1. 为什么一般情况下对离散图象的直方图均衡并不能产生完全平坦的直方图?

- 离散图像的灰度值通常是整数（例如8位图像的0-255），而直方图均衡的映射结果需要四舍五入到最近的整数。这种**量化误差**会导致多个原始灰度级被映射到同一输出灰度级，造成部分灰度级的像素数量堆积，而其他灰度级可能空缺。
- 假设图像总像素数为*N*，灰度级数为*L*，理想情况下每个灰度级应有*N*/*L*个像素。但*N*未必能被*L*整除，导致某些灰度级多1个像素，另一些少1个像素。
- 若某些灰度级在原始图像中完全没有像素（直方图出现“空洞”），直方图均衡无法凭空创造这些灰度级的像素，导致输出直方图仍存在不连续区域。

---

2. 设已知直方图均衡化技术对一幅图象进行了增强，试证明再用这个方法对所得结果增强并不会改变其结果*

对于原始图像灰度级 $r \in [0, L-1]$，其概率密度函数为 $p_r(r)$，累积分布函数CDF为：

$$\mathrm{CDF}_r(r) = \int_0^r p_r(w)\, dw. \tag{1}$$

直方图均衡化的变换函数为：

$$s = T(r) = (L-1) \cdot \mathrm{CDF}_r(r). \tag{2}$$

通过变量替换公式，均衡化后图像的概率密度函数 $p_s(s)$ 满足：

$$p_s(s) = p_r(r) \cdot \left| \frac{dr}{ds} \right|. \tag{3}$$

由于 $\frac{ds}{dr} = (L-1) \cdot p_r(r)$，代入得：

$$p_s(s) = \frac{p_r(r)}{(L-1) \cdot p_r(r)} = \frac{1}{L-1}. \tag{4}$$

这表明均衡化后的图像灰度级 (s) **服从均匀分布**，即 $p_s(s) = \frac{1}{L-1}$。

若以 s 作为新的输入，其CDF为：

$$\mathrm{CDF}_s(s) = \int_0^s p_s(t)\, dt = \int_0^s \frac{1}{L-1}\, dt = \frac{s}{L-1}. \tag{5}$$

再次应用直方图均衡化的变换函数为：

$$s' = T'(s) = (L-1) \cdot \mathrm{CDF}_s(s) = (L-1) \cdot \frac{s}{L-1} = s. \tag{6}$$

因此，第二次变换后的灰度级 (s') **恒等于原灰度级** (s)，图像未被改变。

---

3. 讨论用于空间滤波的平滑滤波器和锐化滤波器的相同点、不同点以及联系。

相同点：

1. 两者均属于空间域滤波技术，直接作用于图像的像素矩阵，通过邻域操作（如卷积）实现对图像局部特征的增强或抑制

2. 均通过调整图像中不同频率分量的强度来达到处理效果：平滑滤波器抑制高频分量（如噪声和边缘），锐化滤波器增强高频分量（如边缘和细节）
3. 均依赖特定的滤波核（模板）进行像素值的加权计算，例如均值滤波核（线性）或拉普拉斯核（非线性）

不同点：

1. **核心目标**
    - 平滑滤波器：以去噪和模糊为主要目的，通过抑制高频噪声（如高斯噪声、椒盐噪声）来提升图像整体平滑度，常用于预处理阶段
    - 锐化滤波器：以增强边缘和细节为目标，通过突出像素值的突变（如物体轮廓）来提升图像清晰度，常用于后处理阶段
2. **频率响应特性**
    - 平滑滤波器：属于低通滤波器，保留低频分量（如大范围结构），削弱高频分量
    - 锐化滤波器：属于高通滤波器，保留高频分量（如细节），削弱低频分量

联系：

两者的效果互为补充：从原图中减去平滑滤波后的结果可得到锐化所需的高频分；反之，原图减去锐化结果可恢复平滑效果。

---

4. 有一种常用的图象增强技术是将高频增强和直方图均衡化结合起来以达到使边缘锐化的反差增强效果，以上两个操作的先后次序对增强结果有影响吗？为什么？

两者顺序对增强结果有显著影响，必须先进行高频增强，再执行直方图均衡化。若先进行直方图均衡化，可能导致图像亮度分布两极分化（例如暗区像素被漂白）。此时再叠加高频增强，会因亮度对比失衡导致边缘不突出，甚至放大噪声。

5. 编程实现对lena.bmp分别加入高斯噪声和椒盐噪声，再进行局域平均和中值滤波。

```
# ---------------------- 添加噪声 ----------------------
def add_gaussian_noise(img, mean=0, sigma=25):
    """添加高斯噪声[4,5](@ref)"""
    noise = np.random.normal(mean, sigma, img.shape).astype(np.int16)
    noisy_img = np.clip(img + noise, 0, 255).astype(np.uint8)
    return noisy_img

def add_salt_pepper_noise(img, prob=0.05):
    """快速椒盐噪声[2,3](@ref)"""
    mask = np.random.choice([0, 255], size=img.shape, p=[1-prob, prob])
    salt_mask = (mask == 255)
    pepper_mask = (mask == 0)
    noisy_img = img.copy()
    noisy_img[salt_mask] = 255
    noisy_img[pepper_mask] = 0
    return noisy_img
# ---------------------- 添加噪声 ----------------------
def add_gaussian_noise(img, mean=0, sigma=25):
    """添加高斯噪声[4,5](@ref)"""
```

```
    noise = np.random.normal(mean, sigma, img.shape).astype(np.int16)
    noisy_img = np.clip(img + noise, 0, 255).astype(np.uint8)
    return noisy_img

def add_salt_pepper_noise(img, prob=0.05):
    """快速椒盐噪声[2,3](@ref)"""
    mask = np.random.choice([0, 255], size=img.shape, p=[1-prob, prob])
    salt_mask = (mask == 255)
    pepper_mask = (mask == 0)
    noisy_img = img.copy()
    noisy_img[salt_mask] = 255
    noisy_img[pepper_mask] = 0
    return noisy_img
```