316 Data Structures

Assignment 4 – Web Page Search

(Due: Nov. 11, 2012)

Internet search engines such as *Google* (http://www.google.com) returns a sequence of web pages (URLs) based on keywords provided by a user searching for information on the web. One or more keywords are provided and the web pages returned are typically ordered according to some likelihood of importance. Some preprocessing is often done on current web content to ensure that succeeding search operations are both efficient and fairly accurate.

Assignment:

For this assignment, you will write a C++ program that implements a simplified Web search engine. There will be one keyword per search and the relative importance of a web page is predetermined. The program will take two input files.

Web page data file. This file represents current web content and the program will begin by reading in this file and incorporating all the data in its search engine (the preprocessing stage). It will contain several lines of text, where each line contains a web page URL, followed by keywords that are associated to that web page. The URLs appear in this file according to their order of importance. The format of a line in this file is summarized below:

<URL> < keyword-1> <keyword-2> … <keyword-i>

The URL and the keywords are strings of characters and they are separated in the text line by one or more spaces. Note that it is possible to have no keywords associated to a URL; i.e., no words follow the URL field in the data line.

Search file. This file will have lines containing exactly one keyword each. Each keyword will represent a search performed on the engine. For each keyword search, the program will print a sequence of web pages which are associated to that keyword.

Output will be directed to a file. Each line in the search file will correspond to a sequence of lines in the output file. The format for this sequence of lines is as follows:

<keyword>: <j> web pages found, <stats> keyword lookups.
<URL-1>
<URL-2>
…
<URL-j>

Where **<keyword>** is a string indicated in the search file, and **<URL-1>** through **<URL-j>** are the associated web pages. The associated web pages must be in the same relative order as they appear in the web page data file. The **<stats>** field indicates the number of keyword lookups performed during the search. Finally, after processing all keywords in the search file, a summary

line is printed that provides the total number of keywords processed, the total number of web pages returned, and the total number of keyword lookups performed. The format for this last line is given below:

TOTALS: <k> keywords, <w> web pages returned, <l> keyword lookups.

Data Structures:

You are required to use the following data structures for this project:

- You will use an array to store the keywords, and these will be stored in sorted order, so that you can later perform binary search on the array as you process on the search file. This means that as a new unique keyword is encountered in the web page data file, it is "inserted" into the array, so that correct order is preserved. This will be relatively inefficient but tolerable in practice since it will occur during the preprocessing state. You may assume a maximum of 1000 distinct keyword. The **<stats>** field for a keyword in the output file is the number of array lookups that are required to find the keyword in the binary search.
- Each array element above will be associated with web pages and these will be represented as a linked list of nodes ordered according to importance (the order in which the web pages occur in the data file). Since it is common that a web page is associated to more than one keyword, you should have a separate linked list that stores all unique web pages, so that the list associated with a keyword have nodes that just point to the appropriate node in the separate list (to avoid having duplicate web page data).

Figure 1 describes these data structures and their interrelationships.

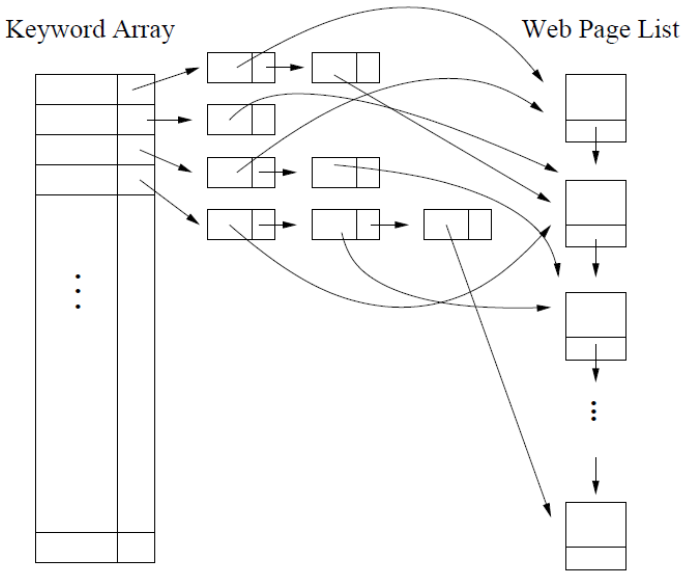

Figure 1: The keyword array on the left has a maximum size of 1000 and is sorted by keyword. The web page list on the right contains the pages as they appear in the data file. Each list in the center is associated with a keyword and its nodes link to the web page nodes on the right.

Program Invocation:

The program is invoked as follows:

websearch **<datafile> <searchfile> <outputfile>**

If the command line contains an insufficient number of arguments or if any of the specified input files does not exist, the program should print an appropriate error message and exit.

Submission: follow the instructions at http://cs.uakron.edu/~echeng/Submission_How.html. Use course number 3460:316.

Sample Input and Output Files:

In the example below, **out1.txt** will be produced from the input files **data1.txt** and **search1.txt**.

data1.txt

```
http://cs.uakron.edu/~echeng/ds/index.html structures algorithms
http://www.sorting-algorithms.com/ algorithms bubble heap
http://www.buildings.com/ cement structures building architecture
http://www.snpp.com/ simpsons cartoon
http://geeks.net/ cartoon algorithms
```

search1.txt

```
cartoon
algorithms
data
building
heap
```

out1.txt

```
cartoon: 2 web pages found, 1 keyword lookups.
   http://www.snpp.com/
   http://geeks.net/
algorithms: 3 web pages found, 3 keyword lookups.
   http://cs.uakron.edu/~echeng/ds/index.html
   http://www.sorting-algorithms.com/
   http://geeks.net/
data: 0 web pages found, 3 keyword lookups.
building: 1 web pages found, 4 keyword lookups.
   http://www.buildings.com/
heap: 1 web pages found, 2 keyword lookups.
   http://www.sorting-algorithms.com/
TOTALS: 5 keywords, 7 web pages returned, 13 keyword lookups.
```