

Disciplina: Analise e Síntese de Algoritmos

Professor: Dr. Luís Antônio Brasil Kowada

Aluno: Flávio Miranda de Farias

**RELATÓRIO**

Comparativo dos métodos de multiplicação utilizando a técnica Karatsuba e Padrão.

A multiplicação de Karatsuba (Karatsuba, 1995) é uma técnica recursiva para multiplicar vetores de números inteiros, muito diferente da técnica tradicional da escola primária. A multiplicação de Karatsuba é frequentemente ensinada nas aulas de ciência da computação e analisada teoricamente. Para números muito grandes, pelo menos teoricamente, entrega o melhor resultado no geral.

No entanto, para ter como método avaliativo, é preciso fazer algumas comparações empíricas. Para isto, foi escrito uma implementação usando código em C++ e comparando a uma técnica apelidada de “Multiplicação Padrão”, no qual se assemelha em parte a técnica de Karatsuba, porem, dividindo as tarefas em mais partes.

Posteriormente será detalhado em mais detalhes o teste comparativo.

**MOTIVAÇÃO**

Esta pesquisa foi estimulada em sala de aula na disciplina de Analise e Síntese de Algoritmos da Pós-graduação em Computação da UFF em 2019-2 pelo professor Dr. Luís Antônio Brasil Kowada. Que instigou aos alunos praticarem uma análise de métodos de multiplicação de Strings numéricas, tendo como base comparativa o algoritmo de Karatsuba, como concorrente, foi sugerido outro algoritmo recursivo. A seguir a atividade proposta na integra.

“Tarefa 1

Multiplicação de dois números em uma base b qualquer com uma quantidade arbitrária de dígitos.

* Multiplicação padrão
* Multiplicação Karatsuba

A implementação deve ser recursiva (para ambas multiplicações).

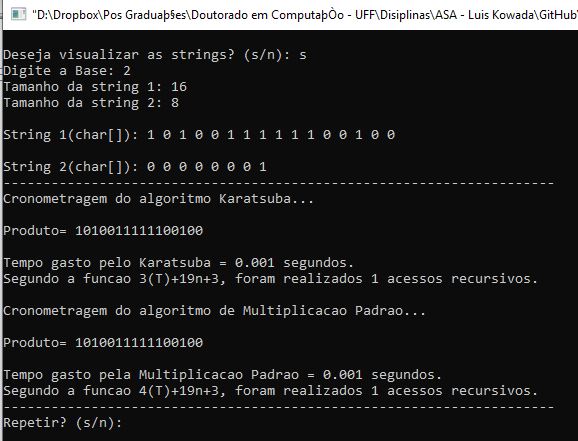
O programa, contendo ambas funções, pode ser feito em Python, C, C++, Java, Pascal ou Fortran.

Acompanhando o programa (fonte e executável), deve haver um relatório, descrevendo o programa e fazendo uma simulação para diversos tamanhos de entrada diferente numa escala linear (p. ex. com crescimento de 10 em 10 dígitos) e numa escala exponencial (p. ex. com quantidade de dígitos variando de 1, 2, 4, 8, 16, 32, ...). A comparação deve ser feita quanto à quantidade de operações multiplicações elementares e quanto ao tempo de execução (em segundos).

A entrega deve ser feita através de um arquivo compactado (.zip) incluindo o programa e o relatório.”

**IMPLEMENTAÇÃO**

Para implementação foi decidido usar a linguagem C++, em ambos os métodos (Karatsuba e Padrão), é solicitado na assinatura da função duas entradas de Strings (s) com valores inteiros de tamanho n, sendo n>=0 e s1 e s2 podem ser de tamanhos diferentes como pode ser visto na imagem a seguir, mais um inteiro representando a base de tamanho t>0. Como saída, retorna uma String s3 também de inteiros representando o produto da multiplicação s3=s1\*s2.



No método main do programa, foi implementada algumas decisões do usuário para tornar o programa mais prático, sendo elas:

* Escolha de base;
* Escolha do tamanho da primeira e segunda String;
* Escolha se deseja ver impresso as Strings de entrada e saída na tela, para ambos os métodos;
* Opção de repetir os testes.

Ainda no método main, com a decisão do tamanho das Strings, são gerados valores inteiros positivos de forma randômica para preencher as entradas.

Na saída, é cronometrado e exibido o tempo de duração de cada método de multiplicação, além do número de acessos recursivos a cada método.

O método Karatsuba, baseada no algoritmo de mesmo nome, nesta implementação possui o custo de 3(T)+19n+3, por possuir 3 chamadas recursivas internamente e 19 operações de custo assintótico n, mais a constante. Enquanto o método Padrão possui o custo de 4(T)+19n+3, muito semelhante ao método anterior, se diferencia basicamente por possuir 4 chamadas recursivas.

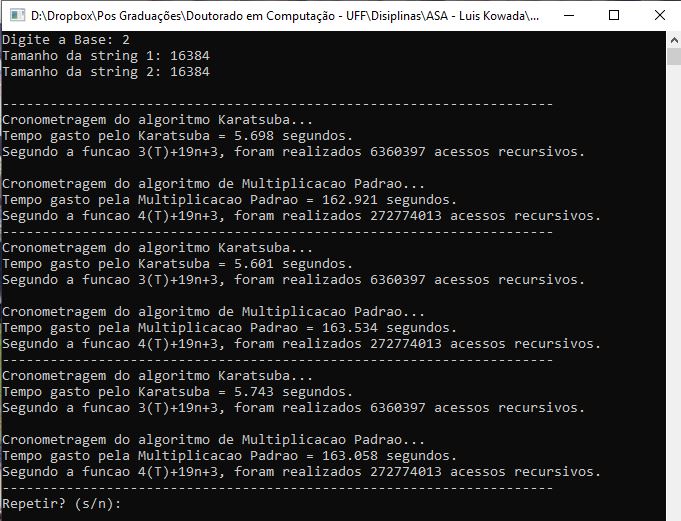
Como metodologia comparativa, foi inserido um contador que informa ao final o número de recursões realizadas, além de comentários dentro do código explicando o custo assintótico de cada operação.

**RESULTADOS**

Os testes foram realizados no sistema operacional Windows 10 com processador i5-7200 com 2,71GHz e 8 Gb RAM e SSD.

Para os testes foi decidido trabalhar nas bases numéricas populares, 2,8,10,16, mesmo sendo possível trabalhar em qualquer valor de tamanho t. Em relação as entradas de String s de tamanho n, foi decidido trabalhar com s1=s2 para todos os testes, sendo s nos tamanhos exponenciais de base 2, sendo n=2^0 até 2^14.

Todos os testes foram realizados 3 vezes consecutivas para cada base t e cada tamanho de par de String n, realizando 180 execuções de cada método ou 360 no total. Foi optado por repetir 3 vezes com objetivo de usarmos o menor tempo, pois o tempo poderia sofrer alterações devido à concorrência com sistemas concorrentes na pilha do sistema operacional, o número de acessos recursivos não alterava pois o número era o mesmo para os 3 testes, como pode ser visto na imagem capturada a seguir.



Todo este projeto como código fonte, tabulação de dados, entre outras, então disponíveis na pasta do GitHub (Farias, 2019). A seguir estão os dados dos testes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Base 2 | | | | | |
| Acessos | | | **Tempo** | | |
| Tamanho | **Karatsuba** | **Padrão** | **Tamanho** | **Karatsuba** | **Padrão** |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 2 | 1 | 1 | 2 | 0 | 0 |
| 4 | 10 | 13 | 4 | 0 | 0 |
| 8 | 34 | 65 | 8 | 0 | 0 |
| 16 | 112 | 285 | 16 | 0 | 0 |
| 32 | 226 | 753 | 32 | 0 | 0 |
| 64 | 1075 | 5093 | 64 | 0 | 0,001 |
| 128 | 2740 | 16101 | 128 | 0,001 | 0,001 |
| 256 | 8479 | 65597 | 256 | 0,001 | 0,021 |
| 512 | 26101 | 261153 | 512 | 0,013 | 0,164 |
| 1024 | 78877 | 1074453 | 1024 | 0,052 | 0,574 |
| 2048 | 236581 | 4304597 | 2048 | 0,2 | 2,689 |
| 4096 | 703807 | 17091517 | 4096 | 0,648 | 10,528 |
| 8192 | 2124637 | 68265201 | 8192 | 1,866 | 40,682 |
| 16384 | 6360397 | 272774013 | 16384 | 5,601 | 162,921 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Base 8 | | | | | |
| Acessos | | | **Tempo** | | |
| Tamanho | **Karatsuba** | **Padrão** | **Tamanho** | **Karatsuba** | **Padrão** |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 2 | 4 | 5 | 2 | 0 | 0 |
| 4 | 22 | 33 | 4 | 0 | 0 |
| 8 | 58 | 125 | 8 | 0 | 0 |
| 16 | 154 | 389 | 16 | 0 | 0 |
| 32 | 526 | 2053 | 32 | 0 | 0,001 |
| 64 | 1591 | 7761 | 64 | 0,001 | 0,001 |
| 128 | 4792 | 31629 | 128 | 0,002 | 0,014 |
| 256 | 14659 | 122325 | 256 | 0,003 | 0,061 |
| 512 | 43816 | 497197 | 512 | 0,024 | 0,31 |
| 1024 | 132685 | 2002765 | 1024 | 0,104 | 1,307 |
| 2048 | 396490 | 7901989 | 2048 | 0,323 | 4,99 |
| 4096 | 1189975 | 31809285 | 4096 | 1,055 | 20,407 |
| 8192 | 3578794 | 127218509 | 8192 | 3,151 | 79,652 |
| 16384 | 10727824 | 509246373 | 16384 | 7,231 | 257,69 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Base 10 | | | | | |
| Acessos | | | **Tempo** | | | |
| Tamanho | **Karatsuba** | **Padrão** | **Tamanho** | **Karatsuba** | **Padrão** | |
| 1 | 1 | 1 | 1 | 0 | 0 | |
| 2 | 4 | 5 | 2 | 0 | 0 | |
| 4 | 22 | 33 | 4 | 0 | 0 | |
| 8 | 70 | 70 | 8 | 0 | 0 | |
| 16 | 178 | 457 | 16 | 0 | 0 | |
| 32 | 541 | 1985 | 32 | 0 | 0,001 | |
| 64 | 1690 | 8077 | 64 | 0,001 | 0,001 | |
| 128 | 5026 | 32633 | 128 | 0,001 | 0,005 | |
| 256 | 15286 | 131797 | 256 | 0,001 | 0,066 | |
| 512 | 45559 | 513269 | 512 | 0,03 | 0,318 | |
| 1024 | 136924 | 2068069 | 1024 | 0,097 | 1,278 | |
| 2048 | 410113 | 8211149 | 2048 | 0,365 | 5,269 | |
| 4096 | 1230313 | 3296613 | 4096 | 1,02 | 20,411 | |
| 8192 | 3691360 | 131846025 | 8192 | 3,292 | 86,712 | |
| 16384 | 11076770 | 527138981 | 16384 | 7,72 | 275,497 | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Base 16 | | | | | |
| Acessos | | | **Tempo** | | |
| Tamanho | **Karatsuba** | **Padrão** | **Tamanho** | **Karatsuba** | **Padrão** |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 2 | 1 | 1 | 2 | 0 | 0 |
| 4 | 16 | 25 | 4 | 0 | 0 |
| 8 | 70 | 121 | 8 | 0 | 0 |
| 16 | 202 | 593 | 16 | 0 | 0 |
| 32 | 553 | 1933 | 32 | 0 | 0,001 |
| 64 | 1816 | 8645 | 64 | 0,001 | 0,001 |
| 128 | 5203 | 33233 | 128 | 0,001 | 0,011 |
| 256 | 15619 | 135241 | 256 | 0,006 | 0,078 |
| 512 | 47323 | 540857 | 512 | 0,025 | 0,338 |
| 1024 | 141787 | 2166329 | 1024 | 0,125 | 1,441 |
| 2048 | 430222 | 8711053 | 2048 | 0,377 | 5,69 |
| 4096 | 1282849 | 34794741 | 4096 | 1,149 | 23,196 |
| 8192 | 3849670 | 138964061 | 8192 | 3,255 | 88,486 |
| 16384 | 11526418 | 555077101 | 16384 | 8,199 | 285,749 |

Todas as Strings de entrada foram geradas de forma aleatória, portanto, podem variar um pouco o tempo de processamento, pois dependendo da base escolhida e número gerado, é necessário menos acessos recursivos, como exemplo, números com mais zeros, são processados recursivamente mais rápido que números com nenhum zero.

**CONCLUSÕES**

Como é possível observar nos dados, para Strings numéricas de tamanhos baixo, aproximadamente n<1024, é insignificante a diferença entre os métodos, sendo que fatores esternos podem influenciar e muito nos resultados. Mas a partir de n>=1024, há uma separação evidente nos gráficos que mostra um crescimento exponencial acelerado para o método Padrão.

Isto é facilmente explicado pelo livro do Cormen (Cormen, 2012), no qual aborda os conceitos teóricos sobre o custo dos algoritmos de recorrência. E no caso do Karatsuba que possui 3 chamadas recursivas, é mais eficiente que o Padrão que possui 4 chamadas recursivas, neste caso, com um ponto inicial provável em 1024, tendendo ao infinito.

Como trabalhos futuros recomendo comparar Karatsuba a ouros métodos interativos que possivelmente podem apresentar melhor custo em números baixos.

**REFERÊNCIAS**

Cormen, T. H. (2012). *Algoritmos : teoria e prática*. Campus. Retrieved from https://books.google.com.br/books?id=6iA4LgEACAAJ&dq=cormen+algoritmos+teoria+pratica&hl=pt-BR&sa=X&ved=0ahUKEwjHspTOutPkAhW3H7kGHfacDlMQ6AEIKTAA

Farias, F. Mi. de. (2019). Karatsuba String C++. Retrieved September 15, 2019, from https://github.com/fmflavio/Karatsuba

Karatsuba, A. A. (1995). The complexity of computations. *Proceedings of the Steklov Institute of Mathematics-Interperiodica Translation*, *211*, 169–183.