Resolução – Lista 2 (Projeto e Análise de Algoritmos)

Novembro - 2022 / Leticia Bossatto Marchezi – 791003

Questão 1

Explique o que é a estratégia dividir para conquistar e descreva o seu funcionamento.

Resolução:

A estratégia dividir para conquistar é uma técnica usada para resolução de problemas complexos ou de alto custo e consiste em separar um grande problema em pequenos casos que podem ser resolvidos mais facilmente.

A execução da estratégia geralmente é constituída por 2 passos:

- Caso base Definição de um caso simples o suficiente que pode ser resolvido com força bruta

- Passos recursivos: Consiste na divisão de um problema em subproblemas menores, resolvendo-os e combinando com a solução do problema original. Eventualmente os subproblemas alcançarão o caso base e o problema maior será resolvido recursivamente.

□

Questão 2

Seja um algoritmo A com a recorrência a seguir:

$$T(n) = 2T(\frac{n}{3}) + cn^2 \tag{2.1}$$

Construa a árvore de recursão e calcule a complexidade do algoritmo em questão.

Resolução:

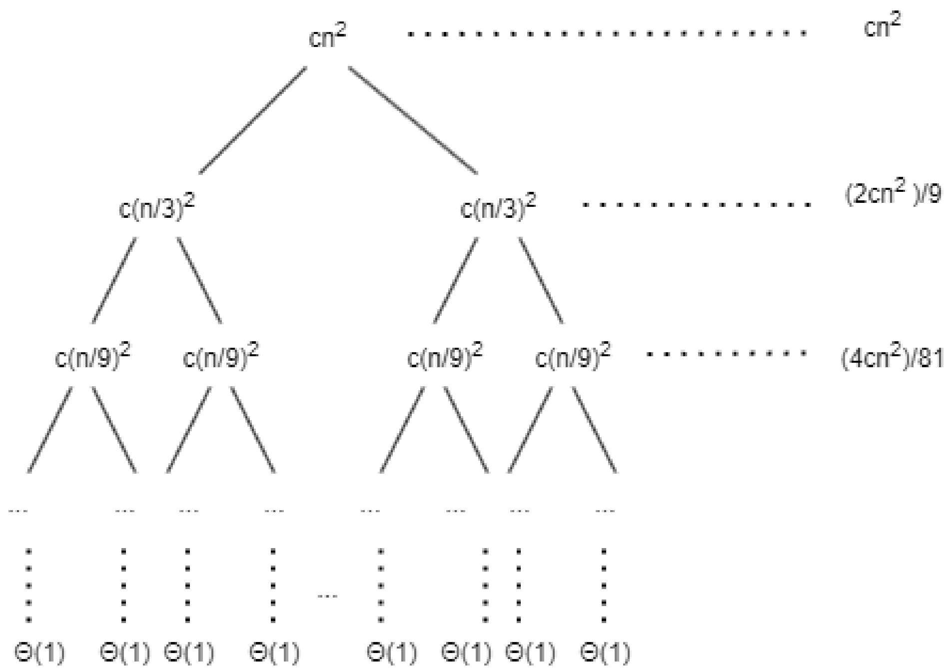

Figure 1: Árvore de recursão

A altura da árvore pode ser definida a partir do cálculo de qual nível terá custo constante (O(1)). Ou seja:

$$\frac{n}{3^k} = 1 \Rightarrow k = \log_3 n \tag{2.2}$$

Já a somatória dos custos em cada nível é expressa pela equação

$$T(n) = cn^2 + \frac{2}{9}cn^2 + (\frac{2}{9})^2cn^2 + ... + (\frac{2}{9})^{log_3 n}cn^2 \tag{2.3}$$

Que é correspondente ao seguinte somatória:

$$cn^2 = cn^2 \sum_{i=0}^{log_3 n} (\frac{2}{9})^i \tag{2.4}$$

Este somatório representa uma Progressão Geométrica de razão 2/9, ou seja, razão menor do que 1. Como buscamos um limitante superior, estudaremos o somatório para infinito:

$$T(n) < cn^2 \sum_{i=0}^{\infty} (\frac{2}{9})^i \tag{2.5}$$

Assim, a PG tem infinitos termos e sua soma pode ser calculada a partir da equação:

$$S_\infty = \frac{1}{1 - 2/9} = \frac{1}{7/9} = \frac{9}{7} \tag{2.6}$$

Ou seja:

$$T(n) \leq \frac{9}{7}cn^2 \Rightarrow O(n^2) \tag{2.7}$$

□

Questão 3

Explique como a estratégia dividir para conquistar pode ser aplicada no problema da multiplicação de inteiros. Explique porque a estratégia dividir para conquistar simples não consegue ser melhor que o algoritmo padrão.

Resolução:

A estratégia dividir para conquistar é utilizada no programa da multiplicação de inteiros com n dígitos ao particionar os números multiplicados em partes superiores(a_1 e b_1) e inferiores(a_2 e b_2).

Sendo A e B números com n dígitos(e n uma potência de 2, o cálculo da multiplicação é feito a partir da equação:

$$a * b = A * 10^{n/2} + (B + C) * 10^{n/2} + D \tag{3.1}$$

em que os termos A, B, C e D são:

$$A = a1 * b1$$

$$B = a2 * b2$$

$$C = a2 * b1$$

$$D = a2 * b2$$

Partindo de n^2 multiplicações para 4 problemas de tamanho $n/2$ mais o próprio particionamento que tem custo O(n).

$$T(n) = 4T(\frac{n}{2}) + O(n) \tag{3.2}$$

A cada nível da árvore de recursão até um nível k temos algo parecido com:

$$cn, \frac{4cn}{2}, \frac{16cn}{4} ... \frac{4^k cn}{2^k} = 2^k cn \tag{3.3}$$

A altura da árvore é definida pela altura do nó com custo constante(1). Ou seja:

$$\frac{n}{2^k} = 1 \Rightarrow \log_2 n \tag{3.4}$$

Assim, o custo do algoritmo é descrito pelo somatório

$$T(n) = cn \sum_{i=1}^{\log_2 n} 2^i = cn \sum_{i=1}^{\log_2 n} 2^{i+1} - 2^i \tag{3.5}$$

Formando uma soma telescópica e sendo resolvida pela subtração entre seu último termo e o primeiro:

$$T(n) = cn(2^{log_2 n+1} + 2^0) = cn(2n - 1) = 2cn^2 - n \tag{3.6}$$

Portanto temos que:

$$O(n^2) \tag{3.7}$$

Por isso, essa divisão de subproblemas não é o suficiente para alcançar um melhor custo do que a versão anterior, resultando também em $O(n^2)$.

□

Questão 4

Descreva o algoritmo de Karatsuba para a multiplicação de inteiros. Calcule sua complexidade e compare com o método tradicional.

Resolução:

O Algoritmo de Karatsuba para multiplicação de inteiros utiliza a técnica de dividir para conquistar com um aprimoramento.

Em vez de realizar 4 operações, as 2 multiplicações que interpolam as partes superiores e inferiores dos números podem ser reescritas de forma que não serão necessárias mais 4 multiplicações, e sim 3. Dessa forma, diminuimos em 1 operação o algoritmo. O cálculo de A e D permanecem inalterados.

$$A = a1 * b1$$

$$D = a2 * b2$$

Na equação do método de dividir e conquistar é possível resumir o termo B+C e eliminar uma multiplicação de tal forma:

$$B + C = (a_1 + a_2)(b_1 + b_2) - a_1 b_1 - a_2 b_2$$

Como foram aplicados no total 3 passos de custo $(n/2)$ e um particionamento O(n):

$$T(n) = 3T(\frac{n}{2}) + O(n) \tag{4.1}$$

A cada nível da árvore de recursão até um nível k temos algo parecido com:

$$cn, \frac{3cn}{2}, \frac{9cn}{4} ... \frac{3^k cn}{2^k} \tag{4.2}$$

Assim como o método de dividir e conquistar simples, a altura da árvore é $\log_2 n$, portanto:

$$T(n) = cn \sum_{i=1}^{\log_2 n} (\frac{3}{2})^i \tag{4.3}$$

Esta somatória representa uma Progressão Geométrica de razão > 1 e é finita, podendo ser calculada a partir da fórmula de soma de uma PG:

$$S_n = a_1 \frac{(q^n - 1)}{q - 1} \tag{4.4}$$

$$S_n = 1 \frac{(\frac{3}{2}^{\log_2 n + 1} - 1)}{\frac{3}{2} - 1} \tag{4.5}$$

$$S_n = \frac{(\frac{3}{2}^{\log_2 n + 1})}{\frac{3}{2}} = n^{\log_2 \frac{3}{2}} \tag{4.6}$$

Ou seja, a complexidade do método de Karatsuba é $O(n^{\log_2 \frac{3}{2}})$ ou $O(n^{1,58})$. Como a notação Big O representa o limitante superior, o pior caso do algoritmo de Karatsuba tem custo menor do que o método tradicional, de complexidade $O(n^2)$.

□

Questão 5

Explique como a estratégia dividir para conquistar pode ser aplicada no problema da multiplicação de matrizes. Explique porque a estratégia dividir para conquistar recursiva simples não consegue ser melhor que o algoritmo padrão

Resolução:

Na multiplicação de matrizes a estratégia de dividir para conquistar é executada a partir da divisão das duas matrizes de tamanho n em submatrizes de tamanho n/2 sucessivamente, até obter matrizes de um único elemento.

Assim, resultam 8 multiplicações de matrizes n/2 x n/2 e 4 adições de matrizes n/2 x n/2. Cada etapa tem custo n/2 e o particionamento tem complexidade $O(n^2)$, resultado na equação de recorrência:

$$T(n) = 8T(\frac{n}{2}) + O(n^2) \tag{5.1}$$

A cada nível da árvore de recursão até um nível k temos algo parecido com:

$$cn^2, \frac{8cn^2}{4}, \frac{64cn^2}{16}, ... \frac{8^k cn^2}{2^k} = 2^k cn^2 \tag{5.2}$$

Expressando em um somatório:

$$T(n) = cn^2 \sum_{i=1}^{\log_2 n} 2^i \tag{5.3}$$

Como demonstrado na questão 2, $\sum_{i=1}^{\log_2 n} 2^i$ resulta em $(2n-1)$, logo:

$$T(n) = cn^2(2n-1) = 2cn^3 - cn^2 \tag{5.4}$$

Assim, a complexidade do algoritmo de multiplicação de matrizes simples é $O(n^3)$, igualmente ao método tradicional, de mesma complexidade e não traz benefícios.

□

Questão 6

Descreva o algoritmo de Strassen para a multiplicação de matrizes. Calcule sua complexidade e compare com o método tradicional.

Resolução:

O algoritmo de Strassen para multiplicação de matrizes consiste na formação de novas matrizes executando apenas cálculos de soma e subtração, de complexidade $O(n^2)$.

$S_1 = B_{12} - B_{22}$,

$S_2 = A_{11} - A_{12}$,

$S_3 = A_{21} + A_{22}$,

...

$S_{10} = B_{11} + B_{12}$

Depois, os passos são subdivididos em 7 operações de custo n/2 envolvendo produto matricial, obtendo $P_1, P_2, P_3, ..., P_7$

$P_1 = A_{11}S_1 = A_{11}B_{12} - A_{11}B_{22}$

$P_2 = S_2B_{22} = A_{11}B_{22}A_{12}B_{22}$

...

$P_7 = S_9S_{10} = A_{11}B_{11} + A_{11}B_{12} - A_{21}B_{11} - A_{21}B_{12}$

Por fim, os elementos da matriz resultante são calculados apenas com operações de subtração e adição:

$C_{11} = P_5 + P_4 - P_2 + P_6 = A_{11}B_{11} + A_{12}B_{21}$

$C_{12} = P_1 + P_2 = A_{11}B_{12} + A_{12}B_{22}$

$C_{21} = P_3 + P_4 = A_{21}B_{11} + A_{22}B_{21}$

$C_{22} = P_5 + P_1 - P_3 - P_7 = A_{22}B_{22} + A_{21}B_{12}$

Para calcular a complexidade do algoritmo, estudaremos a função de recorrência:

$$T(n) = 7T(n/2) + O(n^2) \tag{6.1}$$

A árvore terá altura de:

$$\frac{n}{2^k} = 1 \Rightarrow k = \log_2 n \tag{6.2}$$

Os níveis da árvore de recursão serão:

$$c^2n, \frac{7cn^2}{4}, \frac{49cn^2}{16} ... \frac{7^k cn}{4^k} = (\frac{7}{4})^{\log_2 n} cn^2 \tag{6.3}$$

Dessa forma, a soma dos termos podem ser representados pelo somatório:

$$T(n) = cn^2 \sum_{i=0}^{\log_2 n} (\frac{7}{4})^i \tag{6.4}$$

Calculando pela fórmula da soma de PG:

$$S = (7/4)^{1+\log_2 n}/(7/4) = (\frac{7}{4})^{\log_2 n} = n^{log_2 7} n^{-2} \tag{6.5}$$

Assim:

$$T(n) = cn^2 n^{\log_2 7} n^{-2} = cn^{\log_2 7} \tag{6.6}$$

$\log_2 7$ é aproximadamente 2.8, ou seja, $O^(n^{2,8})$, então o algoritmo de Strassen possui um limitante superior melhor do que o algoritmo tradicional, e melhor desempenho no pior caso. $\square$