**Seam Carving算法说明**

在得到图片像素点能量值的二维数组energy\_map之后，我们需要每次在竖直方向上寻找一条能量总和最短的路径seam然后删除，循环直至图片缩放到指定尺寸。下面说明用动态规划算法寻找seam的方法。

下图是一个energy\_map示例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 4 | 5 | 9 |
| 8 | 4 | 9 | 8 | 4 |
| 4 | 3 | 6 | 3 | 5 |
| 1 | 10 | 1 | 10 | 5 |
| 8 | 5 | 2 | 1 | 6 |

**算法思路：**

可以将每一行看作是一段，将能量值看成边长，寻找seam的问题就转成从第一行到最后一行的多源最短路径问题。为此构建二维数组dp，dp[i][j]表示到达该位置最小的能量总和。由于(i, j)只能由(i - 1, j - 1)、(i - 1, j)、(i - 1, j + 1)三个位置到达，因此到(i, j)的最小能量总和就是：到三个位置的最小能量总和分别加上(i, j)处的能量，再取最小值。而显然到第一行所有位置的最小能量总和就是他们自己的能量，因此得到递推公式：



为了记录取到最小能量总和时是从哪个方向下来的，以便后续寻找seam的具体坐标，构建二维数组k。k[i][j]表示到达位置(i + 1, j)时，在最小的能量总和下，上一个位置在哪里。由于上一个位置只能有左上、正上、右上三种情况，因此k[i][j]只能有-1、0、1三个值，对应三种情况。每次填上dp[i][j]时，看看最小能量总和来自哪个方向并记录在k[i - 1][j]中。由于第一行没有上一个位置，因此k只有i - 1行。

下面对示例进行求解：

初始化dp和k：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 4 | 5 | 9 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

下面对(1, 0)位置进行填表：由于在第一列，因此只有正上和右上两个情况。将这两个位置加上(1, 0)位置自己的能量值并取最小值。4+8=12，3+8=11，最小能量和为11且来自右上，因此dp[1][0] = 11，k[0][0] = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 4 | 5 | 9 |
| 11 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

根据以上规则，从左到右、从上到下填完两表得到：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 4 | 5 | 9 |
| 11 | 7 | 12 | 12 | 9 |
| 11 | 10 | 13 | 12 | 14 |
| 11 | 20 | 11 | 22 | 17 |
| 19 | 16 | 13 | 12 | 23 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | -1 | -1 | -1 |
| 1 | 0 | -1 | 1 | 0 |
| 1 | 0 | -1 | 0 | -1 |
| 0 | -1 | 0 | -1 | 0 |

接下来检查最后一行，发现12最小，因此以位置(4, 3)为终点的seam可以达到最小能量总和，因此从这里开始回溯寻找seam。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 4 | 5 | 9 |
| 8 | 4 | 9 | 8 | 4 |
| 4 | 3 | 6 | 3 | 5 |
| 1 | 10 | 1 | 10 | 5 |
| 8 | 5 | 2 | 1 | 6 |

回溯方法：回溯到(i, j)时，看k[i - 1][j]。若为-1，则上一个位置在左上；若为0，则上一个位置在正上；若为1，则上一个位置在右上。例如终点是(4, 3)，则查看k[3][3]，为-1。则上一个位置在左上，是(3, 2)。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 4 | 5 | 9 |
| 11 | 7 | 12 | 12 | 9 |
| 11 | 10 | 13 | 12 | 14 |
| 11 | 20 | 11 | 22 | 17 |
| 19 | 16 | 13 | 12 | 23 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 4 | 5 | 9 |
| 8 | 4 | 9 | 8 | 4 |
| 4 | 3 | 6 | 3 | 5 |
| 1 | 10 | 1 | 10 | 5 |
| 8 | 5 | 2 | 1 | 6 |

循环直到找到第一行：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 4 | 5 | 9 |
| 8 | 4 | 9 | 8 | 4 |
| 4 | 3 | 6 | 3 | 5 |
| 1 | 10 | 1 | 10 | 5 |
| 8 | 5 | 2 | 1 | 6 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 4 | 5 | 9 |
| 11 | 7 | 12 | 12 | 9 |
| 11 | 10 | 13 | 12 | 14 |
| 11 | 20 | 11 | 22 | 17 |
| 19 | 16 | 13 | 12 | 23 |

这就是能量总和最小的seam，将其删除得到：

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 4 | 5 | 9 |
| 8 | 9 | 8 | 4 |
| 4 | 6 | 3 | 5 |
| 1 | 10 | 10 | 5 |
| 8 | 5 | 2 | 6 |

重复以上过程直到图像缩放到指定尺寸即可。

**算法分析：**

设原始图像宽高分别为w、h。总共要经过w/2轮，第i轮填一个h\*(w - i)和(h - 1)\*(w - i)的表，加起来是：



由于在第i轮需要一个h\*(w - i)和(h - 1)\*(w - i)的表，需要的空间。

综上：**时间复杂度：** **空间复杂度：**

**使用方法：**

文件结构按以下方式布局：

├── in（自行创建的文件夹）

│   └── image.png（待处理的图片）

│

├── out（自行创建的文件夹，输出图片将会在这里生成）

│

├── main.py

└── seam\_carving.py

保证已经创建了in和out文件夹，且待处理的图片的文件名和main中input\_filename一致的情况下，运行main.py即可。