**北 京 邮 电 大 学**

**计算机应用编程实践**

**实验报告**



|  |  |
| --- | --- |
| 题目： | 网站页面分析器 |
| 姓名： | 田昌昊 |
| 学号： | 2019110680 |
| 班级： | 2019111307 |

计算机学院

2019年12月

目录

[一、实验任务与实验内容 3](#_Toc27072901)

[1. 实验任务： 3](#_Toc27072902)

[2. 实验意义： 3](#_Toc27072903)

[3. 程序要求 3](#_Toc27072904)

[二、实验过程 4](#_Toc27072905)

[1. 主要的数据结构和流程 4](#_Toc27072906)

[1.1 爬虫程序 4](#_Toc27072907)

[1.2 页面分析程序 5](#_Toc27072908)

[2. 实验过程 5](#_Toc27072909)

[三、遇到的问题 5](#_Toc27072910)

[1. Bloomfilter过滤失效 5](#_Toc27072911)

[四、结果指标 5](#_Toc27072912)

[五、结论和总结 6](#_Toc27072913)

# 一、实验任务与实验内容

## 1. 实验任务：

**目标**

设计一个网站页面分析系统

爬取十几万个页面的网站，构建页面链接网络，计算pagerank最高的前10个页面

**编程技能**

C语言编程

网络编程与HTTP协议

稀疏矩阵与图结构

随机过程

特征值与特征向量

PageRank算法

## 2. 实验意义：

**各种实际应用**

科研或产品研究都需要从互联网抓取数据

基于图结构分析复杂网络，例如社交网络、科研合作网络、欺诈网络等

如何利用矩阵来建模图并理解矩阵的作用

## 3. 程序要求

**输入数据**

大约16万个网页，不存在的链接过滤掉

**程序要求**

**爬虫程序**

基本版本：bcrawler firsturl url.txt

url.txt：

* 一行一个url及其编号
* 空行
* 后面是url链接关系，一行两个编号表示链接关系

**页面分析程序**

基本版本：./pagerank url.txt top10.txt

Top10.txt：

* 一行一个，rank倒序排列
* 每行：url rank值

# 二、实验过程

## 1. 主要的数据结构和流程

### 1.1 爬虫程序

数据结构队列Queue：

typedef struct QN{

char\* url;

struct QN\* next;

}QueueNode;

typedef struct {

int size;

QueueNode\* ql;

QueueNode\* tail;

}Queue;

数据结构布隆过滤器Bloomfilter：

typedef struct

{

char\* bits;

} BF;

数据结构三叉树TrieTree：

typedef struct TNode

{

char data; //节点存储的字节数据

struct TNode \*left, \*right; //同级节点TSNode

struct TNode \*next; //下一级节点

int ID; //编号id

} TNode, \*TNodePtr;

typedef struct

{

TNodePtr root; //三叉树根节点

int uNum; //插入链接的个数

int uLen; //插入链接的长度

char\* searchURL; //待搜索的链接

TNodePtr currentNode; //当前搜索到的节点

} TREE;

数据结构网页爬虫：

typedef struct

{

char \*urlName;

int fd;

} Ev\_arg;

Queue结构用来存储识别得到的需要继续爬取的URL链接，在每轮请求时，从队列中获取一定量的URL，并行发出请求，然后通过Ev\_arg结构进行事件监测，在得到服务器的返回数据后，进行处理并将识别到的新的URL加入Queue。其中，每次添加至Queue前会通过Bloomfilter结构进行判重，从而防止重复爬取。

三叉树Tree结构用来存储URL和对应的ID，并在完成爬取任务后，将链接关系ID-URL识别并转化为ID-ID的对应关系。

### 1.2 页面分析程序

通过数组结构进行计算和存储。

主要函数如下：

getValueforCoo(); // 存储当前的连接关系，并存储总连接个数

CalPageRank(); // 计算pagerank

findTop10(); // 遍历寻找前10个最高分数的URL

## 2. 实验过程

在实验过程中，首先编写bcrawler.c程序，利用之前实验编写的Queue结构和Bloomfiler结构进行封装，然后提供给本程序使用。之后实现了三叉树结构进行URL到ID的映射关系，实现快速查询URL对应的ID。

对于状态自动机识别URL链接的过程比较繁琐，老师的PDF里面存在漏洞，需要补足。所以调整BUG浪费了较多时间。

之后参考网上的例子，完成了pagerank的计算过程和逻辑，比较顺利。

# 三、遇到的问题

## 1. Bloomfilter过滤失效

在编写过程中，得到服务器返回的html信息，并识别到URL链接后。在加入队列前需要通过Bloomfilter进行URL判重，并去掉已经爬取的URL，将未爬取的URL加入Queue并加入Bloomfilter。

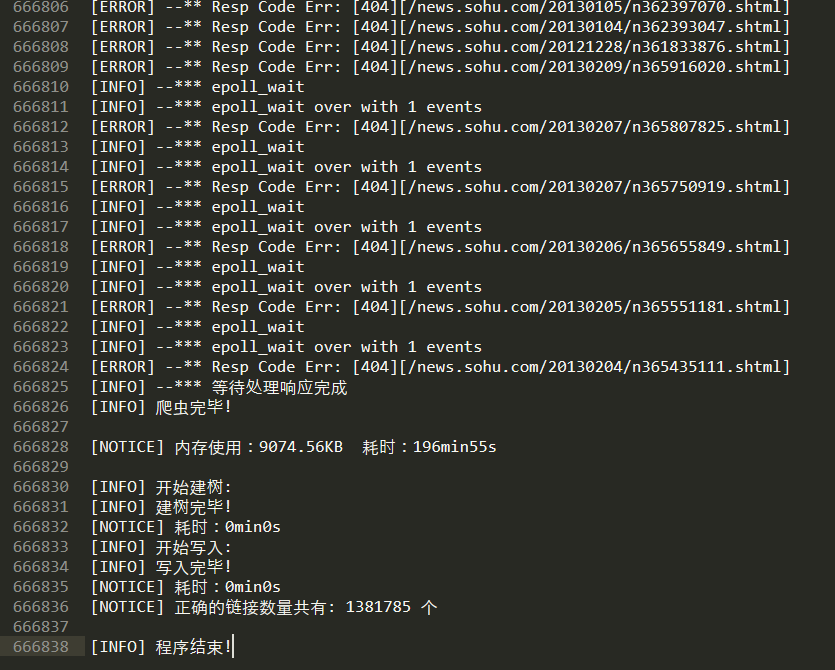
而在加入Queue和Bloomfilter前，需要将URL前面加上“/news.sohu.com”字段，所以设置的时候，其实加入的是“/news.sohu.com”+URL，而判定是否爬取，使用的是URL，导致所有的URL都判空，全部加入了队列，使得队列不断增大，爬取不完。。。

后续查出BUG并将URL拼接放在判定前，问题解决。

# 四、结果指标

|  |  |  |
| --- | --- | --- |
| **版本** | **内存使用（kb）** | **运行时间** |
| **bcrawler** | 9074.56 | 196min55s |
| **pagerank** | 29378 | -- |

运行结果展示如下：





上述分别为bcrawler可执行程序运行结束的日志内容；以及pagerank可执行程序运行结束的命令框输出。

# 五、结论和总结

本次实验比较庞大，以前用C编写的都是比较简单的程序，而本次程序的复杂度较大，涉及多线程爬取、事件处理、队列使用、布隆过滤器去重、状态自动机识别文本，三叉树加速查询URL对应ID的速度、构建稀疏矩阵等方面，实现的工程能力要求比较高，也在网上查询了很多知识点。

爬取很难，bug不好调整，而且爬取的时候，服务器有时候很不稳定，x.x.x.162经常断连。。。