中文分词系统的设计与实现

一、 设计目标

本题要求设计和实现一个中文分词系统，将任意给定的一段中文切分成一个一个单独的词。按所推荐的参考算法和数据结构通过编程实现系统，使学生进一步强化 C 语言程序设计能力，掌握动态存储分配、文件读写等高级编程，掌握 Hash 表、Trie 树等数据结构的使用方法，提高综合运用各门课程知识解决实际问题的能力。

二、 设计要求

中文分词是将一段中文的字序列切分成词序列的过程。比如，“我是一名大学生”切分的结果为“我|是|一名|大学生”。中文分词系统所采用的算法较多，一种基本的分词算法为基于字符串匹配的分词方法，这种方法需要用到词典。分词过程中需要频繁地对词典进行查找和匹配，为了提高分词速度，需要使用特殊的数据结构对词典进行索引和存储。

本系统需实现以下基本功能：

1. 词典维护功能：可从文件导入词典，对词典进行增加、删除和修改，将维护好的词典保存到文件，数据存放的格式可以自定义。

2. 索引维护功能：为了提高词典的查找和比对速度，需要对词典建立各种索引。因此，当词典发生变化后，索引需相应更新。系统启动后，索引被加载到内存，索引更新后，应采用合适的数据存放格式将索引存到硬盘，以便下次启动系统时，被加载的索引与系统上次运行时的索引完全一致。

3. 待处理中文文本的输入：待处理中文文本可以有多个段落，每个段落可以有多个句子，每个句子字数不限，还可以带各种标点符号。待处理中文文本可以通过键盘输入，也可以从文本文件导入。

4. 对输入的中文文本进行分词处理：分割后的词两两之间用“|”分隔，保持原文次序。提供保存到文本文件的功能。

5. 用户界面功能：系统需要有良好的用户界面，方便操作。用户界面可以采用简单文本菜单界面、下拉式文本菜单界面和图形菜单界面。

在实现上述功能基础上，如果系统还能正确切分中文文本中的英文单词、各种数字，则可以酌情给予附加分奖励。

三、 参考算法

为了充分发挥大家的编程能力，题目对系统实现所采用的算法和数据结构没有限定，也就是说，只要分词结果足够准确、切分速度足够快，任何算法和数据

结构都可以采用。事实上，合适的算法和数据结构可以大大提高切分准确性和速度。这里给出的算法和数据结构是有效的，但不一定是最优的，仅供参考。

【基于字符串匹配的分词方法】又叫做机械分词法，这种方法要事先准备一个“充分大”的词典，然后将待切分的句子按照一定的扫描规则与词典中的词条进行匹配。如果匹配成功，则将这个词切分出来，否则进行其他相关处理。按照扫描方向的不同分为正向匹配和逆向匹配；按照不同长度优先分配的情况，分为最大匹配和最小匹配。常用的基于字符串匹配的方法有如下四种。

(1) 正向最大匹配分词法

之所以称为最大匹配，就是要求每一句的切分结果中词组的总数最少。比如在“我们是中华人民共和国的公民”这句话中，我们可以清楚地判断，如果在词典中进行匹配，只要匹配成功就切分出来，那么这句话可能被切分成“我们|是|中华|人民|共和国|的|公民”，该结果中一共包含 7 个词。但是，为了实现最大匹配，我们将把“中华人民共和国”作为一个整体的词进行处理。因此就要求将上面这句话切分为“我们|是|中华人民共和国|的|公民”，一共是 5 个词，根据最大匹配的原则，我们选择后面这种分词结果。

正向最大匹配法又分为增字匹配法与减字匹配法。增字匹配法需要一种特殊结构的词典，这种方法能够达到非常高的分词效率，在此只介绍减字匹配法。减字匹配法的流程如图 1 所示，首先将句子读入，然后将句子在词典中进行查找匹配，如果没有匹配成功则在句子末尾减去一个字，再在词典中进行查找匹配，重复上述过程，直到匹配上词典上的某个词组或只剩下一个字符，接着将句子剩余的部分重复上述流程，直到将句子全部分解成原子或词典中存在的词组。

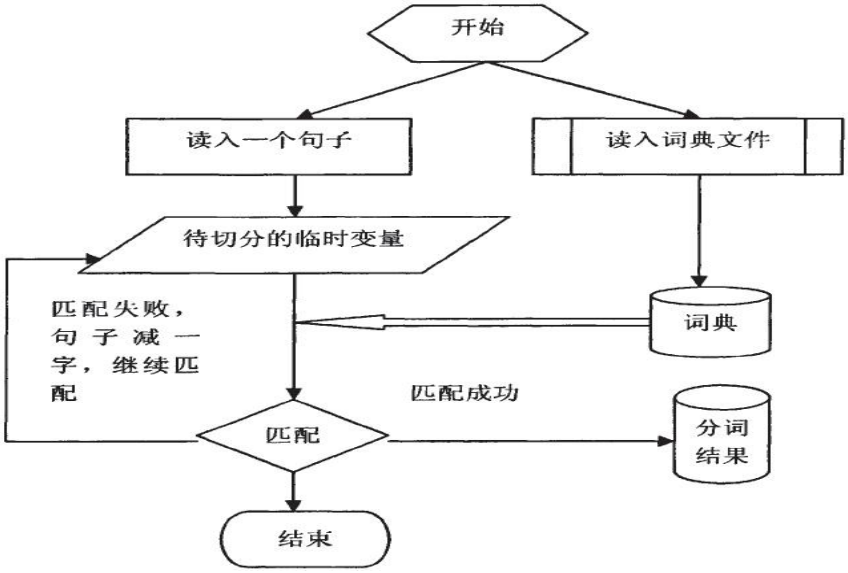

图 1 减字分词法处理过程

图 1 是减字法的分词过程，为了说明该方法是如何分词的，我们利用减字法进行正向最大匹配分词，以“我们是中华人民共和国的公民”为例。如果事先知

道词典的最长词长，那么将减少很多步骤，从而提高分词速度。此处假设词典中最长词长为7。那么实例的整个匹配过程如下所示：

步骤	操作的句子	操作	分词结果
1	我们是中华人民共和国的公民	只取7个字	
2	我们是中华人民	词典匹配失败	
3	我们是中华人	词典匹配失败	
……	……	……	
	我们	词典匹配成功	我们
	是中华人民共和	减词并取前7个字	
	是中华人民共和	词典匹配失败	我们
	……	……	……
	是	词典匹配成功	我们\是
	中华人民共和国	减词并取前7个字	
	中华人民共和国	词典匹配成功	我们\是\中华人民共和国
	……	……	……
	NULL	减词	我们\是\中华人民共和国\的\公民

(2) 逆向最大匹配分词法

逆向最大匹配分词的扫描方向与正向最大匹配分词相反，是从句子的结尾开始扫描，直至句首。统计结果表明，单纯使用正向最大匹配的错误率为1／169，单纯使用逆向最大匹配的错误率为1／245，显然逆向最大匹配分词法在切分的准确率上比正向最大匹配分词法有较大的提高。比如待切分字串为“瑞星以技术和服务开拓网络安全市场”，那么正向最大匹配的结果为“瑞|星|以|技术|和服|务|开拓|网络|安全|市场”，逆向最大匹配的结果为“瑞|星|以|技术|和|服务|开拓|网络|安全|市场”，而前者是错误的，后者才是正确的结果。当然，这也只是个特例，一定会有正向最大匹配比逆向最大匹配更准确的情况。

关于正向最大匹配算法和逆向最大匹配算法相应的词典组织结构，我们做如下设想：设置正向最大匹配的词典中的词是正序的，而逆向最大匹配的词是逆序的，比如前者词典中的一个词为“见义勇为”，则后者词典中为“为勇义见”。这样可以省略处理待匹配字符串顺序的工作，从而节约时间。

(3) 最少切分分词法(使每一句中切出的词组数目最小)

(4) 双向匹配法(将正向最大匹配法与逆向最大匹配法组合)

最大匹配法分词存在如下缺陷：首先，词典词长限制，词长过短，长词就会被切错；词长过长，查找匹配效率就会比较低。其次，掩盖分词歧义，不能发现

交叉型歧义。最后，最大的匹配并不一定是想要的分词结果。但这种方法的优点在于实现简单，而且切分速度快。

四、 参考数据结构

中文分词主要关心两项指标：切分速度和切分精度。由于在分词过程中，需要频繁地在词典中进行信息查找匹配，因此要求设计出高效的数据结构来组织词典，以提高查找匹配的速度，这样才能满足上层应用的性能要求。使用索引来组织数量庞大的文件是一种高效的方法。这里推荐两种用于组织词典的索引方法：Hash 索引和 Trie 索引树。

1. Hash 索引

Hash 函数是一个映像，其将关键字的集合映射到某个地址的集合。用 Hash 表的方法构造词典就是将关键字与表项的存储位置建立一个对应的函数关系。以首字 Hash 词典机制的原理为例，据汉字机内码的编码规律可知，我们就可以通过一对一映射的 Hash 函数实现词首字的快速查找。根据 Hash 函数的定义可知，Hash 函数一般都无法避免冲突，所以通常还要有相应的冲突处理方法，因此对于词组中的剩余字串最快的只能通过二分查找来进行查找。我们的思想是基于 Hash 索引的词典机制就是构造一种 Hash 函数来计算词语的 Hash 值，将 Hash 值相同的词组放入一个通常称之为“桶"的集合内。匹配时先计算待查词的 Hash 值，得到首字的存储位置，然后再进入相应的 Hash 桶内再进行二分查找。

2. Trie 树

键树又称单词查找树、前缀书。它是一棵度>=2 的树，树中的每个结点中不是包含一个或几个关键字，而是只含有组成关键字的符号。例如，若关键字是数值，则结点中只包含一个数位；若关键字是英文单词，则结点中只包含一个英文字母。键树中每个结点的最大度 d 和关键字的“基”有关，若关键字是英文单词，则 d=27，若关键字是数值，则 d=11。键树的深度 h 则取决于关键字中字符或数位的个数。若以树的多重链表表示键树，则树的每个结点中应含有 d 个指针域，此时的键树称为 Trie 树。

若从键树的某个结点开始到叶子结点的路径上，每个结点中都只有一个孩子，即为单支树，则可将该路径上的所有结点压缩成一个“叶子”，且在该叶子结点中存放关键字值及指向该元素的指针域。在分支结点中不设置数据域，每个分支结点所表示的字符均由其双亲结点的指针所指向的位置决定。

在 Tile 树中的查找操作过程如下：从根结点出发，沿着与给定值相应的指针逐层向下直到叶子结点，若叶子结点中的关键字与给定值相等，则查找成功；若分支结点中与给定值相应的指针为空，或叶子结点中的关键字与给定值不相等，则查找失败。

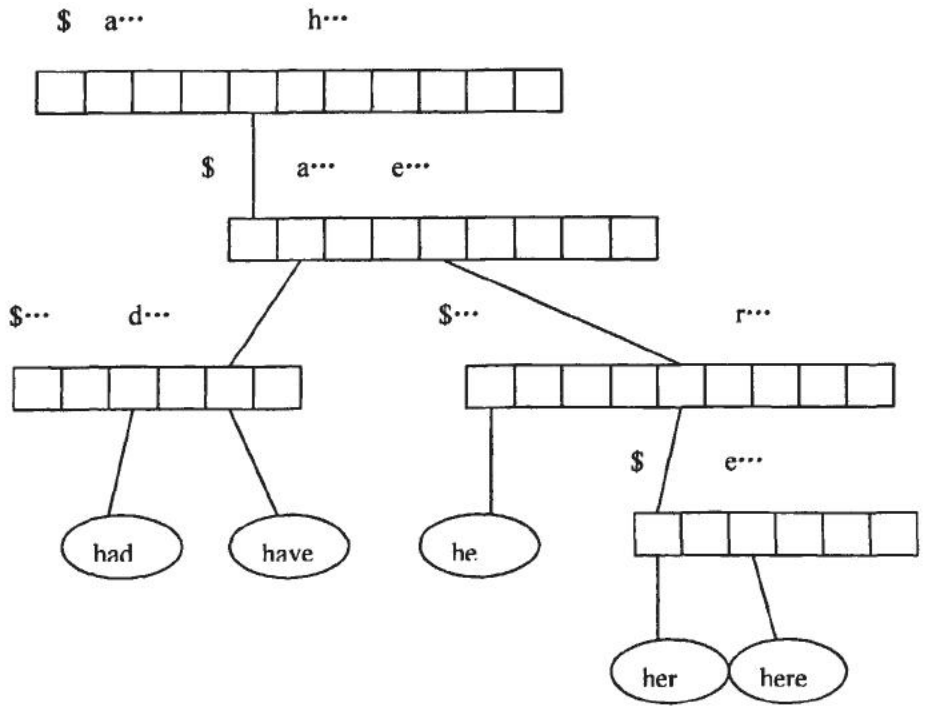

图 2 一个字母类型的 Trie 索引树结构

图 2 是一个字母类型的 Trie 索引树，比如查找单词“have”，则首先在第一级索引中查找字母“h”，接着按照第一级索引中“h”对应的指针找到对应的第二级索引，再在其中查找字母“a"，……，依次查找下去，最后找到叶子结点中的关键字“have”，与给定值相等，说明查找成功。又比如查找单词“hza”，同样首先在第一级索引中查找字母“h”，接着按照第一级索引中“h”对应的指针找到对应的第二级索引，在其中找到字母“Z”，此时的指针为空，查找失败。

3. 基于 Trie 索引树的分词词典机制

基于 Trie 索引树的分词词典机制，如图 3 所示。这种分词词典包括两个部分：首字 Hash 表、Trie 索引树结点。Tile 索引树的优点是在对被切分字串的一次扫描过程中，不需要预先知道待查词组的长度，沿着树链逐字匹配即可；缺点是它的构造和维护比较复杂，而且都是单词树枝(一条树枝仅代表一个词组)，由于词典在进行中文分词时会事先读入内存，因此这种空间浪费了一定的空间。

下面举一个实例进行说明，以查找 S=“你小子大白天还在睡觉”中从“大”字开始的最长词(及所有词)为例，按照图 3 的词典结构，其匹配步骤如下：

(1)首先通过首字散列表得到以“大”字开头的词的 Trie 索引树结点，假设为 T1；

(2)由于结点 T1 中包含关键字“NULL”(表示空字符)，因此“大”字是一个词。在结点 T1 中用二分法查找关键字“白”，其指针指向的目标结点假设为 T2。

(3)结点 T2 中包含关键字“NULL”，因此“大白”也是一个词。在结点 T2 中继续用二分法查找关键字“天”，此时“天”的目标结点已是叶子结点，所以

“大白天”也是一个词，查询结束。最后得到“大白天”为最长词。

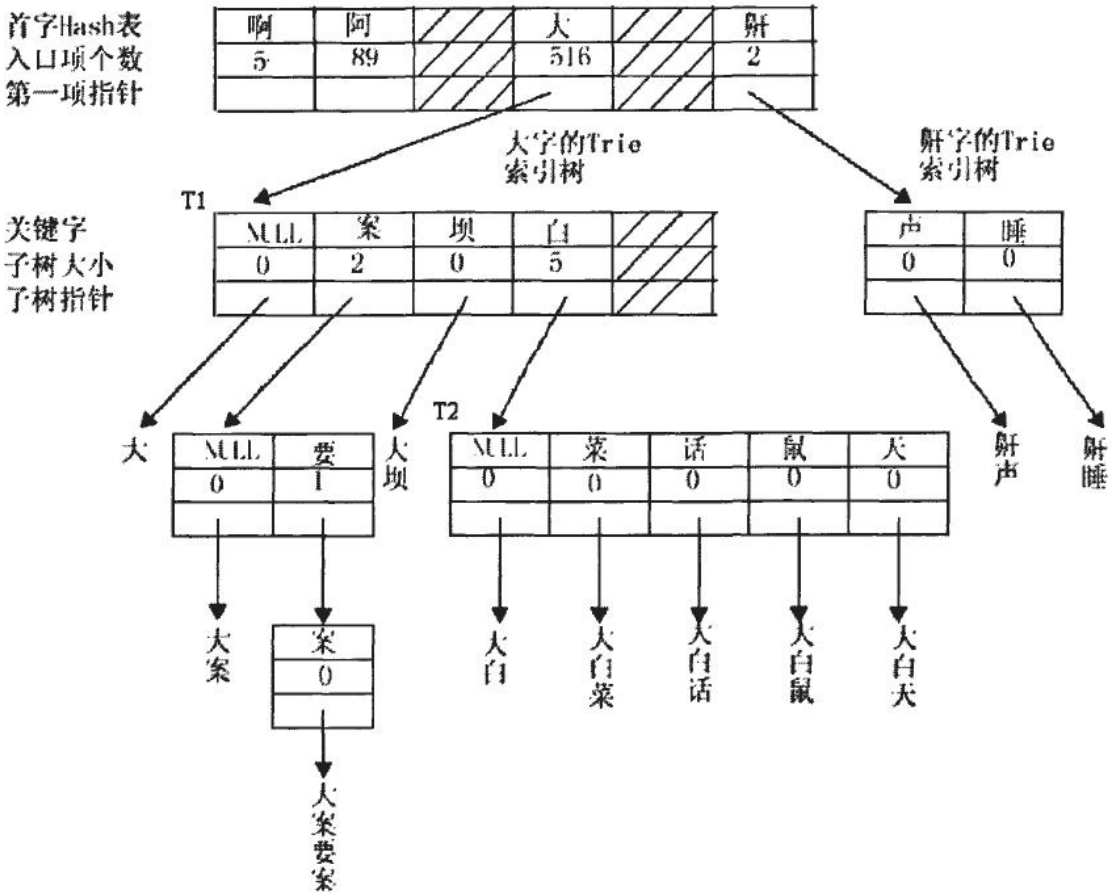

图 3 基于 Trie 索引树的分词词典机制

从上图可以看出，为了节省空间，此处的 Trie 索引树并非如图 2 所示，把所有汉字索引都列出来，而只是列出了词组存在的部分，故上述步骤(2)采取二分查找的方法来进行查找。而在图 2 中是可以直接根据指针去查找是否在该位置是否存在词，虽然图 2 的方法速度更快，但严重浪费了空间。

五、 中文分词参考词典

见文件 dict.txt，也可根据需要自行在网上下载。