目录

[思路 1](#_Toc534111019)

[单元测试 1](#_Toc534111020)

## 思路

1: 首先很明显的全部载入内存做法非常简单，全部载入内存，对一条URL用map进行hash，在统计每条URL的次数，最后插入堆或者数组排序即可(前者可自动排序，后者需要手动排序，排序规则由自己定，差别在于后者的插入速度更快，对于全部插入在询问的情况适合)，取出top100就是答案，我用C++实现了，有点是速度快，缺点是内存占用大。

2: 那么对于内存有限制的我们只能牺牲时间换空间了。大致做法为将大文件的每条URL取出来hash后丢进一些小文件，保证同一个地址会丢进同一个小的文件，然后在对每一个i小文件进行map计数，用堆或数组排序维护top100，再次输入到对应文件中，保证内存够用，最后对这些小文件再进行读取归并取出top100。

具体做法为先将这个100G的大文件每条URL取出来，用一个字符串hash函数hash(常用乘随机种子)成一个数字，将此数字%500+1后输出到相应小文件中(这里的500是表示划分成500个小文件，可按情况增减)，因为相同的hash函数所以同一个URL地址一定会再同一个小文件中。然后对这500个文件进行map统计次数，维护top100，将次数和URL地址再次输出到相应文件中，清空内存。最后再对这500个文件进行取出来进行相同的归并，得出最后的top100的URL地址和对应次数。

优点: 可处理大数据文件，做到大化小，分而治之。缺点: 因为频繁的对文件进行操作，速度很慢

## 单元测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试模块 | 预期结果 | 实际结果 | 备注 |
| URL类排序规则 | 次数多的在前，次数相同的url地址字典序小的在前 | 符合预期效果，如图1所示 | 对几条样例进行测试 |
| FileDivided类划分文件功能 | 能将一个较大文件划分为很多小文件 | 符合预期效果，如图2所示 | 小文件测试结果 |
| MainProcessing类处理中间过程和结果输出功能 | 维护每个小文件的top100，最后归并所有小文件的top100取出最终结果的top结果，并输出到文件中 | 符合预期效果，如图3所示 | 小文件测试结果 |

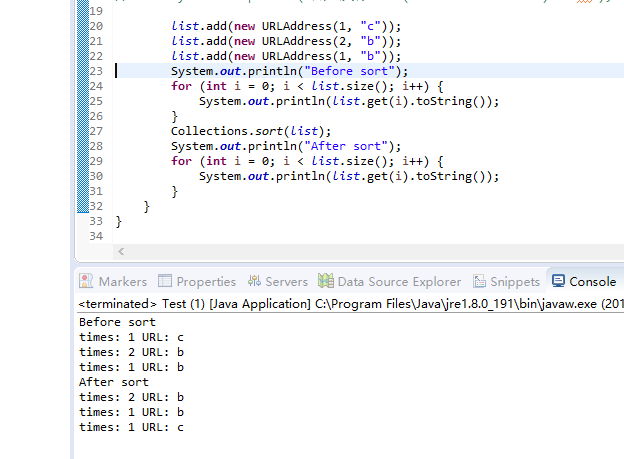


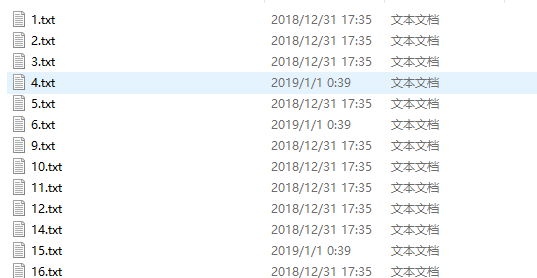
图1 

图2 

图3