****

课程设计报告书

**题目：科学文献管理系统**

**学 院 计算机科学与工程学院**

**专 业 计算机科学与技术专业**

**学生姓名 李泓锦**

**学生学号 201830581039**

**指导教师 张见威**

**课程编号 045101681**

**课程学分 1.0**

**起始日期 2020.02.01**

|  |  |
| --- | --- |
| 教  师  评  语 | 教师签名：  日期： |
| 成  绩  评  定 |  |
| 备  注 |  |

**科学文献管理系统**

**一、选题背景**

在文化和科学技术不断发展的今天，有许许多多的科研工作者在为了各自不同的科研目标而努力研究，而如今也是一个信息共享，信息全球化和透明化世界，有许许多多的优秀的科学文献可以供我们学习，科研工作者们也不例外，为了做出更好和更科学的研究，科研工作者们也会不断借鉴和学习他人的优秀科学文献。所以科研工作者的日常工作离不开查阅科学文献，并对其中的信息进行分析、筛选、挖掘和管理。因此为了方便科研工作者们查阅科学文献，我们致力于制作出一个方便、准确、快速的科学文献管理系统。

我们的科学文献管理系统应解决的主要问题是:

F1. （必须实现） 基本搜索功能。输入作者名，能展示该作者发表的所有论文信息。输入完整的论文的题目，能展示该论文的其他相关信息

F2. （必须实现） 相关搜索。输入作者名，能展示于该作者有合作关系的其他所以作者。

F2. （必须实现） 作者统计功能。输出写文章最多的前100名作者。

F3.  (扩展实现)   热点分析功能。分析每一年发表的文章中，题目所包含的单词中，出现频率排名前10的关键词。

F4.  (扩展实现)  部分匹配搜索功能。给定若干个关键字，能快速搜索到题目中包含该关键字的文章信息

F5.  (扩展实现) 聚团分析。作者之间的合作关系可以看成是一个图，每个作者对应一个顶点，任两个作者之间如果存在合作关系，则在两个顶点之间建立连边。这个图中的每一个完全子图我们称为一个聚团（所谓完全子图指的是该子图的任意顶点都和该子图的其他顶点有连边，完全子图的顶点个数称为该完全子图的阶数），请统计整个图中各阶完全子图的个数。

F6.  (扩展实现) 可视化显示。通过图形化界面，展示作者之间合作关系图及其相关文章信息。、

所以我们需要达到的技术要求首先是选择某一编程语言设计该科学文献管理系统，然后是采用某一数据结构存储文件中的部分信息以便于查询，并且要解决大量数据存储的问题。而且我们需要针对读取文件建立一个类，该类主要实现在读取每一篇文章时要进行的操作。在完成对数据的存储之后，我们还需要完成的技术就是把提取出来的数据保存在文件中，这样就可以通过读取文件中的信息优化读取的时间，还需要达到的技术要求就是使用该编程语言设计界面，界面要通俗易懂，方便操作。

指导思想：通过上学期数据结构的学习，我们学习了许多可以存储数据的数据结构以及快速查找的算法，本科学文献管理系统的实现将基于我们数据结构课程所学习的知识进行设计，并且通过该科学文献管理系统深入理解和掌握所学知识。

**二、方案论证(设计理念)**

本次项目的运行环境：Pycharm编辑器、Win/Mac系统

在通过对本次实验项目的分析之后，我们通过对已给的数据文件进行分析，因此XML文件极为庞大，所以我们使用了SAX方式对XML文件进行解析。对于语言的选取上因python语言的功能强大，且我们运用此语言较为熟练，因此我们使用了python语言进行了程序的编写。之后我们又发现我们此次数据量极为庞大并且我们需要进行的主要操作是查询功能，所以我们需要选择对应的数据结构来进行数据的存储并能够尽量在最短的时间内获取对应信息。而通过我们过去《数据结构》课程的学习，我们发现B+树对于量很大的数据集有着极为便利的操作。因此在本项目之中主要使用了B+树进行数据的存储。在我们将XML文件中的数据读取到B+树中之后，我们已经实现的主要操作有:

（1）基本搜索功能。输入作者名，能展示该作者发表的所有论文信息。

输入完整的论文的题目，能展示该论文的其他相关信息

（2） 相关搜索。输入作者名，能展示于该作者有合作关系的其他所有作者。

（3） 作者统计功能。输出写文章最多的前100名作者。

（4）  热点分析功能。分析每一年发表的文章中，题目所包含的单词中，出现频率排名前10的关键词。

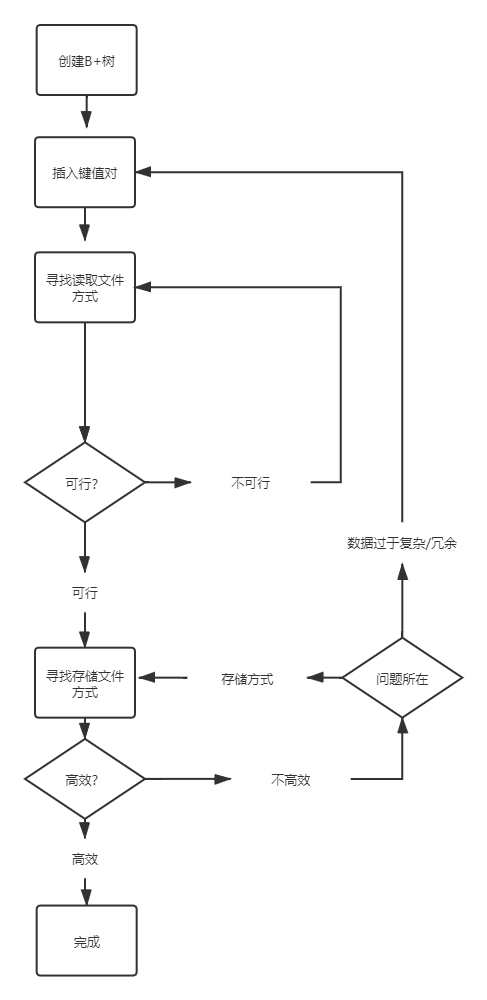
（5）部分匹配搜索功能。给定若干个关键字，能快速搜索到题目中包含该关键字的文章信息

（6）可视化显示。通过图形化界面，展示作者之间合作关系图及其相关文章信息。

之后为了减少每一次进行加载的时间。我们将B+树存到txt文件中，以方便下一次直接进行加载，防止再出现因再次构建B+树和读取数据造成的加载时间过长，经过多个文件存储方法的比较与实践对比，我们发现pickle库最适合于我们将数据存储，所以最后决定使用pickle库实现文件保存。最后我们对比了两种界面库PyQt5和Python内置的Tkinter库，发现PyQt5的界面更加美观，所以使用了PyQt5来进行界面的撰写。

三、过程论述

本系统的总体思路是通过B+树存储文章索引，使用SAX方式对XML文件进行解析并获取索引，用pickle模块写入二进制文件，采用PyQt5模块进行界面的编写。

****

A. 键值对

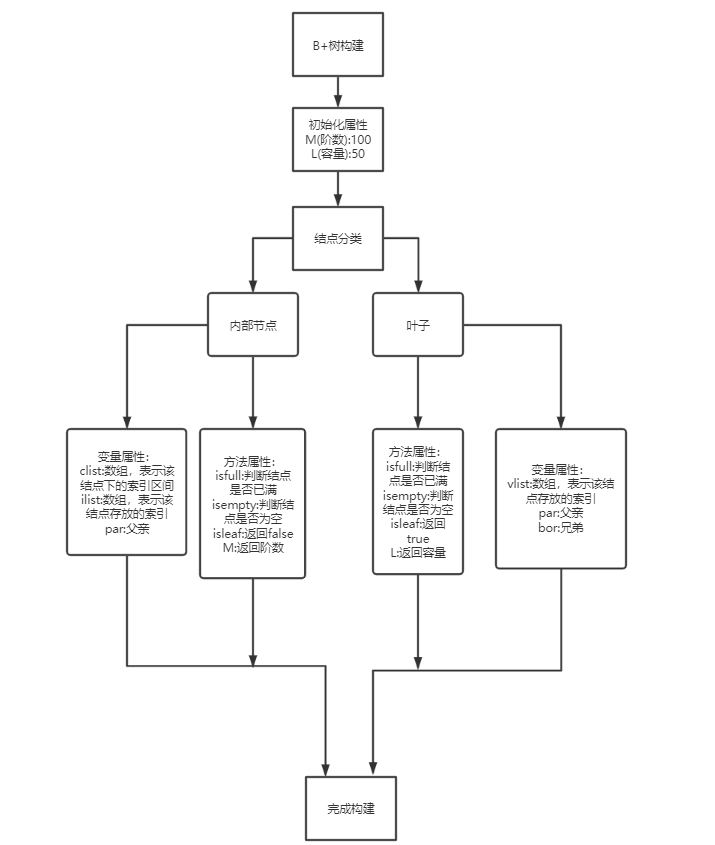
首先我们需要定义一个KeyValue类，用来插入到B+树之中。在类中我们定义了私有化函数大于小于等于，重载相关运算符。

定义两个KeyValue对象比较的是键而不是值。本类的初始化中会有两个值分别是 key,value。

而我们还需要编写\_\_str\_\_函数，使此类用print打印时能按照如下方式进行输出。至此我们B+树中元素的类已经定义完毕。

B. B+树

之后我需要实现我们的B+树，其具体流程如下：



因我们使用python语言，所以没有相关的类型的说明，而因此对于值类型是可变的（Value值）。在我们的B+树之中我们一共又定义了两个类分别是InterNode,Leaf。下面对这两个类进行分别的阐述，而其相应的流程图如下：

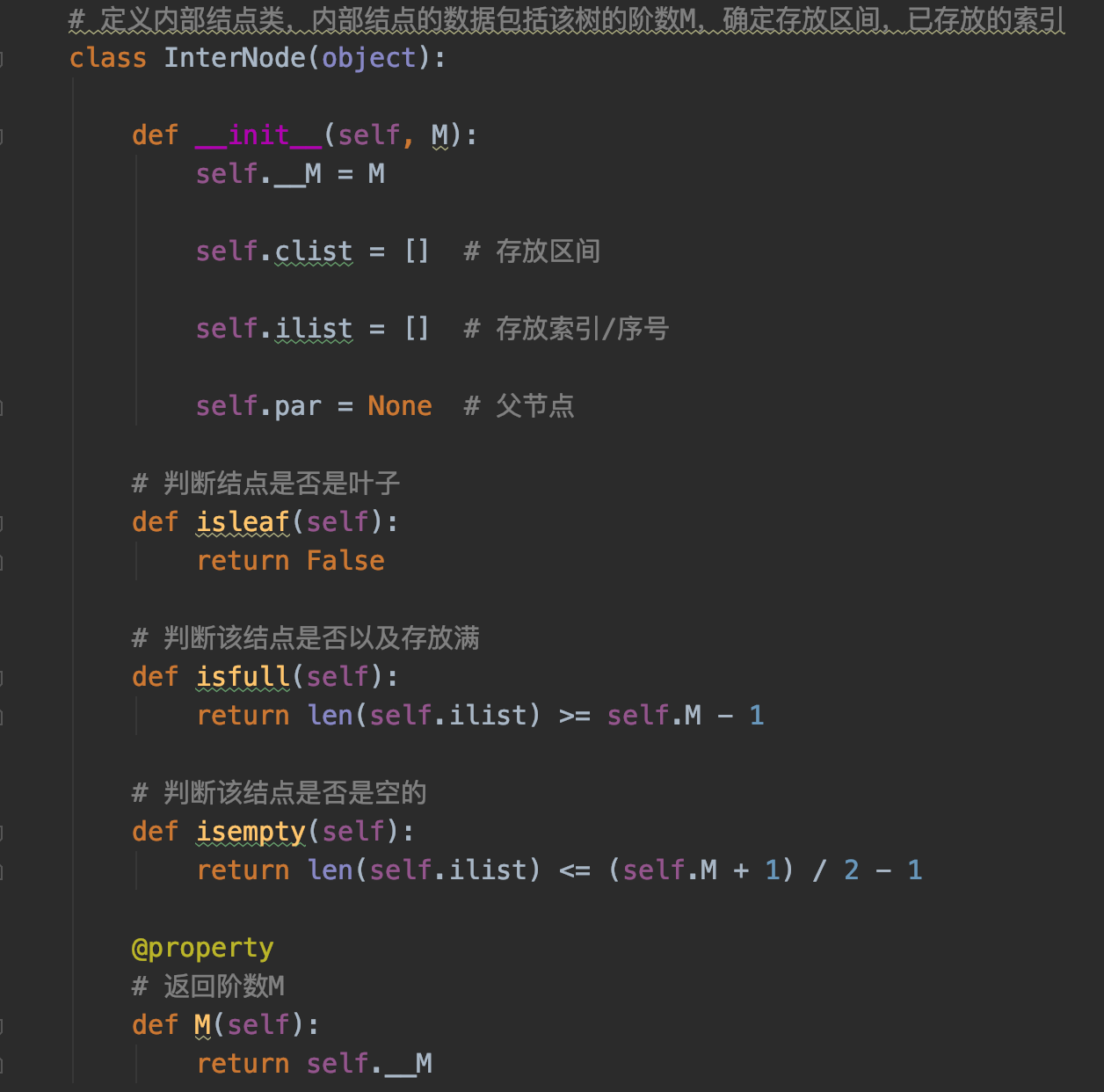


图3 InterNode的定义

如上图3InterNode类用来定义在B+树中的内部节点，此类中需要M作为阶数用来定义一个内部节点最多存储的键值对，并且按照B+树的定义所以又不能使得内部节点存储元素过少，使其元素个数不能少于阶数的二分之一。并且其中clist存储的是该节点下(包括孩子)含有的索引，vlist存储的是该节点含有的索引。书写了相关函数用来判断此节点是否为空或满。

无独有偶，Leaf类用来定义在B+树中的叶子节点。与类InterNode一样，我们同样使用L来定义我们叶子节点的阶数，并且vlist列表存储的是本节点所有的索引(因其没有孩子所以不包含孩子的节点)。如下图4



图4 Leaf类的定义

在我们声明了上述两个类的构造之后，我们可以进行大类的初始化。我们在初始化对象的时候需要传入两个参数分别为内部节点和叶子节点的阶数。并初始化一个根节点，为以后B+树的构造做下铺垫。并且我们定义类相关的函数用来返回两种节点的阶数。

我们已经初始化此类并进行了相关函数来返回对应属性，下面进行具体功能的实现。因本项目不要求删除和修改功能，因此我们一共有四个功能：

1. 向B+树中插入值 insert
2. 从B+树中按照键来搜索值 search
3. 遍历叶子的所有存储元素（输出所有及对应高度）
4. 输出所有叶子的键和值（返回一个列表）

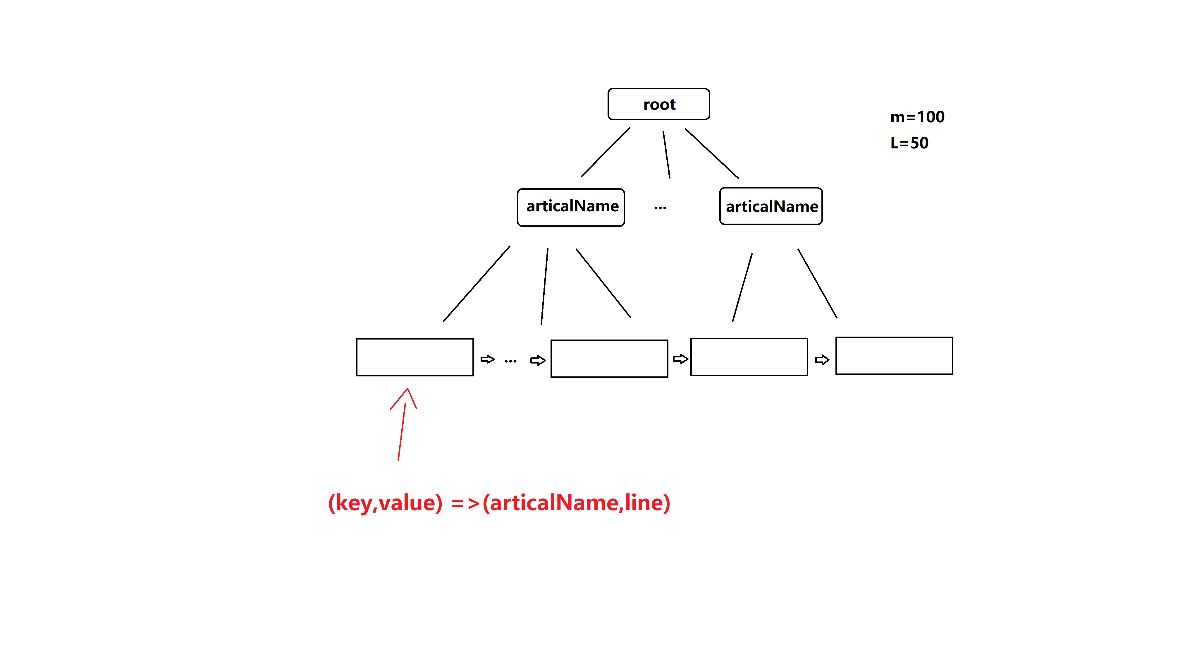
一．插入功能：

在本功能之中，我们定义了一个辅助函数如下图，我们对此函数进行了装饰，使其能够使用其他的函数。对于不同情况调用不同的函数。我们取根节点进入插入节点函数。插入对应的节点，首先需要判断是否是叶子节点，如果是的话，就通过调用bisect\_right方法进行二分查找，找到对应的位置并进行insert()方法(之后叙述)插入，如果叶子节点满了就将其分裂，反之直接退出。如果是内部节点，那么我们首先判断是否满，如果满就把节点n分裂，利用上面写的节点分裂函数，在将其进行插入，如果没有满则利用二分查找找到对应位置进行插入（进行递归调用）。如下图6 insert\_node函数的定义。



图6 insert\_node函数的定义

我们将节点分裂分为了两部分，分别是叶子节点的分裂和内部节点的分裂，如下图7所示为叶子节点的分裂。在本函数中，我们首先确定一个临界点，用来判断是当叶子存储对应临界值。并且创建了一个新的叶子对象，将多余的元素通过切片的方式复制给了另外一部分。因此，我们原来的节点分为了现有的节点和新节点。之后进行判断，当叶子的节点父亲不为空的时候，将现有节点的最后一个索引对应的值插入到原节点的父亲节点中，在将新生成的节点第一个索引向父亲节点提交，并将原节点的父亲节点复制给新节点的父亲节点属性。而当原节点的父亲节点为空的时候，新建立一个节点，并将其clist赋值为儿子对应的节点，并提交现有节点的最大索引值作为索引，将新的生成的newroot作为父亲节点和树的根结点。最后将原节点的vlist值改为之前的节点值减去新节点对应的索引值将其兄弟变为新生成的叶子节点。如下图7叶子节点的分裂。



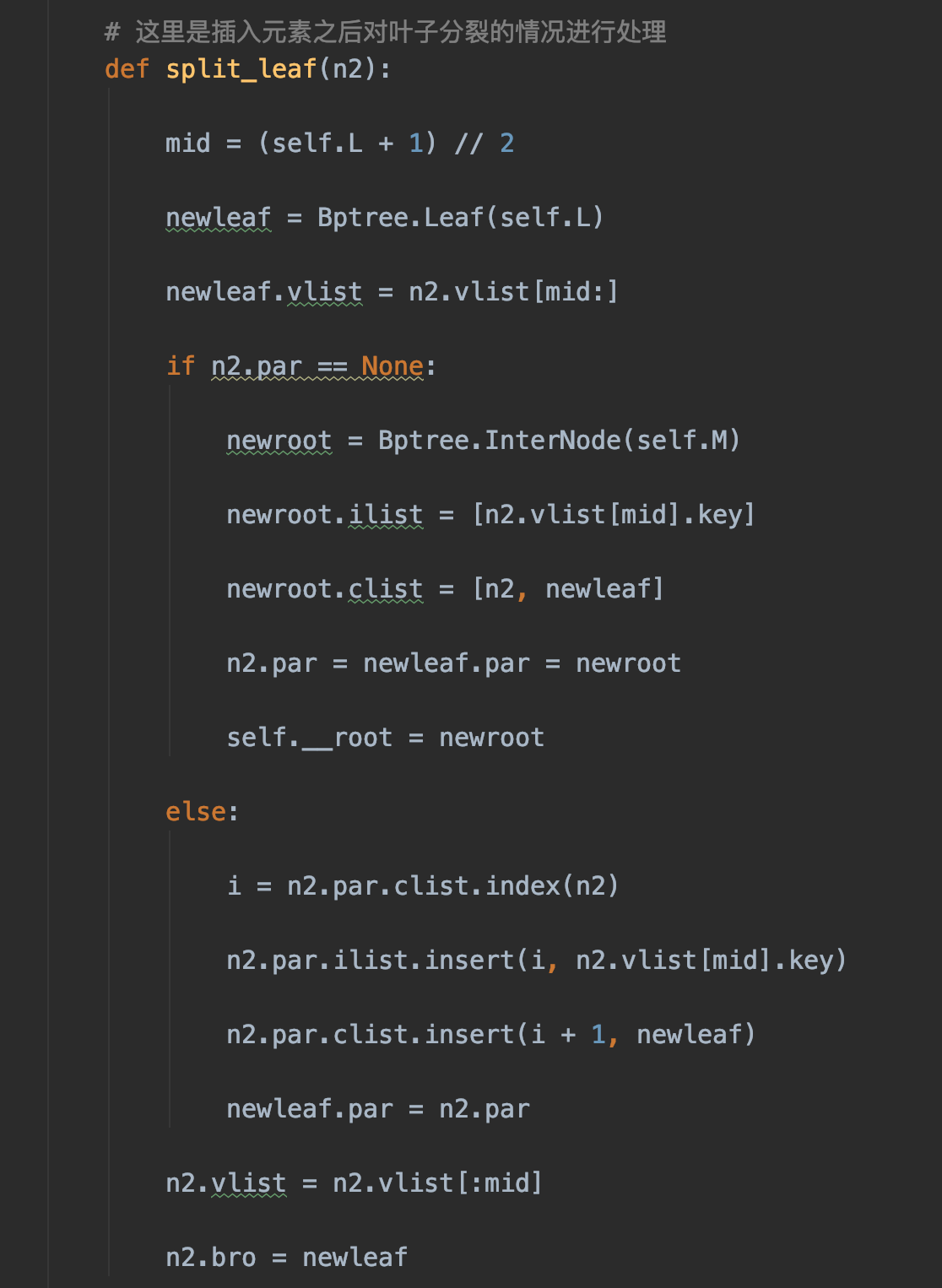
、

图7 叶子节点的分裂

而对于内部节点的分裂类似于叶子节点的分裂，之后不再赘述。

1. 搜索功能

我们在传入参数的时候规定输入两个参数，分别为键的上界和下界。首先我们需要定义一个result用来返回最后的结果。编写search\_key()函数，此函数共有两个参数，分别是节点和要搜索的key值。此函数能够实现对于某一节点中键的查找。在此函数中首先需要判断此节点的类型，如果是叶子节点则进行二分查找查找到对应值进行返回，如果是内部节点则在本内部节点中搜索并找到对应的位置，通过此位置再次使用搜索键的方法进行递归调用直至找到对应节点。如下图9



图9 搜索函数

之后我们回到函数的主体之中，首先按照其上下界分为了三种类型：1.有下界无上界 2.无下界有上界 3。有上界有下界。

1. 如果下界是空的，那么就不断的从所有叶子节点中寻找小于此key值的所有元素，加到result中，直到无叶子节点。 如图10

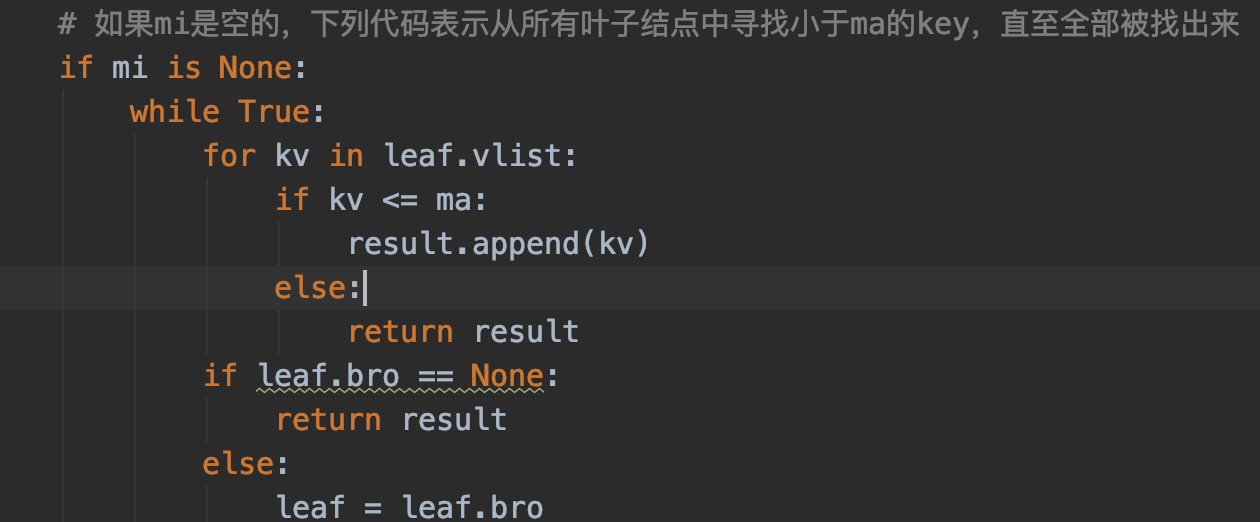


图10 无下界的搜索

1. 如果上界是空的，那么就从等于下界的下标位置处开始进行添加，不断将元素加入到列表中，与上同。如图11

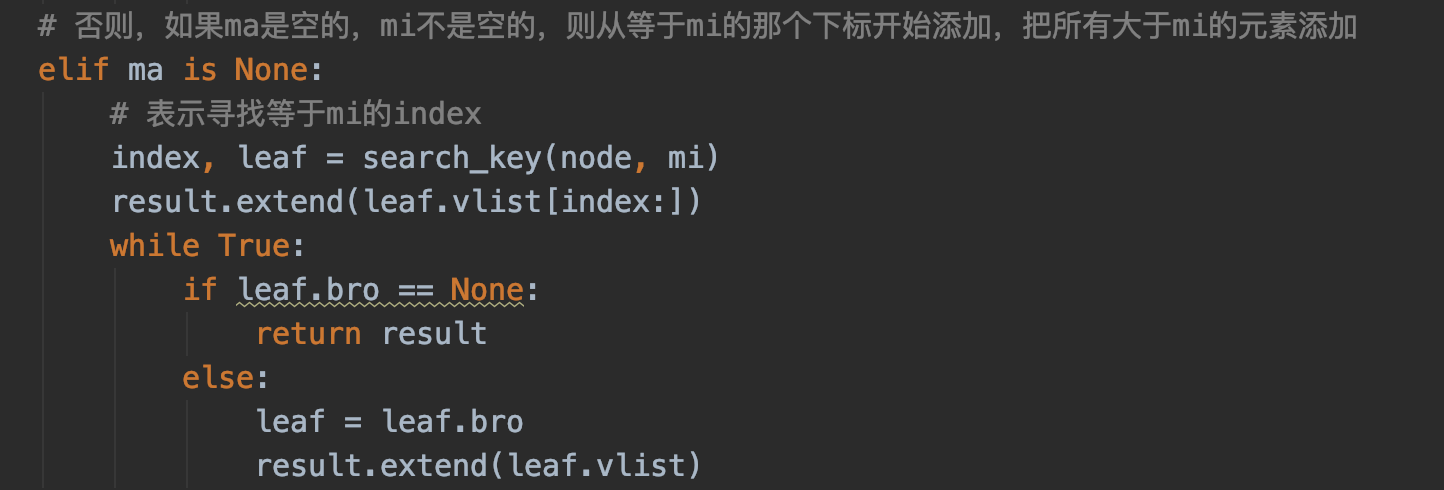


图11无上界的搜索

1. 如果上界和下界都存在的时候，我们首先判断上下界的值是否相等，若相等的话就将其看为对一个元素的搜索，仅找对应键即可，使用search\_key来找到对应应该存在的位置和节点，之后试着去判断对应的位置是否有值以及是否相等，如果不相等就直接返回，想等就将此值加入到列表之中之后进行返回。如图12

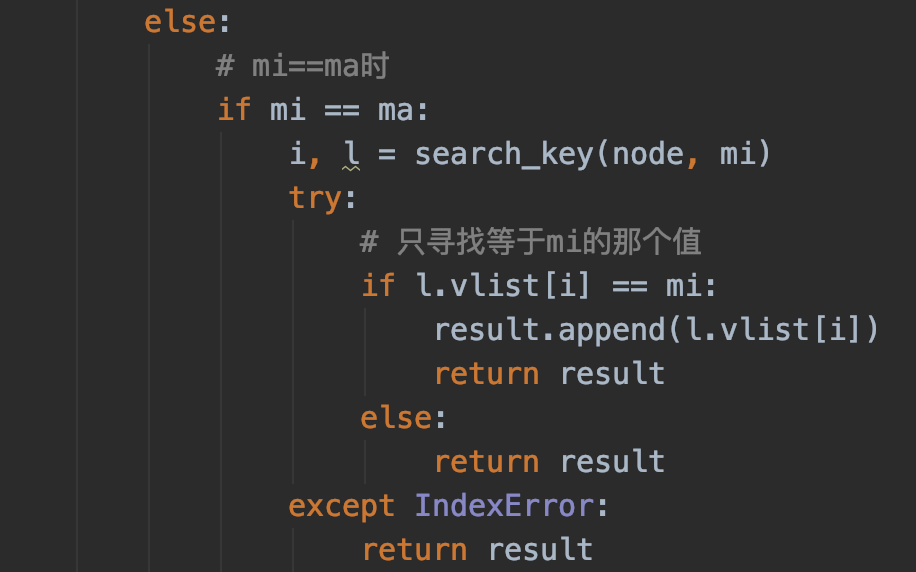


图12 上下界一样时的搜索

当下界和上界不一样的时候。同样的通过search\_key来找到对应的上界节点位置和节点和下界节点位置和节点。如果是同一个节点就直接利用切片加入到结果之中，如果不是就通过类比第一二种情况，不断访问兄弟节点直至为相对应的节点或者是为空，最后直至返回result结果。如图13

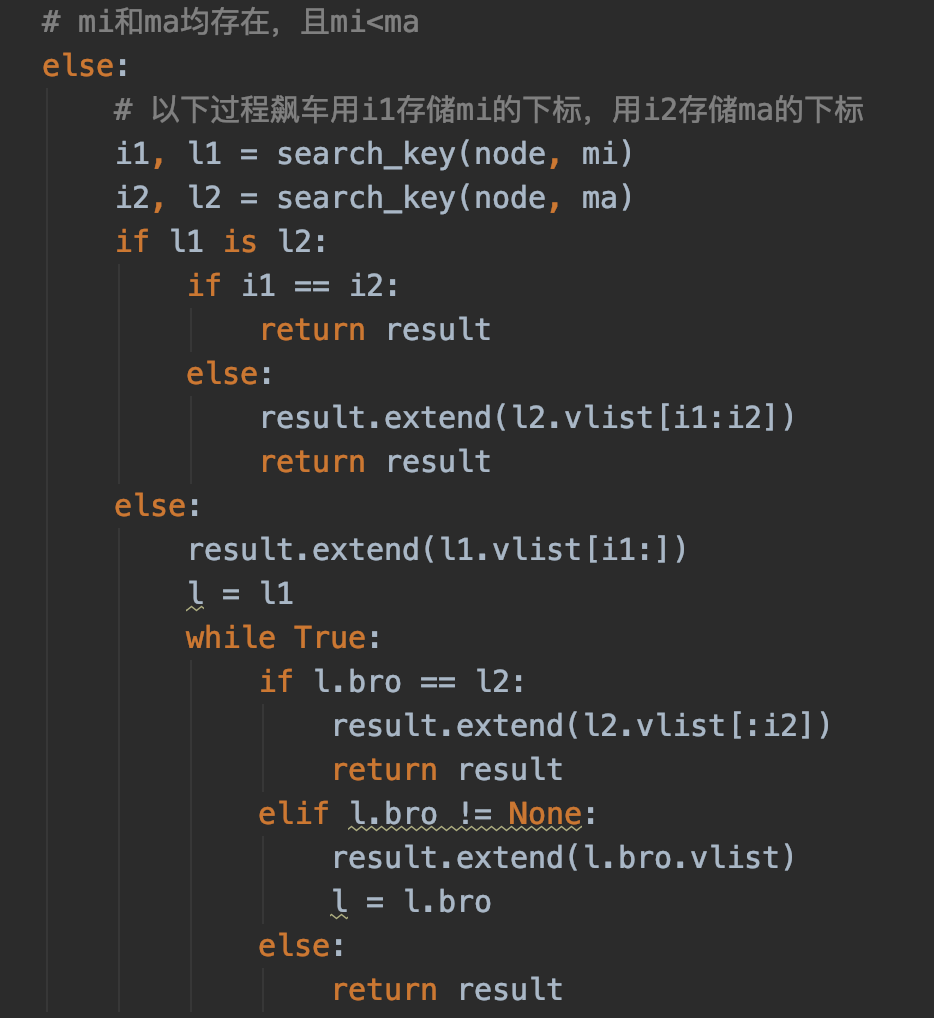


图13 上下界不一样的范围搜索

1. 遍历功能

定义一个result，将所有叶子节点中的键值对加进这个列表中，返回列表，较为简单。

1. 输出所有叶子值

此功能对于我们代码调试阶段十分有用，可以通过show函数直接进行对于元素的输出，对于检查树的正确有着很明显的帮助。声明一个队列和高度，首先将根节点压入队列，之后不断进行弹出左元素。如果弹出来的节点不是叶子，那么将输出本节点对应的高度，以及本节点包含索引的列输出出来，高度加一并将此节点中的clist(包含其孩子的所有索引)以列表的方式加入到队列中。如果此节点为叶子节点，通过遍历vlist叶子本身包含的所有索引，将对应的索引和高度进行输出。如图15

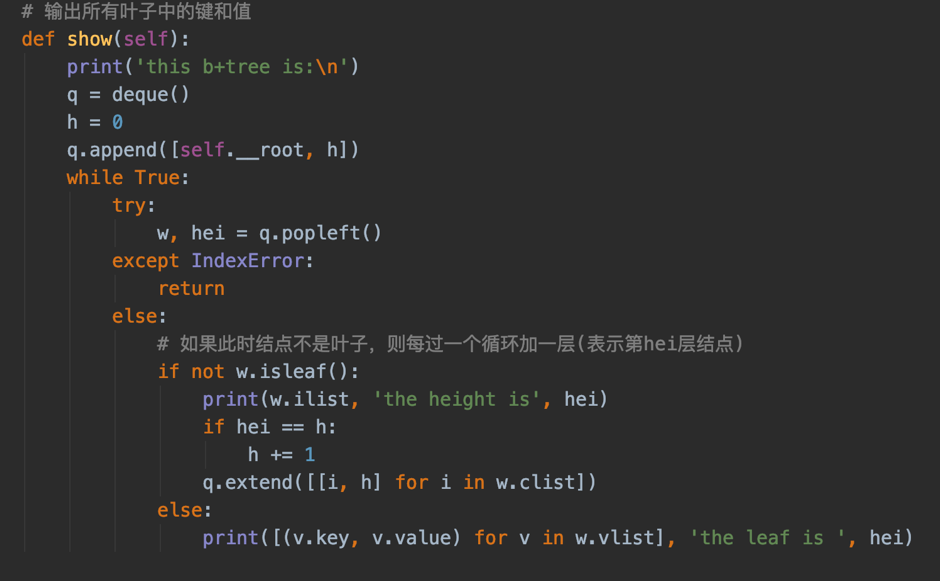


图15 打印输出所有叶子的键值

至此我们所有的B+树的构建和函数含义已经叙述结束，下面进行总类的构造。

C. MovieHandler总类的构造：

一．准备工作

首先我们通过分析整个XML文件，发现只有SAX方式对超大XML文件进行解析并读取相关信息，而比如ElementTree都不支持大文件的读取。在我们确定使用了SAX进行解析之后首先我们需要定义一个XML解析类，并继承于sax，并且我们需要实现我们的回调函数分别是：startElement，endElement，characters.这一共三个函数。在sax解析XML文件时，遇到<>会调用startElement，遇到</>会调用endElement，遇到文本内容的时候会调用characters进行读取。而对于dblp本身的文件，我们发现其中有类似于&nbsp;等之类的符号，上网查找我们需要一个dtd文件进行约束，对于上述字符可以进行解析(诸如希腊字母、法语、中文汉字等)。至此准备工作已经结束。

手机屏幕截图

描述已自动生成

二．初始化属性

而在MovieHandler初始化函数中，总共初始化如下属性。如下图16

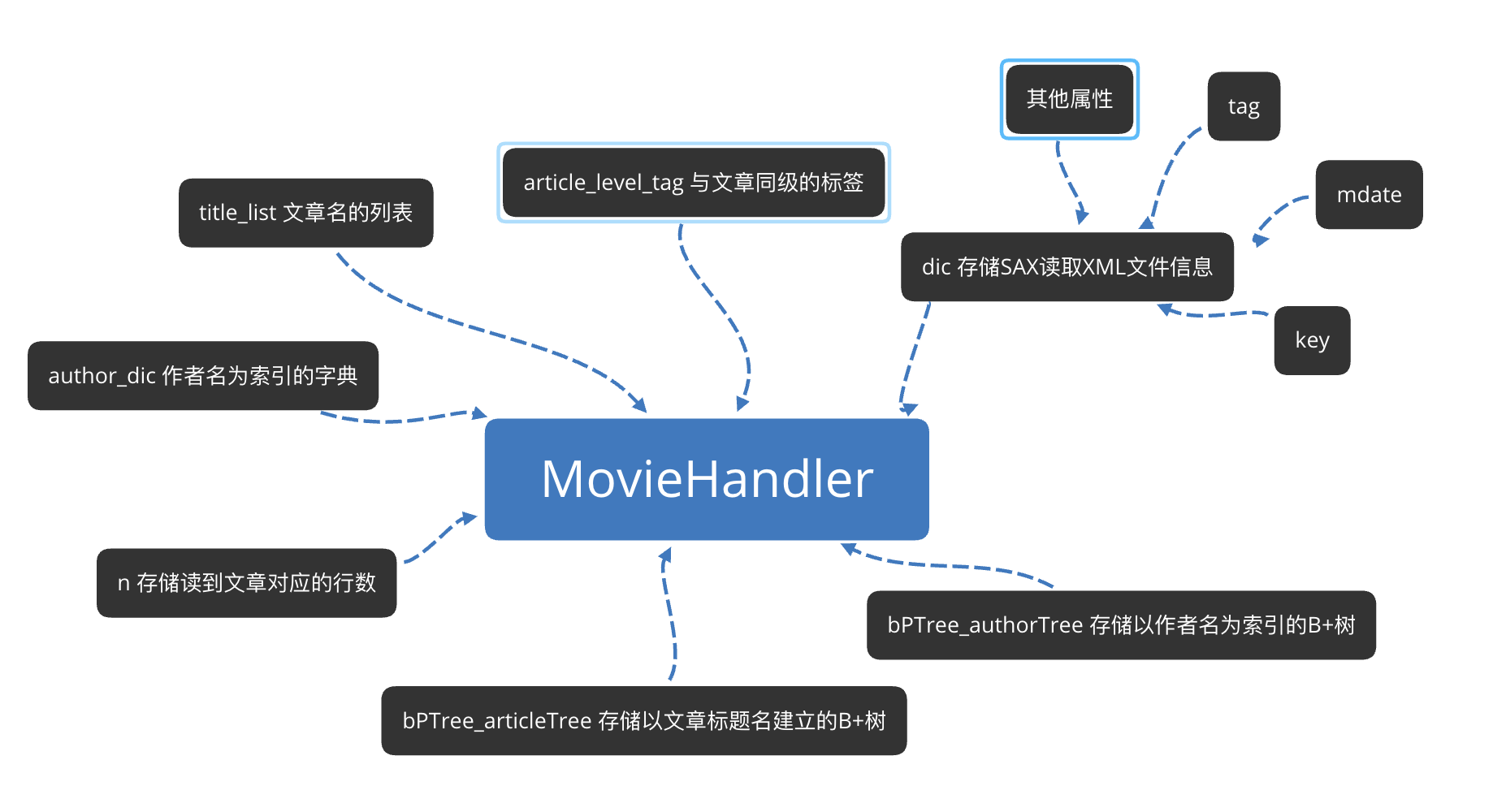


图16 总类中成员变量

之后我们分别解释对应属性所代表的含义：

1. 在我们进行文章读取之前需要寻找出目前为止共有多少和文章标签同级的标签，我们定义了article\_level\_tag列表用来存储和文章标签同级的标签，如果遇到了这个标签，则代表着一个新的文章信息的读取，那么就需要对字典清空，并将这个标签包含的属性进行初始化。如下图17

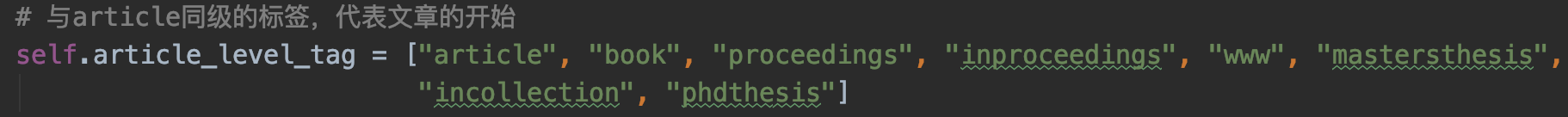


图17文章标题同级标签

1. 我们定义了ignore列表和hotword进行热点词的判断，通过我们日常生活经验以及上网查找，搜索出对应的介词、不定代词等等无关紧要的词汇，在进行热点词汇的分析时，首先需要判断单词是否在此列表中出现，出现则不进行操作，否则在hotword中寻找并将对应频率加1。如下图18

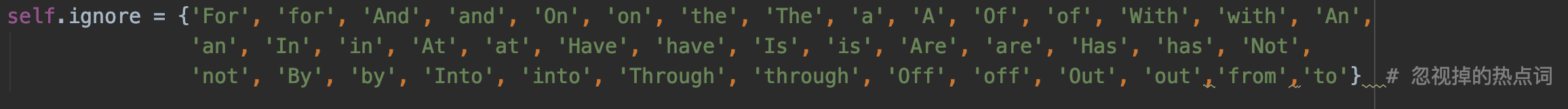
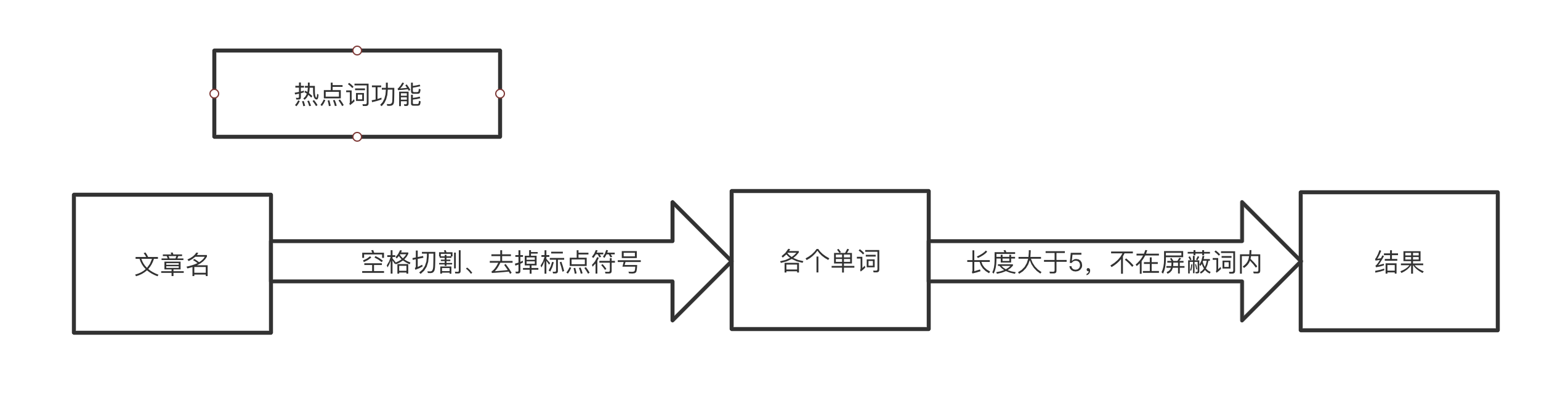
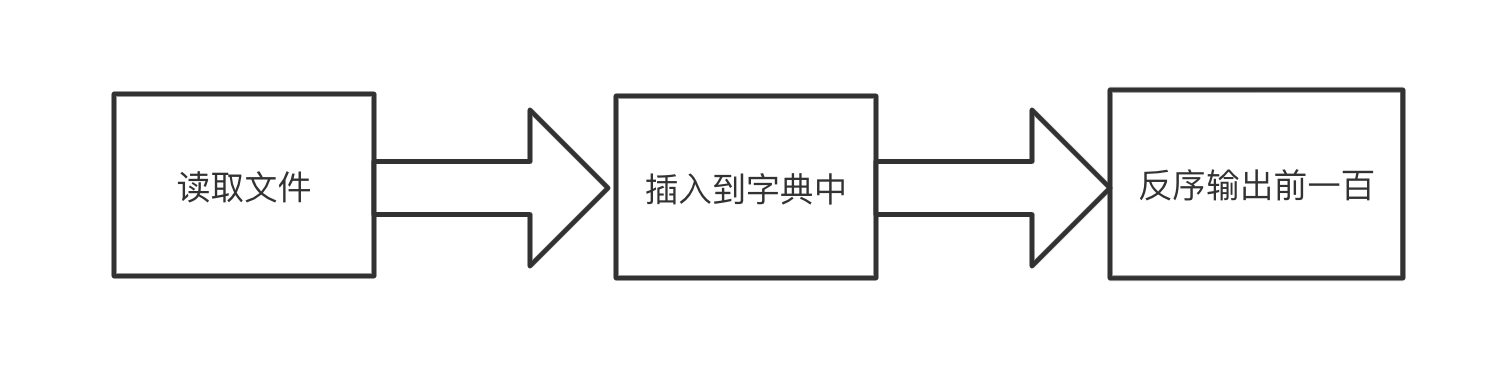


图18 忽略掉的热点词汇



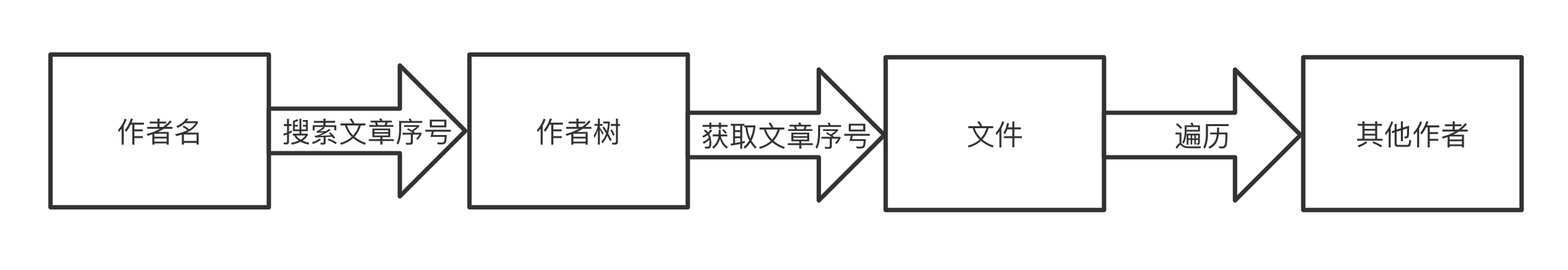
1. 我们定义了一个整数n,初始化为0，n代表这我们读取到了第几篇文章，如果遇到了article\_level\_tag同级的标签就加1，n代表着本文章是在XML文件中的第几篇文章。n作为值存储到两颗B+树之中，从而进行信息的搜索。在我们使用B+树的查找功能时，会返回一个以n为元素的列表，我们用这个列表去XML文件中寻找即可得到相应信息。
2. 之后我们定义了一个author\_dic的字典，此字典用来存储作者名为键，写过文章对应序号为值。在我们直至读到文件的结束之前都会不断向其中插入值，当遇到文件结束标签的时候会将此字典中所有值以键值对的方式插入到以作者名为键的B+树中。
3. author同样是一个以作者名为键的字典，不过值是一个整数，用来存储所有作者的论文数。

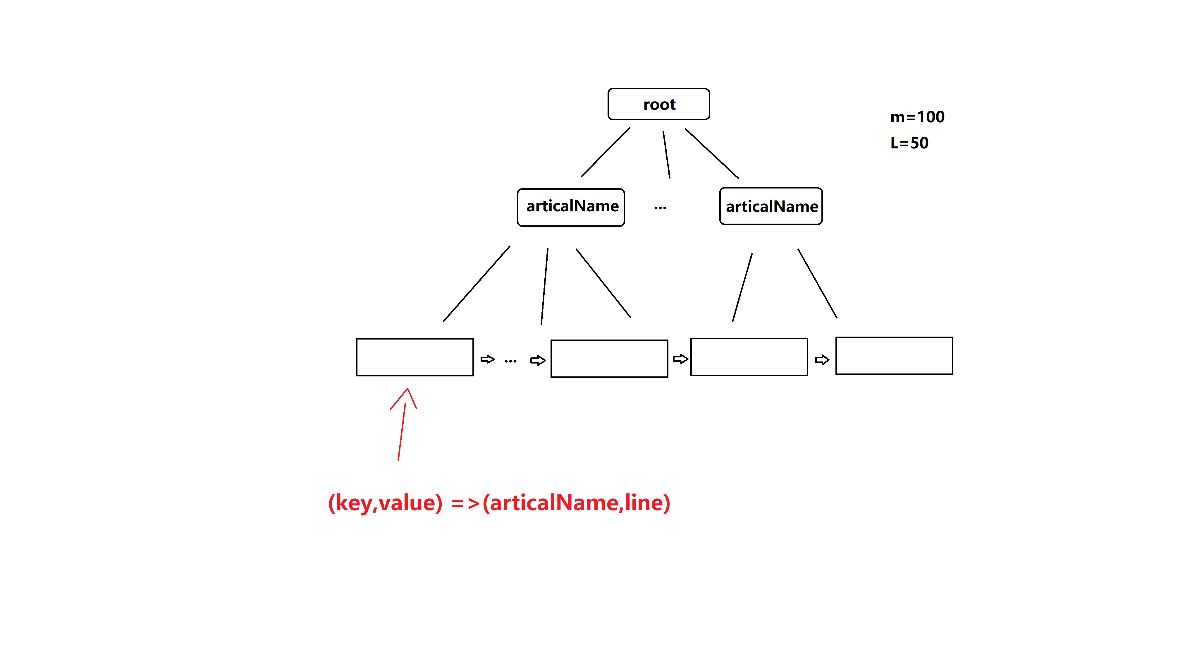


1. 我们定义了两颗树分别是bPTree\_aritcle和bPTree\_author。前者是以文章名为索引，文章序号列表为值建立B+树(我们发现title的值是可以进行重复的)，因此我们将文章对应的序号加入到这个列表中。以作者名为键的B+树同理。

卡通人物

描述已自动生成





1. 我们定义了title\_list用来存储我们所有文章的标题名，当我们进行部分匹配的时候会用到，此为一个字典，以文章名为键，文章序号为值(因文章标题可以重复，所以值可以为多个)。
2. CurrentData用来存储现在读到的标签名，在开始标签函数中，将tag赋值给此变量。
3. name\_string title\_siring 用来存储对应的文章和标题字符串，因为我们发现出现不合法字符时会自动停止，所以需要用一个字符串将断开的字符串进行拼接
4. 字典dic，用来监视并存储现在已经获取到目前标签的信息，在我们遇到与article同级的标签结束标签时，我们可以获取这个字典中的所有信息，此信息即对应文章的所有信息。
5. 回调函数的实现
6. startElemen函数的实现。我们已知传入两个参数，分别是标签和标签中包含的属性，通过查阅文档发现属性有mdate和key，因此从attribute中获取属性并将其赋值给字典，在我们这个函数之中我们通过对于标签的判断，如果遇到了和article同级的标签我们就对n进行+1操作，代表本篇文章在XML文件之中是第几篇文章。如下图19



图19 开始标签调用的回调函数

1. characters函数的实现。当我们在读XML文件开始标签函数调用结束之后进行调用此函数。首先判断我们读到的标签是否是与文章同级的标签，是则不用。如若不是则进行进一步的读取。之后我们需要判断是否是换行符，如果是的话就需要进行忽略，再判断标签是否是author,title标签，通过拼接字符串的方式将这两个标签中的内容进行拼接防止为止的错误发生。最后将我们读到的内容加入到字典dic之中。如下图20



图20 读取字符时调用的回调函数

1. endElement函数的实现，在我们进行结束标签的时候调用此函数。首先需要对年份进行拼接，因为我们在代码调试过程中发现有的年份分成了两部分(1916分成[19,16])，于是我们用过此逻辑将列表转化为字符串。之后我们判断结束标签是否是与文章同级的标签，我们分为三种情况，分别是文件的结束标签、与文章同级的结束标签、其他标签。
2. 其他标签。包含了author,title 标签。1.title标签，我们通过此结束标签将title标签的缓存即title\_siring清零，判断标签存在与否，如果不存在则初始化，存在就加入列表中。2.author标签，除了同样要实现title标签对应的功能外，为了实现功能统计文章数前100的人，所有如果是作者标签的话，首先判断author字典中是否存在作者，存在就+1，不存在初始化为1，这样能够统计出每一个作者写的文章数。如图21



图21 其他标签结束时的处理

b.与文章同级的标签。当遇到此标签时代表一篇文章已经结束，此时dic中对应的所有信息代表了文章相应的信息。首先我们需要确定字典不为空，并对title和author属性进行不同的操作。如图22 与文章同级的标签处理

若字典中有title标签：1.首先在以文章名为键的B+树中进行搜索，结果不为空则加到列表中，为空则使用键值对的方式插入到B+树之中。并在我们的title\_list中加入对应的标题名。之后若有year标签，则对应进行热点词的统计。我们首先使用映射的方式去掉对应的标点符号。之后将转化的字符串按空格进行分割。我们将对应年份对应的单词加入到字典中。同理无为1，有加1。

若字典中有author标签，将对应的作者逐个加入到author\_dic的总字典中，还是若存在则加到之前的列表中，不存在则初始化列表并赋值。

c. 文件结束标签时，此时所有作者已经读取完毕，以键值对的形式，作者名作为键，文章序号组成的列表作为值插入到 bPTree\_author的树之中。如图23 文件结束标签的处理

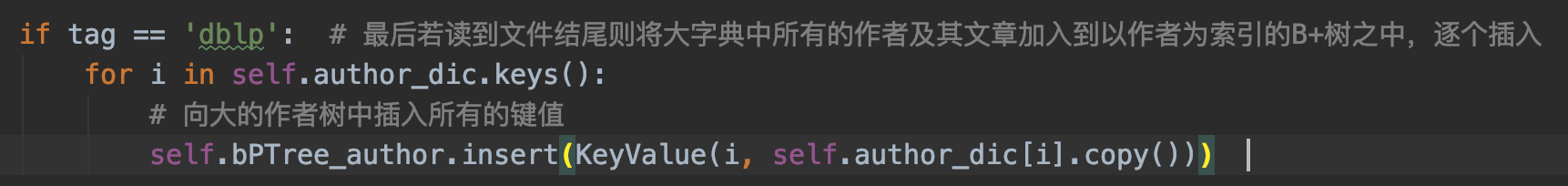


图23 文件结束标签的处理

至此主类的构造已经结束，下面进行主程序的实现。

D. 主程序的实现

首先使用sax 方式对于XML文件进行解析，获取主类对应的对象，解析路径。如下图24 创建XMLReader

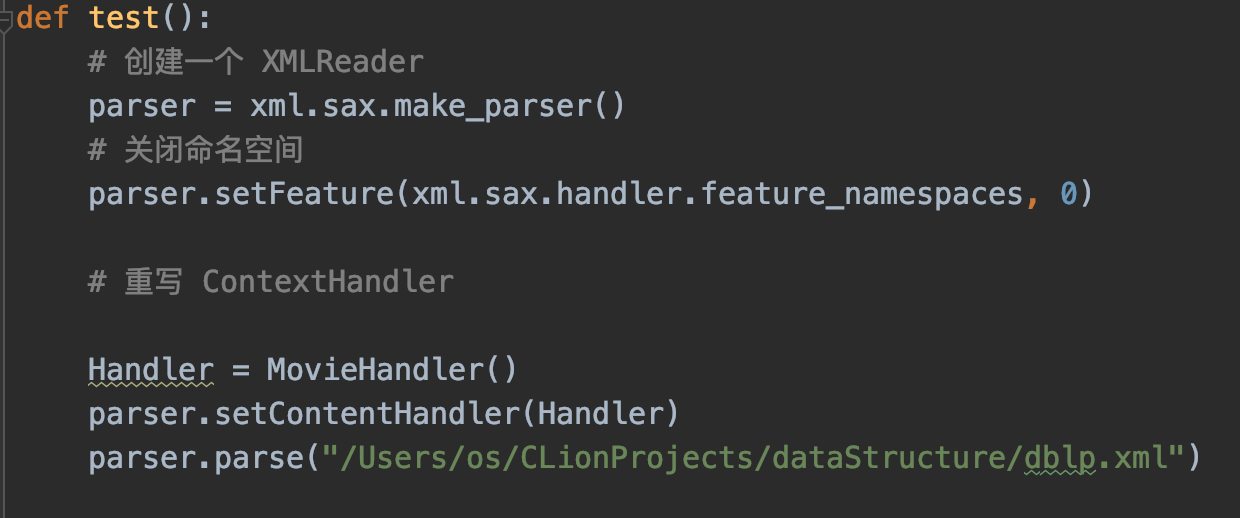


图24 创建XMLReader

进行解析之后使用pickle模块将B+树，字典以二进制的形式存储到对应txt文件之中。如下图25 使用pickle模块



图25 使用pickle模块进行数据的存储。

至此写入数据已经写入完毕。

E. 读取数据

在我们通过load加载数据之后，使用bPTree中的搜索功能返回一个序号的列表，我们将此列表传入getContent函数之中，进行信息的读取。首先在函数中定义一个Read类，此类和主类MovieHandler类相似，都是使用了SAX方法对于dblp文件进行解析。我们首先将getContent的lis传入此类构造的对象，之后我们带着这个列表去XML文件中寻找对应的信息，如果对应的文章序号在target之中，就进行读取，否则就进行跳过操作。每一次读取成功之后就将对应的序号加到列表中。每次进行判断是否已经读到的文章数目已经和target相等，若相等则不需要再往下进行读取，直接抛出异常(节省读取的时间)，如下图26抛出异常。具体代码类似主类，不再粘贴。

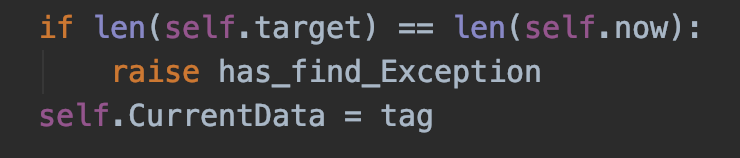
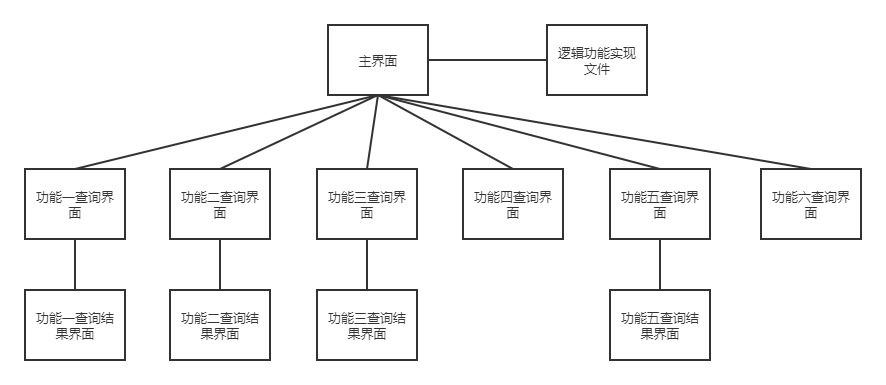


图26 长度相等抛出异常

我们抛出异常之后不对异常进行处理，直接返回对应的字典即可读取完毕所有信息。

F. 界面的编写

主界面使用了网格布局作为布局方式，这样可以让界面自由拉伸。界面中有六个pushbotton和一个label，每个pushbotton的clicked槽分别连接到六个功能的函数。在主界面的初始阶段，会使用pickle模块中的load方式读取保存在本地中的文件信息并加载进主界面的类中，在调用各种功能的时候传递这些信息。

在我们功能1的实现之中，我们定义了对应的页面显示函数，并且包含了pushbotton和textedit，最后如图又有对应的按返回时结束此页面并返回到主程序之中。如图29 功能1的页面

四、结果分析

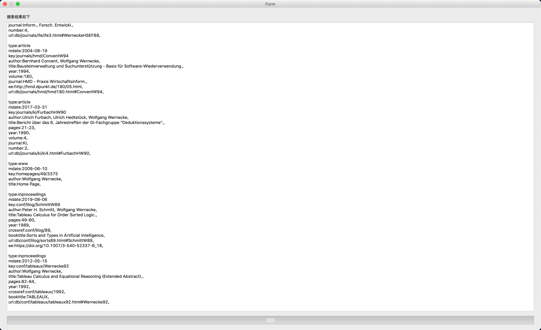
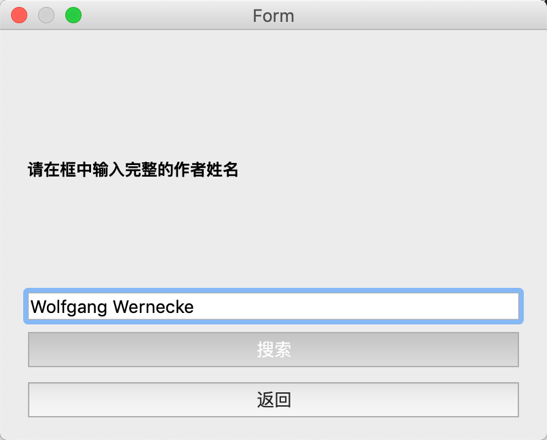
对研究过程中所获得的主要的数据、现象进行定性或定量分析，

通过上面代码的实现我们可以得到如下结果：

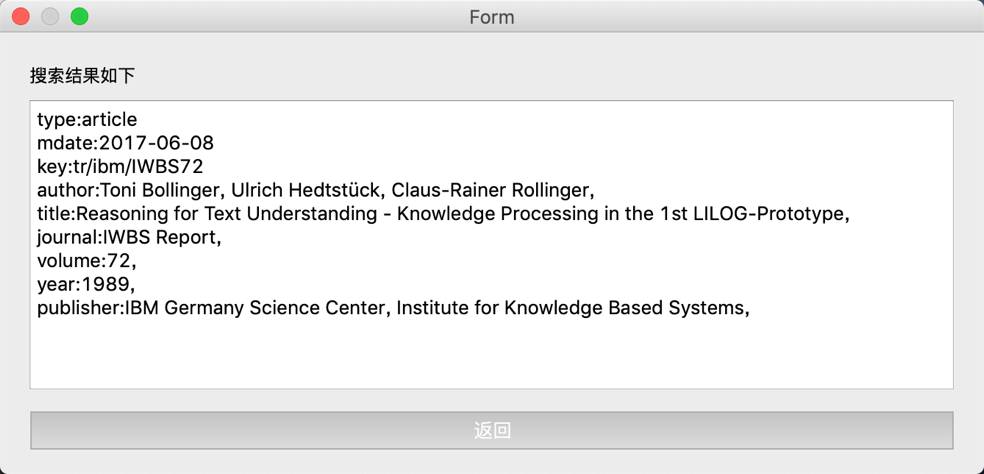
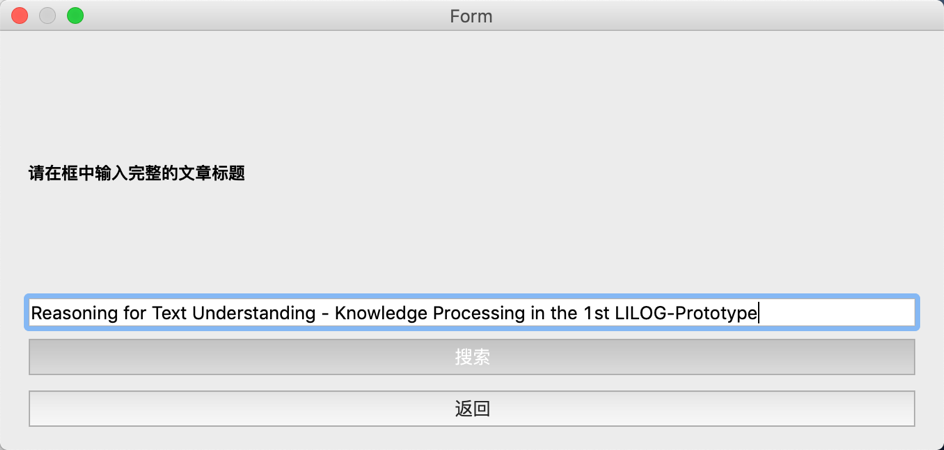
1. 主界面中有六个按钮，分别对应着六个不同的功能。



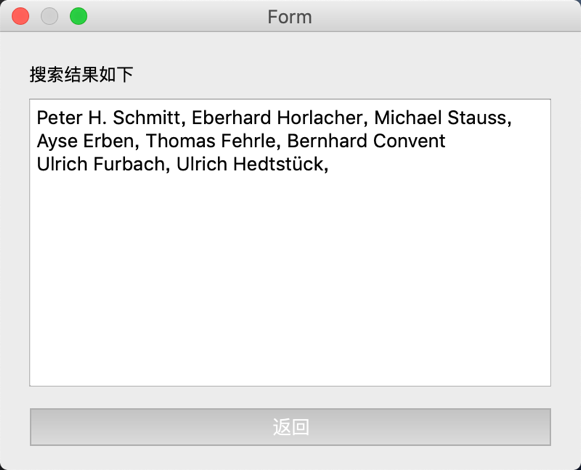
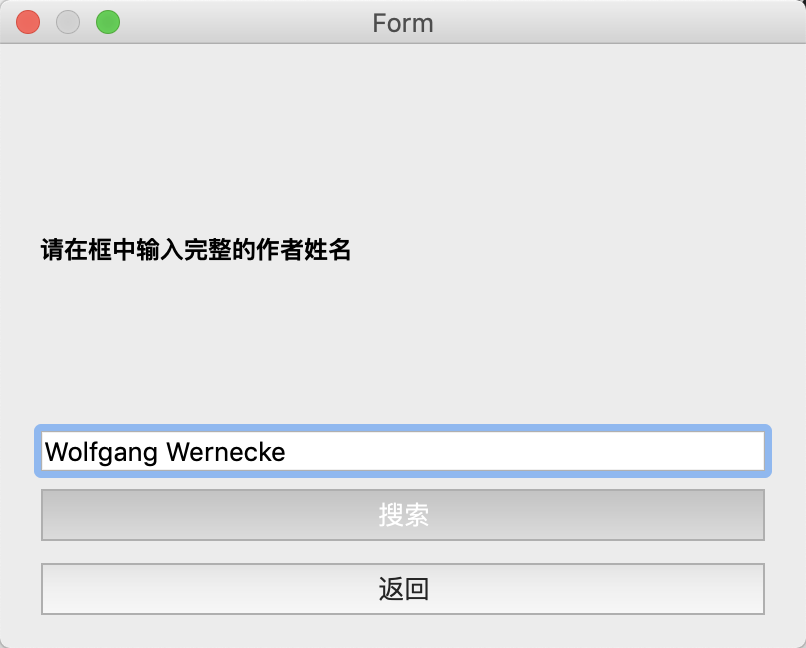
1. 第一个功能，按照作者名搜索对应文章，并输出所有文章详细信息。



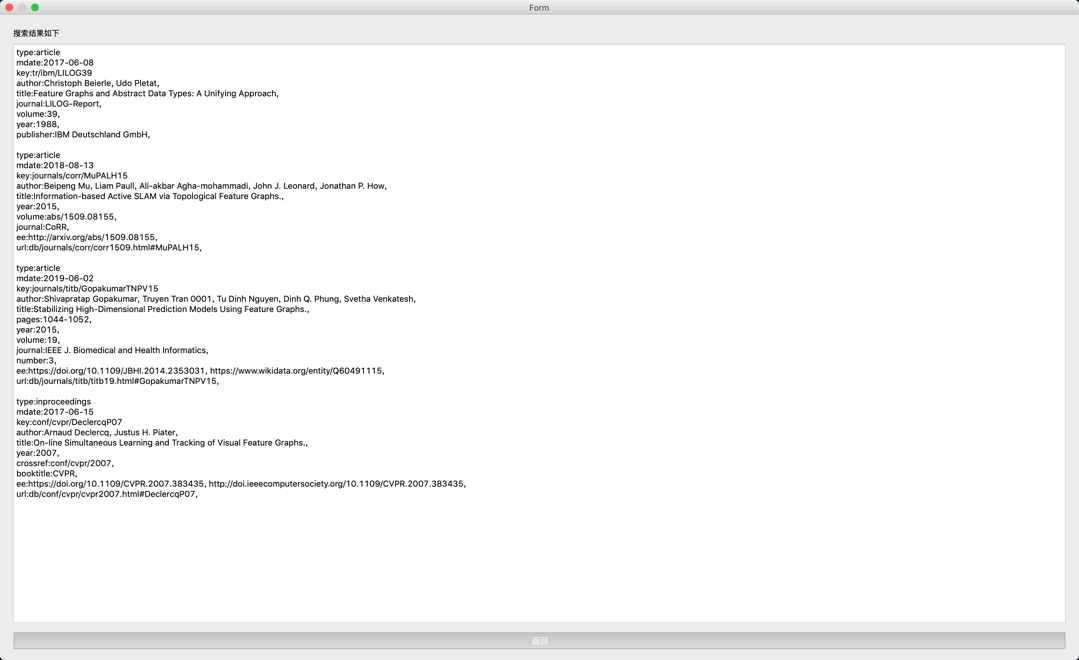
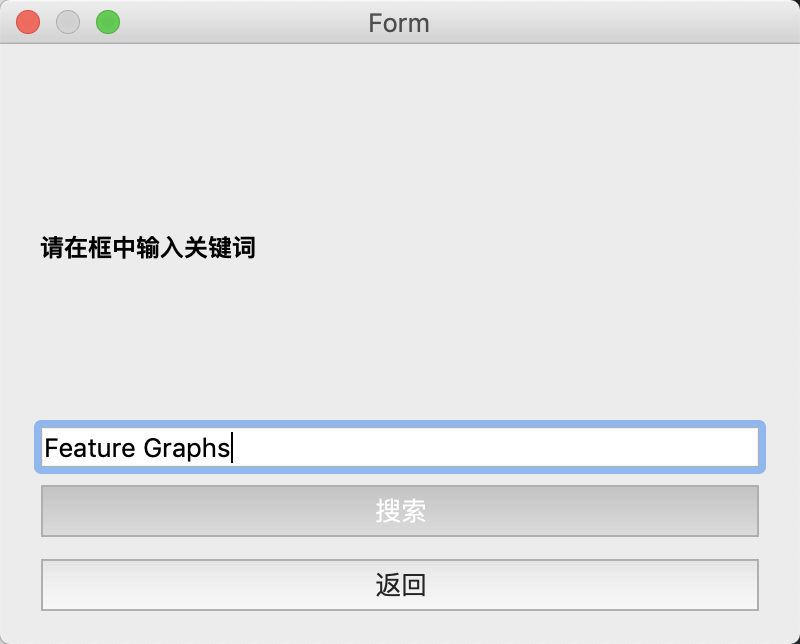
1. 第二个功能，按照文章名，搜索文章对应的详细信息。



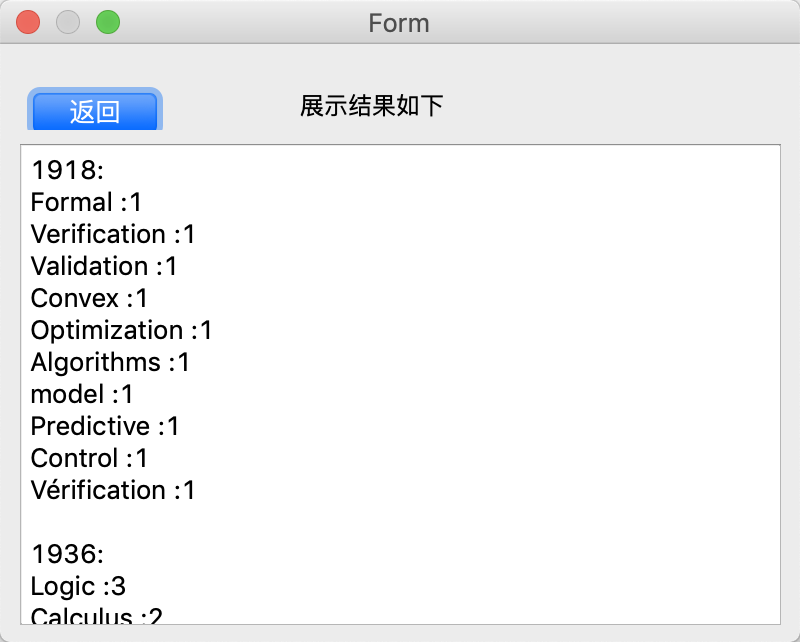
1. 第三个功能，搜索作者合作关系，即输出此作者的所有合作作者



1. 第四个功能，展示文章数最多的前一百名作者
2. 第五个功能，部分匹配搜索。



1. 第六个功能，展示各年热点词



最终我们所有功能都如上图，此系统具备对应六个功能。

五、课程设计总结

1. 遇到问题：读取XML文件方式，minidom等方式不支持XML的大文件读取方式。解决问题：使用SAX库对XML文件进行解析，此为事件驱动形式的对文件读取，因此可以读取超大XML文件。收获：通过上网寻找资料极为关键，因此类知识在课没有涉及到，所以上网信息检索能力很重要，网上有解决问题的方法。并且在我们对于陌生的模块进行使用时需要学会看技术文档并理解相关的函数使用方式，以及参数等。
2. 遇到问题：使用对应的SAX库，最初想法是记下来读取的信息对应位置行数，之后将行数作为值，标题名和作者名分别作为键，直接对于XML文件进行读取操作，但因SAX流不是ElementTree样，无法直接获取对应在XML文件中的位置。解决问题：第一次读取之中，将每一个读取的作者名放入一个字典之中，若不存在键置为一，存在则自增。将每一个读出的信息放入一个大的字典之中。键为标签，值为列表(可对应多个)。收获：此为处理最直接的思路，即读到一个信息就插入一个信息，但梦想总是美好的，见下面问题 ，此方法依然不行。
3. 遇到问题：发现作者名之中夹杂着 &auml；等特殊字符，将作者名分裂开，导致输出前一百个作者名错误。解决问题：上网查找，存在dtd文件约束，即将上述特殊字符转化成正常编码格式。将原本XML文件前两行替换成对应另外的一个文本文件中的内容。收获：在我们项目的撰写之前需要详细进行准备工作，不能盲目开始。
4. 遇到问题：在我们进行树的输出的时候我们遇到了一个问题，发现我们的文章树之中存在重复的信息。通过一步步的调试发现，我们所有的文章名作为键进行存储的对应值都是一样的。解决问题：经过查找之后发现我们插入的时候以键值对的形式进行插入，键为字符串，值为字典。但是我们之前插入的字典都是同一个字典，也就是地址是相同的。在我们每次再遇到同等级的标签的时候先进行了一个清空，然后再进行读取，这样子会导致每一次再读取下一个文章信息的时候会将之前所有的值进行一个覆盖。（所有的键都指向了同一个字典），解决方法是在我们进行插入的时候值参数为字典的一个复制copy()方法。收获：我们需要知道python之中字典的形式，对于字典列表等充分理解，其形式类似地址，不能直接进行赋值，否者使用的是一个地址。（但此方法在之后进行了舍弃，改为了list，具体原因见 ）
5. 遇到问题：在我们进行小文件所有功能已经测试完毕之后，我们在大文件中进行测试所有的功能，发现输出和一个作者相关的合作者的时候，程序进行报错显示其KeyError也就是字典不存在对应的'author'属性。但是在小文件中测试成功，于是排除掉逻辑问题，改为寻找大文件的问题。解决问题：经过苦苦查找和不断的debug，发现我们输出的最后一条记录之中没有作者的标签。于是我们很惊讶的发现，这篇文章竟然没有作者，但是它是如何被加到这个作者写的这篇文章之中的呢。原来我们在进行读取的时候读出来了作者写过的文章名，于是又调用了搜索功能，这个功能所搜的文章名。所以我们将这个没有作者标签文章的key进行复制与查找，发现这个文章没有作者，再将其文章名进行搜索发现有许多重名的文章，因此难怪字典中有时没有'author'标签，于是先搜素对应的文章，若有作者则进行输出，若无就不进行输出。收获：需要考虑文件中的内容问题，在如此庞大数据量的同时，需要考虑到什么可以作为键(唯一的)，什么是可以重复的。（但是此方式依然在之后进行了废弃）
6. 遇到问题：在我们功能日臻完善的同时，我们发现程序加载时间变的越来越长，于是我们思考如何将整个B+树存储起来，这样可以在下一次的程序加载的时候直接使用。否则我们每一次运行程序之前都需要进行B树和字典的构造，极为麻烦并且时间长达一个多小时。

解决问题：首先我们查找是否有对应的模块进行书写我们自定义数据结构的对象。之后我们找到了pickle模块，将B+树、字典序列化并写入二进制文件。但是我们在存储bPTree\_article的时候遇到了错误，程序意外退出。于是我们思考是不是我们程序本身的逻辑错误，但是我们在小文件(小文件为大文件的一小部分)都是跑过的，于是我们来排查是不是文件的错误，思考是不是txt文件存储不了我们这么大的B+树。我们按照其字母顺序分成了26个字母，对应的字母存储到不同的B+树之中。我们之后又发现这26个字母中有以H和Unknown开头字母的B+树存储时还是遇到了意外退出的情况。我们通过记元素的个数来进行比较，发现我们整个H开头的B+树是大概在2.7G，而我们最大已知能够存储的是770MB。所以我们对于以字母H开头的树再次进行了分割，发现H树下一位还是没有太大的改变（大多数数据以Ho开头）。至此我们已经几近崩溃的边缘。于是我们思考是不是我们的思路有着什么样的错误。上网寻找mmap内存映射等都是看不太懂。而一个偶然的机会我们发现linecache模块可以对大文件进行随机访问，再通过测试时间发现读取第100000行和第30行时间差别不大。于是我们使用了linecache模块，将我们B+树中的值转化为两个整数相信空间大小会缩减很多。但是我们又遇到了一个问题就是我们的XML文件没有进行格式化。即没有规律，仅仅通过函数调用也无法知道，所以我们无法获取对应的行数。于是我们使用了sublime中的Intent xml插件来对整个文件进行格式化（运行大概三小时）。至此已经将文件进行了格式化，所以我们可以获取对应的行数，将原本的字典转换为一个列表（文章开始位置、文章结束位置）。于是我们再次进行将树用pickle模块存储进txt文件中。这次我们的空间大小缩减了五倍，但是我们的H开头的树还是存储不进去。但是这样子是没有道理的，我们之前都已经能够存储770MB，按理我们H树应该比这个小，所以我们对整个树进行检查。通过输出整个树我们发现这个文章数和原本插入是对不上的。又想起之前我们发现的错误就是title是可以有重名的，所以我们发现是H树的问题。为了解决这个问题，我们在插入树之前首先进行文章名的搜索，如果为空则直接插入，不为空则用append来把这个加入到列表中。之后我们再次将这个写入txt文件中，于是发现可以运行且不会像之前一样意外退出。为了再次排查错误，我们将一整个树不分26个字母写入txt文件之中，这次终于可以运行成功，即将文章的B+树写入到了txt文件中，下次使用可以直接加载出来。在通过了上面的尝试之后，我们加载树并进行信息的读取，发现我们在行数方面出现了问题。因为我们将整个XML文件进行了格式化，所以类似于<sup>标签也是进行了格式化，这就导致了规律的不符合(因为我们之前将文件的格式看为article同级的标签独占两行，其余标签独占一行，但是通过这个我们不能够判断类似<sup>标签是独占一行还是两行，这就导致了位置定位的不准确)。通过上面的尝试，直接存储行数就无法确定，也就无法进行下去。之后我们再次转变思路，我们将此篇文章是在XML文件中是第几个文章存入到B+树中当作这树的值。将树还是用pickle模块进行写入，之后读取的时候进行加载，在输入之后即可得到对应的序号，使用序号组成的列表进行搜索。

六、使用到的类库：

1.pickle模块，用来将我们生成好的B+树写入到txt文件中

2.SAX读取超大XML文件方法

3.PyQt5用来书写图形化界面

**参考文献**

# ［1］Mark Allen Weiss. 数据结构与算法分析C++语言描述（第四版）. 北京：电子工业出版社, 2016. 135.