**主类（含main方法）：**

Main.java

**命令说明：**

**find**  查找某个单词

**delete** 只用一个表删除指定文件

**newdelete**  借助表2删除指定文件

**help** 查看所有命名

**对比效果：**

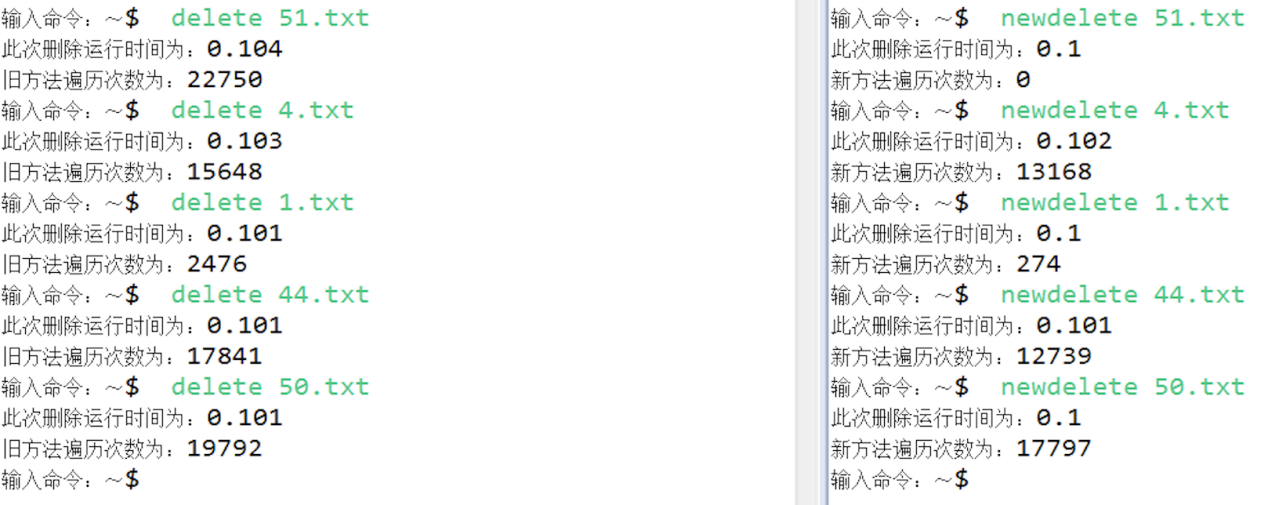


图1 效果图

此结果是经过上千次查询得到的结果，可知时间差异不大，但能够根据不同情况不同程度上降低遍历次数。

**实现代码：**

**public static int deleteFile(String filename){**

**// 需要首先删除数据源**

**int flag=0;**

**int count=0;**

**long start=System.currentTimeMillis();**

**try {**

**for (int i = 0; i <10 ; i++) {**

**Thread.sleep(10);**

**}**

**}catch (Exception e){**

**e.printStackTrace();**

**}**

**for(Files file:files) {**

**if (file.toString().equals(filename)) {**

**files.remove(flag);**

**break;**

**}**

**flag++;**

**}**

**//开始遍历mainMap中的数据 最后从mainmap中移除filename代表的文件 注意存在性问题**

**List<Words> inmapList=mainMap.get(filename);**

**if (inmapList==null) return 0;**

**mainMap.remove(filename);**

**count=remove(inmapList,filename);**

**long end=System.currentTimeMillis();**

**System.out.println("此次删除运行时间为："+(end-start)/1000F);**

**return count;**

**}**

**// 根据传入的list 进行删除操作 ，两个删除方法体现在传入的list差异**

**private static int remove(List<Words> list,String filename){**

**int count=0,fileflg=0;**

**for (Words word: list) {**

**for (String str : word.files) {**

**if (str.equals(filename)) {**

**word.files.remove(fileflg);**

**break;**

**}**

**fileflg++;**

**count++;**

**}**

**fileflg=0;**

**}**

**return count;**

**}**