Análise Comparativa: Algoritmo de Karatsuba vs. Multiplicação Ingênua

Roney Nogueira de Sousa[1], **Leonardo Adriano Vasconcelos de Oliveira**[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará
Av. Treze de Maio, 2081 - Benfica, Fortaleza - CE

nogueiraroney453@gmail.com

Abstract. *In this work, we perform a comparative analysis between the naive multiplication algorithm and the Karatsuba algorithm, highlighting their differences in terms of complexity and computational efficiency. The naive multiplication algorithm, with complexity $O(n^2)$, uses a direct approach to calculate the product between two large numbers. The Karatsuba algorithm, based on divide and conquer, reduces the number of multiplication operations, with a complexity of $O(n^{\log_2 3})$. Both approaches were implemented and evaluated for performance, demonstrating that the Karatsuba algorithm is significantly more efficient for large numbers.*

Resumo. *Neste trabalho, realizamos uma análise comparativa entre o algoritmo de multiplicação ingênua e o algoritmo de Karatsuba, destacando suas diferenças em termos de complexidade e eficiência computacional. A multiplicação ingênua, com complexidade $O(n^2)$, utiliza uma abordagem direta para calcular o produto entre dois números grandes. Já o algoritmo de Karatsuba, baseado em divisão e conquista, reduz o número de operações de multiplicação, com uma complexidade de $O(n^{\log_2 3})$. Ambas as abordagens foram implementadas e avaliadas quanto ao desempenho, demonstrando que o algoritmo de Karatsuba é significativamente mais eficiente para números grandes.*

1. Introdução

Ambos os algoritmos foram implementados em Python para análise comparativa. As implementações foram projetadas para lidar com inteiros de precisão arbitrária. A abordagem ingênua utiliza loops aninhados, enquanto a implementação de Karatsuba emprega chamadas recursivas.

A multiplicação de inteiros grandes é uma operação fundamental na ciência da computação e na matemática. Tradicionalmente, esse problema é resolvido usando o método "ingênuo", que possui uma complexidade de tempo de $O(n^2)$. No entanto, algoritmos mais eficientes, como o algoritmo de Karatsuba, foram desenvolvidos para reduzir o custo computacional. Neste relatório, comparamos a implementação e o desempenho dessas duas abordagens.

2. Descrição dos Algoritmos

2.1. Algoritmo de Multiplicação Ingênua

O algoritmo de multiplicação ingênua envolve uma aplicação direta da definição de multiplicação. Dados dois inteiros de n dígitos, o algoritmo calcula o produto somando os produtos parciais para cada par de dígitos:

- Complexidade de Tempo: $O(n^2)$.
- Complexidade de Espaço: $O(1)$.

2.1.1. Pseudocódigo Ingênua

Algorithm 1: Multiplicação Ingênua

```
Input: Dois vetores vec1 e vec2 de tamanho n.
Output: Vetor resultante da multiplicação ingênua.
n ← tamanho de vec1;
result ← vetor de zeros de tamanho 2n;
for i ← 0 to n − 1 do
    for j ← 0 to n − 1 do
        result[i + j] ← result[i + j] + (vec1[i] ×
         vec2[j]);
    end
end
return result;
```

2.2. Algoritmo de Karatsuba

O algoritmo de Karatsuba, introduzido por Anatolii Alexeevitch Karatsuba em 1960, é um método de divisão e conquista que reduz o número de multiplicações recursivas. Para dois inteiros de n dígitos x e y, ele funciona da seguinte forma:

1. Divide x e y em duas metades de aproximadamente $n/2$ dígitos cada.
2. Calcula três produtos: $P_1 = A \times C$, $P_2 = B \times D$ e $P_3 = (A + B) \times (C + D)$.
3. Deriva o produto final usando:

$$x \times y = P_1 \times 10^n + (P_3 - P_1 - P_2) \times 10^{n/2} + P_2.$$

4. Complexidade de Tempo: $O(n^{\log_2 3})$.
5. Complexidade de Espaço: Maior que a do algoritmo ingênuo devido às chamadas recursivas.

2.2.1. Pseudocódigo de Karatsuba

Algorithm 2: Algoritmo de Karatsuba

Input: Dois vetores `vec1` e `vec2` de tamanho n.
Output: Vetor resultante da multiplicação usando Karatsuba.

```
n ← tamanho de vec1;
if n = 1 then
    return [vec1[0] × vec2[0]];
end
mid ← ⌊n/2⌋;
Divida vec1 em low1 e high1;
Divida vec2 em low2 e high2;
z0 ← KARATSUBA(low1, low2);
z1 ← KARATSUBA(low1 + high1, low2 + high2);
z2 ← KARATSUBA(high1, high2);
result ← vetor de zeros de tamanho 2n;
for i ← 0 to |z0| − 1 do
    result[i] ← result[i] + z0[i];
end
for i ← 0 to |z1| − 1 do
    result[mid + i] ← result[mid + i] + z1[i] − z0[i] −
     z2[i];
end
for i ← 0 to |z2| − 1 do
    result[2 * mid + i] ← result[2 * mid + i] + z2[i];
end
return result;
```

3. Resultados

Para visualizar o crescimento da complexidade computacional, foi plotado o comportamento assintótico dos dois algoritmos. A Figura 1 ilustra as diferenças no tempo de execução em função do tamanho da entrada n.

4. Conclusão

O algoritmo de Karatsuba demonstra melhorias significativas de desempenho em relação à abordagem ingênua para inteiros grandes. Embora o algoritmo ingênuo seja mais simples de implementar, sua complexidade de tempo $O(n^2)$ limita sua praticidade para problemas em larga escala. Por outro lado, a estratégia de divisão e conquista de Karatsuba oferece uma redução substancial no custo computacional, tornando-o adequado para aplicações que envolvem inteiros grandes.

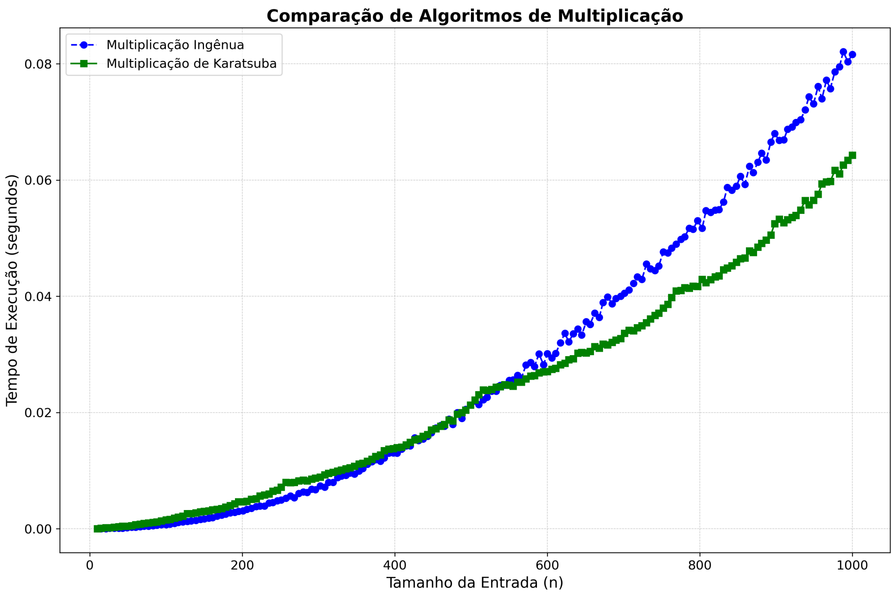

Figure 1. Crescimento assintótico da multiplicação ingênua ($O(n^2)$) e do algoritmo de Karatsuba ($O(n^{\log_2 3})$).