Logan Adams/Danny Nelson

CS 303 – Crenshaw

Homework #4

April 2, 2013

Extra Credit:

For the extra credit, we did not alter our program, but rather wrote a program that would allow us to generate our own large test files for the purpose of testing our code on. Our code to create the additional test files can be seen in Appendix A. This program created 6 files of the needed sizes for testing, and used random values to set the names, descriptions, URLs, zip codes, and locations of the companies. These files were then tested using the time utility on Cygwin, and the values are listed in Table 1. After analyzing the data, we have determined that lists containing 10, 100, 1000, and 10000 all can be properly printed under the verbose tag with Cygwin. However, the 100000 and 1000000 item test files do not print all of the nodes in their lists, but the time utility gave us a result for when the program terminated. We discovered this error by noticing that the number of nodes in the list was not printed upon completion, and after searching through the actual text file we discovered that it only printed the first part of the list for each. Verbose mode times increased logarithmically, although this might not take into account the complete opening and printing of each file. Additionally, the merge sort switch worked faster than verbose mode, and when plotted appears to be linear. Insertion sort was expected to be the slowest of the files, appeared to increase even faster than verbose mode for the first few files that it successfully ran on, and would not complete on files containing 10000 or more nodes. This shows how slow insertion sort is compared to the others. The labeled plot made from the data in Table 1 is listed in Appendix B.

**Table 1. Linked List Sorting Algorithms and the Time utility**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Insertion** | **Merge** | **Verbose** |
| **10** | 0.006 | 0.003 | 0.003 |
| **100** | 0.034 | 0.027 | 0.027 |
| **1000** | 4.95 | 0.142 | 0.091 |
| **10000** | x | 2.147 | 3.034 |
| **100000** | x | 3.746 | 15.253 |
| **1000000** | x | 16.625 | 34.365 |

Appendix A:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <sys/types.h>

#include <unistd.h>

/\*

\*

\* Iovation

Real-time device reputation and fraud detection

http://www.iovation.com/

111 SW 5th Avenue

Suite 3200

Portland

OR

97204

45.522105

-122.676129

\*

\*/

void get\_random(char \*s, unsigned len);

int main(){

//array of values

int array[] = {10,100,1000,10000,100000,1000000};

//array of names of files to create

char \* arrays[] = {"10.txt","100.txt","1000.txt","10000.txt","100000.txt","1000000.txt"};

int i = 0;

int j = 0;

//file pointer

FILE \*file = NULL;

//loop to create the proper number of nodes in each file

for(i = 0; i < 6; i++){

//open file

file = fopen(arrays[i],"a+");

//seeds random generator

srand((unsigned int) time(0) + getpid());

//loop through the

for(j = 0; j < array[i];j++){

//company name

char st[100];

get\_random(st,10);

//put company name to file

fprintf(file,"\* %s \n",st);

//company descriptipon

char s[50];

get\_random(s, 40);

//put comapny description to file

fprintf(file,"%s \n",s);

//put website to file

fprintf(file,"http://%s.com \n",st);

//address to file

fprintf(file,"%s \n","111 SW 5th Avenue");

fprintf(file,"%s \n","Suite 3200");

fprintf(file,"%s \n","Portland");

fprintf(file,"%s \n","OR");

//zipcode

int zip = rand()%10000;

char \* string[100];

//zipcode to file

sprintf(string,"%d",zip);

fprintf(file,"%s \n",string);

//lat and lon

double lat = rand() % (90-0+1) + 0;

double lon = rand() % (180-0+1) + 0;

//lat and long to file

fprintf(file,"%f \n",lat);

fprintf(file,"%f \n",lon);

fprintf(file,"\n");

}

//close file when done

fclose(file);

}

return EXIT\_SUCCESS;

}

//function derived from: http://ubuntuforums.org/showthread.php?t=1979310

void get\_random(char \* string, unsigned length){

/\* ASCII characters 33 to 126 \*/

unsigned int num\_chars = length - 1;

unsigned int i;

for (i = 0; i < num\_chars; ++i)

{

string[i] = rand() % (126 - 33 + 1) + 33;

}

string[num\_chars] = '\0';

}

Appendix B: