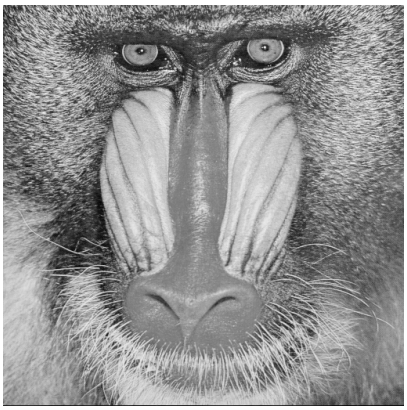
MC102 - Algoritmos e Programação de Computadores

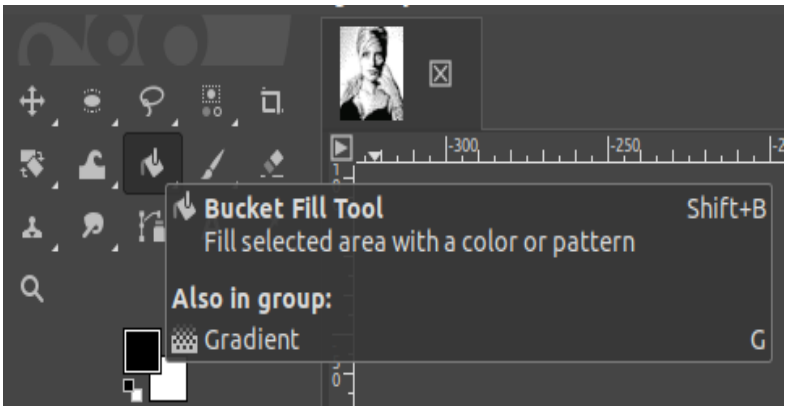
Lab 13

Data da Primeira Chance: 30 de junho de 2023
Peso: 3

Nos últimos meses, o time de *designers* da sua empresa tem recebido centenas de pedidos e projetos envolvendo a criação de desenhos, *banners* e artefatos visuais em preto e branco e escala cinza, no formato Portable Gray Map (PGM). A criação desses é extremamente custosa, uma vez que exige a alocação de um *designer*, um desenhista, um ilustrador digital e outros tantos profissionais envolvidos. Entretanto, o gerente do departamento notou que muitos destes projetos estão se tornando repetitivos, e que o trabalho poderia ser mais rapidamente executado se seu time tivesse algumas ferramentas automáticas disponíveis.

Alguns softwares estão disponíveis no mercado, porém não podem ser empregados aqui devido a altos custos associados, restrições em suas licenças ou incompatibilidade com o ambiente de criação do time. O gerente então contatou o time de desenvolvimento, do qual você faz parte, para a construção de uma ferramenta interna que auxiliasse seus ilustradores em seus trabalhos.





Imagens são representadas em um meio digital por matrizes de pixels, onde cada pixel é um inteiro no intervalo de 0 a 255 (um *byte*), representando a intensidade de luz emitida pela fonte. Para imagens em escala cinza, 0 representa a completa ausência de cor (preto), enquanto 255 representa a ativação completa (branco). Valores intermediários de pixel descrevem ativações intermediárias (cinza), em uma interpolação linear entre o completo preto e o completo branco.

Imagens no formato *Portable Gray Map* (PGM) são armazenadas como um arquivo texto, contendo em suas linhas (1) o identificador “P2”, referente a versão do formato PGM empregada; (2) um *header* de descrição do arquivo, iniciado por #; (3) o número de colunas

e linhas na imagem; (4) o valor de píxel máximo contido no arquivo (usualmente 255) e (5) os bytes indicadores da imagem.

Abaixo, encontra-se um exemplo mínimo de uma imagem PGM:

```
P2
# Created by GIMP version 2.10.30 PNM plug-in
5 3
255
0   0  50 127   0
0  50 255 255   0
0   0 200 200 200
```

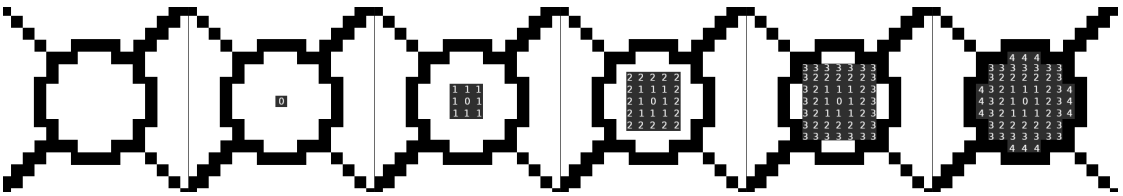
Programas de edição de imagem frequentemente implementam ferramentas que auxiliam os ilustradores e designers na execução de seus trabalhos, como a ferramenta de preenchimento de balde de tinta e o seletor automático de regiões por similaridade de cores. Estas ferramentas possuem algo em comum: a identificação de **regiões conexas** à região semente (*seed*), isto é, de cor igual à região *seed*, salvo por um limiar de tolerância *t* (isto é, (abs($pixel_{atual}$ - $pixel_{seed}$) <= *t*) que podem ser alcançadas a partir desta, considerando as oito direções possíveis (esquerda, direita, cima, baixo e as quatro diagonais), sem que quaisquer fronteiras ou traços de diferentes cores as separem.

Na imagem de exemplo abaixo, os píxeis com intensidades 224, 226 e 221 pertencem a uma única região conexa ($pixel_{seed}$ = 224, *t=5*), já que *abs(226-224)* e *abs(221-224)* são menores ou iguais ao limiar 5:

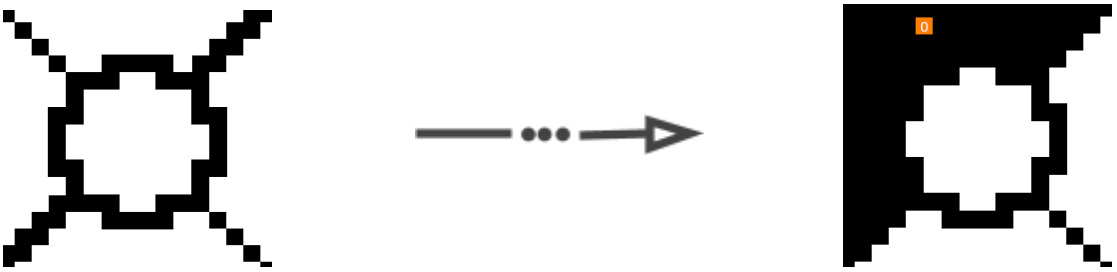
224*	226
114	221

Em um primeiro momento, você foi incumbido de implementar um programa de edição de imagens contendo as seguintes funcionalidades:

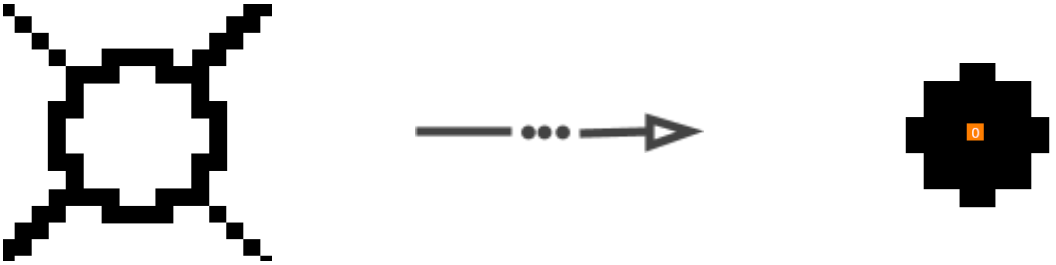
1. Preenchimento com balde: preencher as regiões conexas a uma região semente com a cor específica **C**.
   Importante: píxeis com intensidade exatamente igual à **C** não devem ser expandidos, isto é, não deve ter sua cor alterada e seus vizinhos não devem ser avaliados;



2. Negativo: inverter as cores das regiões conexas a uma região semente (intensidade de píxel máxima - $pixel_{atual}$);



3. Máscara complementar: retornar uma imagem binária (máscara) contendo 0 nos píxeis pertencentes a região conexa contendo o píxel semente, e 255 nas demais regiões não-conexas.



As três ferramentas (preenchimento com balde, negativo e máscara complementar) devem ser **necessariamente** implementadas como funções recursivas ou composições de funções recursivas.

Finalmente, o programa também deve ser capaz de ler e escrever imagens binárias do disco, no formato *Portable Gray Map* (PGM).

Dica: você pode ajustar o número máximo de chamadas recursivas de um programa Python da seguinte maneira:

```
import sys
sys.setrecursionlimit(16385)
```

Entrada

Um arquivo de tarefa é composto das seguintes informações:

1. Caminho para uma imagem PGM;
2. O número de operações **O** a serem aplicadas sobre a imagem lida;
3. Serão então apresentadas **O** linhas contendo o nome da operação a ser realizada, seguida de seus argumentos;

As seguintes operações são válidas:

Nome	**Modelo**	**Argumentos**	**Exemplo**
Preenchimento com balde de tinta	bucket c t col row	c: int, cor utilizada no preenchimento t: int, limiar de tolerância col, row: *tuple[int, int]*, região semente	bucket 127 5 32 42
Negativo de cor	negative t col row	t: int, limiar de tolerância col, row: *tuple[int, int],* região semente	negative 5 12 7
Máscara complementar	cmask t col row	t: int, limiar de tolerância col, row: *tuple[int, int],* região semente	cmask 5 19 241
Salvar imagem	save		save

Saída

Ao receber o comando *save*, seu programa deve imprimir a imagem no formato PGM na tela (utilizando o comando *print* ou escrevendo no buffer *sys.stdout*). A informação impressa no stdout será considerada na avaliação.

Importante: o arquivo de imagem impresso deve sempre conter o *header* de descrição "# Imagem criada pelo lab13", como nos exemplos a seguir.

Exemplos

Exemplo 1:

Entrada

```
images/sample.pgm
3
bucket 25 5 4 0
bucket 127 5 4 2
save
```

Arquivo images/sample.pgm

```
P2
# Created by GIMP version 2.10.30 PNM plug-in
5 3
255
0   0  50 127   0
0  50 255 255   0
0   0 200 200 200
```

Saída

```
P2
# Imagem criada pelo lab13
5 3
255
0 0 50 127 25
0 50 255 255 25
0 0 127 127 127
```

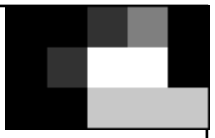
Exemplo 2:

Entrada

```
images/sample.pgm
4
negative 5 3 1
negative 5 1 1
negative 5 3 2
save
```

Arquivo images/sample.pgm

```
P2
# Created by GIMP version 2.10.30 PNM plug-in
5 3
255
0   0  50 127   0
0  50 255 255   0
0   0 200 200 200
```



Saída

```
P2
# Imagem criada pelo lab13
5 3
255
0 0 50 127 0
0 50 0 0 0
0 0 55 55 55
```

Exemplo 3:

Entrada

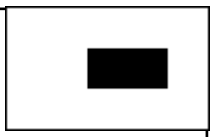
```
images/sample.pgm
2
cmask 5 2 1
save
```

Arquivo images/sample.pgm

```
P2
# Created by GIMP version 2.10.30 PNM plug-in
5 3
255
0   0  50 127   0
0  50 255 255   0
0   0 200 200 200
```

Saída

```
P2
# Imagem criada pelo lab13
5 3
255
255 255 255 255 255
255 255 0 0 255
255 255 255 255 255
```



Regras e Avaliação

Nesse laboratório, você não pode usar bibliotecas (isto é, o comando *import*), exceto pelas bibliotecas *typing* para melhorar a clareza e escrita do código e *sys* para (e somente para) ajustar o tamanho da pilha de recursão do python ou escrever no buffer *sys.stdout* (caso seja necessário).

Seu código será avaliado não apenas pelos testes do CodePost, mas também pela qualidade. Dentre os critérios subjetivos de qualidade de código analisaremos nesse laboratório: as ferramentas (preenchimento com balde, negativo e máscara complementar) devem ser **obrigatoriamente implementadas de forma recursiva**, o uso apropriado de funções, e de documentação; a escolha de bons nomes de funções e variáveis; a ausência de diversos trechos de código repetidos desnecessariamente. Note, porém, que essa não é uma lista exaustiva, pois outros critérios podem ser analisados dependendo do código apresentado visando mostrar ao aluno como o código poderia ser melhor.

Os casos de testes estão disponíveis através do link.

Submissão

Você deverá submeter no CodePost, na tarefa Lab 13, um arquivo com o nome `lab13.py`. Você pode enviar arquivos adicionais caso deseje para serem incluídos por `lab13.py`.