第三周实验报告

一、 实验概览

本实验旨在理解时间复杂度概念、学习哈希散列的原理与实现、学习布隆过滤器的数学原理，理解其实现和内部参数选择，并基于框架实现布隆过滤器；理解并发编程的概念和原理并基于 threading 库实现并发爬虫。

二、 实验环境

Docker: SJTU-EE208

三、 解决思路

1. 练习一

修改提供的 Bitarray.py，利用类实现布隆过滤器。

将 GeneralHashFunctions.py 文件中提供的 BKDRHash 函数修改为可传入 seed 的 hash 函数

```python
def BKDRHash(key, seed):
    hash = 0
    for i in range(len(key)):
        hash = (hash * seed) + ord(key[i])
    return hash
```

在初始化中定义过滤器中传入的字符串数量并根据参考文献得出对应的哈希函数个数和位数组的长度，初始化种子列表用于确定哈希函数

```python
class BloomFilter:
    def __init__(self, capacity=1e8, error = 1e-5) -> None:
        self.capacity = capacity
        self.m = math.ceil(-(capacity * math.log(error) / pow(math.log(2), 2)))
        self.k = math.ceil(0.7 * self.m / self.capacity)
        self.bitarray = Bitarray(self.m)
        self.seed_lst = [random.randint(1, 1000) for _ in range(self.k)]
```

加入关键字：根据哈希函数个数得到每个关键字对应置 1 的位并修改位数组

```python
def add(self, keyword):
    for seed in self.seed_lst:
        idx = BKDRHash(keyword, seed) % self.m
        self.bitarray.set(idx)
```

查找：利用上述哈希函数得到目标字符串的对应置 1 的位并检索位数组，如果对应位上的值均是 1 则判断该字符串在关键字中

```
def lookup(self, target):
    for seed in self.seed_lst:
        idx = BKDRHash(target, seed) % self.m
        if not self.bitarray.get(idx):
            return False

    return True
```

测试：设计随机生成字符串的函数（测试中设置字符串长度为 2-20 位），并测试错误率

```
def get_random_str():
    length = random.randint(1, 20)
    letters = '1234567890qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM'
    letters = list(letters)
    res_str = ''.join([random.choice(letters) for _ in range(length)])
    return res_str
```

```
bloom_filter = BloomFilter(num)
wordlst = []
for _ in range(int(1e5)):
    text = get_random_str()
    wordlst.append(text)
    bloom_filter.add(text)

count = 0
for i in range(int(5e5)):
    print(i)
    text = get_random_str()
    res = bloom_filter.lookup(text)
    if res != (text in wordlst):
        count += 1

p = count / (5e5)
print(p)
```

2. 练习二

修改 lab2 中的 crawler.py 文件，引入 threading 库，将工作分布给四个线程

```
def crawl(seed, max_page):
    tocrawl = queue.Queue()
    tocrawl.put(seed)
    crawled = []
    graph = {}
    # count = 0
    NUM = 4
    for i in range(NUM):
        t = threading.Thread(target=crawl_per_thread, args=(tocrawl, crawled,
        graph, max_page))
        t.setDaemon(True)
        t.start()

    t.join()
    return graph, crawled
```

利用 threading.lock 和 queue 处理并发程序中访问同一变量（queue 自带并发处理，其中 crawled 和 graph 变量利用 Lock 类处理）的问题

```
def crawl_per_thread(tocrawl, crawled, graph, maxpage):
    if not tocrawl:
        time.sleep(0.5)
    while len(crawled) <= maxpage:
        try:
            page = tocrawl.get()
            if page not in crawled:
                print(page)
                content = get_page(page)
                add_page_to_folder(page, content)
                outlinks = get_all_links(content, page)
                for outlink in outlinks:
                    tocrawl.put(outlink)
                lock.acquire()
                crawled.append(page)
                graph[page] = outlinks
                lock.release()
        except:
            pass

    return
```

四、 实验结果

练习一：经检验错误率约为 7e-5，满足使用需求

```
499998
499999
7e-05
```

练习二：结果详见 index.txt 和文件夹 html，下图为 index.txt 的部分截图

```
'http://www.sjtu.edu.cn'        httpwww.sjtu.edu.cn
'https://mp.weixin.qq.com/s/xw_a1eEpk_0rBFiKzf-sjA'     httpsmp.weixin.qq.comsxw_a1eEpk_0rBFiKzf-sjA
'https://mp.weixin.qq.com/s/X578_wp2JKFRmBEtyKhe1w'     httpsmp.weixin.qq.comsX578_wp2JKFRmBEtyKhe1w
'https://mp.weixin.qq.com/s/7O2Q7cK1dCLjFjD0o4tUpw'     httpsmp.weixin.qq.coms7O2Q7cK1dCLjFjD0o4tUpw
'https://mp.weixin.qq.com/s/3bHAyVN2FT3OgmdD0gCyAw'     httpsmp.weixin.qq.coms3bHAyVN2FT3OgmdD0gCyAw
'https://news.sjtu.edu.cn/jdyw/20231003/188866.html'    httpsnews.sjtu.edu.cnjdyw20231003188866.html
'https://mp.weixin.qq.com/s/0AaKhFOAeimYcozEuL7esw'     httpsmp.weixin.qq.coms0AaKhFOAeimYcozEuL7esw
'https://news.sjtu.edu.cn/jdyw/20231007/188893.html'    httpsnews.sjtu.edu.cnjdyw20231007188893.html
'https://news.sjtu.edu.cn/jdzh/20231005/188872.html'    httpsnews.sjtu.edu.cnjdzh20231005188872.html
'https://news.sjtu.edu.cn/jdyw/20231001/188833.html'    httpsnews.sjtu.edu.cnjdyw20231001188833.html
'https://news.sjtu.edu.cn/mtjj/20231007/188887.html'    httpsnews.sjtu.edu.cnmtjj20231007188887.html
'https://news.sjtu.edu.cn/jdyw/20230929/188812.html'    httpsnews.sjtu.edu.cnjdyw20230929188812.html
'https://mp.weixin.qq.com/s/H7AdI6PQnf0WrKapNuxtUg'     httpsmp.weixin.qq.comsH7AdI6PQnf0WrKapNuxtUg
'https://news.sjtu.edu.cn/jdyw/20230928/188733.html'    httpsnews.sjtu.edu.cnjdyw20230928188733.html
'https://news.sjtu.edu.cn/jdyw/20230929/188809.html'    httpsnews.sjtu.edu.cnjdyw20230929188809.html
'https://news.sjtu.edu.cn/jdyw/20231002/188852.html'    httpsnews.sjtu.edu.cnjdyw20231002188852.html
'https://news.sjtu.edu.cn/jdyw/20230928/188721.html'    httpsnews.sjtu.edu.cnjdyw20230928188721.html
'https://news.sjtu.edu.cn/jdzh/20231001/188834.html'    httpsnews.sjtu.edu.cnjdzh20231001188834.html
'https://news.sjtu.edu.cn/zhxw/20230928/188786.html'    httpsnews.sjtu.edu.cnzhxw20230928188786.html
'https://news.sjtu.edu.cn/jdyw/20230929/188808.html'    httpsnews.sjtu.edu.cnjdyw20230929188808.html
'https://news.sjtu.edu.cn/zhxw/20230928/188784.html'    httpsnews.sjtu.edu.cnzhxw20230928188784.html
'http://www.sjtu.edu.cn/tg/20230926/188597.html'        httpwww.sjtu.edu.cntg20230926188597.html
'http://www.sjtu.edu.cn/tg/20230911/187902.html'        httpwww.sjtu.edu.cntg20230911187902.html
'https://news.sjtu.edu.cn/index.html'   httpsnews.sjtu.edu.cnindex.html
'http://www.sjtu.edu.cn/tg/20230926/188604.html'        httpwww.sjtu.edu.cntg20230926188604.html
```

五、 分析与思考

线程和进程：进程是一个正在执行的程序的实例，包括程序计数器、寄存器和程序变量的当前值。进程之间是相互独立的，有自己的内存空间，而线程之间可以共享进程中的内存（因此存在访问同一变量的问题），线程具有的独立部分在于栈区和所用到的寄存器。多线程中抢占变量的问题在于处于临界区的线程存在地址冲突。