一个小型搜索引擎的设计与实现

姜文渊1) 陈宥希1) 赵天畅1)

1)(东北大学计算机科学与工程学院,沈阳 110169)

摘 要 提出一个基于C++ 编程语言和Python编程语言的小型搜索引擎.通过对目前搜索引擎核心原理及相关技术成果的研究分析，将搜索引擎的搜集、分析、索引、查询四个部分形成了网页获取子系统、索引生成子系统、检索子系统三大系统.该搜索引擎支持中文内容和英文内容.分析了系统所采用的核心算法，并在实验中验证了算法设计的复杂度.在进行临时索引文件的排序中，使用了败者树的数据结构，相比于采用蛮力搜索进行多个文件的归并，在文件数为500的情况下，败者树数据结构能将排序性能提高21%.

关键词 搜索引擎；索引；检索；外部排序；网页获取

中图法分类号 TP391.1

Design and implementation of a light search engine

JIANG Wen-Yuan1) CHEN You-Xi1) ZHAO Tian-Chang1)

1)(School of Computer Science And Engineering, Northeastern University, Shenyang 110169)

**Abstract** A small search engine based on C++ and Python. Through the research and analysis of the core principles of the current search engine and related technical achievements, the four parts of the search engine's collection, analysis, index and query form a webpage acquisition subsystem , index generation subsystem, retrieval subsystem. The search engine supports Chinese content and English content. The core algorithm used by the system is analyzed, and the complexity of the algorithm design is verified in the experiment. In the sorting, the data structure of the loser tree is used. Compared with the brute force search for merging multiple files, when the number of files is 500, the loser tree data structure can improve the sorting performance by 21%.

**Key words** Search Engine; Indexing; Searching; External Sort; Webpage Fetching

# **1** 系统架构

搜索引擎系统由网页获取子系统、网页分析子系统、索引生成子系统、检索子系统四个模块构成.作为数据的收集处理系统,网页获取子系统采用python编程语言编写,通过递归方法进行每个网页的网址、标题、文章内容三部分信息的获取,并返回存入至news.csv文件中;该文件送入网页分析子系统,经网页内容抽取、分词、构建临时索引三个步骤后，得到初步的单词索引编号、网页索引编号、临时索引文件;临时索引文件被送入索引生成子系统,经过采用败者树数据结构的外部排序后,先得到有序的临时索引文件,并据此生成倒排索引;检索子系统根据倒排索引文件和带查询的关键词分词后的单词编号,找出与本次查询文本相关性最强的若干个网页内容,并赋予其相关度一个权值,最终在结果显示部分显示最相关的20个网页.

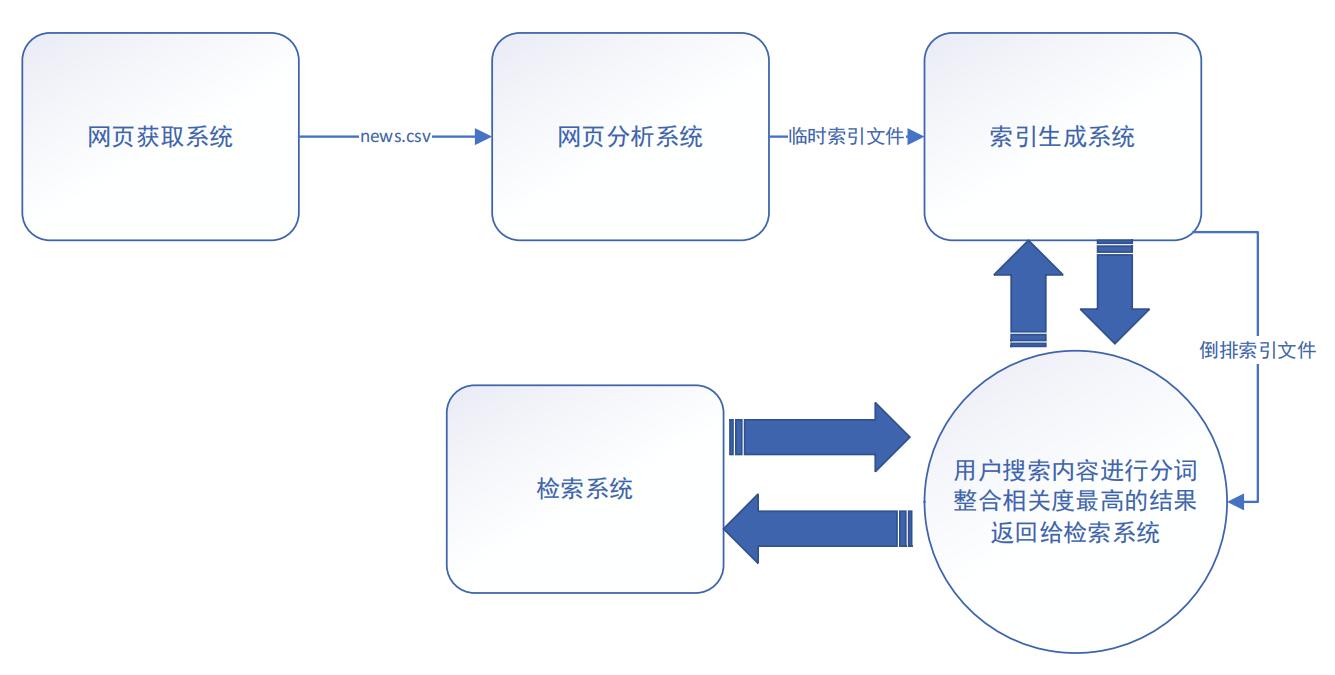


图1 系统架构图

# **2** 子系统

## **2.1** 网页获取子系统

### 2.1.1 类与函数列表

本子系统的类与函数列表如表1所示.

表1 网页获取子系统类与函数列表

|  |  |  |
| --- | --- | --- |
| 类/函数名 | 参数 | 功能 |
| NewsSpider2 | name(存放文件名),start\_urls(起始网址) | 继承scrapy.Spider,爬取中文页面信息 |
| parse() | self, response | 递归获取网页内容，并将内容以三元组返回并存入news2.csv文件中 |

### 2.1.2 文件格式定义

news2.csv: 存放{网页url,网页标题,网页文章内容}的三元组,每行存放一个三元组.

### 2.1.3 爬虫部分的伪代码

算法**1**. 基于初始网页递归爬取网页内容.

输入：start\_urls(初始网址)

输出：triples(三元组序列)

1. FOR *news2* IN response.vss(‘div.main’) DO
2. IF news2.css('h1::text').get() 不为空; THEN
3. 更换递归主体,接收底层递归的返回值;
4. ELSE IF news2.css('h1::text').get() 为空;
5. return triple{url,title,news\_content}
6. END IF
7. END FOR
8. RETURN triples

### 2.1.4 时间复杂度与空间复杂度分析

网页数据爬取部分采用的是递归式爬取，但其递归形式仅为了能够自动获取相关网页。对于爬取部分的时间复杂度，与读取的网页个数成正相关，同时与每个网页文章内容的字数成正相关。空间复杂度方面，所需空间为获取的网页内容三元组所占空间。

## **2.2** 网页分析子系统

### 2.2.1 类与函数列表

本子系统的类与函数列表如表2所示。

表2 网页分析子系统类与函数列表

|  |  |
| --- | --- |
| 类名/函数名 | 功能 |
| Trie | 给出了Trie树结点结构的定义 |
| findWord(word) | 根据输入的参数word，返回其唯一的单词编号（按第一次出现顺序编号），以及该单词是否为第一次出现 |
| jieba.lcut\_for\_search(s) | jieba库函数，根据给定的中文字符串s，输出其分词之后得到所有单词的列表 |
| main() | 读取爬虫得到的输入文件news.csv，将其中每个网页的URL、标题和正文内容区分开来，并调用分词函数及单词编号函数，将得到的相关信息分别写入网页编号文件、单词编号文件、临时索引文件中 |

### 2.2.2 文件格式定义

news2.csv: 存放{网页url,网页标题,网页文章内容}的三元组,每行存放一个三元组.

doc\_id.txt: 存放{网页编号,网页URL,网页标题}的三元组,每行存放一个三元组.

term\_id.txt: 存放{单词,该单词的编号}的二元组,每行存放一个二元组.

temp\_index.txt: 存放{单词,该单词的编号}的二元组,每行存放一个二元组.

### 2.2.3 伪代码

算法**2**. 使用Trie树获得给定单词的唯一编号.

输入：给定单词word

输出：单词的编号ret\_id，以及标识是否为新单词的标志位

class TrieNode:

chr = str() # chr为通向此节点的分支所对应的字符

isword = bool()

# 标志位，表示此节点能否表示一个单词（红色/白色）

term\_id = int() # term\_id为此节点表示的单词编号

child = list() # list of TrieNode，指向此节点的所有孩子

root = TrieNode() # Trie树的根节点

term\_id\_count = 0 # 当前单词总数（编号从0开始）

findWord(word: str) -> (int, bool):

# returning (term\_id, is\_new\_word)

i = 0

currentnode = root

WHILE (i < 单词word的长度) DO

IF (不存在与word[i]对应的孩子) THEN

给当前节点增加一个孩子，它的chr = word[i]

currentnode = 这个新的孩子

ELSE

currentnode = word[i]对应的孩子

i += 1 # 指针++

END IF

END WHILE

IF (not currentnode.isword) THEN

# 若当前所在节点还不能表示一个单词

currentnode.isword = True

# 标志位置True，使此节点能表示一个单词

currentnode.term\_id = term\_id\_count

# 为这个单词赋予编号

ret\_id = term\_id\_count # 获得要返回的单词编号

term\_id\_count += 1 # 单词总数++

is\_new\_word = True

ELSE # 若当前所在节点能表示一个单词

ret\_id = currentnode.term\_id

is\_new\_word = False

END IF

RETURN ret\_id, is\_new\_word

算法**3**. 由源文件生成三个目标文件.

输入：爬虫获取到的网页源文件news.csv

输出：网页编号文件doc\_id.txt、单词编号文件term\_id.txt、临时索引文件temp\_index.txt

初始化Trie树，建立根节点;

*mode* = 处理url和标题行;

*当前正在处理的网页编号* = 0;

*this\_line* = 从news.csv读取一行;

WHILE 尚未读到文件末尾 DO

IF *mode* == 处理url和标题行 THEN

将*this\_line*中的url和标题分隔开,并给新网页编号;

将得到的{网页编号、url和标题}三元组写入doc\_id.txt;

*wdl* = 对当前行分词后的所有单词列表;

# 分词方式：中文分词调用jieba.lcut\_for\_search(当前行)；英文分词则直接根据空格划分，并去除标点符号

FOR *w* IN *wdl* DO

调用findWord(*w*),在Trie树中查找它的单词编号，以及是否为新单词;

将{*w*的单词编号,当前正在处理的网页编号}二元组写入temp\_index.txt;

IF *w*为新出现的单词 DO

将{*w*,*w*的单词编号}二元组写入term\_id.txt;

END IF

END FOR

*mode* = 处理多行正文内容;

ELSE IF *mode* == 处理多行正文内容 THEN

WHILE 尚未读到下一网页的url DO

*this\_line* = 从news.csv读取一行;

IF 读到了下一个网页的url和标题 THEN

*this\_line* = 之前读入的所有内容;

BREAK

END IF

END WHILE

*wdl* = 对当前行分词后的所有单词列表;

FOR *w* in *wdl* DO

调用findWord(*w*),在Trie树中查找它的单词编号，以及是否为新单词;

将{*w*的单词编号,当前正在处理的网页编号}二元组写入temp\_index.txt;

IF *w*为新出现的单词 DO

将{*w*,*w*的单词编号}二元组写入term\_id.txt;

END IF

END FOR

*当前正在处理的网页编号* += 1;

*this\_line* = 下一个网页的url和标题;

*mode* = 处理url和标题行;

END IF

END WHILE

### 2.2.4 时间复杂度与空间复杂度分析

中文词分词采用jieba库的函数,对于每个待分词句子,根据前缀词典生成有向无环图,再动态规划查找最大概率路径;对于总字符数为n的文件,需O(n)的时间完成分词.

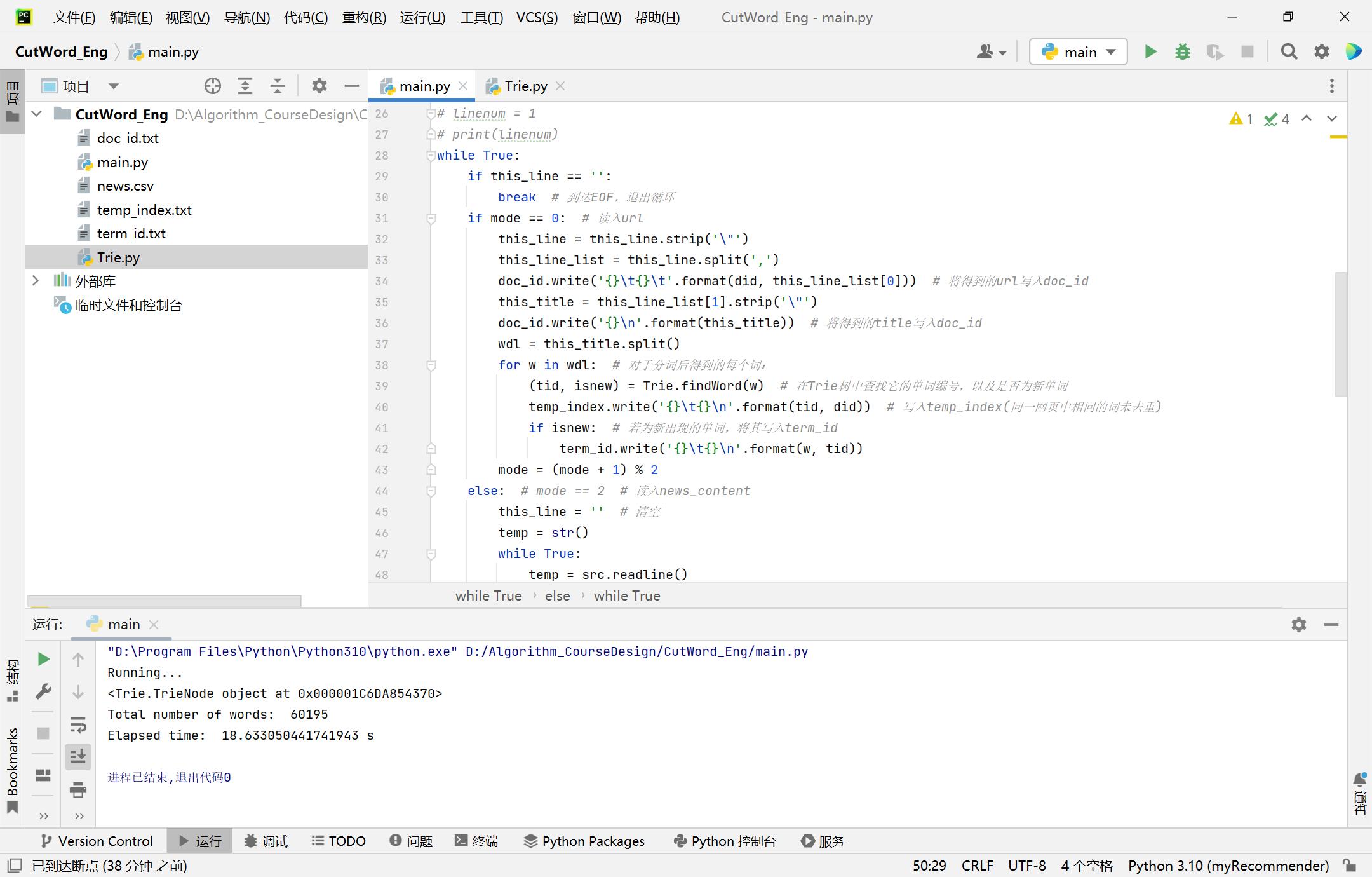
中文单词的字符较多样,因此若采用Trie树编号,会使得树中最初几层的结点占用空间较高(平均占用内存约500MB),影响查找应进入结点的效率,使得到编号的时间大于O(m).对于每个单词,若其字符数为m,则Trie树可在O(m) < t < O(m2)的时间内得到它的编号,即对于总字符数为n的文件,需O(n) < t < O(n2)的时间完成编号.

爬取的中文网页文件(336MB)使用经优化消减运算强度后的代码约需2h30m完成生成网页编号文件(15.0MB)、单词编号文件(44.1MB，582383词)和倒排索引文件(832MB) (含读写文件耗时).

英文词分词采用空格分隔的方式,对于总字符数为n的文件,需一遍扫描O(n)的时间完成分词.

由于英文单词的字符种类数是确定的,对于每个单词,若其字符数为m,则Trie树可在O(m)时间内得到它的编号,即对于总字符数为n的文件,需O(n)的时间完成编号.

示例给出的英文网页文件(28.4MB)约需18~20s完成生成网页编号文件(1.55MB)、单词编号文件(936KB,60195词)和倒排索引文件(44.1MB) (含读写文件耗时).



空间复杂度方面，Trie树上一个字符对应分支需要一份存储空间，因此共需(所有单词去除相同前缀字符)份存储空间;对于总字符数为n的文件，空间复杂度O(1) < m < O(n).

## **2.3** 索引生成子系统

### 2.3.1 类与函数列表

本子系统的类与函数列表如表3所示.

表3 索引生成子系统类与函数列表

|  |  |  |
| --- | --- | --- |
| 类/函数名 | 参数 | 功能 |
| QuickSort | r(数组首地址),first(排序头部偏移量),end(排序尾部偏移量) | 在数组r的first到end序列中递归地进行快速排序 |
| Partition | r(数组首地址),first(排序头部偏移量),end(排序尾部偏移量) | 一次划分,选择一个元素作为枢轴,然后将所有比枢轴小的元素放到枢轴的左边,将比枢轴大的元素放到枢轴的右边 |
| swap | r(数组首地址),j(交换数1的偏移量),k(交换数2的偏移量) | 将r数组里的偏移量为j和k的元素互换 |
| f\_in\_init | 无 | 初始化顺串文件名 |
| init | 无 | 初始化文件输入流 |
| get | index(下标) | 获取下标为index的叶子节点的值 |
| set | index(下标) | 设置下标为index的叶子节点的值 |
| adjust | index(下标) | 在败者树中对下标为index的结点开始向上进行调整 |
| initTree | 无 | 对败者树进行初始化 |
| divide | 无 | 读取乱序临时索引文件,分为K个文件进行读取并调用QuickSort进行内排序,之后分别写入K个文件 |
| merge | 无 | 将K个排序好的文件进行归并 |

### 2.3.2 文件格式定义

temp\_index.txt: 存放乱序的临时索引文件,内部为{term\_id,doc\_id}的二元组.

\*.txt: 为临时索引文件拆分后的文件，\*代表从0开始到K-1，共K个文件.

sorted\_temp\_index.txt: 存放排序后的临时索引文件,内部为{term\_id,doc\_id}的二元组.

inverted\_index.txt: 存放倒排索引.

### 2.3.3 采用败者树数据结构的外部排序伪代码

算法**4**. 基于败者树数据结构的外部排序.

输入:temp\_index.txt(乱序临时索引文件)

输出:sorted\_temp\_index.txt(排序后的临时索引文件)

1. 初始化文件输入流
2. 初始化败者树
3. FOR i =0 TO K-1 DO
4. 将第i部分读入;
5. QuickSort(r,0,N/K);
6. 将结果写入第i份文件;
7. END FOR
8. 调用merge(),用败者树将K份文件归并
9. 生成倒排索引文件

### 2.3.4 时间复杂度与空间复杂度分析

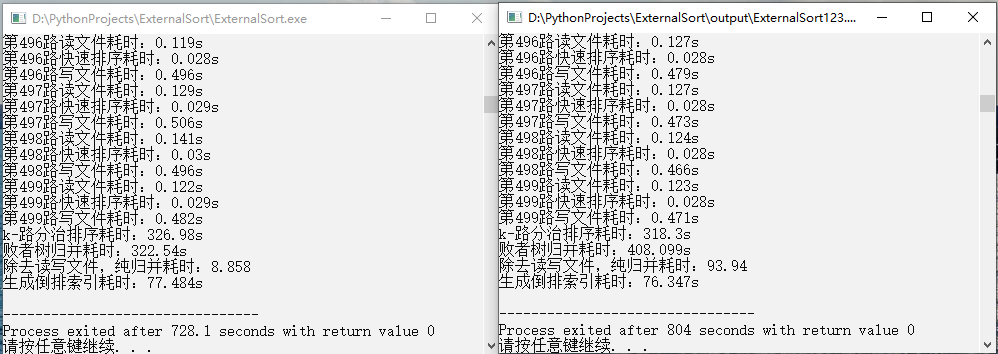
预先声明：N为总数据量,K为归并排序分为的文件个数,则对于内部排序的快速排序来说,数据量n=N/K.

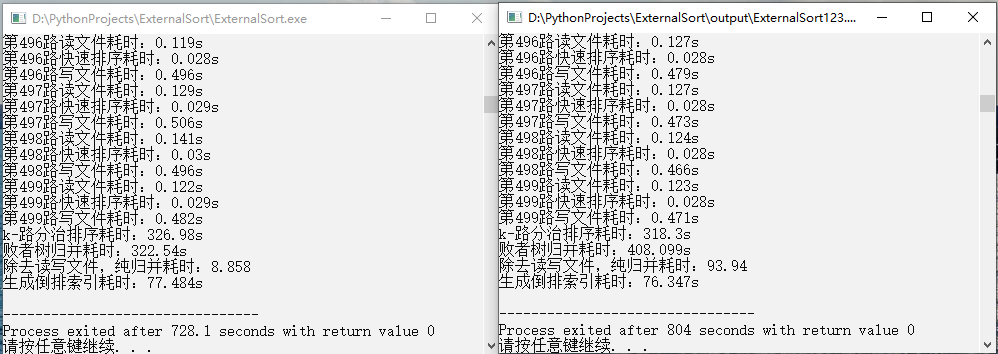
时间复杂度: 读文件 O(N)

写文件 O(N)

内部排序 O(K\*nlogn)

归并部分:由于败者树在进行K个文件的归并时,从K个顺串的首部选择一个最小者的过程是使用二叉树的树形结构，因此和蛮力搜索相比较其时间复杂度从O(N×K)降低为O(N×logK),因此在K等于500的时候,可以明显地看到,除去读写文件的时间,纯归并耗时败者树比蛮力搜索并进行归并要快接近10倍,而总体的归并排序耗时则因此提升了21%.





空间复杂度:由于将数据量为N的文件分为了K份,因此在程序中若要实现内部排序,仅需开辟一个大小为n=N/K的临时数组,因此空间复杂度为O(N/K).

## **2.4** 检索子系统

按照任务书中的任务要求，我们应根据输入文本生成相应分词，然后根据分词到单词编号文件中查找对应单词编号，然后用所得单词编号到第三步生成的倒排索引文件中得到相关的网页编号列表，由此找出与本次查询文本相关性最强（即在网页编号列表中出现频率最高）的若干网页编号，最后，利用这些编号回到网页编号文件中得到对应的网页链接。

可以看出，在文件之间检索的过程中，查询单词编号文件是为了得到单词编号，查询倒排索引文件是为了得到对应网页编号，查询网页编号文件是为了得到对应网页链接，而我们在文件之间检索的纽带正是各自文件中对应的索引，从而我们想到可以利用python中的list数据结构每个数据项的索引来替换这个过程中各个纽带所起到的作用。

因此，可以预先读取单词编号文件、网页编号文件和倒排索引文件，将各自其中起到被指向作用的数据以数据项的形式保存到三个不同的list中，由此list列表的下标索引属性取代了原有的文件中具有指向作用的数据项所起到的索引作用，在简化了查询操作的同时节省了查找所需的时间。

以下为具体实现步骤：

首先，我们将单词编号文件中的单词编号、网页编号文件中的网页链接和标题、倒排索引文件中的网页编号分别读取后存入3个list列表，在将输入的查询文本进行分词操作之后，遍历单词编号列表term\_list比较并保存所得下标。

在此之后，利用此下标对倒排索引列表index\_list相应的下标进行访问，便可直接得到所要查找的相关网页编号。在对这些网页编号进行权重排行之前，我们可以对他们进行一个简单的快速排序，从而得到一个有序的相关网页编号列表——这样做是为了将相同的网页编号放在相邻的位置，以便于后续对权重值的统计。

接下来我们便可开始进行对网页编号权重的统计及排序。由于相关网页编号是通过分词结果间接得到的，换言之，同一网页编号出现次数越多，说明本次分词结果中与之相关的单词越多，该编号对应的文章与本次查询文本的相关性也就越强。因此在本次权重排行中，我们以网页编号出现次数为依据，通过计数每个网页编号在上一步所得的有序编号列表中出现的次数，对每个编号进行权值统计，并进行去重。根据日常生活经验，我们将推荐结果数目设定为20条。遍历上一步所得有序相关网页编号列表，如果当前网页编号已经出现过则该网页编号对应次数加一；如果未出现过，说明上条网页编号次数已经统计完毕，将该次数与当前已出现过的前20位最大次数进行比较——若大于第20位对应次数，则进行插入排序，置入新的网页编号和出现次数，并继续读入下一条网页编号，否则直接置入新的网页编号和出现次数，并继续读入下一条网页编号。

经过上述操作后，我们已经得到了前20位相关性最强（即出现次数最多）的网页编号。只需利用该编号序列对第一步得到的网页编号列表doc\_list进行对应下标访问并保存对应数据项（网页链接和标题），如此进行一遍访问过后便可得到最终结果。

### 2.4.1 类列表

本子系统的类列表如表4所示.

表4 网页获取子系统类列表

|  |  |  |
| --- | --- | --- |
| 类名 | 成员 | 功能 |
| resdid | did:网页编号  num:用户输入的单词在该网页中出现次数 | 给出了根据网页包含用户输入单词的多少进行排序时的结点定义 |

### 2.4.2 文件格式定义

doc\_id.txt: 存放{网页编号,网页URL,网页标题}的三元组,每行存放一个三元组.

term\_id.txt: 存放{单词,该单词的编号}的二元组,每行存放一个二元组.

inverted\_index.txt: 存放{单词编号,网页编号1, 网页编号2,…}的多元组,每行存放一个单词对应的多元组.

### 2.4.3 伪代码

算法**5**. 根据查询文本给出相关网页链接.

输入: 待查询文本

输出: 与输入文本相关性最强的若干个（最多20个）网页对应链接及其标题

WHILE loopflag DO //主循环，loopflag为是否继续循环输入的判断标志，初值为True

1.预先分别读入网页编号文件并将网页编号存入doc\_list、读入单词编号文件并将单词存入term\_list、读入倒排索引文件并将网页编号存入index\_list

2.调用jieba库对查询文本进行分词，并将分词后结果保存到wdl，同时将分词个数保存到res\_cnt

3.FOR i IN range(term\_list\_len) DO

到term\_list中查询相关分词，将分词所对应的下标保存到res\_list，同时注意去重，当已找到单词数等于分词个数res\_cnt时可提前退出

END FOR

4.FOR y IN res\_list DO

rel\_did.extend(index\_list[y])

根据上步保存的下标直接根据index\_list得到本次查询分词相关的网页编号

END FOR

5.QuickSort(rel\_did)

对这些网页编号进行快速排序

max\_num = [rsd.resdid() for n in range(20)]

对准备用于保存前20位相关性最强的相关网页编号的列表进行定义并初始化

6.统计相关网页编号出现次数并去重

FOR curr IN range(len(rel\_did)) DO

对当前网页编号判断是否已出现过：

若已出现过，则出现次数自增，继续；

若未出现过，将当前网页编号置为新的网页编号，次数置1，同时将上一个网页编号出现次数与当前次数最大的前20个编号最末尾的编号出现次数进行比较，如果上一出现次数大于该末位次数则进行插入排序，继续

END FOR

7.保存对应网页编号

FOR n IN range(len(max\_num)):

tp\_did.append(max\_num[n].did)

END FOR

8.统计查询到的相关网页编号个数并输出网页链接和标题

9.询问是否继续，输入n将loopflag置False,输入y不改变loopflag值

END WHILE

### 2.4.4 时间复杂度与空间复杂度分析

时间复杂度：

读入文件——O(n)

获取分词结果对应下标——O(n)

根据下标获取对应网页编号——O(n)

对相关网页编号快速排序——O(nlogn)

统计权重——O(n)

统计时的20个元素-类插入排序——O(1)

获取网页链接——O(n)

空间复杂度：

读入文件——O(n)

获取分词结果对应下标——O(n)

根据下标获取对应网页编号——O(n)

对相关网页编号快速排序——O(1)

统计权重——O(n)

统计时的20个元素插入排序——O(20)

获取网页链接——O(20)

参 考 文 献

[1] [Mark Allen Weiss](https://book.douban.com/author/425351).Data Structures and Algorithm Analysis in C (2nd Edition), 1996, 198-204 (马克·艾伦·维斯. 数据结构与算法分析 C语言描述 第二版,1996,198-204) 2020年10月第一版. 北京市西城区百万庄大街22号院3号楼1-9层: 机械工业出版社,2020年

附录.

各成员具体工作：

任务量配比:姜文渊40%,陈宥希40%,赵天畅20%.

姜文渊：学习python文档、scrapy网页爬虫框架、CSS和html基础，并确定好后续分工；程序总体框架设计与组织、根据网页结构确定好要爬取的网页，并批量进行数据爬取，同时进行了基础题里的英文页面爬取和提高部分的中文页面爬取，并进行数据清理，去除脏数据；分词阶段确定使用Trie树（字典树）的数据结构，并编写第一次汇报的word文档、进行汇报录屏；编写QuickSort.h头文件，并进行外部排序部分的编写，成功得到排序后的临时索引文件，便于队友进行倒排索引文件的生成，和搜索引擎的实现；对外部排序进行进一步的优化，分析其时间复杂度和空间复杂度；优化并调试程序，制作验收PPT

陈宥希：分工讨论，熟悉使用python；确定负责分析部分的编写，使用Python语言，主要算法为分词及为单词编号，并对此进行相关学习；开始编写英文分析的代码，分词算法采用线性扫描法，遇空格即分词，并过滤标点符号；单词编号算法采用Trie树去重保证一词一号；完成英文分析的编写，并开始编写中文分析的代码，分词算法采用jieba库中的分词函数（前缀词典+动态规划）；单词编号算法采用Trie树去重保证一词一号；完成英文分析的编写，对中英文分析代码进行测试，并调整代码；分析时间复杂度和空间复杂度；完善程序并准备验收；整合各成员编写的报告

赵天畅：整理思路并进行python的学习；初步整理出查询步骤的实现方法及具体实现步骤，并基本完成对python相关部分的学习；开始着手对算法进行设计，完成了事先对单词编号文件、网页编号文件的处理，同时思考并进一步完善后续实现思路；设计利用分词所得结果得到对应单词编号的查询算法，和利用所得单词编号得到对应网页编号的查询算法思路；补充对倒排索引文件进行的事先读取处理，并根据单词编号进一步由上述读取操作得到对应网页编号，并进行初步排序为计算网页权重做准备，计数权重并排序，得到出现频率最高的前20个网页，并经过查询得到网页链接；完善查询部分并进行一定程度的优化

**JIANG Wen-Yuan**, undergraduate, class monitor. His research interests include artificial intelligence and big data analysis.

**Chen You-Xi**, undergraduate. His research interests include artificial intelligence and big data analysis.

**Zhao Tian-Chang**, undergraduate. His research interests include artificial intelligence and big data analysis.

**Background**

With the rapid development of the Internet, search engines have become an indispensable part of people. With its powerful crawling ability, search engines have covered about 20% of the web pages in the whole network at this stage. Google (Google) has about 6 billion page coverage, and Baidu has reached 1 billion, and they have formed their own way of survival (selling technology, PPC, placing ads). Google is known as the greatest company on the Internet, and Baidu has become the second largest Internet company in China, all of which herald the great development of the search era. Search engines have brought great convenience to people who roam the Internet. On the other hand, the search of site portals is becoming more and more popular. When a website becomes bigger and stronger, its content must be very rich, and users want to find the content they need (such as the video they just watched yesterday, but it has been It is not displayed on the homepage), or if you want to see related content in this area, you need the support of search engines. Search engines are no longer a single search point. Taking Baidu as an example, it is not just natural results, but various structured display results. For example, if you search for Liaofan, you will immediately give information such as encyclopedia, movies, pictures, know, Weibo, news, etc. This is the best search result given based on user logs, and each result is obvious. Better than a simple natural result, the result of the hard work of many product people and programmers. Searching for a movie is two words, and a search result that occupies a very large space appears in front of the user. This is the high-quality display technology. The latest, hottest, user-friendly, all kinds of movies for you to choose from. Searching for the Men's Basketball World Cup, all aspects of the game, such as the schedule, live broadcast entrance, game report, statistics, etc., are covered in this small high-quality display. All of the above are beyond the reach of the basic Rank (sorting only by the back-end algorithm) strategy. When a user may only be very interested in one aspect of knowledge and not interested in other aspects of knowledge, topical search is required. For example, a user is only interested in news topical content, and news topical search can obviously better satisfy users. The efficiency of thematic search engines is higher than that of web searches for thematic content, and has received a lot of attention. For example, Baidu News, covering news on the Internet, domestic, international, military, financial and other aspects, users only need to enter their own If you are interested in news topics or content, the news about the news on major news websites will be listed immediately. If the user is just searching for no purpose, just click on the homepage of the special search website, and various news hotspots will be listed. These results are carefully selected by professionals in combination with people's interests, making it difficult for users to leave the page. Baidu knows that it solves all kinds of problems for people. As we all know, the Internet contains all kinds of knowledge, but how to extract it has always been a difficult problem to solve. If you go through the basic rank strategy alone, you may get a lot of relevant content, but the answer is completely guaranteed. Baidu Know appeared to solve this problem, providing a platform for everyone to submit questions, and everyone can answer this question. When the questioner really solves the problem, the answer becomes the accepted answer by netizens, and everyone can clearly feel that the answer is true and credible. Knowing that professionals will also be employed to answer all kinds of different questions gives a good assurance of the quality of the answers. The search side uses machine learning to pick out questions with relatively high quality answers and display them on the search side. When Baidu knows that it has done a lot of work, it can even give answers to questions that do not have accurate answers (such as emotionally ambiguous questions, such as: can pregnant women wear makeup) through the opinions of netizens. length. All these profoundly illustrate the popularity and diversity of the current development of search engines, the era of great development of search engines has come, and the research of search engines is of great significance to people's Internet life.