实验报告

一、 实验目标

基于文档的分词结果，根据输入的关键词进行查询。实践课程中学习的经典数据结构，加深对其的理解。

二、 实验环境

操作系统：Windows 10 64 位
集成开发环境：Visual studio 2015
编程语言：C++，Javascript，Nodejs

三、 抽象数据结构说明

AVL：平衡树，修改时自动调整平衡性，数据域只包括一个树节点指针，树节点为自定义类型。可完成根据特定数据插入、查找、删除节点的操作以及中序遍历树。
TreeNode：树节点类型，包括存储的数据、节点频率和高度等数据。
FileChain：文档链表类，内部数据域中有特征关键词、文档数量和以链式结构存储的文档节点，节点包含的信息有文档内容，文档 ID 以及关键词词频
Trie：hash 字典树，每一个节点的数据域中存储了该节点对应文字，路径构成词是前缀还是完整词语，以及一个 unordered_map 存储下一级节点信息。

四、 算法说明

通过正向匹配分词后得到文章的分词链表，再将文章添加到 AVL 树上每一个对应关键词的节点上，之后按照查询关键词将内容从树中取出，合并后打印。

五、 流程概述

主程序：初始化词典->解析->对解析出来的正文内容分词->挂载到平衡树上->打印内容。
服务器：服务器脚本调用主程序并将主程序的标准输出流重定向到服务器进程。再完成与前端的通信，不是重点就不再细说。

六、 输入输出及操作相关说明

直接运行程序的时候，程序解析同目录下 query.txt 文件，并将结果打印至 result.txt。运行可视化界面的时候，在 cmd 中定位至此目录，并运行“node app.js”并访问 localhost:23333，在服务器就绪时，前端会弹窗提示服务器已就绪。

七、 实验结果

平衡树的建树和查找速度非常快。
简单的普通中文字典树的速度小于普通 hash 小于 hash 字典树

八、 功能亮点说明

前端呈现、可解析类似 243.html 这样含有多段内容的文档（但是此时链接将不可用）。

九、 实验体会

全栈工程师真不好当……
编码问题真坑