}, A_{21}, A_{22}, B_{11}, B_{12}, B_{21}$ y B_{22} ;
  $P_1 \leftarrow$ strassen(A_{11}, $B_{12} - B_{22}$, $n/2$) ;
  $P_2 \leftarrow$ strassen($A_{11} + A_{12}$, B_{22}, $n/2$) ;
  $P_3 \leftarrow$ strassen($A_{21} + A_{22}$, B_{11}, $n/2$) ;
  $P_4 \leftarrow$ strassen(A_{22}, $B_{21} - B_{11}$, $n/2$) ;
  $P_5 \leftarrow$ strassen($A_{11} + A_{22}$, $B_{11} + B_{22}$, $n/2$) ;
  $P_6 \leftarrow$ strassen($A_{12} - A_{22}$, $B_{21} + B_{22}$, $n/2$) ;
  $P_7 \leftarrow$ strassen($A_{11} - A_{21}$, $B_{11} + B_{12}$, $n/2$) ;
  $C_{11} \leftarrow P_5 + P_4 - P_2 + P_6$;
  $C_{12} \leftarrow P_1 + P_2$;
  $C_{21} \leftarrow P_3 + P_4$;
  $C_{22} \leftarrow P_5 + P_1 - P_3 - P_7$;
  $C \leftarrow \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$;
  retornar C // Combinación de C_{11}, C_{12}, C_{21} y C_{22} ;

Análisis del Algoritmo de Strassen

- El paso Divide de las matrices A y B en 8 matrices $n/2 \times n/2$ toma $O(1)$.
- Antes de hacer la llamada recursivas se deben hacer computar 10 sumas y restas de matrices de tamaño $n/2 \times n/2$, esto se hace en $O(n^2)$.
- El paso Conquer lo componen las 7 llamadas recursivas.
- Para el paso Combine se llevan a cabo 8 sumas y restas de matrices de tamaño $n/2 \times n/2$, esto toma un tiempo $O(n^2)$.
- Sea $T(n)$ el número de operaciones de la función de `strassen`, tenemos que:

$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 7T(n/2) + O(n^2) & \text{en caso contrario.} \end{cases}$$

- En consecuencia `strassen` es $O(n^{\log_2 7}) = O(n^{2,81})$.

Problemas vistos

- El problema del subarreglo máximo
- Multiplicación de matrices.
  - Solución con método clásico
  - Solución Divide and Conquer
  - Solución con método de Strassen

Referencias

T. Cormen, C. Leirserson, R. Rivest, and C. Stein.
Introduction to Algorithms.
McGraw Hill, 3ra edition, 2009.