LETRA A – Método Pedestre

1) O algoritmo realiza a multiplicação de matrizes. Para isso:

Percorre cada linha e da M1, onde M1 é a 1a matriz e multiplica cada vetor-linha da M1 por cada vetor coluna da M2. O resultado da multiplicação da linha i de M1 por coluna j de M2 é inserida na posição i,j da matriz resultante M3.

2) Para isso temos que fazer 3 For's aninhados para selecionar cada linha da M1, cada coluna da M2 e o terceiro vem de fato para percorrer e multiplicar.

3) O algoritmo vem do direto do método de multiplicação de matrizes. Já que as matrizes são quadradas, cada For, vai iterar N vezes e como são 3 Fors aninhados temos O(n^3).

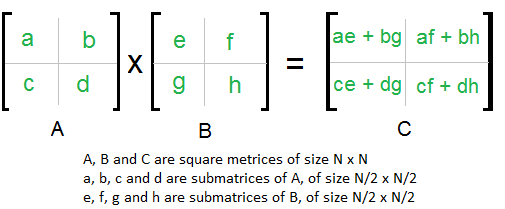
Segue abaixo os resultados:

|  |  |
| --- | --- |
| Size NxN | Tempo(s) |
| 16 | 0,0001 |
| 32 | 0,0002 |
| 64 | 0,0015 |
| 128 | 0,0107 |
| 256 | 0,0860 |
| 512 | 0,9050 |
| 1024 | 12,5116 |
| 2018 | 138,9660 |

LETRA B - Método de Strassen

1) O algoritmo realiza a multiplicação de matrizes. Para isso:

Utiliza uma abordagem de divisão em conquista com um truque matemático observado por Strassen. A divisão e conquista por si só, particiona a matriz em 4 pedaços iguais de tamanho SIZE /2 e soma recursivamente os valores de ae + bg, af + bh, ce + dg and cf + dh conforme pode ser visto na imagem abaixo:



Dessa forma, nós fazemos 8 multiplicações recursiva e 4 adições. Adição de matrizes tem complexidade O(N^2), então a complexidade desse algoritmo pode ser escrita como:

T(N) = 8T(N/2) + O(N2)

Utilizando o Teorema Mestre, temos: a=8,b=2,k=2 e portanto 8 > 2^2 e assim O(n^log(8,base=2)) = O(n^3).

E assim não há melhora em relação ao método pedestre. Mas a observação de Strassen era de que era possível chegar no produto de duas matrizes através das seguintes fórmulas:

P1= A \* ( F – H )  
P2= H \* ( A + B )  
P3= E \* ( C + D )  
P4= D  
P5= ( A + D ) \* ( E + H )  
P6= ( B – D ) \* ( G + H)  
P7= ( A – C ) \* ( E + F)



E com isso, diminuir o número de recursões para **7.**  Com isso, temos: T(N) = 7T(N/2) + O(N2), e portanto aplicando o teorema mestre temos a=7,b=2e k=2, gerando o resultado final de O(n^log(7,base=2)) = O(n^2.8074).

Etapas do algoritmo:

O caso base foi definido como 64x64 e não uma matriz de 1x1. O motivo dessa escolha foi porque quando iniciamos o algoritmo com o caso base de uma matriz 1x1, vimos computador dar crash algumas vezes. Pesquisando online foi observado que isso é decorrente de um overhead por chamar a função recursiva até 1 elemento.

De acordo com esse estudo:

<http://ce.u-sys.org/Veranstaltungen/SiWiR1%20(Ru%CC%88de)/Uebungen%20WS0910/Julian2_Steffi_Balthi/ex01-presentation.pdf>

O breakeven para o algoritmo de Strassen foi uma matriz de 64x64 que usamos como nossa referência.

Apesar do Strassen ter uma menor complexidade tem um Big-O constant bem maior. É como comparar n^3 com n^2.8 + 100n^2 e portanto não vale a pena descer até 1 elemento.

Ao atingir 64, estamos fazendo o Fall Back para o algorítimo pedestre que performa melhor em instância menores.

O segundo passo do algoritmo segue a seguir:

Particiona a matriz em sub matrizes n/2.

Começa a calcular os Ps a partir de:

Faz a submatriz para o espaço específico aonde a soma ou subtração de Ps ocorrerão.

Executa a soma ou subtração, por exemplo: a+b ou f-h

Chama a Multiplicação recursivamente pela partição conforme a formula de Strassen, por exemplo a\*(f-h).

Aloca a soma o “p” dentro do seu quadrante específico.

Executa os passos acima de P1 a P7.

Seguem os resultados abaixo:

|  |  |
| --- | --- |
| NxN | Tempo(s) |
| 16 | 0 |
| 32 | 0,0002 |
| 64 | 0,0026 |
| 128 | 0,0128 |
| 256 | 0,0938 |
| 512 | 0,5547 |
| 1024 | 3,807 |
| 2048 | 30,6102 |

Comparativo:

|  |  |  |
| --- | --- | --- |
| Size NxN | Pedestre | Strassen |
| 16 | 0,0001 | 0 |
| 32 | 0,0002 | 0,0002 |
| 64 | 0,0015 | 0,0026 |
| 128 | 0,0107 | 0,0128 |
| 256 | 0,0860 | 0,0938 |
| 512 | 0,9050 | 0,5547 |
| 1024 | 12,5116 | 3,807 |
| 2018 | 138,9660 | 30,6102 |

Para o cálculo do tempo de execução foram rodadas 10 vezes para o mesmo tamanho da amostra e calculado uma média no final. Para o cálculo do tempo foi usado a biblioteca time.h.