电工导 C 第三次实验报告

姓名: 宋士祥　　学号:521030910013　　班级:F2103001

2022 年 10 月 5 日

1　实验概览

本次实验共学习了三个基本内容，分别为哈希散列，BloomFilter，以及并发编程的三个模块。

如果我们采用 List 的方式进行查找的话，其时间复杂度为 O(n)，因此遍历查找的方式效率不高。基于这样的考虑，我们需要一种数据结构方式来快速查找。Hash Function 给予我们这样的机会。Hash Function 的本质在于将某个字符串映射到一个 $[0, k-1]$ 的区间。对于原本的字符串而言，这样就相当于获取到了字符串的 key，便于查找。对于一个 Hash Table，我们理想的模型是`[[<word>, <word>,…], [<word>, <word>,…],…]`。

BloomFilter 则是在 Hash Function 的基础上，创建一个查询的数组，利用哈希函数将 string 映射到数组某一位上。当我们再次查找时，我们只需要验证某一位是否在数组内即可。当然，假设我们有 m 个字符，哈希函数映射到 $[0, k-1]$，我们要求 $k \gg m$。同时，为了防止哈希碰撞的出现，我们可以采用多个哈希函数的形式来减少这种可能的出现。此处略去数学推导，我们只需要这样做比较可行就可以了。只有多个哈希函数映射到的位均位 1 时我们才会判定字典中已经有这个字了。

并发编程是一种常见的任务分配方式。这样能够使任务同时进行，能够使任务更高效的完成。在 Python 中，`threading` 和`_thread` 提供了两个多线程的库。利用 threading 进行开发的模式在 ppt 和给定的代码中已经给出开发模式。利用 queue 的方式对任务进行排队可以有效完成多线程任务。相应的函数和功能在 ppt 中均已给出。

在上述知识基础上，本次实验需要完成两个任务：

1. 设计一个 BloomFilter 并自行验证错误率。
2. 利用并发式编程对 Lab2 中的爬虫进行改良。

2　实验环境

本次实验采用所需的实验环境如下：

- Docker 中的`sjtumic/ee208` 镜像
- Python3（使用 VSCode 编译）
- `BeautifulSoup4` 扩展以及`lxml` 扩展。
- `threading` 扩展和`queue` 扩展（在 SJTU EE208 中已经给出）。

3 练习 1

3.1 问题重述与代码说明

练习 1 要求我们自行设计一个 Bloomfilter 并自行验证错误率。

考虑`GeneralHashFunctions` 中已经给定了几种哈希的方法，此处不再自己写哈希函数。经过`HashTest.py` 的调用我们发现，给定哈希函数普遍较长，因此我们需要通过取模的形式进行放入。

经验证，我们发现，取四个哈希函数对于 5 万字以下的文档有较好的效果，因此我们的思路为：

- 先申请一个 500 万的数组
- 分别计算某一个字符的 Hash 值。
- 验证 Hash 值处是否有占位
- 均无占位记作新单词

核心代码展示如下：

```
def my_hash(words:list())->int:
  answer = 0
  my_bit_array = Bitarray(5000000)
  for word in words:
      elf = ELFHash(word) % 5000000
      dek = DEKHash(word) % 5000000
      djb = DJBHash(word) % 5000000
      fnv = FNVHash(word) % 5000000
      #elf = 1
      #dek = 1
      #djb = 1
      #fnv = 1
      # 验证是否都存在
      if my_bit_array.get(elf)==1 and my_bit_array.get(dek)==1 and my_bit_array.get(fnv)
                                                    ==1 and my_bit_array.get(djb)==1:
          continue
      else:
          answer += 1
          my_bit_array.set(elf)
          my_bit_array.set(dek)
          my_bit_array.set(djb)
          my_bit_array.set(fnv)
  return answer
```

3.2 实验对比与结果展示

我们分别采用了四篇文档，分别是每日邮报对英国女王去世新闻的报导、五个核武器国家领导人关于防止核战争与避免军备竞赛的联合声明、柯南道尔所著的《波西米亚丑闻》（短篇小说）与《血字的研究》（长篇小说）。

结果如下：

```
root@4db74b43060e:/workspaces/lab3-Crawler2/code# python myHash.py
There are 462 words in the the announcement of Her Majstery's death.
There are 245 vocabularies in the the announcement of Her Majstery's death.
Our method calculates 245 words in the the announcement of Her Majstery's death.
There are 0 errors.
There are 337 words in the joint announcement of Permanent members of the United Nations Security Council.
There are 193 vocabularies in the joint announcement of Permanent members of the United Nations Security Council.
Our method calculates 193 words in the joint announcement of Permanent members of the United Nations Security Council.
There are 0 errors.
There are 7319 words in the Conan Doyle's The Five Orange Pips.
There are 2401 vocabularies in the Conan Doyle's The Five Orange Pips.
Our method calculates 2401 words in the Conan Doyle's The Five Orange Pips.
There are 0 errors.
There are 43386 words in the Conan Doyle's A Study in Scarlet.
There are 8866 vocabularies in the Conan Doyle's A Study in Scarlet.
Our method calculates 8866 words in the Conan Doyle's A Study in Scarlet.
There are 0 errors.
There are 51504 words in total pages.
There are 10166 vocabularies in total pages.
Our method calculates 10166 words in total pages.
There are 0 errors.
```

我们可以看到，我们设计的 BloomFilter 在近 5 万字的小说中表现性能依旧可行。

3.3 问题探讨

3.3.1 关于 Hash 函数个数的问题

我们尝试分别将几个函数置 1，情况如下：

置一情况	fnv	djb	dek	elf	elf&dek	elf&fnv	djb&fnv	elf&dek&djb	dek&djb&fnv
test1 错误	0	1	0	0	0	0	1	0	0
test2 错误	0	0	0	0	0	0	0	0	0
test3 错误	0	1	0	0	0	0	1	1	5
test4 错误	0	1	0	0	1	0	2	9	32
total 错误	0	1	0	0	1	0	3	12	37

经过实验，我们发现，当函数个数减少时，不同实验已经或多或少得出现了错误。虽然存在一些理想的情况，但是多少会存在影响。

而函数的增多，由于我们申请了一个 5000000 的内存，导致内存空间过于稀疏（事实上我们申请了 500000 的内存也没有关系），导致函数暂时不会出现误差。但值得肯定的是，当字典数目增加时，如果哈希函数过多，一定会存在影响。

3.3.2 如果申请内存过少

当我们只申请 100000 的内存，运行结果如下：

```
There are 462 words in the the announcement of Her Majstery's death.
There are 245 vocabularies in the the announcement of Her Majstery's death.
Our method calculates 245 words in the the announcement of Her Majstery's death.
There are 0 errors.
There are 337 words in the joint announcement of Permanent members of the United Nations Security Council.
There are 193 vocabularies in the joint announcement of Permanent members of the United Nations Security Council.
Our method calculates 193 words in the joint announcement of Permanent members of the United Nations Security Council.
There are 0 errors.
There are 7319 words in the Conan Doyle's The Five Orange Pips.
There are 2401 vocabularies in the Conan Doyle's The Five Orange Pips.
Our method calculates 2401 words in the Conan Doyle's The Five Orange Pips.
There are 0 errors.
There are 43386 words in the Conan Doyle's A Study in Scarlet.
There are 8866 vocabularies in the Conan Doyle's A Study in Scarlet.
Our method calculates 8844 words in the Conan Doyle's A Study in Scarlet.
There are 22 errors.
There are 51504 words in total pages.
There are 10166 vocabularies in total pages.
Our method calculates 10130 words in total pages.
There are 36 errors.
```

可以发现，随着字数的增多，已经出现了哈希碰撞，而且字数越多，出现的越多。

4 练习 2

4.1 问题重述与代码说明

本题需要我们自行设计一个多线程爬虫。本题的`crawler` 将与 Lab2 中的`crawler` 进行对比。

本次实验的核心代码在于`working()` 函数，也是线程的 target。由于变量的作用域问题，我们需要先设置变量为全局变量。同时考虑到避免队列被重复读取，我们需要设置锁来避免队列被重复读取的问题。同时考虑到只有当队列被清空时任务才会结束，我们需要在达到我们需要爬取页面的任务后清空队列。

核心的`working()` 代码如下：

```
count = 0
def working():
    global count
    while count < max_page:
        page = q.get()
        if page not in crawled:
            content = get_page(page)
            add_page_to_folder(page, content)
            outlinks = get_all_links(content, page)
            if varLock.acquire():
                for link in outlinks:
                    q.put(link)
                graph[page] = outlinks
                print(page)
                crawled.append(page)
                count += 1
                varLock.release()
            q.task_done()
    else:
        # 强制清空队列
        while not q.empty():
            q.get()
            q.task_done()
```

4.2 实验对比与结果展示

我们在北京时间 2022 年 10 月 5 日 16 时在上海交通大学闵行校区主图书馆使用 SJTU 校园网分别对两个 crawler 进行测试。考虑到多线程爬虫类似 bfs，我们分别采用 bfs 对两种爬虫进行测试：

与 Lab2 类似，`crawl_thread.py` 不可以直接运行，运行方式与 Lab2 类似（即采用 sys 输入，新建 python 文件，运行 Python IDLE）

多线程：

```
root@4db74b43060e:/workspaces/lab3-Crawler2/code# python crawler_sample_thread.py
线程0正在进行
线程1正在进行
线程2正在进行
线程3正在进行
https://www.sjtu.edu.cn
https://news.sjtu.edu.cn/jdyw/20220929/174670.html
https://mp.weixin.qq.com/s/MPnPqJi3bCx1_NY31CtheQ
https://mp.weixin.qq.com/s/zHH4uJkMJgLEc17gFbm-4w
https://mp.weixin.qq.com/s/1vJulQpzchz5QTouJwfzmQ
https://news.sjtu.edu.cn/jdyw/20220930/174711.html
https://news.sjtu.edu.cn/jdyw/20221002/174769.html
https://mp.weixin.qq.com/s/M__a_6amNMWysCh5BkkMnw
https://news.sjtu.edu.cn/jdyw/20221003/174794.html
https://news.sjtu.edu.cn/jdyw/20221003/174800.html
https://news.sjtu.edu.cn/jdyw/20221003/174785.html
https://news.sjtu.edu.cn/jdyw/20220930/174731.html
https://news.sjtu.edu.cn/jdyw/20220930/174749.html
运行时间：0.3389749526977539
```

单线程：

```
root@7898d403f408:/workspaces/lab2-Crawler/code# python sjtu_crawler.py
https://www.sjtu.edu.cn
https://mp.weixin.qq.com/s/MPnPqJi3bCx1_NY31CtheQ
https://news.sjtu.edu.cn/jdyw/20220929/174670.html
https://mp.weixin.qq.com/s/1vJulQpzchz5QTouJwfzmQ
https://mp.weixin.qq.com/s/zHH4uJkMJgLEc17gFbm-4w
https://news.sjtu.edu.cn/jdyw/20220930/174711.html
https://mp.weixin.qq.com/s/M__a_6amNMWysCh5BkkMnw
https://news.sjtu.edu.cn/jdyw/20221002/174769.html
https://news.sjtu.edu.cn/jdyw/20221003/174794.html
https://news.sjtu.edu.cn/jdyw/20221003/174800.html
https://news.sjtu.edu.cn/jdyw/20221003/174785.html
运行时间：0.5047271251678467
```

可以看出多线程的效果优于单线程。

4.3 问题讨论

4.3.1 一个报错问题

在我们爬取 B 站的时候，系统出现了一个报错：

```
encoding error : input conversion failed due to input error
```

出现这一报错的原因是因为编码错误。beautifulsoup 的默认编码是 utf-8，但是 B 站的编码不完全是 UTF-8，出现了问题。

4.3.2 关于多出的网页问题

通过观察我们发现，我们在爬取的时候多爬取了几个网页。这是因为我们的条件判断是`count < max_page`，而当爬到第八个或第九个时，所有的线程都会同时爬取，这样会导致爬取的页面较多。

4.4 空的 Queue 锁着的问题

在第一个版本中，queue 虽然是空列但无法继续执行，这是因为没有执行`q.task_done()`，导致任务仍在继续执行中。queue 中的每一个元素都需要进行`task_done()` 以避免锁住。

5 拓展思考

5.1 每个线程爬取网页的速度相同吗？

我们把每个网页的线程打印出来：

https://www.sjtu.edu.cn
Thread id：线程 0
0.2970860004425049
https://mp.weixin.qq.com/s/1vJulQpzchz5QTouJwfzmQ
Thread id：线程 3
0.3757476806640625
https://news.sjtu.edu.cn/jdyw/20220929/174670.html
Thread id：线程 2
0.38355517387390137
https://mp.weixin.qq.com/s/MPnPqJi3bCx1_NY31CtheQ
Thread id：线程 1
0.410036563873291
https://mp.weixin.qq.com/s/zHH4uJkMJgLEc17gFbm-4w
Thread id：线程 0
0.18369460105895996
https://news.sjtu.edu.cn/jdyw/20220930/174711.html
Thread id：线程 3
0.1512281894683838
https://mp.weixin.qq.com/s/M___a_6amNMWysCh5BkkMnw
Thread id：线程 2
0.16092324256896973
https://news.sjtu.edu.cn/jdyw/20221002/174769.html
Thread id：线程 1
0.17873597145080566
https://news.sjtu.edu.cn/jdyw/20221003/174794.html
Thread id：线程 0
0.20467877388000488
https://news.sjtu.edu.cn/jdyw/20221003/174800.html
Thread id：线程 3
0.19912099838256836
https://news.sjtu.edu.cn/jdyw/20221003/174785.html
Thread id：线程 2
0.20090174674987793
https://news.sjtu.edu.cn/jdyw/20220930/174731.html
Thread id：线程 1
0.1788196563720703
运行时间：0.768399715423584

我们能发现，4 个线程的运行时间几乎大致相同，区别在于不同网页的响应时间有所差异。

5.2 N 个多线程爬取网页所需的时间是单线程爬取的 1/N 吗？

十条看不太出来，我们尝试爬 200 条网页（爬的是 bilibili，交大官网会爬到 science 官网导致速度显著下降）：
单线程用时 42s，多线程用时 12s。

可以看出，多线程几乎是单线程的 $\frac{1}{N}$。但是时间不同的点在于网页的爬取速度不同，一些站点的爬取时间较长，这导致单线程时间较长。而线程的切换也有一定时间，这些都导致了线程时间变长。

因此，这个结论并不是很严谨。计算不是完全的严格，但是结论还是可信的。

5.3 目前的 crawler.py 还有什么可以进一步提升速度的方法？

考虑我们每次都要做大量的出栈，我们可以引入一个参数来计算队列还可以添加的个数。当到达后停止添加。经测试爬取 150 条新浪网相关新闻可以提升约 2s 的运行速度。

与此同时，我们也可以通过正则表达式回避一些大文件的爬取如 apk 等，这样也能避免爬取到一个大文件，从而提升运行速度。