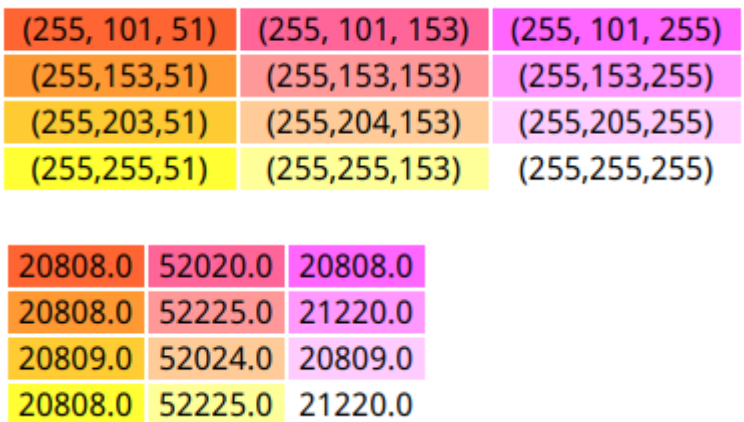
程序运行说明：

- 编译源代码的时候注意包名，是`seamcarving`
- 程序运行时候从命令行（终端）接受输入参数
  - 如果是`IDE`运行就从`Run Configuration`输入参数
  - 直接运行程序会报错（因为你没有指定图片）
- 图片的位置要求和程序的工作路径在同一目录下（即bin）
  - 如果是`IDE`运行则在`Project`根目录下
    - 如果`IDE`的配置文件对工作目录有修改，服从配置文件的修改
- 支持常见图片格式（`png`、`jpg`等）
- 区分大小写

程序运行效果图：

能量密度的正确性

能量密度的计算结果与样例完全一致

> 样例
>
> 
>

> 我的计算结果
>
> ```
> 20808.0 52020.0 20808.0
> 20808.0 52225.0 21220.0
> 20809.0 52024.0 20809.0
> 20808.0 52225.0 21220.0
> ```

最短路径的正确性

最短路径的计算与样例完全一致

> 样例

(78,209, 79)	(63,118,247)	(92,175, 95)	(243, 73,183)	(210,109,104)	(252,101,119)
(224,191,182)	(108, 89, 82)	(80,196,230)	(112,156,180)	(176,178,120)	(142,151,142)
(117,189,149)	(171,231,153)	(149,164,168)	(107,119, 71)	(120,105,138)	(163,174,196)
(163,222,132)	(187,117,183)	(92,145, 69)	(158,143, 79)	(220, 75,222)	(189, 73,214)
(211,120,173)	(188,218,244)	(214,103, 68)	(163,166,246)	(79,125,246)	(211,201, 98)

The corresponding pixel energies are shown below, with a minimum energy vertical seam highlighted in pink. In this case, the method `findVerticalSeam()` returns the array { 3, 4, 3, 2, 2 }.

57685.0	50893.0	91370.0	25418.0	33055.0	37246.0
15421.0	56334.0	22808.0	54796.0	11641.0	25496.0
12344.0	19236.0	52030.0	17708.0	44735.0	20663.0
17074.0	23678.0	30279.0	80663.0	37831.0	45595.0
32337.0	30796.0	4909.0	73334.0	40613.0	36556.0

我的计算结果

```
57685.0 50893.0 91370.0 25418.0 33055.0 37246.0
15421.0 56334.0 22808.0 54796.0 11641.0 25496.0
12344.0 19236.0 52030.0 17708.0 44735.0 20663.0
17074.0 23678.0 30279.0 80663.0 37831.0 45595.0
32337.0 30796.0 4909.0 73334.0 40613.0 36556.0
[3, 4, 3, 2, 2]
```

输出图片的正确性

输出图片完成裁剪的目的，且保留了图片最重要的部分

宽度压缩1%、高度压缩2%

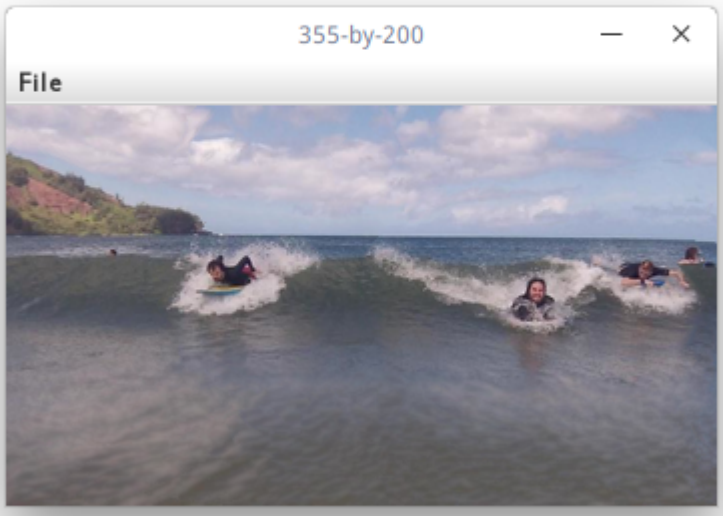
用时492毫秒

```
Cutting 1 and 2 percent.
Begin:Sat Nov 11 19:52:28 CST 2017
Done:Sat Nov 11 19:52:29 CST 2017
Finish in 492ms.
```

宽度压缩30%、高度压缩30%

用时5244毫秒

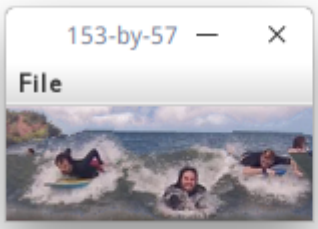
```
Cutting 30 and 30 percent.
Begin:Sat Nov 11 19:49:55 CST 2017
Done:Sat Nov 11 19:50:01 CST 2017
Finish in 5244ms.
```



宽度压缩70%、高度压缩80%

用时6625毫秒

```
Cutting 70 and 80 percent.
Begin:Sat Nov 11 19:51:46 CST 2017
Done:Sat Nov 11 19:51:53 CST 2017
Finish in 6625ms.
```



必要的说明

从上面的结果可以看出，不管裁剪比例是多少，最终都能保留最重要的部分（即中间冲浪的3个人）

但是很明显的是，裁剪掉的越多，程序运行所用的时间越长

程序运行的时间不仅与图片本身的长宽有关

还与裁剪量的多少有关

如果只裁剪1条个像素，一秒钟就好了

但是如果要裁剪掉一半（例如本例的裁剪70%，裁剪掉了300多条像素），花费就要几秒，甚至十几秒

这个原因有两个

一个是因为每次都要算一下新的最短路径

相当于每次都要遍历一次二维数组，求能量密度、求最短路径

另一个原因是每次裁剪一条像素

等于是后面的像素平移一个单位，并且改变图片本身的大小

这个也是占用内存、耗费时间的

更多的测试图片

原始图片

宽度裁剪20%、高度裁剪40%

```
Cutting 20 and 40 percent.
Begin:Sat Nov 11 20:26:20 CST 2017
Done:Sat Nov 11 20:26:51 CST 2017
Finish in 31166ms.
```

从结果可以看出，“华东师范大学”上下多余的空白，导航栏上下的高度、以及图片中的天空，都被认为是不重要的东西，被裁剪了。

而最重要的是草地上的一群学生，仍旧被保留了。

但是由于图片本身的原因，也可能是我的裁剪要求设置太高了（高度直接去掉了一半），所以导致有些地方有点变形。

不过无所谓了。

同时可以看到，程序运行时间已经长达半分钟。