《数据结构》实验报告

班 级: 0601200 姓 名: 薛茵午 学 号: 2020303828

E-mail: 814680344@qq.com 日期: 2023/5/8

**◎实验题目: 使用邻接十字链表为存储结构实现文本的关键词查找算法**

**◎实验内容：**

本次实验以五篇英文文章为内容, 使用哈希链表为头邻接节点的十字链表作为存储结构，实现了英文文章关键词的自动分词，十字链表矩阵的建立, 显示以及查找算法

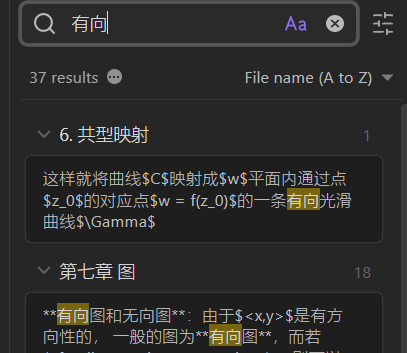
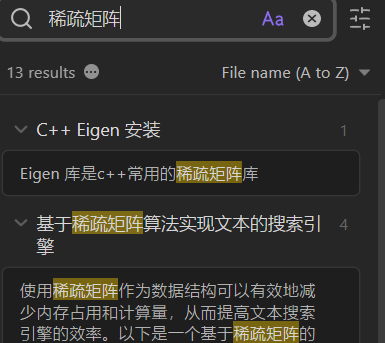
**(另外中文分词可以借助第三方库来进行实现，因为配置太麻烦没有进行中文分词实现)**

1. **需求分析**

文章的题目确立过程是我使用ChatGPT建议的多个结果中我进行选题得到的，对应src文件夹中的五篇文章均使用ChatGPT生成，但在编程过程中我未使用GPT辅助编程，代码均为自主完成（有的看了CSDN的一点文章）。

在我们一般的文章的词汇查找中，往往需要进行关键词的整篇文章或多篇文章的搜索，如果是在大批文章如数据库中进行搜查，除了使用关键词来进行，我们还可以使用一个稀疏矩阵，存储每一篇文章内容中出现的词汇，即将文章进行分词之后存储在矩阵中，然后每次搜索时，仅需要在矩阵中搜索对应词汇，并确定其出现的文章即可。

下面附上Obsidian笔记软件中自带的全局搜索功能界面图片。



我们将每篇文章的关键词存储到矩阵中，搜索时检查搜索字符串与每一个关键词字符串的匹配程度， 然后打印不同字符串的匹配结果，文章中的出现次数以及匹配程度，由于期中考试时间紧张，这里仅对核心存储，遍历和查找功能进行实现。

对于多篇文章分词的存储，每一篇文章的词汇往往重复较多而总词汇量并不多，因此我们可以使用稀疏矩阵的格式， 其存储结构为每一行存储一篇文章的关键词，每一列存储一个关键词，在本程序中，节点仅存储关键词的重复数量， 这样可以有效的节省存储内存。

使用稀疏矩阵存储搜索结果有很大优势，其一是在查找到对应的词语时， 只需要向下遍历即可查完所有的词语，二是查找某个文件中的关键词，只要向右遍历即可。另外，在删除某一文件时，我们只需删去稀疏矩阵中的一行(删除算法由于时间我没有实现)。

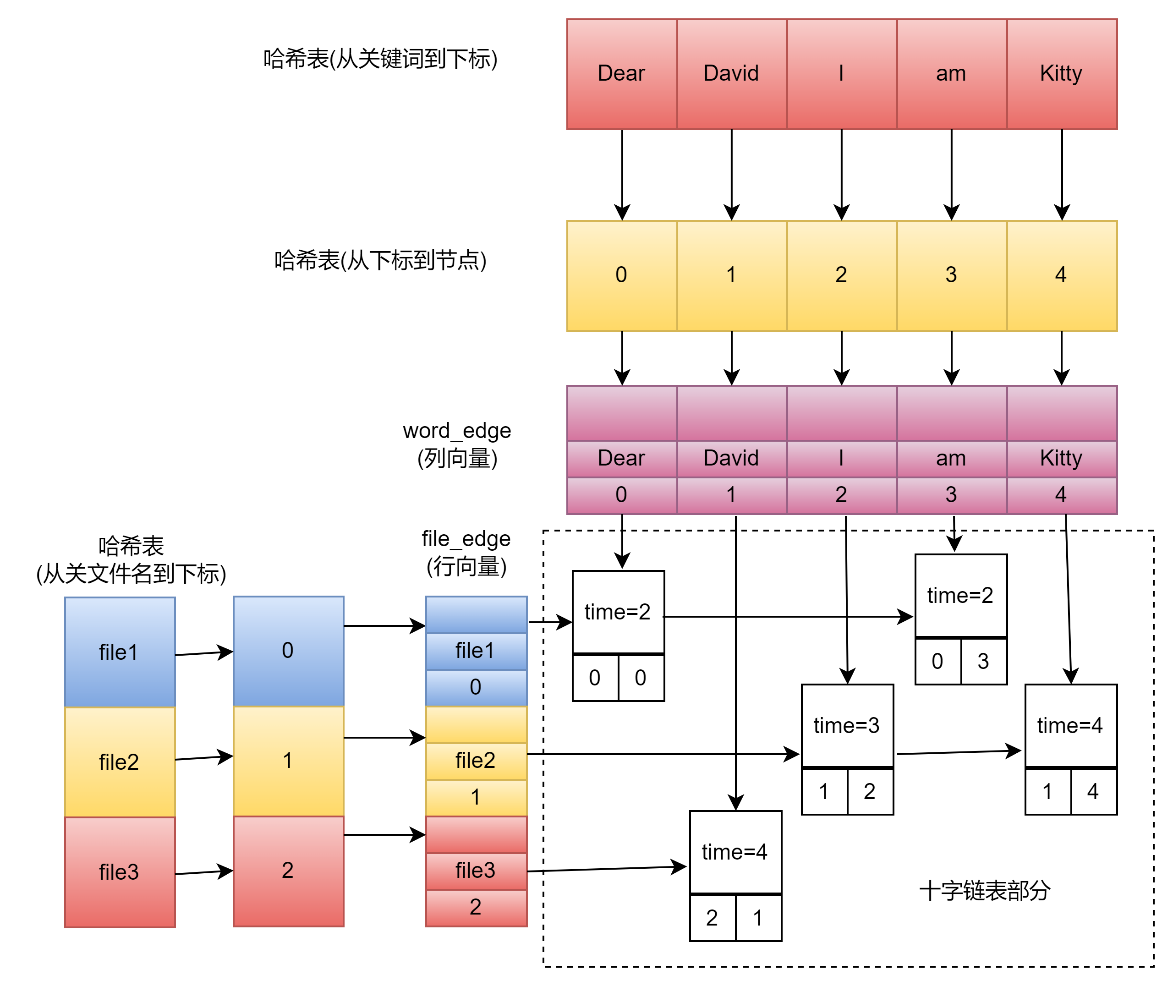
详细的需求和设计分析部分会放在实现流程部分中。

1. **实现流程以及设计**

首先为了实现分词算法，需要建立对应文件夹以及进行文件的读取，在源码中，主程序为textsearcher.cpp, 五篇关键词的文章(仅支持英文分词) 放在src文件夹中， 并使用相对路径进行访问(textsearcher.cpp必须和src文件夹放在同一目录下)。

程序中使用Extension\_func.h中的refine\_word函数进行单词处理，在存储单词时截去头尾部分的符号(如, . [ ] 等)， 仅在稀疏矩阵中存储纯单词(有的可能有部分其他存储)。

1. 由于每一次使用的文章数量是不同的，这就要求矩阵的大小可以进行动态分配，因此借助于c++提供的向量vector和哈希表，实现了动态分配矩阵的大小， 而不使用定义MAX\_SIZE初始化较大空间的方法，有效地节省了内存，另外，对于删除节点时（程序里没有写删除节点的部分，可以缩短向量长度和清除哈希表元素）
2. 为了实现关键词访问的高效插入和删除，使用哈希链表结构进行访问， 而为了实现双向访问，我对每一个词语建立对应的哈希表可以实现由关键词到下标的访问，并对每一个下标建立到edge\_vector的哈希访问关系便于快速插入。而在edge\_node中又定义了对词汇和下标的存储。因而我们在查找时，可以方便由单词找到对应的列，并获取文件名。由节点获取对应的文件和单词时，只需要使用节点存储的行列即可。存储结构图示如下（其中edge和node是两个不同的结构体）



1. 十字链表的建立方法不详述, 在建立链表时，定义一个访问函数，每访问一次，如果节点已经存在， 则词频增加1，如果不存在则新建节点并初始化词频为1
2. 整个输出矩阵的元素时，可以按文件进行输出，也可以按单词进行输出，这个思想和查找是类似的，对于每个文件，只要向右进行遍历， 然后通过节点对应的行列获取对应单词即可实现输出。
3. 查找算法的实现： 对于在多篇文章中查找某一单词，我们是使用模糊方法进行查找的。对于某一指定的字符串，程序采用了最大匹配子串长度的方法来进行， 即使用最大匹配子串长度除需要匹配字符串的总长度方法来进行。
4. 另外还需要说明一下部分没有实现的功能， 在输出查找结果时，可以按照字符串匹配程度(在原本程序中输出的similarity项)和访问频率(repeat time项)进行先排序，将similarity最高的前置，similarity相同使用repeat time进行排序后，将repeat time 多的结果前置后输出， 另外可以设置一个截断误差(如0.4), 如果匹配度低于0.4则不输出结果(认为匹配不足)。 这些想法由于实现起来比较简单和时间紧等原因我没有做，程序中输出所有可能性>0的结果， 并不进行排序。
5. **设计说明**

在代码中定义Sparse\_Matrix类为整个稀疏矩阵，并使用哈希链表进行动态存储节点的值。其定义的类等等， 分为两个头文件Extension\_Func.h和Sparse\_Matrix.h。

Listdir函数用于获取src下的所有文件，在Extension\_Func中。

读取文件的函数在Sparse\_Matrix中内嵌，需要完整文件路径作为传入参数

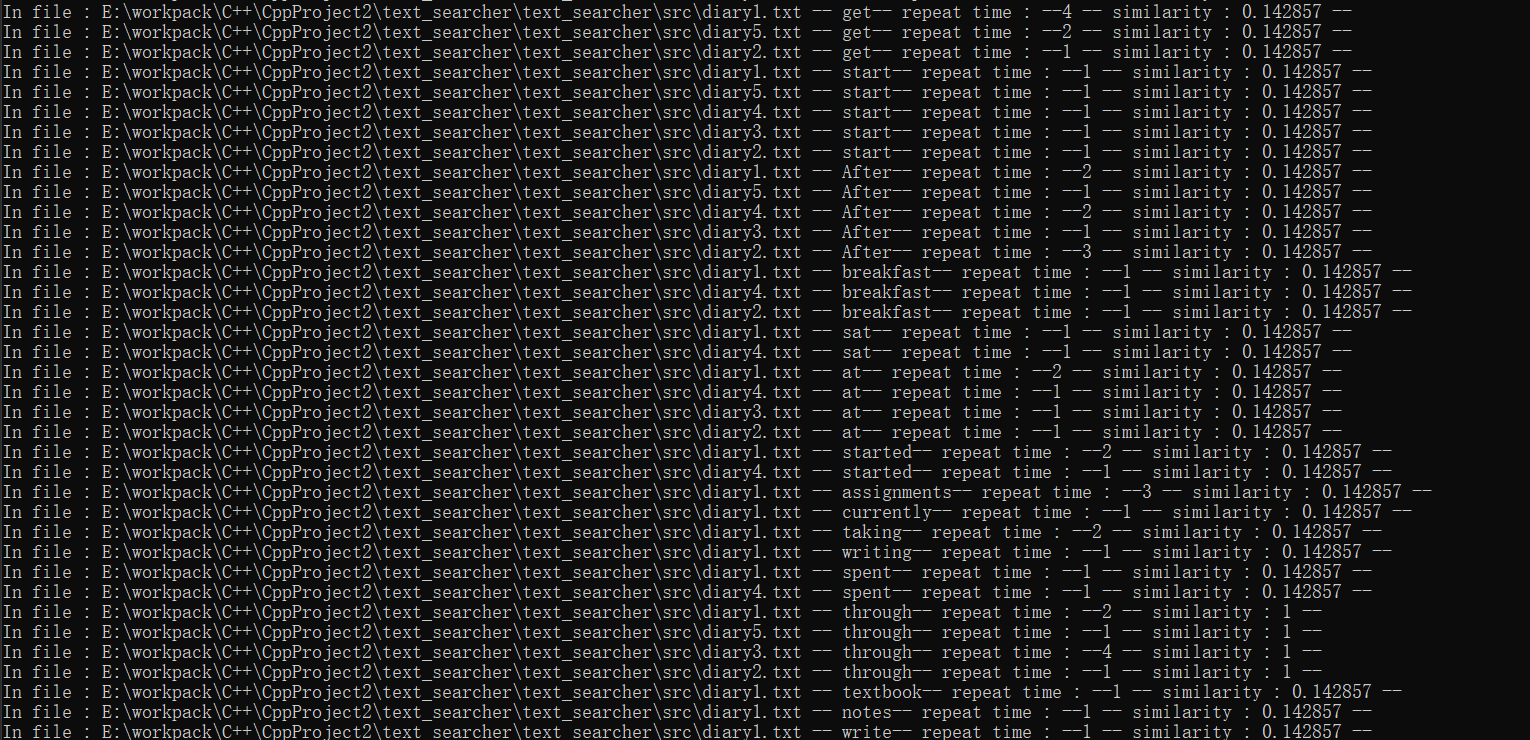
稀疏矩阵的访问和方法均已经放在Sparse\_Matrix.h中

其余主要部分请参考源代码，不详细叙述

1. **使用说明**

本函数会统计src文件夹中的英文文章的所有单词(你可以将里面txt文件的英文文章改成其他txt文本内容, 但是尽量纯单词，可以加标点，但不要带公式什么的, 按照空格分词, 且不支持中文).

打开text\_search.cpp , 初始时24行 m.find("through"); 默认查找的是五篇文章中through的内容, 输出如下图(里面similarity是相似度，这是最重要的一项，里面有几个1的就是完全匹配的查找结果，程序会输出所有有匹配字符的内容, 相似度越高匹配越好)， 文件… .txt 后面的一项(第一行是get)是出现的匹配单词。你可以将through改成其他的，查找其它单词的匹配项。



另外也可以把主程序里面的m.show(true);取消注释，这个会输出整个稀疏矩阵中的内容，true按文件输出，false按单词输出, 使用true输出整个矩阵结果如下:

