Assignment4

截止时间：2023.5.31 23：59

提交内容：文件压缩包(学号_姓名_Assignment4)，内含一份 pdf 文件(学号_姓名.pdf)，两个文件夹 code1, code2(文件夹内容见题)

提交邮箱：zheng_pengyu@163.com(邮件主题：**学号_姓名_Assignment4**;若未收到自动回复邮件，请重新发送；重复提交以最后一次提交为准)

1. **（10 分）最短路径数量** 国际象棋中的车可以水平或竖直移到棋盘中同行或同列的任何一格。将车从棋盘的一角移到另一对角，有多少条最短路径？路径的长度由车所经过的方格数（包括第一格和最后一格）来度量。使用下列方法求解该问题。
   a) 动态规划算法
   b) 基本排列组合

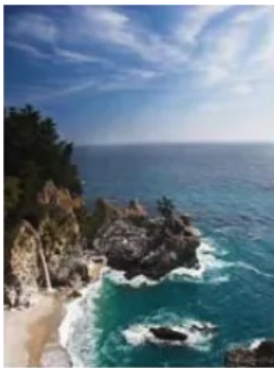
2. **（10 分）**a.对下面的数据构造一套哈夫曼编码：

| 字符 | A | B | C | D | - |
|---|---|---|---|---|---|
| 出现概率 | 0.4 | 0.1 | 0.2 | 0.15 | 0.15 |

   b.用 a 中的编码对文本 ABACABAD 进行编码。
   c.对于 100010111001010 用 a 中的编码进行解码。

**编程题（80 分）**

1. **（40 分）**Seam Carving 算法是一种处理图像缩放的算法，可以尽可能的保持图像中的“重要区域”的比例，避免由于直接缩放造成的“失真”，效果如下图：

算法的原理是计算每个像素点的梯度（可以将图像视为离散型的函数，此时在 x 方向上的梯度可以简化为相邻像素点的差值，y 方向同理，该像素点的梯度为了简便运算，可以使用 x 和 y 方向的梯度绝对值之和代替，当然这只是计算梯度的一种方式，在 OpenCV 中定义了多种算子进行梯度计算，如 Sobel 算子，具体的实现可以参考已实现的算法）作为该像素点的“能量”。然后在对应方向上寻找能量最小的一条缝隙（路径），进行删除。

缩放算法的步骤可以总结如下：

1. 计算像素点能量；
2. 找到相应方向上能量最小的路径，称为 seam；
3. 移除 seam，得到新图像；
4. 重复执行步骤 1 到步骤 3，n 次，得到缩放后的图像。

其中，查找最小能量路径的问题可以看作是最短路径问题，将每个像素视为一个节点，像素（i, j）与上方（i-1, j-1）(i-1, j)(i-1, j+1)和下方（i+1, j-1）(i+1, j)(i+1, j+1)相连通，此时问题就转换为求从第一行像素出发到最后一行像素的能量最短路径。可以使用动态规划方法进行高效求解。Seam Carving 算法还可以进行图像放大，目标保护

和去除等应用，感兴趣的同学可以自行拓展。

要求：实现 Seam Carving 算法的图像缩放功能（其他功能不做加分处理）并横向缩放一张图片为原图的二分之一，编程语言不限。除寻找能量最小路径的算法需要自行完成外，其余可以参考已实现的算法。

提交内容：源码；说明文档（包含 Usage，寻找能量最小路径算法的时间复杂度和空间复杂度）；缩放前和缩放后的图像(任意)；不需要可执行文件；打包成文件夹命名为 code1

参考链接：https://dl.acm.org/doi/10.1145/1276377.1276390（论文）
https://github.com/vivianhylee/seam-carving (pyhton 实现)
https://github.com/krthk/SeamCarving (C++实现)
https://github.com/aidan-n/seam-carving (java 实现)

2. **（40 分）**在求解复杂最优化问题，尤其是非凸函数时，常常无法通过直接计算导数为 0 来计算出最值点，此时，经常会用到梯度下降法来代替，梯度下降法的思想是通过将一步计算分解成多步计算，不断迭代进行近似求解。具体方法为设置一个初始位置$x_0$，在当前位置选择下一步的迭代方向和步长，移动到新的位置$x_1$,继续在当前位置选择下一步的迭代方向和步长，重复迭代 n 次，得到近似最优解$x_n$。显而易见，迭代方向和步长的选择是影响梯度下降法效果的重要因素。对于这一问题，梯度下降法选择的是贪心算法的思想，即找到当前位置下降最快的方向作为下一步的迭代方向。对于可微函数而言，梯度方向的负方向就是当前点下降最快的方向，正是基于这一原因，将其命名为梯度下降法。梯度下降法的迭代公式为$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - t\nabla f(\boldsymbol{x})$，其中$\nabla f(\boldsymbol{x})$是当前点的梯度，$t$是步长(学习率)，迭代预先设定的迭代次数 n 后，就得到了算法的解，当然，一般也会设定一个阈值$\eta$，当迭代前后$\boldsymbol{x}$的插值$|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}| < \eta$时，也会提前退出迭代。使用不同的迭代次数 n，步长$t$以及阈值$\eta$，往往也会有不同的算法效果。现在有一个无约束的最优化问题：

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$

假设初始点位于(0, 0)，请使用梯度下降法来求出该问题的最小值$f^*$和对应的$(x_1^*, x_2^*)$。精度取到小数点后 10 位。编程语言不限。

思考内容（不做加分处理）：对于凸函数而言，使用迭代方法最后会收敛到最小值点，而对于非凸函数，一定会收敛到最小值点吗？初始点的设置对于非凸函数极值点的求解有多大影响？一般有哪些初始点设置方法？步长又有多少影响？步长要怎样设置？梯度下降法的缺点有哪些？针对这些缺点，有哪些改进的迭代方法？

提交内容：源码；运行结果截图；思考内容(选做)；打包成文件夹命名为 code2
运行结果截图实例：

```
x1*= 1 x2*= 1
f*= 1
```