Trabalho 01 - Multiplicação de Matrizes

Data de entrega: 04/10/2024

Importante:

- **Não** olhem códigos de outros grupos ou da internet. Exceto o que é fornecido.
- Os trabalhos poderão ser feitos individualmente ou em duplas. Em qualquer caso é necessário preencher esse formulário para a formação de grupos (mesmo que seja um grupo de 1 só pessoa): https://forms.gle/YACY7KDAz5dSCPRGA
- TODOS os membros do grupo devem participar e compreender completamente a implementação.
- Em caso de plágio, fraude ou tentativa de burlar o sistema será aplicado nota 0 na disciplina aos envolvidos.
- Alguns alunos serão solicitados para explicar com detalhes a implementação.
- Passar em todos os testes não é garantia de tirar a nota máxima. Sua nota ainda depende do cumprimento das especificações do trabalho, qualidade do código, clareza dos comentários, boas práticas de programação e entendimento da matéria demonstrada em possível reunião.
- O líder do grupo deverá submeter, até a data de entrega, o seu código na plataforma runcodes.hokama.com.br.
- É esperado e faz parte do aprendizado vocês terem algumas dificuldades como Problemas com Falha de Segmentação, Loops Infinitos, etc. Então não deixe para os últimos dias!

Esse trabalho deverá ser realizado individualmente ou em duplas, com os grupos já definidos na disciplina. Neste trabalho o seu grupo deverá implementar o algoritmo de **Strassen** para multiplicação de matrizes, que utilizaremos para aplicação de transformações e filtros em imagens. (Sugestão: normalmente algoritmos de divisão e conquista são implementados de maneira recursiva, o que facilita bastante o trabalho).

Neste trabalho usaremos arquivos de imagem em formato PPM, um formato de imagem livre, não é muito otimizado, mas a vantagem dele é ser bem simples de ler, entender e manipular. Existem algumas codificações do formato PPM, aqui usaremos o formato P3, ele tem a seguinte forma:

- começa com o nome da codificação utilizada “P3”;
- depois a largura e altura, no nosso trabalho iremos considerar apenas imagens quadradas, ou seja, n x n, e ainda n é sempre uma potência de dois, o que vai facilitar a implementação;
- Um inteiro com a escala máxima de cor, normalmente é o valor 255;
- depois existem n linhas cada linha contém n pixels;

Cada pixel é representado em um formato rgb, composto por 3 valores representando a quantidade de vermelho, verde e azul respectivamente. Sendo que a escala de cada cor é

um valor entre 0 e 255. Por exemplo um pixel totalmente vermelho tem a seguinte codificação:

```
255 0 0
```

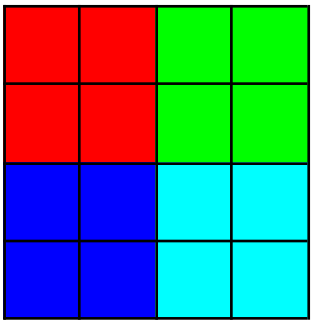
Já um pixel magenta, tem um pouco de vermelho e um pouco de azul:

```
240 0 240
```

Um pixel preto não tem nada de nenhuma cor

```
0 0 0
```

Exemplo: Considere a seguinte imagem 4 x 4 com 16 pixels:



em formato PPM teria a seguinte codificação:

```
P3
4 4
255
255 0 0 255 0 0 0 255 0 0 255 0
255 0 0 255 0 0 0 255 0 0 255 0
0 0 255 0 0 255 0 255 255 0 255 255
0 0 255 0 0 255 0 255 255 0 255 255
```

Uma forma comum de aplicar transformações e filtros em uma imagem é multiplicar a matriz de uma imagem pela matriz de um filtro. Neste trabalho o filtro vai ser um matriz de *n* x *n* de “pixels”, cada pixel do filtro tem apenas valores 0 ou 1. Para multiplicar um pixel da imagem por um pixel do filtro, basta multiplicar cada cor deles separadamente: (a, b, c) x (d, e, f) = (a*d, b*e, c*f). Por exemplo um filtro de 4 x 4 seria esse:

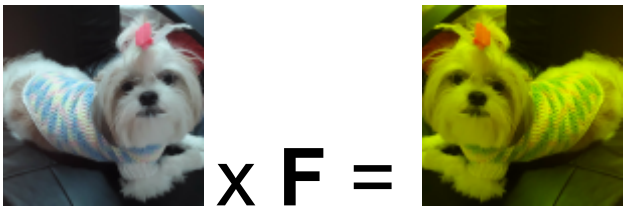
```
1 1 1 0 0 0 0 0 0 0 0 0
```

```
0 0 0 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1
```

Esse filtro é a identidade **I**, ele não faz nenhuma alteração na imagem, ou seja,

Agora considere o seguinte filtro F, que inverte a imagem na horizontal e retira o azul (note que os pixels não nulos são (1, 1, 0) e dessa forma os valores de azul de cada pixel da imagem são multiplicados por 0.

```
0 0 0 0 0 0 0 0 0 1 1 0
0 0 0 0 0 0 1 1 0 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0
```



O seu programa deverá ler da entrada padrão do sistema um texto equivalente a uma imagem no formato ppm seguida de uma matriz de mesmo tamanho que é um filtro

Veja um exemplo de entrada:

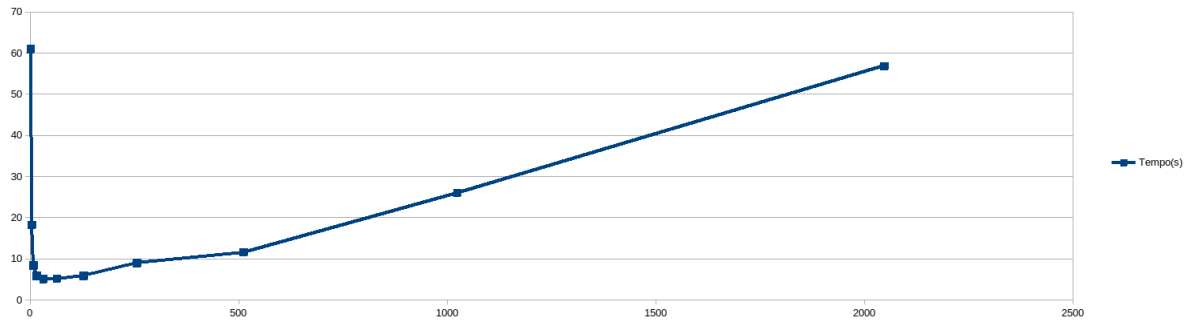
```
P3
4 4
255
255 0 0 255 0 0 0 255 0 0 255 0
255 0 0 255 0 0 0 255 0 0 255 0
0 0 255 0 0 255 0 255 255 0 255 255
0 0 255 0 0 255 0 255 255 0 255 255
0 0 0 0 0 0 0 0 0 1 1 0
0 0 0 0 0 0 1 1 0 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0
```

A saída esperada para seu programa será um texto equivalente a uma imagem em formato ppm. (Para facilitar cada linha da matriz tem um espaço no final)

```
P3
4 4
255
0 255 0 0 255 0 255 0 0 255 0 0
0 255 0 0 255 0 255 0 0 255 0 0
0 255 0 0 255 0 0 0 0 0 0 0
0 255 0 0 255 0 0 0 0 0 0 0
```

Caso Base

Você vai notar que o algoritmo de Strassen vale muito a pena quando a dimensão da matriz é grande. porém ele tem constantes muito grandes, e por isso ele pode não valer muito a pena quando a dimensão é pequena. Isso vale particularmente para o caso base. Se o seu caso base for uma matriz de 1x1, o algoritmo funciona, porém ele vai resolver Strassen até o último nível, e isso pode ser ineficiente. Por isso é útil aumentar o tamanho do caso base (tente valores como 16x16 ou 32x32 ou 64x64), e quando atingi-lo você utiliza a multiplicação tradicional. Veja o gráfico abaixo onde no eixo x temos o tamanho do caso base e no eixo y o tempo em segundos que o algoritmo de Strassen demora para executar em uma instância particular;



- Você deverá implementar seu programa em linguagem C.
- Seu programa deve executar no run.codes em menos de 1 segundo.
- Você não pode usar bibliotecas especializadas, e só poderá executar operações em tipos primitivos (int). Por exemplo, você não pode utilizar uma operação que multiplica uma lista ou um vetor, a menos que você a implemente.
- Você deve liberar adequadamente a memória alocada.
- Se você não tiver certeza se alguma coisa é permitida ou não no trabalho, não hesite em perguntar ao professor!