****

**计算机视觉课程**

**实验2——Graph Cut算法实验报告**



实验时间：2022年3月26日-2022年4月4日

**学 院 智能与计算学部**

**专 业 计算机与科学技术**

**学 号 3019244140**

**姓 名 郭思齐**

# 一、Graph Cut算法实验

# 实验目标

本次实验主要围绕Graph Cut分割算法进行。

1. 熟练掌握Graph Cut分割算法原理及代码实现，在理解并掌握Graph Cut分割算法原理的基础上，能够对已实现算法进行灵活调用，通过调用Graph Maker类及部分函数、编写代码实现对图像分割，获取分割结果。
2. 熟练掌握图像分割的各种评价指标，例如：dice、hd95、sensitivity、specificity、accuracy等，对分割结果可视化的同时，针对分割结果和其对应的Ground Truth图像进行**评价指标的计算。**
3. 最后对指标结果进行统计分析；撰写实验报告。
4. 学习U-Net网络结构及标注信息，加深对U-Net的理解，估算《模型参数量及计算量FLOPs计算（以U-Net为例）》中BCDE操作的参数量和计算量FLOPs.

# 2. 实验一 Graph Cut 算法分割图像

## 2.1 算法实现说明

Graph Cut代码结构说明：

在整理算法原理之前，先简单地对老师提供的代码框架做以下梳理：

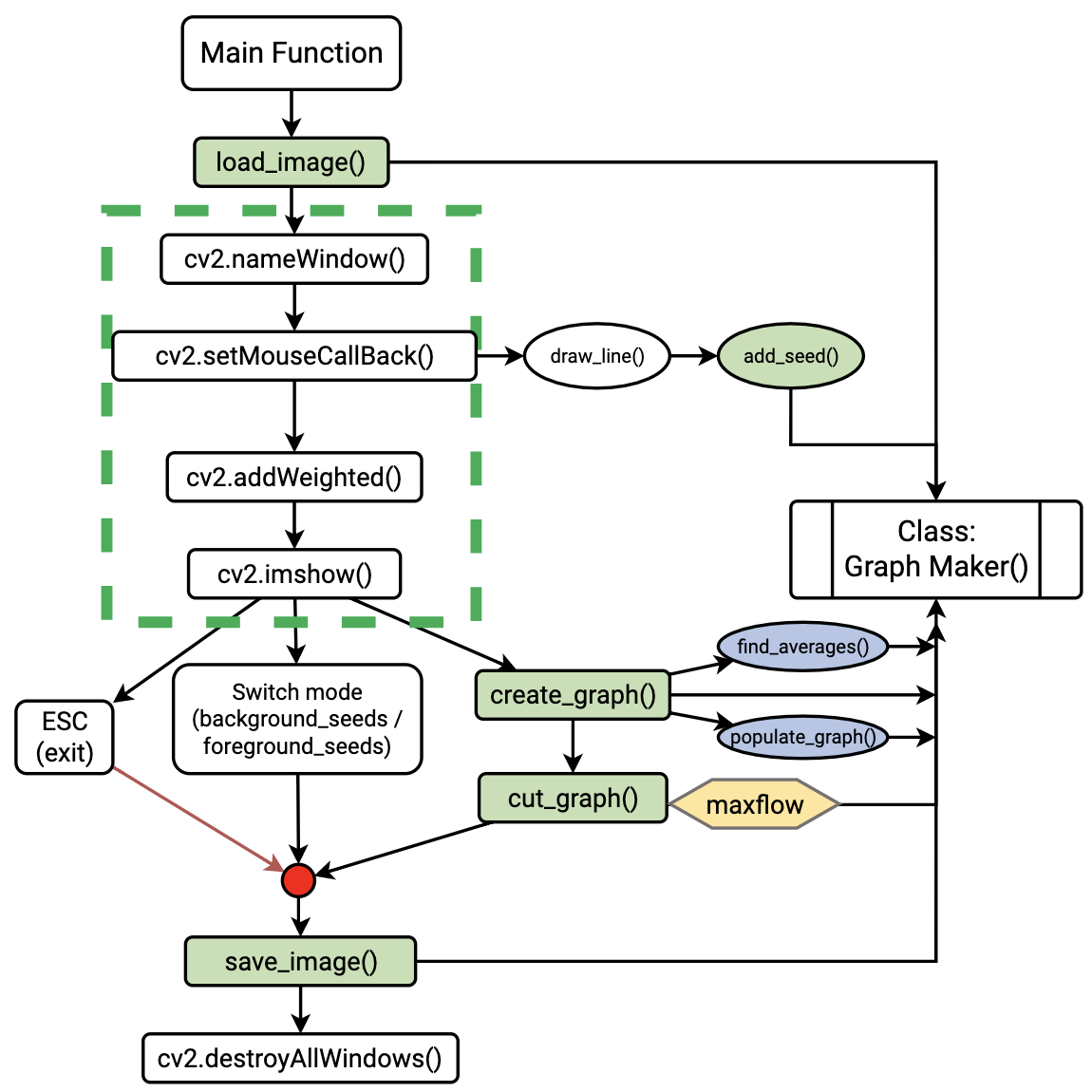


图 1 代码实现部分，算法整体的框架（自己嵌入了交互取SEED的部分）

Graph Cut算法：

在面对图像处理领域中前后背景分割、抠图、立体视觉等应用，Graph Cut是一种非常流行的算法；此类算法把图像分割问题与图的最小割问题关联起来。

**总的来说，**Graph Cut在普通的图 的基础上，添设了两个终端顶点（source：源点）和 （sink：汇点），其他所有的顶点都必须和这两个顶点相连形成编辑和中的一部分；使得图拥有两种顶点，两种边。

第一种：普通顶点对应于图像中的每个像素。每两个邻域顶点（对应于图像中每两个邻域像素）的连接就是一条边。这种边也叫 。

第二种：除图像像素外，另外两个终端顶点和 ；每个普通顶点和这2个终端顶点之间都有连接，组成第二种边。这种边也叫 。

下图为Graph Cut的 示意图。

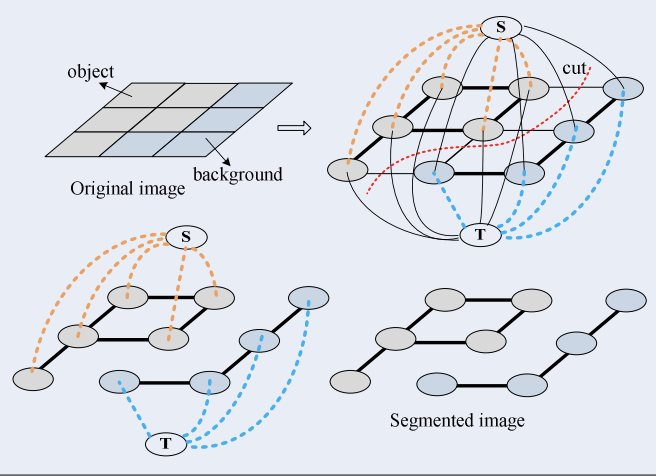
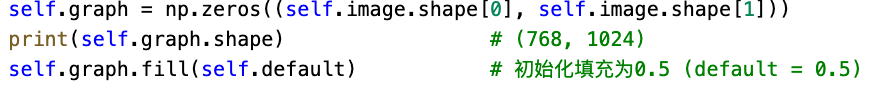


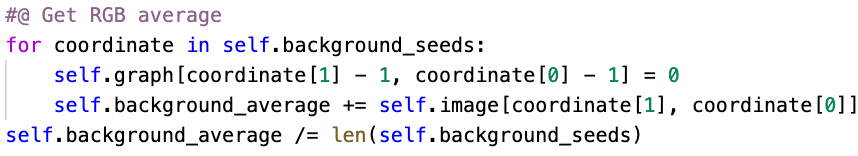
图 2选自 [IEEE Xplore Full-Text PDF](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6223428) 每个像素对应图中一个相应的顶点

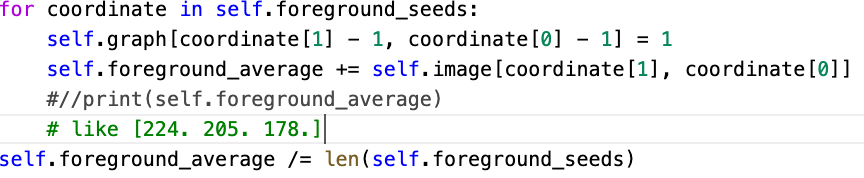
这一部分，在具体实现中主要是由create\_graph()来实现的：

create\_graph()主要分为了两部分，**首先**在确定已经人为的取得了分别属于background集合和foreground集合的seed之后，find\_average() 对graph进行了赋值:

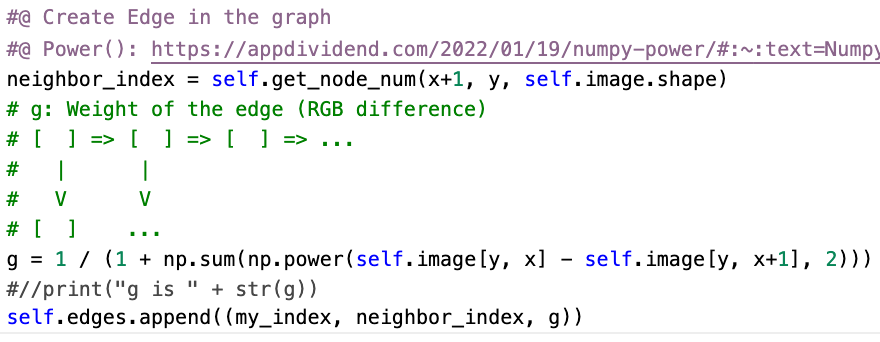


并且计算了两个种子集合中各自的RGB均值。



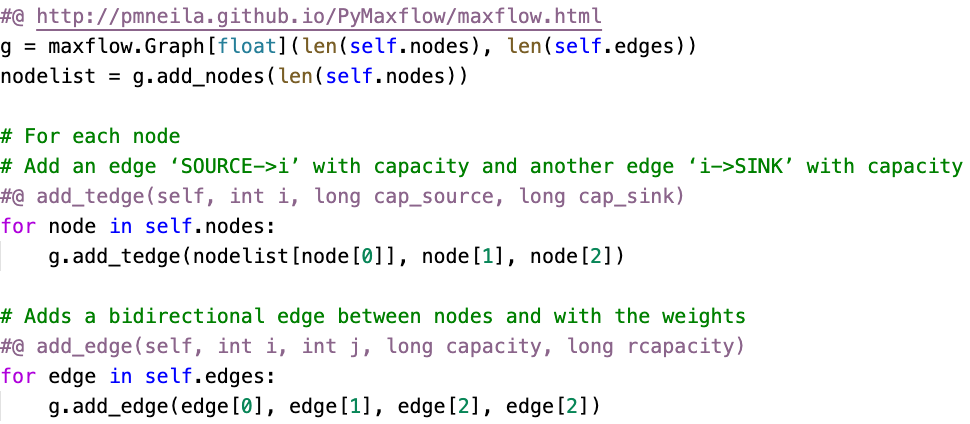


**第二部分populate\_graph()** 为每个seed像素点设置入度出度，以及 示意图中的，边的权重。根据图2参考论文，最终cut的消耗可以有以下公式表示：

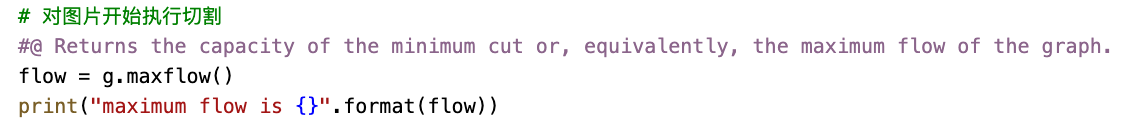


**对于创建好的 图，cut\_graph() 进行切割操作**， 同样核心由两部分：

调用maxflow函数库创建Graph的以及源点和汇点；



以及之后的切割操作：

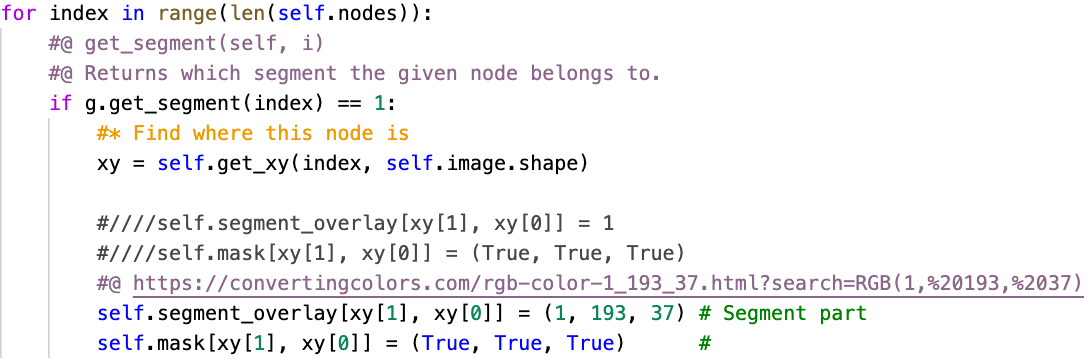


涉及到maxflow**寻找图的最小割**也可以转化为求其最大流。在图像切割利用最大流最小割的过程中，定义了这样的能量方程：

希望最小化左式， 是关于区域信息， 是关于边界信息，具体表达公式可以参考论文，有对其进一步解释。

在本次实验的具体实现中，分配好图的边容量等信息后，直接调用了maxflow()函数计算图的最大流，使用get\_segment () 来获取每一个像素点到底属于背景分割块还是属于切割出来的分割快。

得到的结果可以确定出哪些点属于切割部分，哪些点属于背景部分；最后利用可以利用掩码来记录那些且各部分的像素位置：



## 2.2 结果比较及分析

分割结果可视化：（结果分析见3. 2）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 原图 | mask | 分割结果 | |
| 7 |  |  | |  |
| 14 |  |  | |  |
| 16 |  |  | |  |

# 3. 实验二 分割指标计算

调用metric包，对每一个结果图计算以下分割指标：

表格 1 分割指标计算结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 评测指标 | **sensitivity** | **specificity** | **accuracy** | **dice** | **hd** | **hd95** |
| out7.png | 0.851513 | 0.999433 | 0.998773 | 0.860978 | 16.970562 | 5.656854 |
| out14.png | 0.980613 | 0.968755 | 0.973028 | 0.963233 | 72.006944 | 1.0 |
| out16.png | 0.945312 | 0.999264 | 0.97348 | 0.97149 | 39.293765 | 0.0 |

## 3.1 分割指标理解

分割指标的含义及理解：

TP: 被判定为正样本，判定正确。 TN: 被判定为负样本，判定正确。

FP: 被判定为正样本，判定错误。 FN: 被判定为负样本，判定错误。

**Sensitivity：**

**Specificity:**

**Accuracy:**

**Dice:**

Dice相似系数是一种集合相似度度量指标，通常用于比较计算两个样本的相似度，致的反胃0-1，分割结果越好，越近于1。

**hd95:**

hausdorff distance是量度度量空间中真子集之间的距离，对分割出的边界比较敏感，计算Hausdorff\_95（hd95）目的是为了消除离群值的一个非常小的子集的影响。

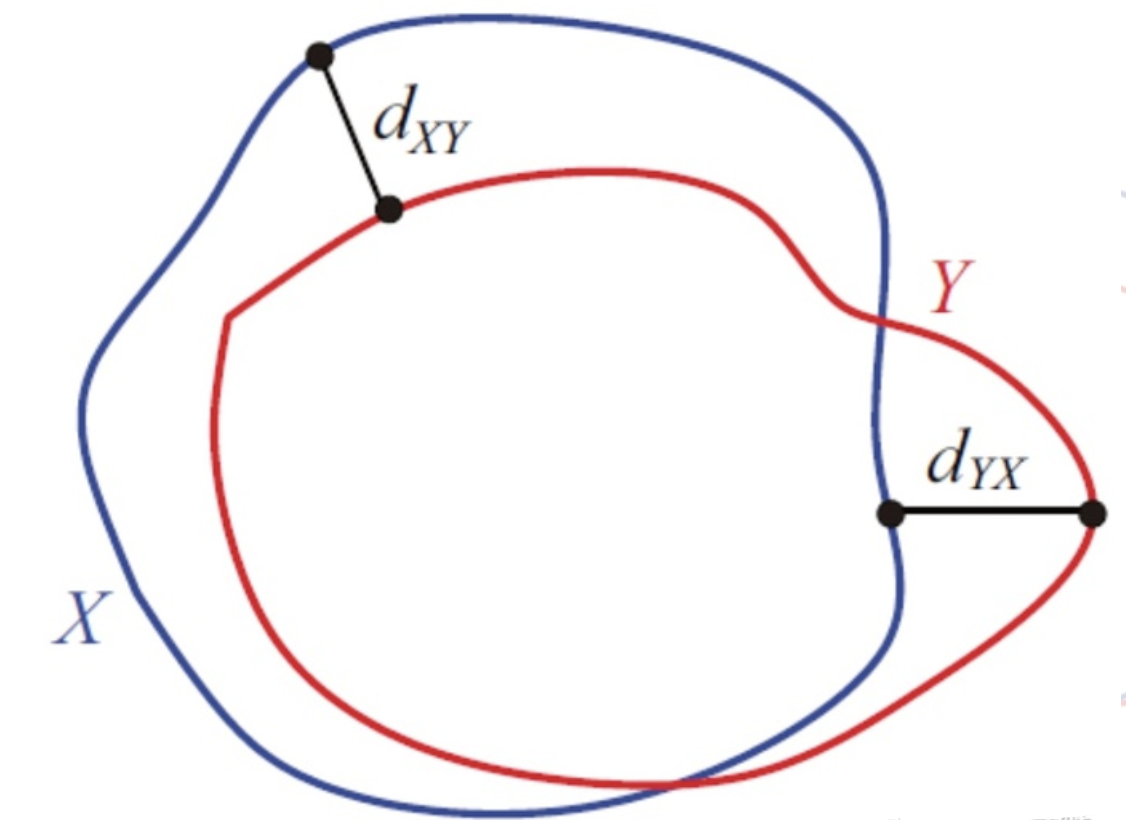
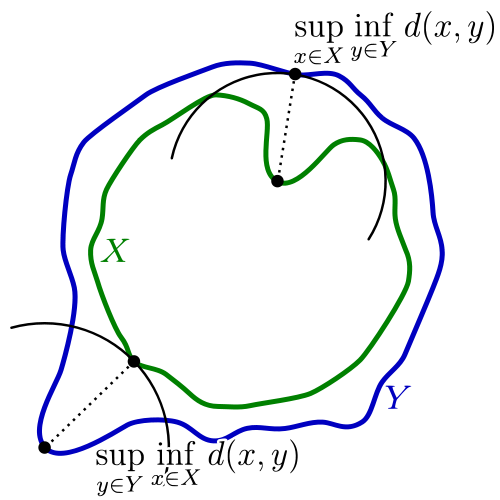
 

图 3 HD示意图 ([Hausdorff 距离 - 简书 (jianshu.com)](https://www.jianshu.com/p/24ec42a61ace) )

## 3.2 结果分析

最后对指标和分割结果进行分析：

这样的结果表现出分割效果比较不错。可以看到第一张图7号的sensitivity和dice比较低，不难想象，因为第一张图想要切割的物体切割面积比较小，难以把控。

但是总体而言sensitivity、accuracy、specificity和dice都是比较理想的。接近于一。至于hausdorff distance和hd95指标一般情况是为了比较边缘切割的效果，hd95在hausdorff distance的基础上消除离群值的一个非常小的子集的影响，可以看到实验结果第一个大约在5左右，后两个都趋近于0，因此总的结果不错；第一个还是因为切割物体较小，像素也比较集中，效果略差。

我发现选取的seed其实对结果也有很大的影响，因为我使用了交互式采点的设计，因此尝试了很多组seed点集合。记录下了他们的平均值，统计在前面表格中。seed点需要采得足够多，尽量均匀的分布在前景（待切割出的物体）和背景中。在点数不够多、分布不够均匀等情况下、最后的分割结果可能不够理想、不够完整。

# 二、 U-Net网络相关参数计算

图示

描述已自动生成

B：Conv(3,64,kernel\_size=(3,3))+Relu(\*)+Norm(64)

B：Conv(64,128,kernel\_size=(3,3))+Relu(\*)+Norm(128)+maxpooling(kernel\_size=(2,2), stride=2)

C：Conv(128,256,kernel\_size=(3,3))+Relu(\*)+Norm(256)+maxpooling(kernel\_size=(2,2), stride=2)

D：Conv(256,512,kernel\_size=(3,3))+Relu(\*)+Norm(512)+maxpooling(kernel\_size=(2,2), stride=2)

E：Conv(512,1024,kernel\_size=(3,3))+Relu(\*)+Norm(1024)+maxpooling(kernel\_size=(2,2), stride=2)

**1. 估算参数量**

根据给出U-Net网络结构及标注信息，计算U-Net encoder部分 B、C、D、E操作各自的参数量。

对B而言，Conv 输入为64通道，输出为128通道，采用3 \* 3的kernel，默认stride=1，Norm(128)参数量128；因此B操作总参数量64 \* 3 \* 3 \* 128 + 128 = 73856

对C而言，Conv 输入为128通道，输出为256通道，采用3 \* 3的kernel，Norm(256)参数量256；因此B操作总参数量128 \* 3 \* 3 \* 256 + 256 = 295168

对D而言，Conv 输入为256通道，输出为512通道，采用3 \* 3的kernel，Norm(512)参数量512；因此B操作总参数量256 \* 3 \* 3 \* 512 + 512 = 1180160

对E而言，Conv 输入为512通道，输出为1024通道，采用3 \* 3的kernel，Norm(1024)参数量1024；因此B操作总参数量512 \* 3 \* 3 \* 1024 + 1024 = 4719616

1. **估算计算量（FLOPs）**

根据给出U-Net网络结构及标注信息，计算U-Net encoder 部分B、C、D、E操作各自的FLOPs。

对B而言，Conv 输入为64通道，输出为128通道，采用3 \* 3的kernel，默认stride=1，Norm(128)参数量128(输入尺寸\*1/2=输出尺寸)；因此B操作FLOPs = 128\*128 \* 64\*3\*3\*128 + 128\*128\*128 = 1210056704

对C而言，Conv 输入为128通道，输出为256通道，采用3 \* 3的kernel，默认stride=1，Norm(256)参数量256(输入尺寸\*1/4=输出尺寸)；因此B操作FLOPs = 64\*64 \* 128\*3\*3\*256 + 64\*64\*256 = 1209008128

对D而言，Conv 输入为256通道，输出为512通道，采用3 \* 3的kernel，默认stride=1，Norm(512)参数量512(输入尺寸\*1/8=输出尺寸)；因此B操作FLOPs = 32\*32 \* 256\*3\*3\*512 + 32\*32\*512 = 1208483840

对E而言，Conv 输入为512通道，输出为1024通道，采用3 \* 3的kernel，默认stride=1，Norm(1024)参数量1024(输入尺寸\*1/16=输出尺寸)；因此B操作FLOPs = 16\*16 \* 512\*3\*3\*1024 + 16\*16\*1024 = 1208221696

# 三、总结

课程实验收获：

在这次的Graph Cut计算机视觉算法实验中，我收获了很多。

一步步的分析，让我对Graph Cut的实现方法有了较为深入的认识，也明白了论文中复杂晦涩的数学公式是怎么实际应用在现实问题中的。对切割指标的理解和掌握为将来阅读分析图像切割等问题的论文做好了铺垫，让我能够认识如何评价图像切割结果。

最后对U-net的学习，让我感受到卷积网络是如何学习的，感受到可解释性的重要性。也对U-net内部有了自己的理解，剥开黑盒，深入到卷积的内部。

课程建议：

这门课程使我对CV领域中图像分割技术有了进一步认识，但是实验部分感觉颗粒性很强，如果可以的话希望能够将实验分成更多的小阶段分阶段完成，有可能的话可以亲自编写一部分算法具体的实现的函数；这样的话，应该可以让我们对算法有更深入的学习和体验，