EXERCÍCIO COMPARAÇÃO DE ALGORITMOS

Data: 02 de outubro de 2017

Matheus Francisco Batista Machado Matrícula : 14202492

Disciplina: Projeto e Análise de Algoritmos de Controle

Exercicio 4: Comparação dos algoritmos de multiplicação de números grandes

A multiplicação de números inteiros vista no ensino básico é simples e usualmente eficiente para inteiros relativamente pequenos, porém para números inteiros de n algarismos, porém dado um numero de m e n dígitos o tempo que leva para multiplicar é $m * n$. No caso m = n então temos um tempo proporcional n^2

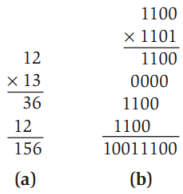

Figure 1: Exemplo de multiplicação

Então tem-se o objetivo de fazer a comparação de tempo para algoritmos de multiplicação de números grandes que usam a ideia de divisão e conquista, com o algoritmo básico. Então a ideia do algoritmo de divisão e conquista implementado é dado dois números x e y, podem ser representados como:

$$X = 2^{n/2} * Xe + Xd$$
$$Y = 2^{n/2} * Ye + Yd$$

então XY:

$$X * Y = (2^{n/2} * Xe + Xd) * (2^{n/2} * Ye + Yd)$$
$$X * Y = 2^n * (XeYe) + 2^{n/2} * [((Xe + Xe) * (Ye + Yd)) - XeYe - XdYd] + XdYd$$

onde podemos chamar $P_1 = XeYe, P_2 = XdYd, P_3 = [(Xe+Xd)*(Ye+Yd)]$, sendo assim o algoritmo de Karatsuba pode ser escrito

Dado o algoritmo pode-se fazer uma analise do tempo de execução onde $T(n) =$ consumo de tempo do algoritmo para multiplicar dois inteiros com n algarismos.

```
1:  procedure KARATSUBA(X,Y,n)
2:      if n < 3 then return XY
3:      else
4:          q ← ⌈n/2⌉
5:
6:          A ← X[q + 1..n]
7:
8:          B ← X[1..q]
9:
10:         C ← Y[q + 1..n]
11:
12:         D ← Y[1..q]
13:
14:         E ← Karatsuba(A, C, ⌊n/2⌋)
15:
16:         F ← Karatsuba(B, D, ⌊⌈n⌉2⌋)
17:
18:         G ← Karatsuba(A + B, C + D, ⌈n/2⌉ + 1)
19:
20:         H ← G − F − E
21:
22:         R ← E ∗ 2^n + H ∗ 2^⌈n/2⌉ + F
23: return R
```

- Linha 2 $\Theta(1)$
- Linha 4 $\Theta(1)$
- Linha 6 até 8 $\Theta(n)$
- Linha 10 até 12$\Theta(n)$
- Linha 14 $T(\lfloor \frac{n}{2} \rfloor)$
- Linha 16 $T(\lceil \frac{n}{2} \rceil)$
- Linha 18 $T(\lceil \frac{n}{2} \rceil + 1)$
- Linha 20 $\Theta(n)$
- Linha 22 $\Theta(n)$

Assim temos $total = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + T(\lceil \frac{n}{2} \rceil + 1) + \Theta(n)$ Utilizou-se o teorema mestre para mostrar que o algoritmo de karatsuba é $\Theta(n^{\log 3})$ então foi implementado e comparado o tempo de execução dos algoritmos básico e de karatsua mostrado no gráfico e tabela abaixo.

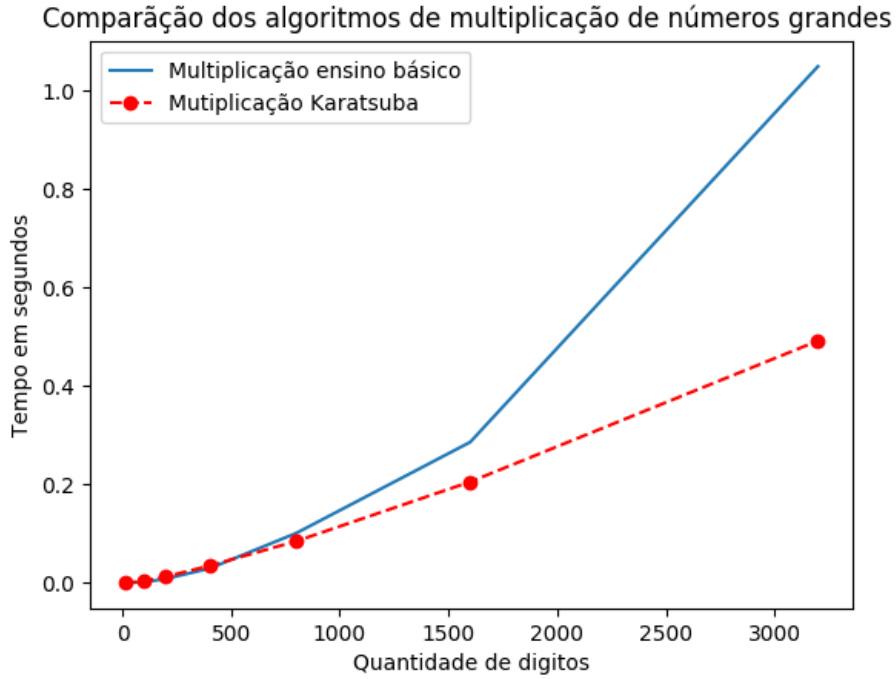

Figure 2: Comparação dos algoritmos de multiplicação de números grandes

Inicialmente os algoritmos não aparentava muita diferença porém com o aumento na quantidade de dígitos os algoritmos começaram a ter uma diferença significativa, mostrado também na tabela abaixo.

Table 1: Tempo em segundo dos algoritmos de multiplicação de números grandes

n	**Ensino Fund.**	**Karatsuba**
10	0.0001	0.00014
100	0.00208	0.004
200	0.00859	0.013
400	0.029519	0.035
800	0.101347	0.085
1600	0.286142	0.2055
3200	1.048996	0.491856

Exercicio 4: Comparação dos algoritmos de multiplicação de matrizes

O produto de duas matrizes nxn X e Y é uma matriz Z = $X * Y$

$$Z_{ij} = \sum_{k=1}^{n} X_{ik} * Y_{kj}$$

A fórmula anterior implica um algoritmo $O(n^3)$ para a multiplicação da matriz: há entradas n^2 a serem computadas, e cada uma tem o tempo $O(n)$. Por um bom tempo, isso foi amplamente acreditado para ser o melhor tempo de execução possível. Porém em 1969 o matemático alemão Volker Strassen anunciou uma eficiência significativamente mais eficiente que o algoritmo de multiplicação básica, um algoritmo baseado na divisão e conquista.

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

Com isso podemos visualizar uma estrategia de divisão e conquista vamos calcular oito produtos do tamanho-n/2 recursivamente AE, BG, AF, BH, CE, DG, CF, DH e, em seguida, fazer alguns adições de

$O(n^2)$ de tempo. Logo o tempo de execução total é descrito pela formula de recorrência.

$$T(n) = 8T(n/2) + O(n^2).$$

Isto resulta em $O(n^3)$ o mesmo que para o algoritmo padrão. Porém Strassen conseguiu dividir utilizando álgebra então XY pode ser dividido em sete $n/2 * n/2$ sub problemas.

$$XY = \begin{bmatrix} P5 + P4 - P2 + P6 & P1 + P2 \\ P3 + P4 & P1 + P5 - P3 - P7 \end{bmatrix}$$

onde

$$\begin{aligned} P1 &= A(F - H) \\ P2 &= (A + B)H \\ P3 &= (C + D)E \\ P4 &= D(G - E) \\ P5 &= (A + D)(E + H) \\ P6 &= (B - D)(G + H) \\ P7 &= (A - C)(E + F) \end{aligned}$$

Reduzindo a formula de recorrência para

$$T(n) = 7T(n/2) + O(n^2)$$

onde o tempo de execução $O(n^{log_2(7)}) = O(n^{2.81})$.

Dado isso foi implementado ambos os algoritmos e comparado o tempo de execução do código para alguns tamanho de matrizes mostrado na tabela abaixo.

Table 2: Comparação dos tempos de execução

Tamanho	**Tempo da Multiplicação Básica**	**Tempo da Multiplicação de Strassen**
10	0.000018 segundos	0.00016 segundos
100	0.007747 segundos	0.006817 segundos
200	0.058816 segundos	0.037906 segundos
400	0.311893 segundos	0.175083 segundos
800	2.01029 segundos	1.231334 segundos
1600	15.703071 segundos	8.628541 segundos
3200	134.070419 segundos	61.003788 segundos

Em seguida plotado um gráfico mostrando os dados da tabela

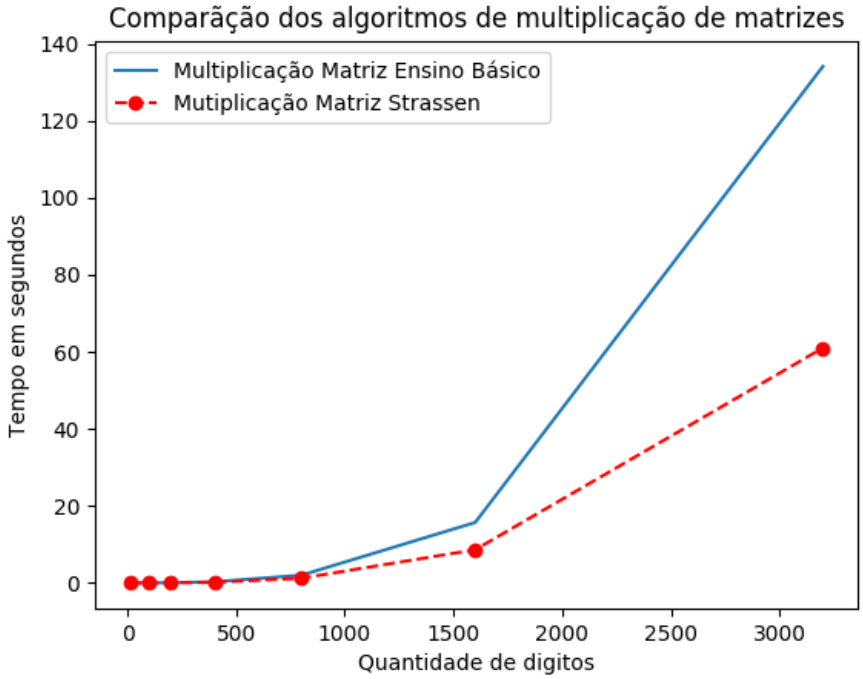

Figure 3: Comparação dos algoritmos de multiplicação de matrizes