Tarea 1: Implementación de algoritmos Divide y Vencerás

Delgado Romero, Gustavo [1]   Torres Reategui, Joaquin[2]

7 de septiembre de 2025

Introducción

En esta presentación, exploraremos cuatro problemas clásicos que pueden ser resueltos eficientemente utilizando el paradigma de **divide y vencerás**. Estos problemas son:

- Moda de un vector
- Multiplicación de enteros grandes
- Multiplicación de matrices
- Subsecuencia de suma máxima

Moda de un vector

2. Moda de un vector
   - Pseudocódigo
   - Análisis de complejidad
   - Conclusión

Problema 1: Moda de un vector

Cálculo de la moda con Divide y Vencerás (Parte I)

```
function MODA_DIVIDE_VENCERAS(arr)
    if longitud(arr) = 1 then
        return { arr[0] : 1 }
    end if
    mid ← longitud(arr) / 2
    freq_izq ← Moda_Divide_Venceras(arr[0:mid])
    freq_der ← Moda_Divide_Venceras(arr[mid:])
    return Merge_Freq(freq_izq, freq_der)
end function
```

Problema 1: Moda de un vector

Cálculo de la moda con Divide y Vencerás (Parte II)

```
function MERGE_FREQ(freq_izq, freq_der)
    combinado ← copia(freq_izq)
    for all (k, v) en freq_der do
        combinado[k] ← combinado.get(k,0) + v
    end for
    return combinado
end function
```

Análisis de complejidad

Consideramos la recurrencia:

$$T(n) = \begin{cases} 2, & n = 1, \\ 2T\left(\frac{n}{2}\right) + cn, & n > 1, \end{cases}$$

donde $c > 0$ es una constante.

Expansión de la recurrencia

Aplicamos el método iterativo (expansión):

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

$$T(n) = 2\Big(2T\big(\tfrac{n}{2^2}\big) + c\,\tfrac{n}{2}\Big) + cn = 2^2\,T\big(\tfrac{n}{2^2}\big) + 2cn$$

$$T(n) = 2^i\,T\big(\tfrac{n}{2^i}\big) + i\,cn$$

Conclusión

Cuando $\frac{n}{2^k} = 1 \implies k = \log_2 n$:

$$T(n) = 2^{\log_2 n}\, T(1) + cn \log_2 n$$

$$T(n) = n \cdot 2 + cn \log_2 n$$

Conclusión:

$$T(n) = \Theta(n \log n).$$

3. Multiplicacion de enteros grandes
   - Pseudocódigo
   - Análisis de complejidad
   - Conclusión

Problema 2: Multiplicación de enteros grandes

Multiplicación de enteros grandes por divide y vencerás

function MULTIPLICACION(x, y)
    if $x < 10$ **or** $y < 10$ **then**
        return $x \times y$ ▷ Caso base: 1 dígito
    end if
    $n \leftarrow \max(\text{len}(x), \text{len}(y))$
    $m \leftarrow \lfloor n/2 \rfloor$
    Dividir x en x_1, x_0 tal que $x = x_1 \cdot 10^m + x_0$
    Dividir y en y_1, y_0 tal que $y = y_1 \cdot 10^m + y_0$
    $z_2 \leftarrow \text{Multiplicacion}(x_1, y_1)$
    $z_0 \leftarrow \text{Multiplicacion}(x_0, y_0)$
    $z_1 \leftarrow \text{Multiplicacion}(x_1 + x_0,\ y_1 + y_0) - z_2 - z_0$
    return $z_2 \cdot 10^{2m} + z_1 \cdot 10^m + z_0$
end function

Recurrencia

La recurrencia asociada al algoritmo es:

$$T(n) = \begin{cases} 4, & n = 1, \\ 3\,T\left(\dfrac{n}{2}\right) + c\,n, & n > 1, \end{cases}$$

donde $c > 0$ es una constante que representa el coste lineal de las operaciones de suma, resta y combinación.

Expansión de la recurrencia

Desarrollamos por expansión iterativa:

$$T(n) = 3T\left(\frac{n}{2}\right) + cn$$

$$T(n) = 3\left(3T\left(\frac{n}{2^2}\right) + c\frac{n}{2}\right) + cn = 3^2 T\left(\frac{n}{2^2}\right) + cn\left(1 + \frac{3}{2}\right).$$

Después de i expansiones:

$$T(n) = 3^i T\left(\frac{n}{2^i}\right) + cn\sum_{j=0}^{i-1}\left(\frac{3}{2}\right)^j.$$

Evaluación de la suma

La suma geométrica es:

$$\sum_{j=0}^{i-1} \left(\frac{3}{2}\right)^j = \frac{\left(\frac{3}{2}\right)^i - 1}{\frac{3}{2} - 1} = 2\Big(\left(\tfrac{3}{2}\right)^i - 1\Big).$$

Por tanto:

$$T(n) = 3^i\, T\left(\tfrac{n}{2^i}\right) + 2cn\Big(\left(\tfrac{3}{2}\right)^i - 1\Big).$$

Sustituyendo en la hoja

Tomamos $i = k$ tal que $\frac{n}{2^k} = 1 \implies k = \log_2 n$.

$$3^k = 3^{\log_2 n} = n^{\log_2 3}, \quad \left(\frac{3}{2}\right)^k = n^{\log_2(3/2)} = n^{\log_2 3 - 1}.$$

Con $T(1) = 4$:

$$\begin{aligned} T(n) &= 3^k T(1) + 2cn\Big(n^{\log_2 3 - 1} - 1\Big) \\ &= 4\, n^{\log_2 3} + 2c\, n^{\log_2 3} - 2c\, n. \end{aligned}$$

Conclusión

Dado que $\log_2 3 \approx 1{,}585$, el término dominante es $n^{\log_2 3}$.

$$\boxed{T(n) = \Theta\left(n^{\log_2 3}\right) \approx \Theta(n^{1{,}585})}$$

Multiplicacion de matrices

4. Multiplicacion de matrices
   - Pseudocódigo
   - Análisis de complejidad
   - Conclusión

Problema 3: Multiplicación de matrices (Parte I)

Multiplicación de matrices por divide y vencerás (MULTdc)

```
function MULTDC(A, B)
    n ← tamaño de A
    if n = 1 then
        return A_11 · B_11
    end if
    (A_11, A_12, A_21, A_22) ← Split_Matrix(A)
    (B_11, B_12, B_21, B_22) ← Split_Matrix(B)
```

Problema 3: Multiplicación de matrices (Parte II)

Multiplicación de matrices por divide y vencerás (cont.)

```
    C_11 ← Add_Matrix(MULTdc(A_11, B_11), MULTdc(A_12, B_21))
    C_12 ← Add_Matrix(MULTdc(A_11, B_12), MULTdc(A_12, B_22))
    C_21 ← Add_Matrix(MULTdc(A_21, B_11), MULTdc(A_22, B_21))
    C_22 ← Add_Matrix(MULTdc(A_21, B_12), MULTdc(A_22, B_22))
    C ← Combine_Matrix(C_11, C_12, C_21, C_22)
    return C
end function
```

Problema 3: Multiplicación de matrices

Multiplicación de matrices por divide y vencerás

```
function ADD_MATRIX(A, B)
    n ← tamaño de A
    for i ← 1 hasta n do
        for j ← 1 hasta n do
            C[i][j] ← A[i][j] + B[i][j]
        end for
    end for
    return C
end function
```

Problema 3: Multiplicacion de matrices

Dividir y combinar matrices

```
function SPLIT_MATRIX(A)
    n ← tamaño de A
    m ← n/2
    A11 ← submatriz superior izquierda
    A12 ← submatriz superior derecha
    A21 ← submatriz inferior izquierda
    A22 ← submatriz inferior derecha
    return (A11, A12, A21, A22)
end function
```

Problema 3: Multiplicacion de matrices

Dividir y combinar matrices

function COMBINE_MATRIX($C_{11}, C_{12}, C_{21}, C_{22}$)
    Formar matriz C uniendo los cuatro cuadrantes
    return C
end function

Recurrencia de la multiplicación de matrices

La recurrencia asociada al algoritmo es:

$$T(n) = \begin{cases} 3, & n = 1, \\ 8\,T\left(\frac{n}{2}\right) + cn^2, & n > 1, \end{cases}$$

donde $c > 0$ es una constante que representa el coste de sumar y combinar matrices.

Expansión iterativa

Expandimos la recurrencia paso a paso:

$$T(n) = 8\,T\left(\frac{n}{2}\right) + cn^2$$

$$T(n) = 8\Big(8\,T\big(\tfrac{n}{2^2}\big) + c\big(\tfrac{n}{2}\big)^2\Big) + cn^2 = 8^2\,T\big(\tfrac{n}{2^2}\big) + 8c\tfrac{n^2}{4} + cn^2$$

$$T(n) = 8^2\,T\big(\tfrac{n}{2^2}\big) + 3cn^2$$

En general, tras i expansiones:

$$T(n) = 8^i\,T\big(\tfrac{n}{2^i}\big) + cn^2 \sum_{j=0}^{i-1} 2^j$$

porque cada nivel de recursión agrega un coste proporcional a $n^2 \cdot 2^j$.

Evaluación en la hoja

Tomamos $i = k$ tal que $\frac{n}{2^k} = 1 \implies k = \log_2 n$.

$$8^k = 8^{\log_2 n} = n^{\log_2 8} = n^3$$

La suma geométrica:

$$\sum_{j=0}^{k-1} 2^j = \frac{2^k - 1}{2 - 1} = \frac{n^{\log_2 2} - 1}{1} = n - 1$$

Por tanto:

$$T(n) = n^3 T(1) + cn^2 \cdot (n - 1)$$

Conclusión

Dado que $T(1) = 3$:

$$T(n) = 3n^3 + c(n^3 - n^2)$$

El término dominante es n^3, proveniente de $n^{\log_2 8}$.

$$\boxed{T(n) = \Theta(n^3)}$$

5. Maxima subsecuencia
   - Pseudocódigo
   - Análisis de complejidad
   - conclusión

Problema 4: Maxima subsecuencia (Parte I)

Función MAX_SUM

```
function MAX_SUM(A, izquierda, medio, derecha)
    suma_izq ← −∞
    total ← 0
    for i ← medio hasta izquierda do
        total ← total + A[i]
        suma_izq ← máx(suma_izq, total)
    end for
    suma_der ← −∞
    total ← 0
    for i ← medio+1 hasta derecha do
        total ← total + A[i]
        suma_der ← máx(suma_der, total)
    end for
    return suma_izq + suma_der
end function
```

Problema 4: Maxima subsecuencia (Parte II)

Función MAX_SUBARRAY

```
function MAX_SUBARRAY(A, izquierda, derecha)
    if izquierda = derecha then
        return A[izquierda]
    end if
    medio ← ⌊(izquierda + derecha)/2⌋
    max_izq ← Max_Subarray(A, izquierda, medio)
    max_der ← Max_Subarray(A, medio+1, derecha)
    max_cruz ← Max_Sum(A, izquierda, medio, derecha)
    return máx(max_izq, max_der, max_cruz)
end function
```

Recurrencia

Consideramos la recurrencia

$$T(n) = \begin{cases} 2, & n = 1, \\ 2\,T\left(\dfrac{n}{2}\right) + c\,n, & n > 1, \end{cases}$$

donde $c > 0$ es una constante que representa el coste lineal de combinar resultados y calcular la suma cruzada.

Expansión iterativa

Desarrollamos la recurrencia por sustitución:

$$\begin{aligned} T(n) &= 2T\left(\tfrac{n}{2}\right) + cn \\ &= 2\Big(2T\big(\tfrac{n}{2^2}\big) + c\tfrac{n}{2}\Big) + cn = 2^2 T\big(\tfrac{n}{2^2}\big) + 2 \cdot c\tfrac{n}{2} + cn \\ &= 2^2 T\big(\tfrac{n}{2^2}\big) + cn + cn = 2^2 T\big(\tfrac{n}{2^2}\big) + 2\,cn. \end{aligned}$$

Tras i expansiones obtenemos la forma general

$$T(n) = 2^i T\big(\tfrac{n}{2^i}\big) + i\ cn.$$

(Observación: en cada nivel de la recursión aparece un término cn, por eso la suma es $i \cdot cn$.)

Evaluación en la hoja y conclusión

Tomamos $i = k$ tal que $\dfrac{n}{2^k} = 1$, es decir $k = \log_2 n$. Sustituyendo:

$$2^k = 2^{\log_2 n} = n, \qquad T(1) = 2.$$

Entonces

$$T(n) = 2^k T(1) + k\,cn = n \cdot 2 + cn \log_2 n.$$

Como término dominante queda $cn \log_2 n$, por lo que

$$\boxed{T(n) = \Theta(n \log n)}$$