INF 112 – Programação II

Trabalho prático I: Multiplicação "rápida" de matrizes!

Seu objetivo neste trabalho será mostrar que você entendeu bem os conceitos de alocação dinâmica, recursividade e introdução à análise de algoritmos. Para isso, você terá que implementar algumas tarefas relacionadas à multiplicação de matrizes.

Sejam A e B duas matrizes com n linhas e n colunas. Você pode assumir que os elementos de A e B são números inteiros com sinal de 64 bits (i.e., `long`). Você terá que implementar duas estratégias para calcular a matriz $C = AB$, ou seja, C é o produto matricial de A e B.

Estratégia 1: Multiplicação pela definição

Você deve implementar o produto matricial computado pela definição. Em outras palavras:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \tag{1}$$

onde c_{ij} é o elemento da linha i e coluna j de C, a_{ik} é o elemento da linha i e coluna k de A e b_{kj} é o elemento da linha k e coluna j de B.

Pode-se mostrar que, utilizando essa abordagem, o número de multiplicações necessárias é $O(n^3)$.

Estratégia 2: Multiplicação rápida

Há um famoso algoritmo na literatura que é capaz de calcular o produto AB com um número significativamente menor de multiplicações, mas com um número maior de operações de soma. Suponha inicialmente que $n = 2^k$, para algum k inteiro e positivo. Assim, pode-se dividir A, B e C em submatrizes de $\frac{n}{2}$ linhas e $\frac{n}{2}$ colunas da seguinte forma:

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}, B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}, \text{ e } C = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}. \tag{2}$$

Só para ficar claro, cada $A_{i,j}$, $B_{i,j}$ e $C_{i,j}$ são matrizes com $\frac{n}{2}$ linhas e $\frac{n}{2}$ colunas, as quais correspondem a uma submatriz de A, B e C, respectivamente.

A ideia principal do algoritmo é calcular as submatrizes $C_{i,j}$ de uma forma mais esperta do que pela definição. Para isso, inicialmente calcule cada uma das seguintes sete matrizes:

$$\begin{aligned}
M_1 &= (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2}) \\
M_2 &= (A_{2,1} + A_{2,2})B_{1,1} \\
M_3 &= A_{1,1}(B_{1,2} - B_{2,2}) \\
M_4 &= A_{2,2}(B_{2,1} - B_{1,1}) \\
M_5 &= (A_{1,1} + A_{1,2})B_{2,2} \\
M_6 &= (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2}) \\
M_7 &= (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2}).
\end{aligned}$$

Então, tem-se que:

$$\begin{aligned}
C_{1,1} &= M_1 + M_4 - M_5 + M_7 \\
C_{1,2} &= M_3 + M_5 \\
C_{2,1} &= M_2 + M_4 \\
C_{2,2} &= M_1 - M_2 + M_3 + M_6
\end{aligned}$$

Repare a estrutura recursiva da solução proposta acima. O produto de duas matrizes AB (cada uma com n linhas e n colunas) pode ser caculado recursivamente por meio dos produtos que definem $M_1, \ldots, M_7$ (cada uma com $\frac{n}{2}$ linhas e $\frac{n}{2}$ colunas).

Pode-se mostrar que essa abordagem requer $O(n^{2.8})$ multiplicações.

ATENÇÃO: quando n não for uma potência de 2 (i.e., quando $n \neq 2^k$ para todo k inteiro e positivo), é necessário fazer uma passo de pré-processamento. Nesse caso, deve-se:

1. encontrar o menor k tal que $2^k \geq n$;
2. alocar duas matriz, A' e B', cada uma com 2^k linhas e 2^k colunas, e inicializar todos seus valores com 0;
3. copiar A para as n primeiras linhas e n primeiras colunas de A' e B para as n primeiras linhas e n primeiras colunas de B';

4. calcular o produto $C' = A'B'$ como descrito acima;

5. copiar as n primeiras linhas e n primeiras colunas de C' para C, o resultado final.

O que deve ser entregue

Você deve entregar apenas dois arquivos, `1.c` e `2.c`. Isso, o trabalho deve ser feito em C (sem o `++`)! O arquivo `1.c` deverá conter a implementação da Estratégia 1 e o arquivo `2.c` deverá conter a implementação da Estratégia 2. Esses dois arquivos devem ser comprimidos no formato `.tar.gz` usando o comando `tar -czvf trab1.tar.gz 1.c 2.c` no Linux.

Entrada

A entrada será feita feita por meio da entrada padrão. Os dois programas terão o mesmo formato de entrada. Primeiro, deve-se ler n, o número de linhas e colunas das matrizes. Após isso, deve-se ler as n linhas da matriz A, cada uma contendo os n elementos das respectivas colunas. Por fim, deve-se ler as n linhas da matriz B, cada uma contendo os n elementos das respectivas colunas.

Nada além do pedido acima deve ser lido.

ATENÇÃO: n poderá ser arbitrariamente grande nos testes que o professor vai fazer. É garantido que as matrizes de entrada e saída sempre caberão na memória principal do computador.

Saída

A saída será feita por meio da saída padrão. Os dois programas terão o mesmo formato de saída. Deve-se imprimir a matriz C, ou seja, a saída terá n linhas, representando respectivamente cada linha da matriz C em ordem.

Nada além do pedido acima deve ser impresso na tela.

Exemplo

A seguir, um exemplo de possível execução no terminal:

```
gcc -o 1 1.c
./1
3
1 2 3
4 5 6
7 8 9
9 8 7
6 5 4
3 2 1
30 24 18
84 69 54
138 114 90
```

No exemplo acima, primeiro o usuário digita o número 3 (n). Após isso, são informados os 9 valores referentes a matriz A e 9 valores referentes a matriz B. Os últimos 9 valores representam a saída do programa, i.e., a matriz C. Repare que o único separador entre as colunas da matriz é um único caractere de espaço. Você deve seguir esse mesmo padrão no seu trabalho.

ATENÇÃO: A mesma saída deve ser produzida ao fornecer a mesma entrada para a Estratégia 2.

Critérios de correção

Os seguintes critérios serão considerados:

- Se usar variáveis globais, nota zero;
- Se usar `goto`, nota zero;
- Se seu trabalho não compilar, nota zero. Se por algum motivo seu código compilar (ou funcionar) no seu computador, mas não no computador do professor, o critério de "desempate" será se o seu trabalho compila (funciona) nos computadores do laboratório CCE 416, considerando o Sistema Operacional Ubuntu. Muita atenção aos usuários de Windows!
- Fração de respostas corretas;
- Organização e modularização do código;

- Legibilidade, i.e., código comentado e nomes intuitivos para as variáveis;
- Presença de vazamento de memória e acesso a posições inválidas de memória. Use `Valgrind` desde os primeiros testes! Em casos extremos, a não observância desse quesito implicará em nota zero;
- Eficiência! Haverá bônus para os trabalhos com as implementações mais eficientes, especialmente para a Estratégia 2.

Como deve ser entregue

Cada dupla deve enviar apenas um trabalho para o e-mail `gcom.tp.sub@gmail.com`, até às 23:59 do dia 17 de Setembro de 2019. O trabalho deve ser enviado por um e-mail de domínio `ufv.br` (e-mails de qualquer outro domínio não serão considerados).

O título (subject) do e-mail deve conter o número de matrícula dos integrantes do grupo (sem o ES), separados por uma vírgula. Se você optar for fazer o trabalho sozinho, esse campo terá apenas seu número de matrícula.

Se a mesma dupla enviar mais de um trabalho, apenas o e-mail mais recente será considerado. O e-mail deve ter em anexo apenas um arquivo comprimido, no formato `.tar.gz`, com nome `trab1.tar.gz`, contendo seus arquivos fontes.

Após baixar o arquivo, o professor digitará os comandos:

```
tar -xzvf trab1.tar.gz
gcc -o 1 1.c
gcc -o 2 2.c
```

ATENÇÃO: a conformidade com os critérios aqui estabelecidos faz parte da avaliação. Se você não entregar o trabalho no prazo, ou se o executável não for gerado da forma indicada, você receberá nota zero.

Algumas outras regras

- O trabalho deve ser implementado em C (compilado com `gcc`);
- O trabalho pode ser feito em dupla (duplas desse trabalho não poderão estar no mesmo grupo nos trabalhos 2 e 3);
- Plágio não será tolerado. O regimento acadêmico será seguido à risca em caso de suspeita.