实验六 :采集网站图像文件

6.1 实验介绍

本实验中实现对某个网站（自己拟定）中的图片进行爬取，由于网站中的图片并不是只存在于某一个单一网页，因此需要对多个网页进行爬取，将网站中的图片下载保存到本地。

6.2 实验目标

掌握非结构化数据采集

掌握 Python 的多线程编程方法

6.3 实验原理与方法

一个站点下一般有一个根页面，一般是网站首页。首页下有多个链接，通过这些链接可以跳转到其他页面。其他页面中也有许多链接，可以继续访问。如此以来，站点中有链接关系的页面可以构成一个有向图。爬虫则需要在这个有向图中的页面中穿梭，抓去有用的信息。类似图的遍历方法，我们可以使用深度优先，也可以使用广度优先。

深度优先方法需要维护一个栈，来保存待访问的 URL。该方法的思路为：

1. 将第一个 URL 入栈
2. 若栈为空，则退出结束，否则将一个 URL 出栈
3. 访问出栈的 URL，获取其中我们感兴趣的信息，并将页面中的所有链接的 URL 入栈
4. 回到第 2 步继续

广度优先方法需要维护一个队列，来保存待访问的 URL。该方法的思路为：

1. 将第一个 URL 入队列
2. 若队列为空，则退出结束，否则将一个 URL 出队列
3. 访问出队列的 URL，获取其中我们感兴趣的信息，并将页面中的所有链接的 URL 入队列
4. 回到第 2 步继续

以上方法都是在循环结构内进行，由于网站的体量难以估计，不建议使用递归方法。对于比较复杂的网站，很有可能在有向图中出现环，因此我们还需要记录一个 URL 是否已经

爬取过，避免陷入死循环。

多线程

对于图片信息，获取所用的时间比普通文本更长。如果使用单线程，就需要等待一个图片下载完，再下载下一个图片，所有图片下载完，才能进入下一个 URL，这样的效率太低，并且一个图片下载失败，会影响整个爬取过程。使用多线程可以很好地解决这些问题。

多线程的使用方法

使用多线程首先需要导入 threading 库，建立一个 Thread 对象：

```
t = threading.Thread(target, args=None)
```

其中，target 是需要多线程执行的函数，args 为函数提供参数。使用 Thread 对象的 start 方法就可以启动线程：

```
t.start()
```

使用 join 方法可以等待其他线程都执行完毕后在继续执行后续操作：

```
t.join()
```

对于多线程中的资源调配，由 threading 库中的 RLock 线程锁对象完成：

```
lock = threading.RLock()
```

该对象中有 acquire()和 release()方法。其中 acquire 方法强迫 lock 获得线程锁，release 方法释放线程锁，如果当前线程调用 acquire 方法，但在此之前已有其他线程调用过 acquire 方法，并且还没有 release，则当前线程阻塞，直到等到线程锁的控制权。在多个线程对一个公共资源进行调用时，可以使用 acquire 和 release 方法，确保同一时刻同一资源只被一个线程调用。

6.4 实验步骤

1. 确定需要爬取的网站，并浏览网站结构和网页源代码。
2. 编写爬虫脚本，对网站中的图片进行下载。
3. 撰写实验报告。

6.5 实验要求

1. 需要爬取同一站点下的多个页面。
2. 学习使用多线程技术。
3. 可以使用之前实验中学习的库。