实验报告

雷怡然　　2016013274

一、实验目标

在完成对铁甲网论坛上发帖信息的提取和分词的基础上，通过建立词库的平衡二叉树和倒排文档实现对关键词的搜索。

二、实验环境

操作系统：windows 7 x64位
IDE: Qt 5.6.1
编程语言: C++

三、抽象数据结构说明

1.平衡二叉树（m_balanced_binary_tree.h）

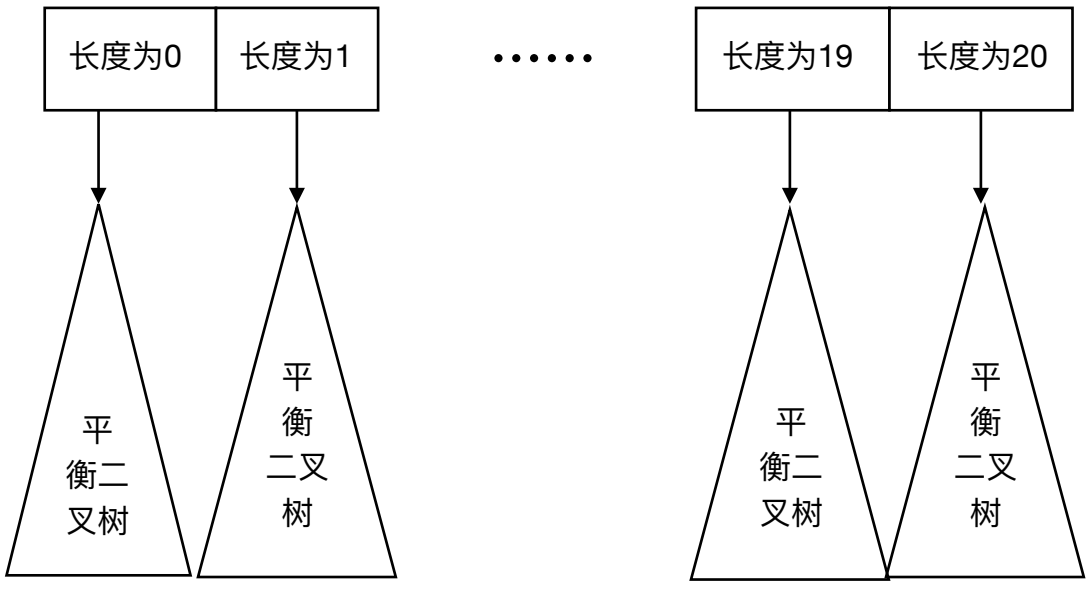
树的节点为：

```
struct node
{
 m_charstring<wchar_t> value;          //单词内容
 int wordID;                           //单词ID
 int Occur;                            //单词总出现次数
 int index;                            //平衡因子
 m_file_list file;                     //文档链表
 node *LChild, *RChild;
};
```

二叉树的每个节点记录了单词的相关信息。实现的功能有插入(Insert)，搜索(Search)和调整二叉树使其平衡(Adjust)。

2.平衡二叉树的优化

最长匹配算法的分词特点是：由最长的待搜寻字符串开始，在词库中搜寻，若未找到，则减短待搜寻长度。根据此特点，我发现将同一长度的单词放置在同一平衡二叉树上，可以提高分词效率，避免重复浪费搜索。结构可以由下图表示：

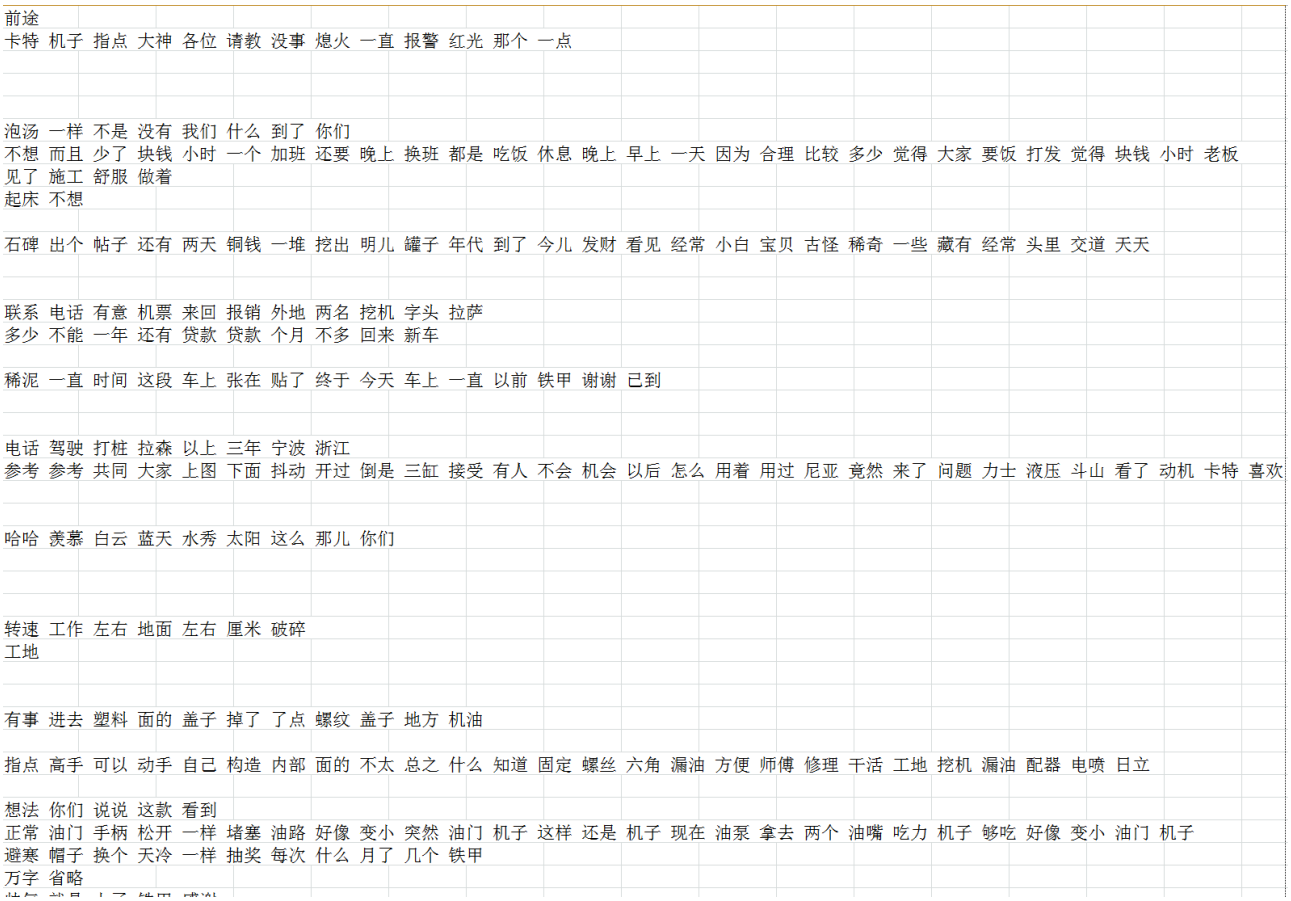


通过这样的方式，在分词判断不同长度的字符串时，我们可以直接查找相应“特定长度”的词库，而不用去遍历全部词库，从而可以大大提高运算效率。

3.词库精简和速度的优化

通过研究词库以及2000个文本对原始全部词库的分词结果。

(下图示例)



发现分词结果多为2个字，这也是比较好解释的，词典中3个字的词语多为“专有名词”，4个字的词语多为“成语”，5个字及5个字以上的词语多为专有名词。在挖掘机论坛这样的环境下，很少会有3个字及以上的词语出现，通过上面建立的根据词语长度排列的平衡二叉树可以发现，3个字及以上的词语占了很多存储空间和运算性能，但是并没有对分词结果起到明显效果。考虑到优化计算速度、精简程序体积，我在初始化词典及判断词时只考虑两个字长度的词语。

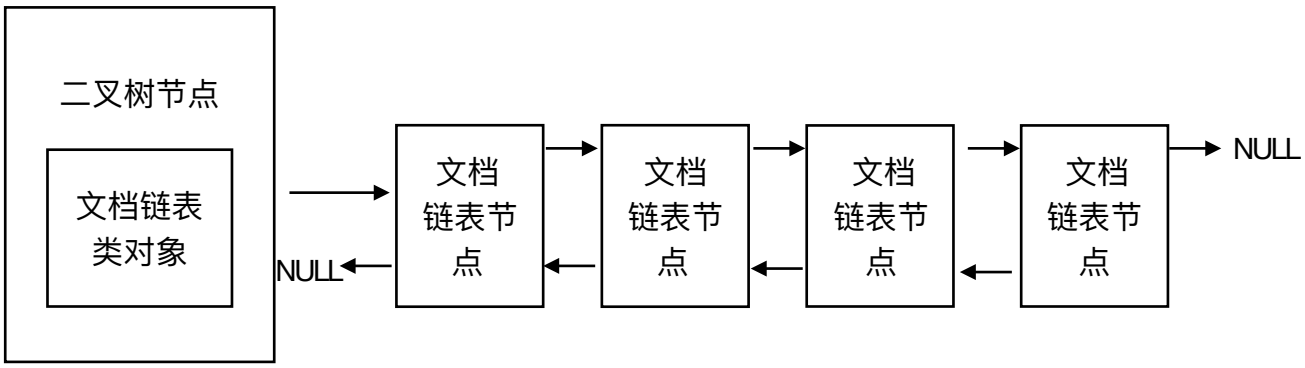
4.文档链表(m_file_list.h)

```
struct list_node
{
    int webID;                                  //网页ID
    m_charstring<wchar_t> website;              //网页网址
    m_charstring<wchar_t> content;              //文档内容(gui程序中有，query程序中没有)
    m_charstring<wchar_t> keyword;              //关键词
```

```
    m_charstring<wchar_t> title;                    //标题
    m_charstring<wchar_t> author;                   //作者
    m_charstring<wchar_t> date;                     //发帖日期
    int times;                                      //单词在该文件出现次数
    list_node *next,*pre;
};
```

文档链表的每个节点记录了单词出现的次数、单词是什么、网页ID、文档内容等信息。
实现了添加文档(add)、搜先文档(search)、编辑文档(edit)、删除文档(Remove)等功能。

词典二叉树中的节点和文档链表类的关系如下图：



搜索平衡二叉树的节点，节点中文件链表的节点就是包含这个词的网页，文档链表是双向链表，是为了方便插入和排序。文档链表的节点顺序按照词语出现的次数，出现次数多的在前面，出现次数少的在后面。因此在类函数中有函数sort()，它是用来调整插入节点顺序的，因为是双向链表，顺序调整十分高效。

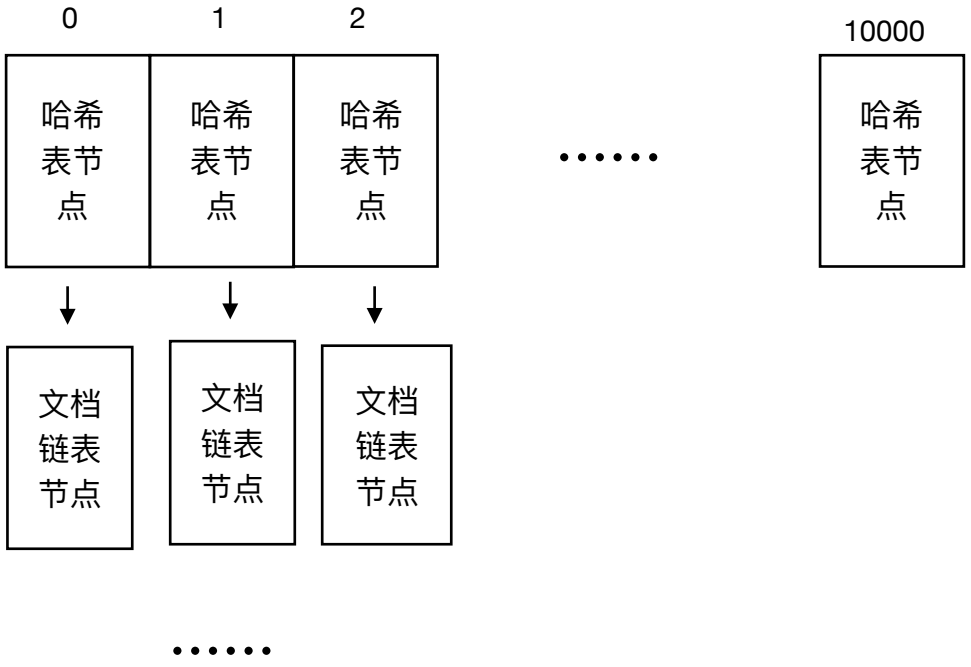
5.输出哈希表和排序链表(query_web_hash_table.h)

用“链式哈希表”和排序链表来整合输出结果，哈希表每个单元表示一个网页，详细的说明在后面介绍输出算法时将详细介绍。

四、算法说明

1.输出整合

对于一些词语的搜索，调用平衡二叉树的Search函数，将返回相应的树节点，每个树节点的文档链表即使包含该词语的全部网页。对于同一网页，可能含有不同的搜索词，因此搜索后会有不同的文档链表节点，这些节点需要整合起来（记录搜索词总出现次数），并且只输出一次。
建立如下图结构的哈希图，最上层是数组，每个数组节点链接一个链表，哈希函数网页ID(webID),用来整合。



当搜索某一词语后，遍历词语树节点的文档链表节点，文档链表中存储了webID,可以根据哈希函数$H(x) = webID$找到最上层位置，然后将文档链表节点指针记录下来，构成链表，并记录搜索词语出现的总次数。同时，将出现的webID记录到一个排序链表中，该链表的所有元素不重复，在此使用了“插入降序排列”。

2.输出顺序

a.优先按照搜索词出现次数

先按照搜索词出现总次数从高到低进行排列。如果出现总次数一样，则按照词语出现“离散程度”排列，搜索词语离散程度越低（即出现次数越相近），则排列位置越靠前，反之越靠后。如果“离散程度”也一样，则按照所搜索词语的先后次序排序（初始搜索排序）。

现对“离散程度”进行说明：

该思想源于“方差”，该值越大，则表明数据越分散，即相互之间差距很大，该值越小，则表明数据之间比较相近。本程序自定义了以下公式来反应“离散程度”：

$$Variance = \frac{1}{n} \times \sum_{i=1}^{n} |x_i - \bar{x}|$$

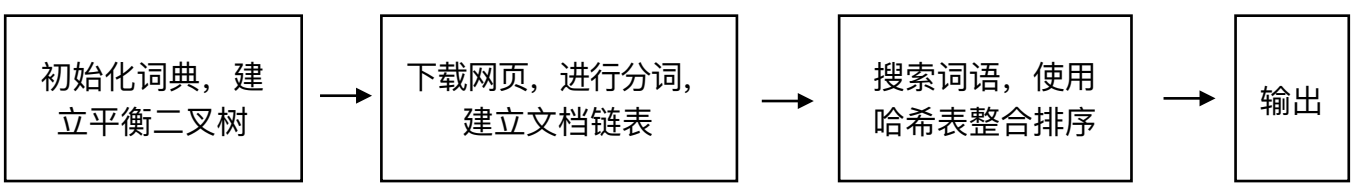
其中n是输入搜索词语的个数，x_i是各搜索词语在网页中出现的次数，$\bar{x}$是各搜索词语出现的平均次数。这个公式相比于方差的平方计算比较简单，运算效率更高。对于输出顺序的改变，是通过改变前面所述的记录webID的链表的节点顺序实现的。

这样的顺序其实也符合我们日常的逻辑，如果搜索多个词语，搜出的结果包含搜索词次数肯定越多越好，同时还应该尽可能包含不同的关键词，这对应于就是“离散程度小。

b.优先按照发帖日期

生活中存在这样一种情况：我们需要提取出“最新”帖子。因此在这种排序下，发帖日期越晚，排列越靠前。当不同帖子发帖日期一样时，再按照搜索词语出现的总次数和“离散程度”进行排列。

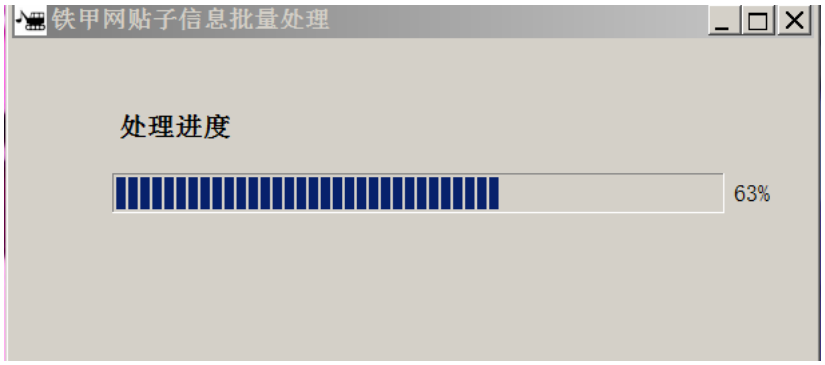
五、流程概述



六、输入输出及操作相关说明

1.exe文件夹中含有两个子文件夹，分别为批量查询和gui查询文件，在两个子文件中，可以在input文件夹中改变url.csv文件，在dictionary文件夹中改变字典文件。对于批量查询，在query.exe同一层级放入query.txt文件即可对其中内容进行查询。
字文件夹中含有的其它文件是程序运行所需要链接的库。

2.对于批量查询，直接打开query.exe程序，等待进度条读完，会显示”处理完成”。读条过程如下图：



3.对于gui查询，打开gui.exe，等待初始化完成后，会显示搜索窗口，在输入框中输入关键词（例如“二手挖机不想睡觉啦啦”），点击“搜索”即可。点击搜索结果中的网页链接，可以直接跳转至原网页。点击不同排序方式的按钮，可以改变搜索结果的排序方式，默认按“搜索词次数”排列。

4.初始化及载入时间在Tsinghua 5G下约10分钟。

七、实验结果

经过调试和验证，发现实验结果符合预期。

八、功能亮点说明

1.Qt网络类的使用，大大提高了网页下载速度，其中还使用了异步操作；gui程序使用了多线程，确保了交互界面的及时刷新。

2.平衡二叉树按词长创建和词库的精简。（详细见前）

3.使用哈希表快速整合搜索结果，以及排序顺序的确立。不同的排序方式。

4.下载2000个网页加处理时间约10分钟左右，程序效率较高。

九、实验体会

感觉整个实验很有趣也很有用，自学了很多东西，包括网页下载，Qt多线程程序，编码和解码等等。我感觉要实现一些实用的功能其实挺复杂的，还需要我不断学习努力。
（吐槽：面向搜索引擎的大作业 + 网络、编码、多线程自学）