Documentação do Desafio de Programação Challenge Meteor

Vinicius de Almeida Lima

Contents

1 Introdução

Nesta documentação, apresentamos o processo completo desde o brainstorming até a implementação e solução do desafio Challenge Meteor promovido no processo seletivo da empresa Tarken. O objetivo é detalhar os passos seguidos para resolver o problema, incluindo a concepção inicial, o desenvolvimento da solução e a codificação final.

2 Brainstorming

2.1 Identificação do Problema

O desafio proposto consiste em analisar uma imagem que contém meteoros, estrelas e um terreno possuindo regiões com água. A tarefa é contar o número de estrelas e meteoros, identificar meteoros que caem na água e descobrir uma frase escondida na imagem.

Figure 1: Imagem desafio challenge meteor

2.2 Discussão de Ideias

As ideias iniciais incluíram:

- Identificar e contar pixels de cores específicas (branco para estrelas, vermelho para meteoros, azul para água).

- Cortar a imagem na linha de água e processar as partes separadamente.

Figure 2: Imagem challenge meteor cortada abixo da linha da agua

- Transformar pixels não permitidos em preto para facilitar a análise.

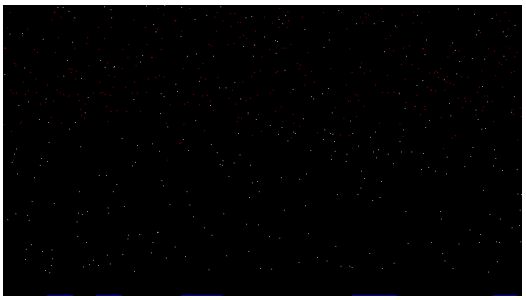

Figure 3: Imagem remoendo pixels nao permitidos para preto

- Criar uma nova imagem binária para decodificar a frase escondida.

Figure 4: Imagem reduzida para duas linhas para decodificar a frase escondida

2.3 Seleção da Melhor Abordagem

A abordagem escolhida foi dividir a tarefa em várias funções, cada uma responsável por um aspecto específico do problema. Isso inclui a contagem de estrelas e meteoros, o corte da imagem na linha de água, a transformação de pixels não permitidos em preto e a geração de uma imagem binária para decodificação da frase escondida.

3 Desenvolvimento da Solução

3.1 Estrutura do Algoritmo

O algoritmo é dividido nas seguintes etapas:

1. Contagem de estrelas e meteoros.
2. Corte da imagem abaixo da linha de água.
3. Transformação de pixels não permitidos em preto.
4. Contagem de meteoros que caem na água.
5. Criação de uma nova imagem com a presença de estrelas e meteoros em linhas separadas.
6. Conversão da nova imagem em uma string binária e decodificação para descobrir a frase escondida.

3.2 Divisão em Módulos

A solução é dividida em funções específicas:

- `contar_estrelas_e_meteoros`: Conta estrelas e meteoros na imagem.
- `cortar_abaixo_da_agua`: Corta a imagem na linha de água.
- `transformar_em_preto`: Transforma pixels não permitidos em preto.
- `contar_meteoros_na_agua`: Conta meteoros que caem na água.
- `reduzir_meteoros_e_estrelas`: Cria uma nova imagem com a presença de estrelas e meteoros em linhas separadas.
- `imagem_para_array_binario`: Converte a imagem em uma string binária.
- `binario_para_texto`: Decodifica a string binária para descobrir a frase escondida.

4 Implementação do Código

4.1 Ambiente de Desenvolvimento

O ambiente de desenvolvimento utilizado foi:

- Linguagem de programação: Python
- Bibliotecas: PIL (Python Imaging Library), numpy

4.2 Código-Fonte

```
from PIL import Image
import numpy as np

def contar_estrelas_e_meteoros(imagem_caminho):
    # Carregar a imagem
    imagem = Image.open(imagem_caminho)

    # Garantir que a imagem est  em modo RGB
    imagem = imagem.convert('RGB')

    # Inicializar contadores
    contador_estrelas = 0
    contador_meteoros = 0

    # Obter dimens es da imagem
    largura, altura = imagem.size

    # Iterar sobre todos os pixels
    for y in range(altura):
        for x in range(largura):
            r, g, b = imagem.getpixel((x, y))

            # Verificar se o pixel    branco (RGB: 255, 255,
               255)
            if r == 255 and g == 255 and b == 255:
                contador_estrelas += 1

            # Verificar se o pixel    vermelho (RGB: 255, 0,
               0)
            elif r == 255 and g == 0 and b == 0:
                contador_meteoros += 1

    # Retornar os resultados
    return contador_estrelas, contador_meteoros
```

Listing 1: Contagem de estrelas e meteoros

```
def cortar_abaixo_da_agua(imagem_caminho,
    imagem_saida_caminho):
    # Carregar a imagem
    imagem = Image.open(imagem_caminho)

    # Garantir que a imagem est  em modo RGB
    imagem = imagem.convert('RGB')

    # Obter dimens es da imagem
    largura, altura = imagem.size

    # Vari vel para armazenar a linha com o primeiro pixel
       azul
    linha_primeiro_azul = None

    # Iterar sobre todos os pixels
    for y in range(altura):
        for x in range(largura):
            r, g, b = imagem.getpixel((x, y))

            # Verificar se o pixel    azul (RGB: 0, 0, 255)
            if r == 0 and g == 0 and b == 255:
                linha_primeiro_azul = y
                break
        if linha_primeiro_azul is not None:
            break

    # Se encontramos uma linha com pixels azuis, cortamos a
       imagem
    if linha_primeiro_azul is not None:
        caixa_corte = (0, 0, largura, linha_primeiro_azul +
           1)
        imagem_cortada = imagem.crop(caixa_corte)
        imagem_cortada.save(imagem_saida_caminho)
```

Listing 2: Corte da imagem abaixo da linha de água

```
def transformar_em_preto(imagem_caminho,
    imagem_saida_caminho):
    # Cores permitidas
    cores_permitidas = [(255, 255, 255), (0, 0, 255), (255,
       0, 0)]

    # Carregar a imagem
    imagem = Image.open(imagem_caminho)

    # Garantir que a imagem est  em modo RGB
    imagem = imagem.convert('RGB')

    # Obter dimens es da imagem
```

```
    largura, altura = imagem.size

    # Iterar sobre todos os pixels
    for y in range(altura):
        for x in range(largura):
            # Obter a cor do pixel
            cor = imagem.getpixel((x, y))

            # Verificar se a cor n o est  na lista de
               cores permitidas
            if cor not in cores_permitidas:
                # Definir a cor como preto
                imagem.putpixel((x, y), (0, 0, 0))

    # Salvar a imagem
    imagem.save(imagem_saida_caminho)
```

Listing 3: Transformação de pixels não permitidos em preto

```
def contar_meteoros_na_agua(imagem_caminho):
    # Carregar a imagem
    imagem = Image.open(imagem_caminho)

    # Garantir que a imagem est  em modo RGB
    imagem = imagem.convert('RGB')

    # Obter dimens es da imagem
    largura, altura = imagem.size

    # Inicializar contador de meteoros na  gua
    meteoros_na_agua = 0

    # Encontrar a linha com pixels azuis (representando a
        gua )
    linha_azul = None
    for y in range(altura - 1, -1, -1):  # Come a do fim da
       imagem para procurar pela linha inferior
        for x in range(largura):
            r, g, b = imagem.getpixel((x, y))
            # Verificar se o pixel    azul (RGB: 0, 0, 255)
            if r == 0 and g == 0 and b == 255:
                meteoros_na_agua +=
                    contar_meteoros_na_cordenada_x(imagem_caminho,
                    x)

        if linha_azul is not None:
            break

    # Retornar o n mero de meteoros na  gua
```

```
    return meteoros_na_agua
```

Listing 4: Contagem de meteoros na água

```
def reduzir_meteoros_e_estrelas(imagem_caminho,
    imagem_saida_caminho):
    # Carregar a imagem
    imagem = Image.open(imagem_caminho)

    # Garantir que a imagem est  em modo RGB
    imagem = imagem.convert('RGB')

    # Obter dimens es da imagem
    largura, altura = imagem.size

    # Inicializar listas para indicar a presen a de
       meteoros e estrelas
    presenca_meteoros = [0] * largura
    presenca_estrelas = [0] * largura

    # Iterar sobre todos os pixels da imagem
    for x in range(largura):
        for y in range(altura):
            r, g, b = imagem.getpixel((x, y))
            # Verificar se o pixel    vermelho (meteoro)
            if r == 255 and g == 0 and b == 0:
                presenca_meteoros[x] = 1
            # Verificar se o pixel    branco (estrela)
            elif r == 255 and g == 255 and b == 255:
                presenca_estrelas[x] = 1

    # Criar uma nova imagem com duas linhas
    nova_imagem = Image.new("RGB", (largura, 2), color=(0,
       0, 0))

    # Definir os pixels na nova imagem de acordo com a
       presen a de estrelas e meteoros
    for x in range(largura):
        # Linha 0 para estrelas
        if presenca_estrelas[x] == 1:
            nova_imagem.putpixel((x, 0), (255, 255, 255))
        # Linha 1 para meteoros
        if presenca_meteoros[x] == 1:
            nova_imagem.putpixel((x, 1), (255, 0, 0))

    # Salvar a nova imagem
    nova_imagem.save(imagem_saida_caminho)
```

Listing 5: Redução de meteoros e estrelas

```
def imagem_para_array_binario(caminho_imagem):
    # Abre a imagem
    imagem = Image.open(caminho_imagem)

    # Converte a imagem para RGB, se necess rio
    imagem = imagem.convert('RGB')

    # Obt m os dados da imagem
    dados = np.array(imagem)

    # Inicializa um array bin rio vazio com o mesmo tamanho
       da imagem
    array_binario = np.zeros((dados.shape[0],
       dados.shape[1]), dtype=int)

    # Percorre cada pixel da imagem
    for y in range(dados.shape[0]):
        for x in range(dados.shape[1]):
            r, g, b = dados[y, x]

            # Verifica se o pixel    preto (R=0, G=0, B=0)
            if r == 0 and g == 0 and b == 0:
                array_binario[y, x] = 0
            # Verifica se o pixel    vermelho (R=255, G=0,
               B=0)
            elif r == 255 and g == 0 and b == 0:
                array_binario[y, x] = 1
            elif r == 255 and g == 255 and b == 255:
                array_binario[y, x] = 1
            else:
                array_binario[y, x] = 0  # Outros casos
                   podem ser tratados como 0 ou de outra
                   forma, se necess rio

    # Retorna o array bin rio como uma string  nica
       concatenada
    return ''.join(str(e) for e in array_binario.flatten())
```

Listing 6: Conversão de imagem para array binário

```
def binario_para_texto(binario):
    if len(binario) % 8 != 0:
        raise ValueError("O comprimento da string bin ria
           deve ser um m ltiplo de 8")

    # Divide a string bin ria em peda os de 8 bits (1 byte)
    bytes_list = [binario[i:i+8] for i in range(0,
       len(binario), 8)]
```

```
    # Converte cada byte em um caractere ASCII e junta-os em
        uma string
    texto = ''.join([chr(int(byte, 2)) for byte in
        bytes_list])

    return texto
```

Listing 7: Conversão de binário para texto

5 Testes

5.1 Cenário de Teste

As funções foram testadas utilizando a imagem `meteor_challenge_01.png`. A imagem passou por cada etapa do processamento, e os resultados foram verificados.

5.2 Resultados dos Testes

- Estrelas contadas: 315
- Meteoros contados: 328
- Meteoros na água: 105
- Frase escondida: "It's not about how hard you hit. It's about how hard you can get hit and keep moving forward. How much you can take and keep moving forward" - Rocky Balboa/Sylvester Stallone

6 Conclusão

Este documento detalhou a abordagem para resolver um desafio de programação envolvendo processamento de imagens. As funções desenvolvidas permitiram contar estrelas e meteoros, cortar a imagem na linha de água, transformar pixels não permitidos em preto, contar meteoros na água, e decodificar uma frase escondida na imagem. A modularização do código facilitou a implementação e teste de cada etapa do processo.