**电工导实验报告2**

F1403023 5140309534 韩坤言

**一、实验目的**

1. **了解**基本的html协议，加深对爬虫的概念
2. 了解哈希散列，Bloomfilter 和并发编程

**二、实验内容**

1. 了解HTML表单，模拟header，模拟post，熟练掌握两种不同的爬取策略（BFS,DFS）
2. 引入时间复杂度的概念，用哈希散列使得查找的效率提高，BloomFilter实现和运用，并发编程来打打提高效率

**三、实验环境**

1. **Firefox + Firebug插件或Chrome**
2. **Python 2.7 + easy\_install + BeautifulSoup**

**四、实验步骤**

首先我们了解了HTTP协议中最重要的部分，HTTP请求，一个HTTP请求起始于用户端向HTTP服务器发送的一个URL请求，从代码的层面了解了是如何登录，发送请求的。接着我们用python模拟get的方式请求网页，有的网页只允许浏览器访问，因而还模拟了header。还有post的请求方式，可以用来模拟帐号的登录。

网络爬虫是一种按照一定的规则，自动抓取万维网信息的程序或者脚本。相应的，爬虫也有多种爬取策略，最常用的就是BFS和DFS，深搜和广搜。C++中我们接触过，所以这部分并不算很难理解。

时间复杂度的概念我们并不陌生，好的算法能使时间复杂度变小，但可能会牺牲空间，两者往往不可兼得，在电脑允许的情况下，减少时间复杂度往往能事半功倍。哈希相当于一个映射关系，让查找更加方便，高效。Bloomfilter是运用多个哈希函数，建立一个二进制的数组，通过0，1一个元素是否出现过。

并发编程非常实用，相当于多线进程，虽然会占用更多的空间，却能大大节省时间，将其运用在爬虫中，能使爬取的效率大幅缩短，令人满意。

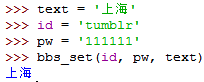
**五、问题及其解决**

**1.使用自己的账号模拟登陆BBS后，修改个人说明档（修改bbs\_set\_sample.py）**

课后练习第一题是使用自己的帐号模拟登陆BBS，并修改个人说明档。

BBS修改个人说明档的页面在 https://bbs.sjtu.edu.cn/bbsplan

其中id为用户名，pw为密码，text为说明档文本。例如：

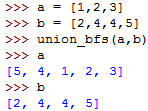


首先，得登录BBS论坛，用前面学习的post请求方式模拟登录就行了，至于就该签名档，原理其实也是通过post请求。但是我的代码能成功修改页面上的说明档，但是再一次从网页爬取下来就是乱码。折腾了半天，发现网页编码方式是gb2312的，网上了解不同的编码方式并寻找解决方案之后，把beautifulsoup后面的参数添加了一个，成为soup = BeautifulSoup(content,fromEncoding="gb18030")后就没有乱码问题了，我对编码方式有了更深的了解。

**2.修改crawler\_sample.py中的union\_bfs函数，完成BFS搜索**

def union\_bfs(a,b):

其中a，b为list，函数将b中的元素插在a之前。注意排除重复元素。 例如：



提示：list的insert操作可以将元素插在指定位置。

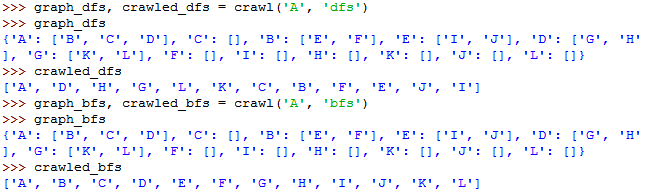
BFS，广搜，提示给到这份上了，轻松搞定。

**3.修改crawler\_sample.py中的crawl函数，返回图的结构**

graph结构与crawler\_sample.py中g的结构相同。

完成后运行graph, crawled = crawl(‘A’, ‘bfs’)

查看graph 中的图结构，以及crawled中的爬取结果顺序。



上一题是为这道题做铺垫的，而这道题又是为爬取网页做铺垫的。问题没有遇到，倒是学习了不少。globals()['union\_%s' % method](tocrawl, outlinks)这个函数运用巧妙，整个函数将bfs和dfs的两个函数的名称的字符串作为参数，通过这个函数调用，使得函数不用加if判断而显得很冗余。

**4.进一步修改函数，完成网页爬虫（修改crawler.py）**

需要修改的函数：

将练习2,3中修改的部分加入crawler.py。

def get\_all\_links(content, page):

links = []

...

return links

输入网页内容content，网页内容所在的网址page，以list形式返回网页中所有链接。建议匹配所有绝对网址和相对网址。

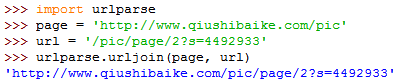
例如，匹配形如

<a target="\_blank" href="http://m.qiushibaike.com"></a>

<a href="/pic/page/2?s=4492933">2</a>

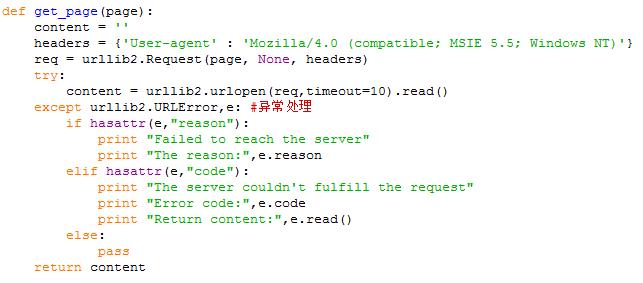
的网址。

提示： soup.findAll(‘a’,{‘href’ : re.compile(‘^http|^/’)}) 可以匹配以http开头的绝对链接和以/开头的相对链接。urljoin可以将相对链接变为绝对链接。



有了之前的铺垫，爬取网页无非就是将之前的g改成一个网址的字符串，改动相应的get\_all\_links。相对链接改为绝对链接之前已经涉及，但实际爬取会遇到很多意想不到的情况，比如有的网址错了，或者无法访问，会导致程序无法运行下去等等问题，所以添加了异常捕获，杜绝隐患。

我在get\_page处加上了如下语句：



把错误说明也加上，顺便把header也模拟了一下，之后就没有遇到程序异常终止或者卡住的情况。遇到问题，如下图所示：

C:\Users\DELL-PC\Desktop\QQ截图20151014144301.png

遇到问题也不会影响程序的正常工作，也会给出原因。

在urlopen时加超时参数timeout，防止爬取有的大型网站时花太长时间，降低工作效率。在函数的部分添加max\_page，否则会无穷无尽爬下去，这就没有必要了。

**5. 实现BloomFilter**

实现一个简单的BloomFilter。

设计一个实验统计你的BloomFilter的错误率(false positive rate)。

提示：可以用函数实现（例如hashtable里，用函数操作table的做法），也可以用类实现（例如Bitarray.py的实现，可以修改Bitarray.py完成Bloomfilter）。

hash函数文件夹中就提供了11个，直接用即可。更据哈希函数个数k、位数组大小m、加入的字符串数量n的关系 ([BloomFilters- the math](http://pages.cs.wisc.edu/~cao/papers/summary-cache/node8.html))。该文献证明了对于给定的m、n，当 k = ln(2)\* m/n 时出错的概率是最小的。 设置了适当的m,n,k，生成若干随机的字符串，并保存下来，之后再生成大量随机字符串，判断是否在之前保存过的list里，计算错误率，错误率大概在0.012%左右。可见BloomFilter的错误率并不高，就算出现误判，相比节省的大量时间，可以接受

**6.实现一个并行的爬虫**

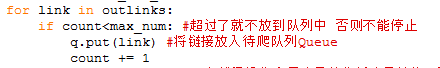
将实验二中的crawler.py改为并行化实现。

需要实现的功能：

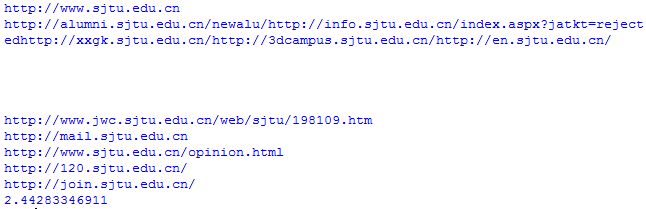
Queue初始时给入一个seed网址，从这个网站开始爬取一定数量的网页。



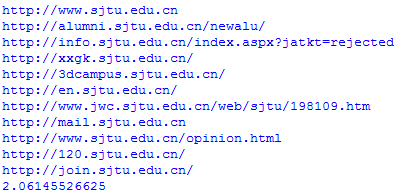
之前没有接触过并行的程序，所以这里遇到了不少问题。大部分都不用改，就是将之前的crawl()函数改为新的函数就行了。第一个问题是我不清楚如何停止。加了很多判断，函数都不停下来，我经过测试，发现终止条件是队列为空时程序才会终止，应为没有东西再去执行了，因此我加入一个全局变量的计数器，当把outlinks里的link放入queue时加一个判断，到了一定数量就不要爬取了，以此达到终止的目的。



第二个问题是格式的问题，输出会有各种空格，如下图所示



推测是并行print的时候产生冲突，顺序混乱，最后的回车也集中到了一块，于是我加了变量锁，print前将其锁住，结束后解锁，问题得到了顺利的解决。



单线程的时候爬10个网页耗时7s多，5线程只要2s左右，如果网页更多，差距相差甚远，这时并发编程的重要性就体现出来了。

**六、实验总结**

这次电工导的课程带给了我们很多，前半部分加深了我们对html和爬虫的了解，而后半部分大都和如何优化相关，大大提升了程序运行的效率。现在，我们已经能更高效地爬取很多网站了，获取信息也更为快速。我真的感到这是一个非常有趣的课程，同时也受益良多。