实验报告

软件22 王华清 2012013317

**实验目的：**

1. 对课堂上的基础数据结构类型进行训练；
2. 把数据结构的基础知识应用到实际的软件开发中，体会数据结构的广泛用途；
3. 通过实验方法的多样性培养学生深入思考，推陈出新，比较不同方法的能力；
4. 通过申请实验分数的方式，培养学生的自我表现能力；

**实验环境：**

C++: VS2012, VS2010

OS: Windows7 64bits

Memory: < 3GB

**实验中用到的数据结构的抽象数据类型说明：**

**实验一：略**

**实验二：**

**文档链表：DocLink**

初始化：void initDocLink();

清空：void clearDocLink();

销毁：void destroyDocLink();

插入文档：void insertDoc(Doc &e);

查找文档：void indexDoc(int &id, Doc &e);

删除文档：void deleteDoc(int &id, Doc &e);

修改文档：void changeDoc(int &id, Doc &e);

输出文档链表：void outputLink();

**大小二叉平衡树倒排文档：SBT**

初始化：void initSBT();

销毁：void deleteSBT(pSBTNode &tree);

左旋：void leftRotate(pSBTNode &tree);

右旋：void rightRotate(pSBTNode &tree);

插入结点：bool insertNode(pSBTNode &tree, DocLink &e);

查找结点：bool indexNode(pSBTNode &tree, DocLink &e);

删除结点：void deleteNode(pSBTNode &tree, DocLink &e, DocLink &r);

修改结点：bool changeNode(pSBTNode &tree, DocLink &e);

维护操作：void maintain(pSBTNode &tree, bool flag);

加入一个网页：void addSBT(int i);

输出文档链表：void outputSBT(pSBTNode &tree);

建立大小二叉平衡树倒排文档：void buildSBT();

**哈希表倒排文档：Hash**

初始化：void initHASH();

销毁：void deleteHASH();

插入结点：void insertNode(DocLink &e);

查找结点：bool indexNode(DocLink &e);

删除结点：void deleteNode(DocLink &e);

修改结点：void changeNode(DocLink &e);

计算哈希值：int hashkey(DocLink &e);

加入一个网页：void addHASH(int i);

输出文档链表：void outputHASH();

建立哈希表倒排文档：void buildHASH();

**实验基本算法和流程：**

**实验一：略**

**实验二：**

建立倒排文档：

SBT/Hash：通过网页分词结果建立/通过倒排文档建立

批量查询：

SBT/Hash查找：找到对应关键词输出

单个查询：

短句分词：中文分词算法

SBT/Hash查找：找到对应关键词输出

图形界面：

MFC：基于对话框的图形界面

**程序输入输出及操作说明：**

配置：

将Stop.txt，Dic.txt，Name.txt，docSBT.txt，docSBTREE.txt，docHash.txt，query.txt，Segmentation文件夹，SinaPage文件夹，放至src\Search\Search\Search文件夹下，编译后运行。（该部分已配置）

由于程序需要加载词语词典，人名词典，停用词典，倒排文档，建议使用Release模式运行（Debug约加载10秒，Release约加载3秒）。

运行后出现图形界面：



批量查询：输入查询文件路径和结果文件路径，单击生成结果文件即可生成结果文件。单击打开查询文件和打开结果文件可以打开对应路径的文件。

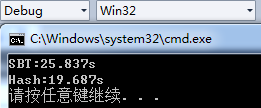
单个查询：输入关键词或短句后单击查询即可在列表中看到结果。

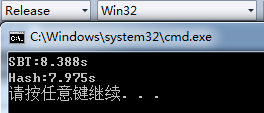
打开网页：输入网页编号，单击打开即可查询对应网页。

**实验测试结果：**

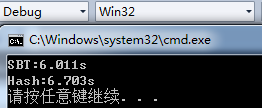
**SBT与Hash对比分析：**

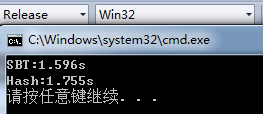
**SBT与Hash在通过网页解析结果建立时耗时：**

Debug：

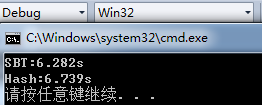
Release：

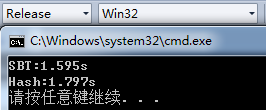
**SBT与Hash在通过倒排文档建立时耗时（先序）：**

Debug：

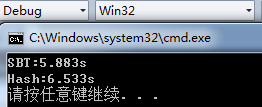
Release：

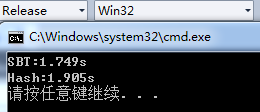
**SBT与Hash在通过倒排文档建立时耗时（中序）：**

Debug：

Release：

**SBT与Hash在通过倒排文档建立时耗时（先序）：**

Debug：

Release：

**SBT与Hash在各个操作中复杂度：**

SBT均为O(nlogn)，Hash取决于Hash函数的优越性。

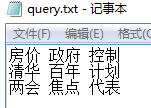
比较发现，SBT总体优越于Hash。在平均时间上SBT比Hash更快，SBT均为O(nlogn)，Hash随机性过大，对于不同数据冲突率不同。在占用空间上SBT与元素大小相同，节约了空间，Hash远超元素个数，浪费了空间。但SBT的运行效率受到元素排布顺序的影响，而Hash因为其随机性不受影响。

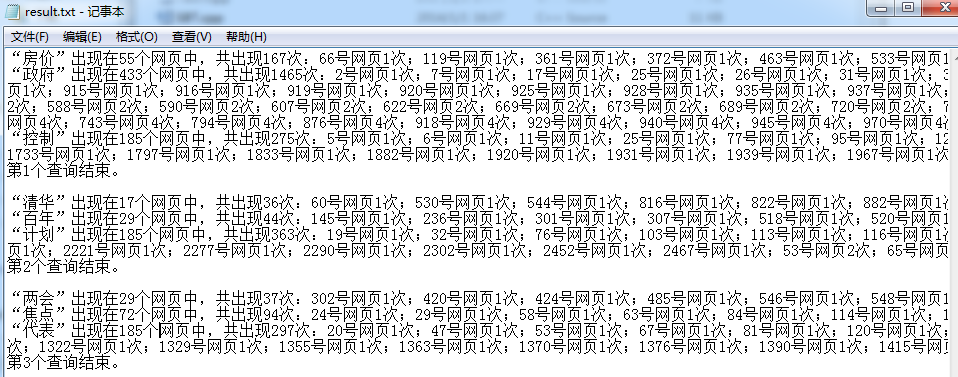
**查询结果展示：**

**单个查询：**

****

**批量查询：**

对****的查询结果：

****

（见query.txt和result.txt。）

**分数申请：**

**字典：20%**

申请理由：实现了SBT和Hash两种字典及基本操作，并做了比较。

检验方法：见代码SBT.h，SBT.cpp，Hash.h，Hash.cpp，见文档实验测试结果。

**倒排文档：10%**

申请理由：实现了基于SBT和Hash的倒排文档的数据结构及基本操作。

检验方法：见代码Doc.h，DocLink.h，DocLink.cpp，SBT.cpp，Hash.cpp。

**建立倒排文档：15%**

申请理由：正确的建立了倒排文档并输出至文件中。

检验方法：见代码SBT.cpp，Hash.cpp，见结果文件docSBT.txt（先序），docSBTREE.txt（中序），docHash.txt（乱序）。

**效率：15%**

申请理由：通过网页分词结果建立倒排文档约7秒（Release）；查询均在1秒以下（Release）。

检验方法：见结果文件 通过网页解析\_Release.PNG；运行后进行查询检验。

**正确性：15%**

申请理由：查询结果正确，可以打开网页检验。

检验方法：批量查询见结果文件 query.txt，result.txt，或运行程序检验；单个查询运行程序检验；打开网页运行程序检验。

**相关文档：15%**

申请理由：按照要求正确提交实验报告，结果和程序，报告中内容正确。

检验方法：见实验报告，结果和程序。

**代码注释：5%**

申请理由：按照要求写详细的代码注释。

检验方法：见代码。

**加分：**

**图形界面：5%**

申请理由：制作了图形界面。

检验方法：运行程序，或见代码SearchDlg.h，SearchDlg.cpp。

**效率提高：5%**

申请理由：效率有所提高，通过网页分词结果建立倒排文档约7秒（Release）；查询均在1秒以下（Release）。

检验方法：见结果文件 通过网页解析\_Release.PNG；运行后进行查询检验。

**实现数据结构：5%**

申请理由：实现优于AVL平衡二叉树的SBT平衡二叉树，相比AVL，SBT空间利用率高，效率不变，编写调试方便。

检验方法：见代码SBT.h，SBT.cpp。

**操作优化：3%**

申请理由：图形界面分为四栏，更容易上手；增加打开批量查询文档功能，方便用户操作；打开网页功能不限于包含查询词的网页，便于检查查询正确性；单个查询增加按回车查询的功能，方便使用。

检验方法：运行程序，或见代码见代码SearchDlg.h，SearchDlg.cpp。

**词库优化：1%**

申请理由：增加了人名词库“Name.txt”，而不失效率。

检验方法：见结果文件Name.txt，见文档配置部分，见代码SearchDlg.cpp初始化部分。

**词库优化：1%**

申请理由：增加了停用词库“Stop.txt” ，而不失效率。

检验方法：见结果文件Stop.txt，见文档配置部分，见代码SearchDlg.cpp初始化部分。