**大规模字符串检索设计文档**

宋其翰 王存

1. **功能分析**

本次作业完成大规模字符串查找功能，主要功能是在1000万个Email地址中找给定的100个Email是否存在。本次作业我们小组采用的压缩Trie树的方法，通过对1000万个Email地址建立一个压缩Trie树，然后查找这100个Email地址。主程序名为strfind，接受三个参数，第一个参数为1000万个Email地址所在文件，每行一个Email地址；第二个参数为100个待查找的Email地址；第三个参数为输出结果文件名。输出结果为待查找Email地址存在情况，找到为yes，没有找到为no。

1. **架构设计**

本系统整体架构如下图1所示，由初始化、建立Trie树、查找Trie树三个部分组成。其中：

初始化：主要实现程序的初始化，用来初始化Trie树。

建立Trie树：从待建树的文件中读入Email地址，并将该Email地址进行逆序，最后将逆序的Email地址插入到Trie树中。

查找Trie树：在建立好的Trie树中查找待查找的Email地址，并输出到输出文件中。

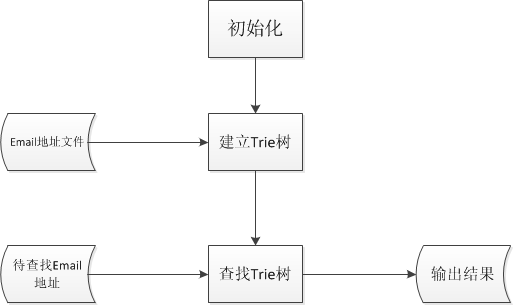


图1 程序架构图

**模块之间的接口：**

初始化模块初始化好Trie树，建立Trie树模块向初始化好的Trie树中插入Email地址，查找Trie树模块在建立好的Trie树中查找对应的Email地址是否存在。

1. **核心模块设计**

本系统的核心模块主要包括建立Trie树和查找Trie树模块，下面将主要介绍这两个模块的设计和核心流程。

* 1. **建立Trie树模块**

本模块是最核心的模块，使用压缩Trie树的数据结构，实现了Email地址的建树功能。

（1）Trie树节点定义

typedef struct trieNode

{

char result; //标明此节点是否为某字符串的终点，为节省空间遂用char

char \*strPointer; //该节点所存储字符串的指针

struct trieNode \*brother; //该节点的兄弟节点

struct trieNode \*son; //该节点的子节点

}trieNode;

图2 压缩Trie树的定义

result中主要存储该节点是否为终止节点，即匹配到这里是否为字符串终止。strPointer表示该节点中存储的压缩Trie树的字符串。brother和son分别表示该节点的兄弟节点和儿子节点，兄弟节点即和该节点字符串具有相同位置的字符串节点，儿子节点即是该节点字符串后面的字符串节点。

（2）Trie树的插入流程

Trie树的插入时，使用一个char指针str指向待插入的字符串（已逆序），t指向头节点。插入操作放在一个while循环中完成，当str指向的字符串为空时，循环结束。while循环中主要比较str指向的字符串和t->strpoint指向的字符串的开始的字符相同的个数。若大于0，则将节点t分为两个，插入str指向的字符串，并将str向前移动相同的个数；否则，直接插入str，循环结束。主要的流程如下图3所示。

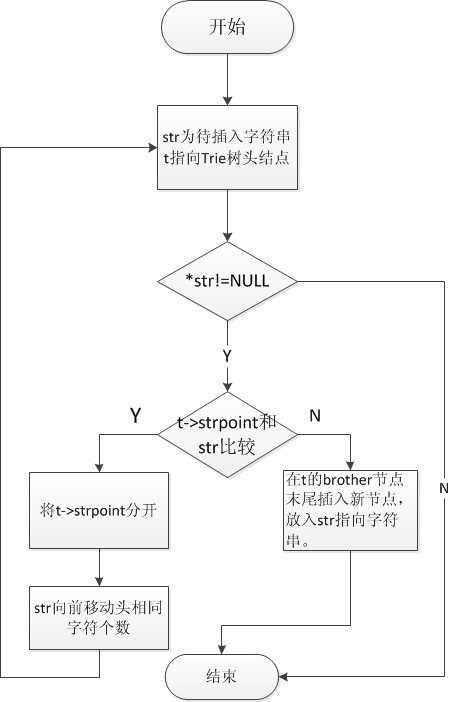


图3 Trie树的插入的主要流程

在插入Trie树时，如果需要将t指向结点分开，分开的示意图如下图4所示。

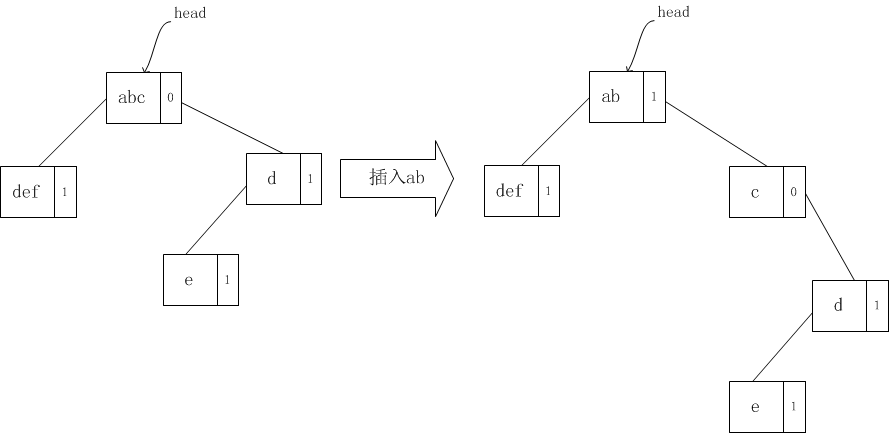


图4 将t分开示意图

图4左边表示插入了abcd，abce，bcd三个字符串后的Trie树，现在要插入ab字符串。右边表示插入了ab字符串后Trie树的样子示意图。图中的Trie树每个节点有两个值，一个是该结点指向的字符串，另外一个是result字段，表示当前是否为一个字符串结束，防止判断出现错误情况。

同时，在插入时，如果发现相同字符串的个数和结点指向的字符串相同，则直接将t指向其孩子节点，str先前移动即可，不用将t节点分开。

* 1. **Trie树查找模块**

当建立好Trie树后，Trie树的查找就比较容易了。运行时，将t指向头节点，str为待查找Email地址。先将Email地址逆序，然后依次比较t中的字符串是否有和str中相同的头节点个数，如果相同，则str向前移动对应的字符个数，t指向其孩子结点。若不相同，则t指向其兄弟节点，继续比较。具体流程图如图5所示。

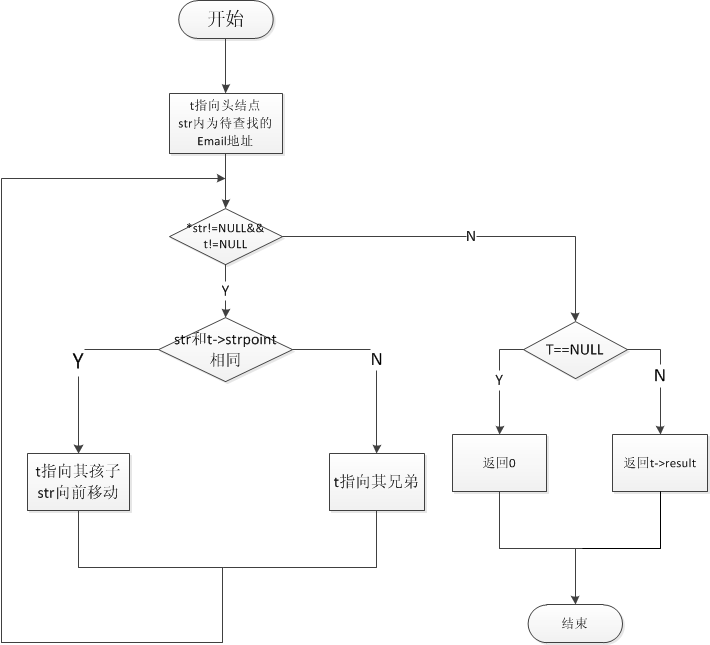


图5 查找Tire树的流程图