# 第五题:搜索引擎的实现

## 1．需求规格说明

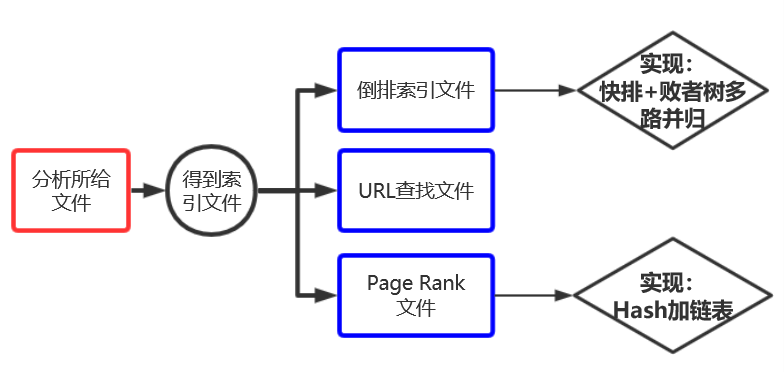
对于给出的大量的网页信息，能够运用多种所学的数据结构知识，进行分析处理，最后根据所给的单词进行快速的网页查询。

## 2．总体分析与设计

### （1）设计思想：

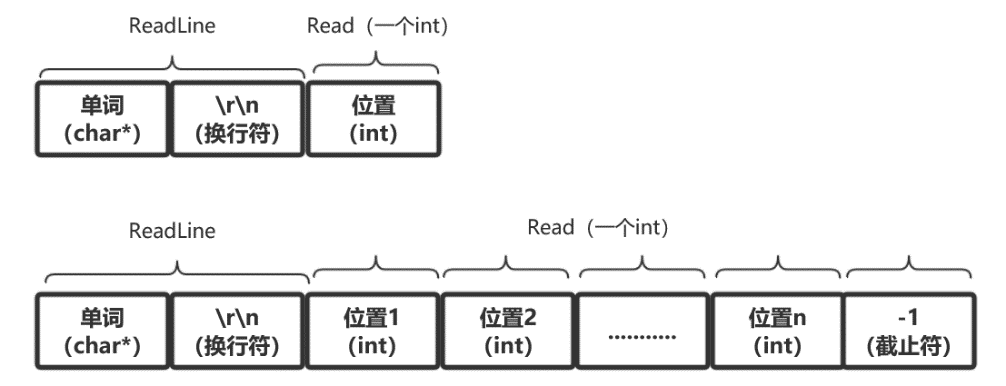
首先需要处理原文件，可知原文件格式为PTQL，PT主要表示网页的主要信息，Q是文件关键字，用于制作倒排索引，L为超链接，用于Page Rank计算排名。因此为了方便之后的查找，减少内存使用，可根据所给文件生成多个辅助文件，当查找网页的时候再打开这些文件，根据文件信息进行查找。因此本题分为两个阶段，第一阶段是制作出多个辅助文件。，第二阶段是处理查询的信息，得到结果。

**第一阶段：**



本阶段需要产出倒排索引文件与Page Rank文件。为了存储方便又加入了URL查找文件。里面主要是存储P与T和此网页的int型编号。

1.倒排索引文件：倒排索引格式主要是先一个单词，然后给出所有的出现的位置。这样通过单词直接知道所在网页的位置。但是一开始单词较为分散，因此解决方法是先一个单词一个位置来写到文件里。然后对这些文件进行排序，可知同名的单词必定在一起，于是可将同名单词的位置合并，另外为了读取方便，在单词最后写入一个-1，截止符。这样可以读到单词的位置已经截止。



为了提高效率，主要还是需要更合理的进行排序，如果是所有的单词都按照一个单词一个位置这样，再进行排序那么排序时间是非常大的，当然内存可能也不太够用。因此采用外部排序的方法，虽然可能相对较慢，但是如果内存较小，仍可以运行。首先将大文件分段。例如每得到10000个倒排索引（单词与一个位置的格式），再进行排序，然后进行初次合并，就将他们写到一个文件里。这样可以得到多个小的有序的文件，然后将多个有序的小文件，合成一个大的文件，最终可以得到有序的大的文件。其中再进行合并，就可以得到所需的倒排索引文件，真正的字典。

其中，初次的10000个倒排索引采用的是快速排序，因为感觉效率是最快的，且内存占用足够。而对于小文件合成大的文件这种算法，采用的是败者树的方法。可以按出这一阶段就是一个归并排序，如果一个一个比较再写入的话时间消耗太多，为了改进，若用最小堆的话会得到一定的改进。但是这只能一次选出一个最值，无法利用比较过的数据信息，这样浪费了一些比较结果的信息。再次改进，可以利用败者树进行改进，败者树相当于一个变种的堆，既得到了当前的最值，又利用了失败者（非当前最值）的信息，这样能够减少比较次数，将时间消耗进一步缩短。

2.URL查找文件：倒排存入的URL地址为int型，可以减少文件占用大小，且格式规整，便于处理。但是需要知道那个数字（int）代表着哪个网页。因此需要制作一个URL的字典。其中，URL的代号为原始文件中P的顺序。存入格式为一个代号（int）一行P，一行T。即代号，URL，时间。

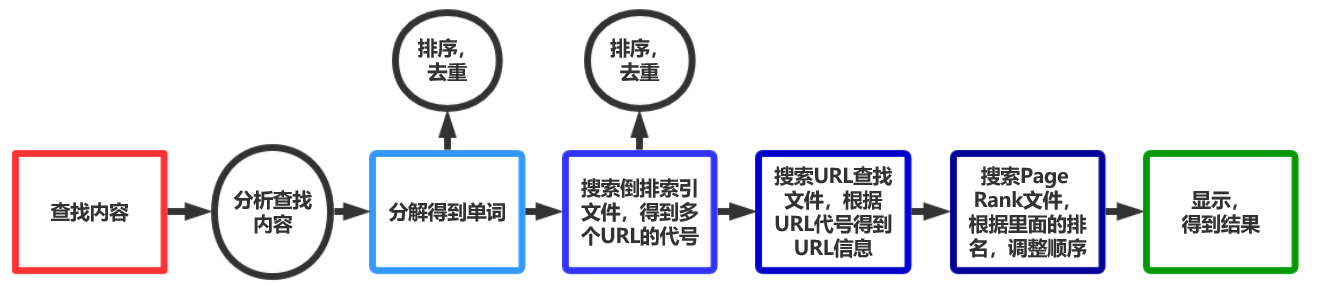
3.Page Rank文件：相当于制作一个n\*n的矩阵，通过利用原始文件中的L，超链接信息。知道当前网页的“选票”投给了哪些网页。遍历一遍后，再将各网页得到的选票加起来，根据得票多少进行排名，按照 网页代号（int） 名次（int）存入文件中。

对于超链接信息，不知道这些URL的代号是什么。于是我们可以在制作倒排的时候建一个URL的Hash链表，这样可以方便查找，即得到超链接URL的代号。知道当前网页的票投给了谁。

根据Page Rank的规则，一个网页只能有一票，因此如果投了多票的话，则进行平均除法。投的票为分数形式。

为了减少内存的使用，且根据计算的规律，不需运用矩阵的方法，只需用两个长度为n的数组，一个存取各网页得票，另一个存取当前网页票数投给谁，这样进行n重的累加，最后得到总票数。再进行快速排序，得到名次。

**第二阶段：**



本阶段主要是根据查找的单词，先根据倒排文件得到所有的网页代号，然后根据URl查找文件得到真正的网页信息，再根据Page Rank文件得到网页的这些网页的相对排名，最后根据排名顺序依次显示。

## 3．编码

遇到的问题：

1.主要是关于倒排文件存入的格式进行了多次改进。最后得出了这种方式。

2.对于败者树的学习，修改bug花费一定的时间。

3.对于Qt的学习，花费了部分时间修改bug，进行调试等。

## 4．程序及算法分析

不足与改进：

1.希望能想到更好的思路来提高查找效率。

2.对于一些需求没有实现与完善。

3.只能运行比较小的数据，对于大数据耗时较慢。主要应该是page rank的计算，应该申请更大的hash表，但是这样再遇到更大文件还是无法根本解决问题。这一方面还需要深入的思考。

4.对于多路归并来说，io操作太多，可能需要一段一段读取，例如每个文件一次性读入500条数据。而在写入的时候当每满10000条数据就写入一次，这样来减少磁头的调换。提高效率。

5.对于第二阶段的查找来说，总感觉设计思路有局限性，因为需要遍历文件进行查找，耗时太多。想到是否能用hash表将大文件拆分成小文件这样进行读取，这样耗时也会降低，但是遇到更大的文件仍不是一个好的解决方法，仍需要进一步思考。

## 5．小结

本题主要是一种数据结构综合的应用，根据所学的知识，建立合理的数据结构来减少时间的消耗。因此，通过制作搜索引擎，我学习到了如何将已知的数据结构知识，针对一种需求进行真正的应用。感觉相当于 一种学期总结一样，制作本题时，想到了许多学到的数据结构，当然也自学了许多其他的衍生的数据结构，来提高程序的效率，确实对于处理大数据来说，我能够更直观的感受到了时间复杂度。更好的感受到一个好的算法的耗时程度。当然本题还有许多不足之处需要改进与学习，一部分需求还没来得及实现，需要进一步的优化。

PS：

第二题：JSON：

修复了一些小的bug：对于空节点能够更好的显示，对于空的数组也可以正常运行。

但是未来得及（之前整天吃喝玩乐，忘了学习）做查找算法。。。。ಠ\_ಠ （这就很尴尬了），想到也用栈来写会更好。

第四题：拼图：

改进了打乱算法，将正确的位置随机移动若干步进行打乱。

同时修复了bug，能够支持16格拼图的A\*寻路。但是好像还是有点慢。。。