数据结构与算法（1）实验一报告

曹张杰 软41 2014013428

一、实验目标

通过解析html，提取信息，中文分词等活动理解并熟练运用栈，树，哈希表，链表等数据结构。并且提高运用算法解决实际问题的能力。

二、实验环境

VM Fusion虚拟机下的中文Windows 64位企业版，双核CPU，4G内存。

三、抽象数据结构说明

栈：模版类，在Stack.h中声明和定义。用数组存储元素，具有length和capacity两个元素分别表示栈中目前元素个数和最大能存储元素个数。
ElemType top() 返回栈顶，只要返回length-1位置的元素。
void push(); void pop() 实现时只要对length操作就可以做栈的push和pop。
该结构的功能主要是html解析时用于存储标签并且匹配标签，保持html嵌套结构；

向量：模版类，在Vector.h中声明定义。这是扩展栈功能的一个数据结构，结构和栈基本相同但
ElemType front(), ElemType operator[](int) 增加了访问头元素以及随机位置元素访问。
这个结构主要用于在树中存储子节点指针。

树：在HtmlTag.h中声明，在HtmlTag.cpp中定义。只有root指针一个数据，指向树的根结点。操作有：
void extractInfo(CharString&) 从html文件解析出html树结构
Vector<HtmlNode*> search(CharString&, CharString&) 输出树的某个节点以及其子节点

static void outputSubTree(HtmlNode*, CharStringLink&) 搜索树中的某个特定元素。
该结构主要用于存储解析完的html树，并支持之后的提取信息等操作。

哈希表：在HashTable.h中声明，在WordTable.cpp中定义。HashTable模版类是哈希表存储数据的结构，用数组指针存储数据。由于hash函数不一定可以把所有元素映射到不同位置，所以数组每个位置可以存一个链表，代表hash到这个位置的所有元素。其中有occupation_num表示被占用的指针数量，并不用于功能，是为了测试hash表的性能，即占用数量 / 总存储元素数量就是性能，越高就是重复hash越少。
void add(ElemType*, unsigned) 在hash表中插入元素
bool search(ElemType, unsigned int) 判断某个元素是否在hash表中

WordTable是一个实际的字符串hash表，包含了hash函数，并用一个字符串HashTable来存储元素。
bool search(CharString&) 先计算字符串的的hash值再存到hash表指定位置。
void initDictionary(CharString&) 用字符串中的词初始化该词典
void divideWords(CharString&, CharStringLink&) 对第一个参数字符串分词并存到第二个参数字符串链表中
static unsigned int hashFunction(const myChar *, int) static unsigned int hashFunction(CharString&) 计算一个字符串的hash值

字符串：在CharString.h中声明，在CharString.cpp中定义。用字符数组存储数据
int indexOf(myChar t, int pos) 查找某个字符第一次出现的位置
int indexOf(const myChar *, int, int) 查找某个字符串中的任意字符在this字符串中第一次出现的位置
int indexOf(CharString&, int) 查找某个字符串第一次出现的位置
int indexOf(const myChar * p, int pos) 查找某个字符串第一次出现的位置
int* compute_prefix(CharString&)
int* compute_prefix(const myChar *, int) kmp匹配算法的辅助结构计算
void strip() 去除字符串前后的空白字符
CharString* substring(int, int) 返回从第一个参数到第二个参数的子串

CharString* concat(CharString&) 将当前字符串与参数中的字符串连接并返回
CharString* operator+(CharString& s) 将当前字符串与参数中的字符串连接并返回
bool operator==(CharString& s) 判断字符串是否相等
bool operator==(const myChar* s) 判断字符串是否相等
myChar operator[](int index) 随机访问字符串元素
friend std::wostream& operator<<(std::wostream&, CharString*) 输出字符串
int htmlHash() 用于哈希html的tag，减少存储量
unsigned int tableHash() 用于词典hash函数，把当前字符串哈希到词典
void utf8ToWchar(const char * s) 把utf8编码转成windows自带wchar编码，由于html是utf8编码
void spaceToSpace(const wchar_t * s) 将字符串所有空白字符都转为空格，方便分词处理

字符串链表：在CharStringLink.h中声明，在CharStringLink中定义。用一个双向循环链表存储字符串数据。
void add(CharString&) 加入字符串
CharString remove(CharString*) 删除某字符串
int search(CharString*) 查找某字符串并返回在链表中的位置
CharString* nextNode(bool) 遍历字符串链表，参数是遍历方向
friend std::wostream& operator<<(std::wostream&, CharStringLink&) 顺序输出字符串链表

四、算法说明

解析html算法：解析html依靠一个栈和一个向量实现，栈中存储读取html文件到当前位置时还没有闭合或者没有闭合标签的标签。向量中存储已经闭合的标签。同时向量中的每一个元素还存了当这个闭合标签生成时栈的深度，这样就可以把所有已经生成的闭合标签存在一个向量中而不至于html嵌套混乱。算法流程就是每一步都从当前位置定位下一个'>'，'/>'，'<'，'</'，当遇到'<'时寻找'>'，如果是寻找到'>'时发现它其实是'/>'就生成该节点并附上栈深度，存入向量，如果是'>'就入栈。当遇到'</'时，

先定位下一个'>'抽取出这个结束标签，然后把这个结束标签和栈中的标签进行匹配，一直弹栈直到找到第一个与该结束标签匹配的标签，然后向量中栈深度比弹出这个结束标签后的栈深度大的标签以及在弹栈过程中弹出的标签设为当前这个闭合的标签的子节点。在解析时当读完'>'时需要判断到下一个'<'之间是否有文字，有文字就应该存成text节点作为入栈。另外为了方便匹配不同html标签，我把html标签的字符串hash为一个值存储。具体就是把html的标签字符串每一个字符的unicode值相加再加100000*字符串长度，这样由于字母的unicode值远小于100000，因此不同长度html标签就可以直接分开。另外同长度html标签的子母unicode值相加是没有重复的（看过html标签表），因此这个hash是一一对应的，可以节省存储空间并且在下面的搜索等操作中加速搜索。

提取关键信息算法：这个就是根据我们需要的信息的标签和属性在html树中做前序遍历的搜索，搜出相关标签然后将其子节点中的text节点的内容提取出来就可以。

中文分词算法：这里首先词典是使用hash表，将不同词hash到hash表数组不同index，这里使用了40000000指针数组存储hash表，然后如果有两个词哈希到了一个位置就用一个链表存储所有在这个位置的值，然后分词用的就是上课讲的最长匹配分词算法，不过加了相连的数字或者相连的拉丁子母直接算作一个词的规则。另外介绍一下hash函数，就是对每个词，从最低位开始，先加上字的unicode值，然后乘以从左算起的位数，及第一个字第一位，第二个字第二位，然后加123456，然后乘1234，然后将目前的hash值左移（位数＋2）位，对词的每一位循环之后算出的hash值对40000000取膜就是它在hash表中的位置。最终测试式这种方法可以让占用总词数的80%个位置，hash效果很不错。

五、流程概述

［载入词库并构建哈希表］->［读取网页］->［构建html树］->［搜索并提取关键信息］->［中文分词］->［输出结果］

六、输入输出及操作的相关说明

命令行输入是文件名，文件名不需要带路径带需要在和Homework.exe的同一个目录下放置input文件夹并且把所有输入的html放置在这些文件夹里。可以输入多个文件名。输出的txt和info文件均在和Homework.exe的同一个目录下output文件夹里（需要预先在这个目录下放output文件夹）。

七、实验结果

html解析的结果：可以把整个html无论有关还是无关的信息都可以建造进入html树，实现了一个通用的html解析器，完全符合预期。

关键信息提取结果：可以提取出所有需要的关键信息，并且通过网页结构和html标签属性等辅助信息的搜索，可以从html树中找到所需要的标签然后从这个标签节点以及其子节点中提取中关键信息。其中两个title部分都保留了标点而对author去掉了两边的标点，这也符合常理，毕竟title都是句子而author是词。content也能全部提取出来没有缺失。

分词结果：分词结果可以滤去所有非数字，字母和中文。中文分词结果大部分都符合中文语言语法以及语义，像“的”，“是”这些常用单字都被分到了一个字，单词和数字也被成功分词出来，总体效果还是很不错的。唯一不符合预期的就是一部分词典中没有的词没有被分开，比如“休赛期”，可能是助教的词库不太完整，第二次实验希望能够用更完整的词库并且允许自己修改词库。

八、功能亮点说明

1、利用栈结构的html解析算法，构建了html树，将提取步骤与解析步骤分离，支持多次查询，并且解析过程中不损失任何信息，不做字符串匹配同时还通过字符串形式保留了属性值，为进一步提取属性值已经后边的信息提取做好了准备。只需扫描一次html文件，时间已经用的最少。
2、本次实验很好地使用了hash函数，html的hash以及词库的hash，hash的好处在于只需计算一次，之后的匹配就方便很多。并且hash函数的设计也针对具体的文本进行设计，html部分的hash充分利用字母值的范围和html标签的特性保证使用一个整数就可以存下一个html标签，而整数是4字节的，html标签每个字母unicode是2字节，因此如果直接存标签存储量

大且操作指针容易引起内存泄漏；词库hash充分体现了随机性，从hash前面介绍的用了词库总数的80%数量的指针就可以看出hash的重复率很低。另外在hash表空位置只存了空指针，占用额外内存较少。
3、分词算法在上课讲的最长匹配的基础上增加了相连的数字或者相连的拉丁子母直接算作一个词这两条规则，增加了算法处理英文和数字的能力，防止出现但英文字母和单数字词。

九、实验体会

编码是windows一个十分坑爹的问题，windows对utf8的支持很差，导致读取html出现很严重的问题，中文字符读入是乱码并且和词典的编码不一致，导致一开始词典hash函数就算错误，无法搜索。后来用了windows自带的解码函数解决了问题。总体来说对于Mac用户，windows的整体开发环境十分不友好，比如编码问题，windows喜欢自搞一套（ansi），不用通用的编码形式utf8，然后windows控制台支持不好，控制台运行程序很多命令不支持。第三就是VS开发环境的配置十分难找，初始配置限制堆栈存储为100MB，在本程序中碰到对内存无法分配问题，但是后来才发现是VS的限制，所以这样的bug是没有什么意义的。所以觉得windows开发还是有一定问题的。

然后是这次实验的体会。实验设计还是非常好的，html解析以及中文分词都是很好的应用，并且将上课的内容结合进去，学到了很多东西。同时词典数据比较大，是一次接触比较大的数据的机会。并且这次实验是对数据结构一次很好的复习，毕竟如何用数据结构存储并快速处理大数据需要对数据结构很深的认识。