

Name:

Sec:

06/27/2006

Lab 4 Serial and Binary Searches

A search is a process where we have a collection of items and we want to find out if 1) the item is in the collection, and if it is 2) the location of the item. For example, if we have a grocery list written by someone else for us to use to pick up some items at the grocery store, we might be passing the milk and we would want to search our grocery list to see if milk is on the list of items to buy. For a small grocery list, you might search the list by scanning from the top item on the page, down the page item by item until you find milk on the list (or not as the case may be). This is essentially a serial or sequential search, where we look at items one by one in order from the top of the page to the bottom.

However, imagine you are instead searching for a phone number in the telephone book. This is another type of search we all perform regularly. If you tried to perform you search for a persons name in the telephone book using a sequential search, you would be there all day, starting with the people who's last name begins with 'A', then the 'B's, etc., especially if you looking for Tom Zimmerman. For this type of a search we instead use a bit of knowledge about the phone book and the types of surnames people have. The important bit of knowledge is, that in this case, the collection of items (the people's names along with their phone numbers) is arranged in a special order. By ordering the names alphabetically by the person's last name, we know that Zimmerman, Tom will appear somewhere near the back of the phone book, while Applewhite, Susan will be near the front, and Smith, John more towards the middle.

The way that most people search for a name in a phone book is similar to what is known as a binary search. We open the phone book to some spot, and based on the names at that spot we will either search for the name further on towards the back of the phone book, if the name we are looking for appears alphabetically after the page we have opened the phone book too. Likewise, if the name we are searching for appears in the alphabet before the names on the current page, we will look further towards the front of the phone book. A Binary search is basically the same as this method, but of course since we need to write it for a computer to perform, we have to be very methodical. So we start by opening the phone book to exactly the middle, and either looking at the front $\frac{1}{2}$ or back $\frac{1}{2}$ respectively based on the name we are searching for. And we repeat this process of eliminating $\frac{1}{2}$ of the remaining names by splitting the unsearched portion of the items in $\frac{1}{2}$ and making a decision.

Objectives

In this lab you will use a serial search and binary search to search a list of English surnames. You will write the serial search implementation yourself, based on the books implementation. You should become familiar with how these searches work and are implemented using a programming language, and the relative speed and power of the searches.

Instructions

1. Copy the file called Search.c, along with the data file called sortednames.dat, from the class web site. Open up the file in your compiler/IDE and compile it. The Search.c source code implements the binary search. It also contains an implementation of a function to input the names from the sortednames.dat data file and put the names in an array of strings, called names. Make sure you can compile the Search.c file and that it reads in the data file and runs correctly. Try searching for various names. How many comparisons does the binary search take to typically search for a name? How many comparisons are needed for an unsuccessful search? Examine the code and make sure you understand both the binary search and the method used to read in the data from the input file.
2. Implement a version of a sequential search. Your sequentialSearch() function should take the same three parameters as the binarySearch() function, an array of strings that is to be searched, an integer that indicates the number of names in the names array, and a string which is the name to be searched for. Also, your sequentialSearch() function should return an integer, just like binarySearch(), which is the index where the searchName appears in the names array if it is found, or -1 if the name is not found in the array.
3. Add a call to your sequentialSearch() function right after the call to the binarySearch() function. If your sequentialSearch() is working correctly, it should give exactly the same answer as the binarySearch() (e.g. the index where the searchName is found, or -1 when not found).
4. Add some code so you can follow the comparisons that are happening in your sequentialSearch(). How many comparisons does it usually take to find a name using the sequentialSearch(). How does this compare to the binarySearch().
5. There are approximately 5,000 names in the sortednames.dat file. What do you think would happen to the number of comparisons that typically are needed to search for a name if there were 10,000 names instead? There is a difference between the sequentialSearch() and binarySearch() in this case, why?

Lab 4 Finished

You have now completed Lab 4. If your program compiles and runs correctly and you have successfully uploaded your source file to the eCollege online submission site, then you are done. At this point, you might want to reread the sections in chapter 8 on the string data type, and in section 9 on using character arrays for string processing if any of the concepts are still fuzzy to you.