Vinícius de Oliveira Guimarães 902431, Vitor Enzo Araújo 902123

LISTA 2 - PAA

① Dividir para conquistar é uma estratégia para projetar algoritmos realmente eficientes, utilizando conhecimentos matemáticos para a resolução de recorrências e recursão; Utilizando-a para simplificar um problema em partes menores e mais simples de resolver.

Estratégia:

1- Caso o meu problema seja pequeno o suficiente, então resolva-o diretamente

2- Caso contrário (Problema grande), Siga meus 3 passos:

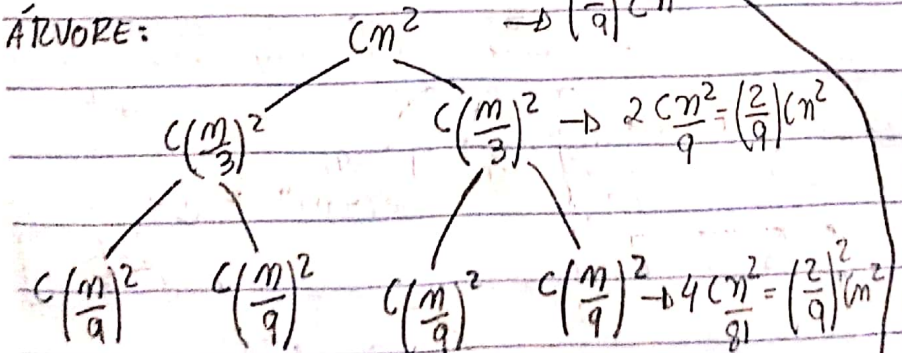
a) Divida o problema em partes menores (Subproblemas) e mais simples

b) Resolva esses subproblemas recursivamente

c) Combine o resultado dos subproblemas para encontrar a solução do problema original

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

ÁRVORE:



$\rightarrow \left(\frac{2}{9}\right)^0 cn^2$

$\rightarrow 2\frac{cn^2}{9} = \left(\frac{2}{9}\right)^1 cn^2$

$\rightarrow 4\frac{cn^2}{81} = \left(\frac{2}{9}\right)^2 cn^2$

Vai continuar até que $\frac{n}{3^k} = 1$, onde k é o nível

$n = 3^k \Rightarrow \log n = k \log 3$

$\Rightarrow k = \frac{\log n}{\log 3} \Rightarrow \boxed{k = \log_3 n}$

$T(n) = \left(\frac{2}{9}\right)^0 cn^2 + \left(\frac{2}{9}\right)^1 cn^2 + \left(\frac{2}{9}\right)^2 cn^2 + \ldots + \left(\frac{2}{9}\right)^k cn^2$

$= \sum_{j=0}^{k} \left(\frac{2}{9}\right)^j cn^2 = cn^2 \underbrace{\sum_{j=0}^{\infty} \left(\frac{2}{9}\right)^j}_{A}$

A soma infinita pois a razão é menor do que 1

$A = \frac{1^{\text{o}}\text{ termo}}{1 - \text{razão}} = \frac{1}{1 - \frac{2}{9}} = \frac{1}{\frac{9-2}{9}}$

$= \boxed{\frac{9}{7}} \Rightarrow \boxed{\frac{9}{7} cn^2} = \boxed{O(n^2)}$

③ Considerando dois números a e b como sendo:

$a = 123.456$, podemos reescrevê-los da seguinte forma: $a = 123.10^3 + 456$

$b = 654.321$ $b = 654.10^3 + 321$

Ou seja, representamos a e b assim:

$a = a_1 \cdot 10^{n/2} + a_2$, onde a_1 e a_2 são as metades inferiores e superiores de a respectivamente (o mesmo vale para b com relação à b_1 e b_2)

$b = b_1 \cdot 10^{n/2} + b_2$

Dessa forma, $(a \cdot b) = (a_1 \cdot 10^{n/2} + a_2) \cdot (b_1 \cdot 10^{n/2} + b_2)$

$= a_1 b_1 \cdot 10^n + a_1 b_2 \cdot 10^{n/2} + a_2 b_1 \cdot 10^{n/2} + a_2 b_2$

(Como $A = a_1 b_1$, $B = a_1 b_2$, $C = a_2 b_1$ e $D = a_2 b_2$) então:

$= \boxed{A \cdot 10^n + B \cdot 10^{n/2} + C \cdot 10^{n/2} + D}$

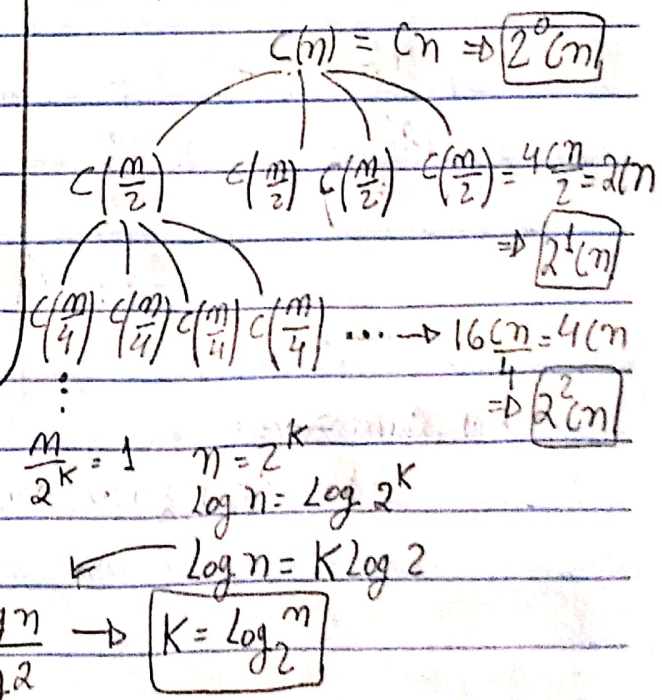
Com isso, pode-se construir o algoritmo:

```
multiply(a, b) {
    if(len(a) <= 1)
        return a.b
    Particionar a e b em { a = a1 . 10^(n/2) + a2 ; b = b1 . 10^(n/2) + b2 }   O(n)
    A = multiply(a1, b1)
    B = multiply(a1, b2)
    C = multiply(a2, b1)        } 4 problemas de tamanho n/2
    D = multiply(a2, b2)
    p = A.10^n + B.10^(n/2) + C.10^(n/2) + D
    return p
}
```

Assim, temos a seguinte recorrência:

$\boxed{T(n) = 4T\left(\frac{n}{2}\right) + O(n)}$

→ MONTAR ÁRVORE DE RECURSÃO



$\frac{n}{2^k} = 1$ $\quad n = 2^k$

$\log n = \log 2^k$

$\log n = K \log 2$

$K = \frac{\log n}{\log 2}$ → $\boxed{K = \log_2 n}$

$$t(n) = \sum_{j=0}^{K} 2^j cn = cn \cdot \sum_{j=0}^{K} 2^j$$ Como tem razão maior do que 1, então é uma PG Finita

$$S_{K+1} = \frac{a_1 \cdot (q^{K+1} - 1)}{q - 1} = \frac{1 \cdot (2^{K+1} - 1)}{2 - 1} = 2^{K+1} - 1 = (2^K \cdot 2) - 1$$

$$= (2^K \cdot 2) - 1 = (\underbrace{2^{\lg_2 n}}_{n} \cdot 2) - 1 = (n \cdot 2) - 1 = \boxed{2n - 1}$$

Assim, temos que

$$cn \cdot \sum_{j=0}^{K} 2^j = cn \cdot (2n - 1) = 2cn^2 - cn = \boxed{O(n^2)}$$

Como o algoritmo padrão de multiplicação de inteiros já é $O(n^2)$, não houve melhoria com esse novo algoritmo.

④ Na estratégia de dividir para conquistar tínhamos

$a = a_1 \cdot 10^{n/2} + a_2$, onde $(a \cdot b) = (a_1 \cdot 10^{n/2} + a_2)(b_1 \cdot 10^{n/2} + b_2) =$

$b = b_1 \cdot 10^{n/2} + b_2$ $= a_1 b_1 \cdot 10^n + a_1 \cdot 10^{n/2} \cdot b_2 + a_2 b_1 \cdot 10^{n/2} + a_2 b_2$

$= A \cdot 10^n + B \cdot 10^{n/2} + C \cdot 10^{n/2} + D$, onde $A = a_1 b_1$, $b = a_1 b_2$, $c = a_2 b_1$,

$D = a_2 b_2$

Agora no algoritmo de Karatsuba temos

$(B + D) = a_1 b_2 + a_2 b_1 = (a_1 + a_2) \cdot (b_1 + b_2) - a_1 b_1 - a_2 b_2$

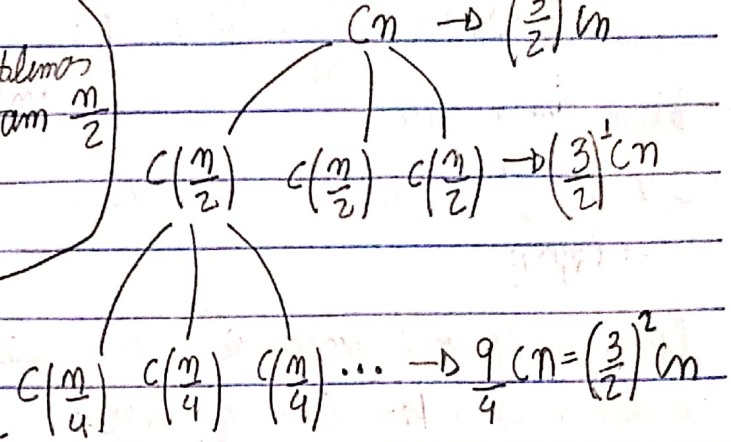
Ficando com o seguinte algoritmo: (PRÓXIMA PÁGINA)

```
Karatsuba (a, b) {
  if (len(a) ≤ 1)
    return a.b
  particiona a e b { a = a1 · 10^(n/2) + a2
                   { b = b1 · 10^(n/2) + b2   } O(n)
  A = Karatsuba (a1, b1)
  B = Karatsuba (a2, b2)
  C = Karatsuba (a1+a2, b1+b2)   } 3 problemas de tam n/2
  P = A · 10^n + (C - A - B) · 10^(n/2) + B
  return P
}
```

$(B+C) = a_1b_2 + a_2b_1 = (a_1+a_2)\cdot(b_1+b_2) - a_1b_1 - a_2b_2$

Assim, $\boxed{T(n) = 3C\left(\frac{n}{2}\right) + O(n)}$

A árvore de recursão fica:



$$T(n) = \sum_{j=0}^{k} \left(\frac{3}{2}\right)^j Cn = Cn \sum_{j=0}^{k} \left(\frac{3}{2}\right)^j$$

↓ RAZÃO

$\frac{n}{2^k} = 1 \rightarrow n = 2^k \rightarrow \log n = k \log 2 \rightarrow$

$\rightarrow k = \frac{\log n}{\log 2} \rightarrow \boxed{k = \log_2 n}$

Como razão é maior do que 1, então é soma de PG finita

$$S_{k+1} = \frac{a_1 \cdot \left(q^{k+1} - 1\right)}{q-1} = \frac{1\left(\left(\frac{3}{2}\right)^{k+1} - 1\right)}{\frac{3}{2} - 1} =$$

$$= \frac{\left(\frac{3}{2}\right)^k \cdot \frac{3}{2} - 1}{\frac{1}{2}} = \left[\left(\frac{3}{2}\right)^k \cdot \frac{3}{2} - 1\right] \cdot 2 =$$

$$= 3\left(\frac{3}{2}\right)^k - 2 = 3\left(\frac{3}{2}\right)^{\log_2 n} - 2 = 3 \cdot n^{\log_2 \left(\frac{3}{2}\right)} - 2$$

$$= 3 \cdot \left(n^{\log_2 3 - \log_2 2}\right) - 2 = 3 \cdot \left(n^{\log_2 3 - 1}\right) - 2$$

$$= 3n^{\log_2 3} \cdot n^{-1} - 2$$

Assim $Cn \cdot \sum_{j=0}^{k} \left(\frac{3}{2}\right)^j = Cn \cdot \left(3n^{\log_2 3} \cdot n^{-1} - 2\right)$

$$= Cn \cdot 3n^{\log_2 3} \cdot n^{-1} - Cn$$

$$= 3C\, n^{\log_2 3} - Cn = 3C \cdot n^{1{,}584} - Cn$$

Como $n^{1{,}584}$ é o termo dominante, então o algoritmo é $\boxed{O(n^{1{,}584})}$

Sendo $n^{1{,}584}$ menor do que n^2, então o algoritmo de Karatsuba apresenta ser mais eficiente do que o algoritmo padrão.

⑤ Dada duas matrizes A e B

$$A = \begin{matrix}\vec{a_1}\\ \vec{a_2}\\ \vec{a_3}\end{matrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$
$\vec{b_1}\ \ \vec{b_2}\ \ \vec{b_3}$

A multiplicação de uma com a outra será:

$$C = A.B = \begin{bmatrix} \vec{a_1}\vec{b_1} & \vec{a_1}\vec{b_2} & \vec{a_1}\vec{b_3} \\ \vec{a_2}\vec{b_1} & \vec{a_2}\vec{b_2} & \vec{a_2}\vec{b_3} \\ \vec{a_3}\vec{b_1} & \vec{a_3}\vec{b_2} & \vec{a_3}\vec{b_3} \end{bmatrix}$$
→ n^2 produtos de vetores de tamanho $n = O(n^3)$

APLICANDO DIVIDIR PARA CONQUISTAR

Assumir que $n = 2^m$ (potência de 2)

Subdividir matrizes A e B em 4 submatrizes $\frac{n}{2} \times \frac{n}{2}$

$$A = \begin{bmatrix} [A_{11}] & [A_{12}] \\ [A_{21}] & [A_{22}] \end{bmatrix} \qquad B = \begin{bmatrix} [B_{11}] & [B_{12}] \\ [B_{21}] & [B_{22}] \end{bmatrix}$$

Assim, o produto das matrizes será

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

Ou seja: 8 multiplicações e 4 Adições

Com isso, o algoritmo fica da seguinte forma:

```
mmDC (A, B, n) {
    if n == 1
        return A.B
    A11, A12, A21, A22 = split (A)
    B11, B12, B21, B22 = split (B)
    P = mmDC (A11, B11, n/2)
    Q = mmDC (A12, B21, n/2)
    R = mmDC (A11, B12, n/2)
    S = mmDC (A12, B22, n/2)
    X = mmDC (A21, B11, n/2)
    Y = mmDC (A22, B21, n/2)
```

operações de ordem $\frac{n}{2}$

CONTINUA NA PRÓXIMA PÁGINA

W = mmDC(A11, B12, n/2)
Z = mmDC(A22, B22, n/2)
} operações de ordem $\frac{n}{2}$

C11 = P + Q
C12 = R + S
C21 = X + Y
C22 = W + Z
} $O(n^2)$

C = monta(C11, C12, C21, C22)
return C
}

Assim, temos:

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

$Cn^2 \rightarrow (2)^0 Cn^2$

$C\left(\frac{n}{2}\right)^2$ $C\left(\frac{n}{2}\right)^2 \cdots \rightarrow \frac{8Cn^2}{4} = 2Cn^2 = (2)^1 Cn^2$

$C\left(\frac{n}{4}\right)^2$ $C\left(\frac{n}{4}\right)^2 \cdots \rightarrow \frac{64 \cdot Cn^2}{16} = 4Cn^2 = (2)^2 Cn^2$

⋮

$\frac{n}{2^k} = 1 \Rightarrow n = 2^k \Rightarrow \log n = k \log 2$

$\Rightarrow k = \frac{\log n}{\log 2} \Rightarrow \boxed{k = \log_2 n}$

Assim

$$T(n) = \sum_{j=0}^{k} (2)^j Cn^2 = Cn^2 \sum_{j=0}^{k} 2^j$$

Soma PG finita, pois razão é maior que 1

$$S_{k+1} = \frac{a_1 \cdot (q^{k+1} - 1)}{q - 1} = \frac{1 \cdot (2^{k+1} - 1)}{1} = 2^k \cdot 2 - 1 = \underbrace{2^{\log_2 n}}_{n} \cdot 2 - 1 = \boxed{2n - 1}$$

Portanto $Cn^2 \cdot \sum_{j=0}^{k} 2^j = Cn^2 \cdot (2n - 1) = 2Cn^3 - Cn^2 = \boxed{O(n^3)}$

Como o algoritmo da multiplicação de matrizes deu $O(n^3)$, então não houve melhora, já que o algoritmo padrão também é $O(n^3)$. Isso se deve às 8 operações de ordem $\frac{n}{2}$, que fazem a árvore de recursão crescer muito rapidamente na quantidade de "subopurações".

⑥ Dado o algoritmo de strassen:

```
strassen(A, B, n) {
    if n == 1
        return A · B
    A11, A12, A21, A22 = split(A)
    B11, B12, B21, B22 = split(B)
    P1 = strassen(A11, B12 - B22, n/2)
    P2 = strassen(A11 + A12, B22, n/2)
    P3 = strassen(A21 + A22, B11, n/2)
    P4 = strassen(A22, B21 - B11, n/2)
    P5 = strassen(A11 + A22, B11 + B22, n/2)
    P6 = strassen(A12 - A22, B21 + B22, n/2)
    P7 = strassen(A11 - A21, B11 + B12, n/2)
    C11 = P5 + P5 - P2 + P6
    C12 = P1 + P2
    C21 = P3 + P4
    C22 = P5 + P1 - P6 - P7
    C = merge(C11, C12, C21, C22)        O(n²)
    return C
}
```

Passamos de 8 para 7 chamadas recursivas

Assim $T(n) = 7T(\frac{n}{2}) + O(n^2)$

$Cn^2 \rightarrow (\frac{7}{4})^0 Cn^2$

$C(\frac{n}{2})^2 \quad C(\frac{n}{2})^2 \quad C(\frac{n}{2})^2 \quad \dots \rightarrow \frac{7}{4}Cn^2 = (\frac{7}{4})^1 Cn^2$

$C(\frac{n}{4})^2 \quad C(\frac{n}{4})^2 \dots \rightarrow \frac{49}{16}Cn^2 = (\frac{7}{4})^2 Cn^2$

$\vdots$

$\frac{n}{2^k} = 1 \rightarrow n = 2^k \rightarrow \log n = k \log 2$

$k = \frac{\log n}{\log 2} \rightarrow \boxed{k = \log_2 n}$

Porisso $T(n) = \sum_{j=0}^{k} (\frac{7}{4})^j Cn^2 = Cn^2 \sum_{j=0}^{k} (\frac{7}{4})^j$

Soma PG finita pois razão é maior do que 1

$S_{k+1} = \frac{a_1 \cdot (q^{k+1} - 1)}{q - 1} = \frac{1 \cdot \left(\left(\frac{7}{4}\right)^{k+1} - 1\right)}{\frac{7}{4} - 1}$

$= \frac{(\frac{7}{4})^k \cdot \frac{7}{4} - 1}{\frac{3}{4}} = \left[\left(\frac{7}{4}\right)^k \cdot \frac{7}{4} - 1\right] \cdot \frac{4}{3}$

$S_{k+1} = \frac{7}{3}(\frac{7}{4})^k - \frac{4}{3} = \frac{7}{3}(\frac{7}{4})^{\log_2 n} - \frac{4}{3} =$

$= \frac{7}{3} n^{\log_2 (\frac{7}{4})} - \frac{4}{3} = \frac{7}{3} \cdot n^{\log_2 7 - \log_2 4} - \frac{4}{3} = \frac{7}{3} n^{\log_2 7 - 2} - \frac{4}{3} =$

$\boxed{\frac{7}{3} n^{\log_2 7} \cdot n^{-2} - \frac{4}{3}}$

Assim

$Cn^2 \sum_{j=0}^{k} (\frac{7}{4})^j = Cn^2 \cdot \left(\frac{7}{3} n^{\log_2 7} \cdot n^{-2} - \frac{4}{3}\right) =$

$= C \cdot \frac{7}{3} n^{\log_2 7} - Cn^2 \cdot \frac{4}{3} = \boxed{C \cdot \frac{7}{3} n^{2,807} - Cn^2 - \frac{4}{3}}$

Como $n^{2,807}$ é o termo dominante, então o algoritmo é $\boxed{O(n^{2,807})}$

EXPLICAÇÃO Como o algoritmo de strassen apresenta 7 chamadas recursivas de ordem $\frac{n}{2}$, ao invés de 8 chamadas recursivas, a árvore de recursão cresce mais lentamente; sendo portanto, mais eficiente que o algoritmo tradicional, já que $O(n^{2,8})$ é menor que $O(n^3)$