Busca em Imagens

Uma imagem digital é a projeção de uma matriz, que representa a imagem, em uma tela. Cada posição `(x,y)` da matriz, onde `x` é o número da linha e `y` é o número da coluna, representa um pixel (*picture element*) `p(x,y)`. O valor de `p(x,y)` indica a cor que deve ser projetada na posição correspondente da imagem. O pixel no canto superior esquerdo da imagem é o `p(1,1)` e o pixel no canto inferior direito da imagem é o `p(n,m)`, para uma imagem com `n` linhas e `m` colunas.

Nas imagens em tons de cinza PGM (Portable Graymap Format), esse valor descreve a intensidade do ponto projetado que varia de 0 (preto) a 255 (branco). As coordenadas da matriz seguem a ordem *raster* (de cima para baixo, da esquerda para direita). Abaixo temos um exemplo do conteúdo de um arquivo PGM.

```
P2
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 51 51 51 51 0 0 51 51 51 51 0 0 119 119 119 119 0 0 119 119 119 119 0
0 51 0 0 0 0 51 0 0 0 0 0 119 0 0 0 0 0 119 0 0 119 0
0 51 51 51 0 0 0 51 51 51 0 0 0 119 119 119 0 0 0 119 119 119 119 0
0 51 0 0 0 0 0 51 0 0 0 0 0 119 0 0 0 0 0 119 0 0 0 0
0 51 0 0 0 0 0 51 51 51 51 0 0 119 119 119 119 0 0 119 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

O cabeçalho do arquivo é formado pelas três primeiras linhas, sendo que a primeira linha possui o valor `P2`, que indica o formato do arquivo; a segunda linha possui dois valores inteiros `m` e `n` que representam o número de colunas e linhas, respectivamente; e a terceira linha possui o valor máximo que um pixel pode assumir, no nosso caso, esse valor sempre será `255`. Depois do cabeçalho, o arquivo possui `n` linhas, onde cada linha possui `m` valores inteiros separados por um espaço em branco. Cada um desses valores representa um pixel da imagem em escala de cinza. Uma imagem no formato PGM pode ser visualizada utilizando este site. Podemos representar as linhas da imagem após o cabeçalho com uma matriz de `n` linhas e `m` colunas, onde cada posição da matriz representa um pixel.

Dizemos que um padrão (imagem a ser buscada) está contido em uma imagem `A`, se existe uma submatriz da imagem `A` igual a matriz do padrão. Neste laboratório,

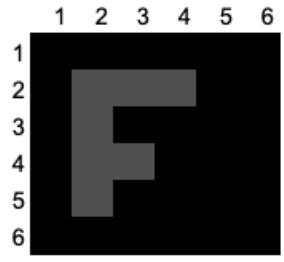
dada uma imagem `A` e uma imagem `B`, você deve indicar se os seguintes padrões estão contidos em `A`:

- imagem original `B`;
- imagem `B` após um **flip** horizontal;
- imagem `B` após um **flip** vertical;
- imagem `B` após uma rotação de 90 graus;
- imagem `B` após uma rotação de 180 graus.

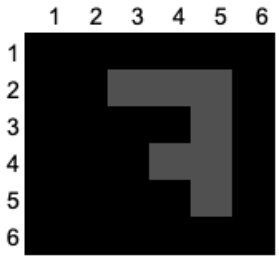
Flip

Flip é a operação que espelha uma imagem na horizontal ou na vertical. Considerando o **flip** horizontal, cada pixel `p(x,y)` da imagem resultante é igual ao pixel `p'(x', y')` da imagem original, sendo que `x' = x` e `y' = m + 1 - y`, onde `m` é a quantidade de colunas. Já para o **flip** vertical, cada pixel `p(x,y)` da imagem resultante é igual ao pixel `p'(x', y')` da imagem original, sendo que `x' = n + 1 - x` e `y' = y`, onde `n` é o número de linhas da imagem.

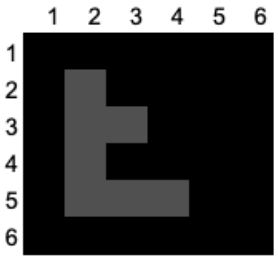
A seguir exemplificamos as operações de **flip** horizontal e vertical. Os números na vertical e horizontal de cada imagem representam o número da linha e da coluna, respectivamente.



Padrão Original



Flip horizontal



Flip vertical

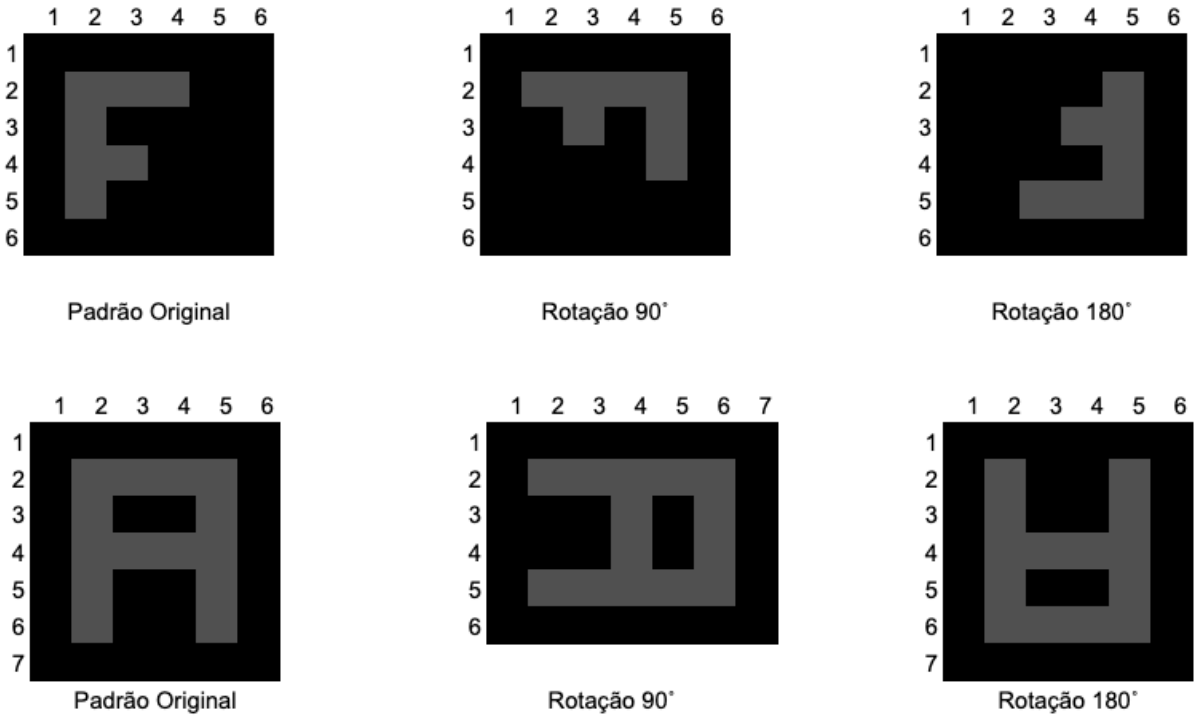
Rotação

Nesta operação, o seu programa deve rotacionar uma imagem em 90 graus ou 180 graus em sentido horário. Note que rotacionar uma imagem em 180 graus é equivalente a aplicar a operação de rotação de 90 graus duas vezes.

Dada uma imagem `A` de `n` linhas e `m` colunas, a imagem resultante `B` da rotação de 90 graus em `A` possui `m` linhas e `n` colunas, sendo que: a `n`-ésima coluna de `B` é igual a primeira linha de `A`; a `(n-1)`-ésima coluna de `B` é igual a segunda

linha de `A`; e, de forma geral, a `i`-ésima coluna de `B` é igual a `(n+1)-i`-ésima linha de `A`.

A seguir exemplificamos as operações de rotação. Os números na vertical e horizontal de cada imagem representam o número da linha e da coluna, respectivamente.



Entrada

O seu programa receberá como entrada a imagem `A` e, em seguida, a imagem `B`, ambas no formato PGM. Cada imagem terá as seguintes linhas (em ordem): uma linha contendo o valor `P2`, que indica o formato do arquivo e deve ser desconsiderada; uma linha contendo dois números `m` e `n`, que indicam o número de colunas e linhas da imagem, respectivamente; uma linha que contém o valor máximo que um pixel pode assumir (no nosso caso, esse valor sempre será `255`); `n` linhas que possuem `m` valores inteiros separados por espaços, representando os valores de cada pixel da imagem em escala de cinza, na ordem *raster* (de cima para baixo, da esquerda para direita).

Saída

A saída deverá indicar, para cada um dos cinco padrões da imagem `B`, se ele está contido na imagem `A`.

```
Original: <contido_ou_nao>
Flip horizontal: <contido_ou_nao>
Flip vertical: <contido_ou_nao>
Rotacao 90: <contido_ou_nao>
Rotacao 180: <contido_ou_nao>
```

Onde `<contido_ou_nao>` deve ser substituído pelas strings "Contido" ou "Nao contido".

Você pode visualizar as imagens no formato PGM usando este site. Para entender o efeito de cada operação, visualize todas imagens dos casos de testes abertos. Note que cada arquivo de entrada possui duas imagens em formado PGM. Logo, você deve separar essas duas imagens para visualização no site.

Código Base

No arquivo auxiliar lab12.py você irá encontrar um código base para dar início ao processo de elaboração deste laboratório. Para facilitar a implementação do seu programa, no código base existem os cabeçalhos das funções correspondentes a operação de *flip* e rotação. Cada função desempenha uma tarefa bastante específica. Usando essas funções é possível obter uma solução para o problema. Os cabeçalhos das funções a serem implementadas são apresentados a seguir.

```
def flip_horizontal(imagem_original):
  ...

def flip_vertical(imagem_original):
  ...

def rotacao_90(imagem_original):
  ...

def rotacao_180(imagem_original):
  ...
```

Exemplos de entradas e saídas esperadas pelo seu programa:

Teste 01

Entrada

```
P2
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 30 30 30 30 0 0 30 30 30 30 0 0 100 100 100 100 0 0 180 180 180 180 0
0 30 0 0 0 0 0 30 0 0 0 0 0 100 0 0 0 0 0 180 0 0 180 0
0 30 30 30 0 0 0 30 30 30 0 0 0 100 100 100 0 0 0 180 180 180 180 0
0 30 0 0 0 0 0 30 0 0 0 0 0 100 0 0 0 0 0 180 0 0 0 0
0 30 0 0 0 0 0 30 30 30 30 0 0 100 100 100 100 0 0 180 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
P2
6 7
255
0 0 0 0 0 0
0 30 30 30 30 0
0 30 0 0 0 0
0 30 30 30 0 0
0 30 0 0 0 0
0 30 0 0 0 0
0 0 0 0 0 0
```

Saída

```
Original: Contido
Flip horizontal: Nao contido
Flip vertical: Nao contido
Rotacao 90: Nao contido
Rotacao 180: Nao contido
```

Teste 03

Entrada

```
P2
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 30 30 30 30 0 0 30 30 30 30 0 0 100 100 100 100 0 0 180 180 180 180 0
0 30 0 0 0 0 0 30 0 0 0 0 0 100 0 0 0 0 0 180 0 0 180 0
0 30 30 30 0 0 0 30 30 30 0 0 0 100 100 100 0 0 0 180 180 180 180 0
0 30 0 0 0 0 0 30 0 0 0 0 0 100 0 0 0 0 0 180 0 0 0 0
0 30 0 0 0 0 0 30 30 30 30 0 0 100 100 100 100 0 0 180 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
P2
8 5
255
```

```
30 0 0 0 0 0 30 0
30 0 0 0 0 0 30 0
30 0 0 0 30 30 30 0
30 0 0 0 0 0 30 0
30 0 0 30 30 30 30 0
```

Saída

```
Original: Nao contido
Flip horizontal: Nao contido
Flip vertical: Nao contido
Rotacao 90: Nao contido
Rotacao 180: Contido
```

Teste 06

Entrada

```
P2
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 80 80 80 80 0 0 80 80 80 80 80 0 130 130 130 130 0 0 80 0 0 80 0
0 80 0 0 80 0 0 0 0 80 0 80 0 130 0 0 0 0 0 80 0 0 80 0
0 80 80 80 80 0 0 0 0 80 0 80 0 130 130 130 0 0 0 80 80 80 80 0
0 80 0 0 0 0 0 0 0 80 80 80 0 130 0 0 0 0 0 80 0 0 80 0
0 80 0 0 0 0 0 0 0 0 0 0 0 130 130 130 130 0 0 80 80 80 80 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
P2
6 7
255
0 0 0 0 0 0
0 80 80 80 80 0
0 80 0 0 80 0
0 80 80 80 80 0
0 80 0 0 0 0
0 80 0 0 0 0
0 0 0 0 0 0
```

Saída

```
Original: Contido
Flip horizontal: Nao contido
Flip vertical: Nao contido
Rotacao 90: Contido
Rotacao 180: Nao contido
```

Teste 08

Entrada

```
P2
24 7
255
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 80 80 80 80 0 0 0 0 0 0 0 130 130 130 130 0 0 80 0 0 80 0
0 80 0 0 80 0 0 0 80 80 80 0 0 80 0 0 0 0 0 80 0 0 80 0
0 80 80 80 80 0 0 0 0 0 80 0 0 80 80 0 0 0 0 80 80 80 80 0
0 80 0 0 0 0 0 0 0 80 80 0 0 80 0 0 0 0 0 80 0 0 80 0
0 80 0 0 0 0 0 0 0 0 80 0 0 80 0 0 0 0 0 80 80 80 80 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
P2
3 4
255
80 80 80
0 0 80
0 80 80
0 0 80
```

Saída

```
Original: Contido
Flip horizontal: Nao contido
Flip vertical: Nao contido
Rotacao 90: Nao contido
Rotacao 180: Nao contido
```

Orientações

- Veja aqui a página de submissão da tarefa.
- O arquivo a ser submetido deve se chamar lab12.py.
- No link "Arquivos auxiliares" há um arquivo compactado (aux12.zip) que contém todos os arquivos de testes abertos (entradas e saídas esperadas).
- O laboratório é composto de 10 testes abertos e 10 testes fechados.
- O limite máximo será de 20 submissões.
- Acesse o sistema SuSy com seu RA (apenas números) e a senha que você utiliza para fazer acesso ao sistema da DAC.
- Você deve seguir as instruções de submissão descritas no enunciado.

- Serão considerados apenas os resultados da última submissão.
- Esta tarefa tem peso 3.
- O prazo final para submissão é dia 21/11/2021 (domingo).