Math 2013 - Introduction to Discrete Mathematics

Project #2 - 2005.11.14

Due Date - 2005.12.07

Interpolation Search

The binary search is a very fast method for searching a sorted array, remember it is $\mathcal{O}(lg(n))$. *Interpolation search* is a searching method which is sometimes faster than the binary search algorithm. For this type of search, two assumptions must be satisfied:

1. Each access must be very expensive compared to a typical instruction. For example, the array might be on a disk instead of memory, and each comparison requires dick access.

2. The data must not only be sorted, but must also be fairly uniformly distributed. For example a phone book is fairly uniformly distributed. If the array looks like $\{1, 2, 4, 8, 16, \ldots\}$, then the distribution is not uniform.

These assumptions on the array are quite restrictive, so you might never actually use an interpolation sort. The idea, however, is that we are willing to make an accurate guess where the item might be. Remember the binary search always uses the midpoint. It would be silly to search for Frank Zappa in the middle of the phone book. So instead of *mid*, we will use *next*.

As an example of what might be good, suppose the range contains 1000 items, the low item in the range is 1000 and the high item is 1000000 and we are searching for a value close to 12000. If the items are uniformly distributed, then one can expect to find a match somewhere near the 12th item. The applicable formula is:

$$next = low + \lceil \Omega \rceil,$$

where

$$\Omega = \frac{key - X_{low}}{X_{high} - X_{low}} \times (high - low - 1)\,.$$

Here *key* is the value to be searched for, and X_k is the k^{th} element of the array X.

The Program

Write a program which performs both the *binary search algorithm* and *interpolation search* on a sorted array and which counts the number of iterations it takes to complete each of the searches.

The Problems

Perform searches on uniformly distributed and ordered arrays of sizes 512, 1024 and 2048. Compare the two search algorithms.
Download the txt files here:
http://www.sosu.edu/faculty/kfrinkle/teaching/current/Math2013/projects/512uniform.txt
http://www.sosu.edu/faculty/kfrinkle/teaching/current/Math2013/projects/1024uniform.txt
http://www.sosu.edu/faculty/kfrinkle/teaching/current/Math2013/projects/2048uniform.txt

Perform searches on non-uniformly distributed and ordered arrays of sizes 512, 1024 and 2048. Compare the two search algorithms.
Download the txt file here:
http://www.sosu.edu/faculty/kfrinkle/teaching/current/Math2013/projects/512nonuniform.txt
http://www.sosu.edu/faculty/kfrinkle/teaching/current/Math2013/projects/1024nonuniform.txt
http://www.sosu.edu/faculty/kfrinkle/teaching/current/Math2013/projects/2048nonuniform.txt

A special thanks to my good buddy Mikey (http://mikey.netdojo.com) for writing a nice little piece of perl code to generate these .txt files! Click below to download the code!
http://www.sosu.edu/faculty/kfrinkle/teaching/current/Math2013/projects/rnd.txt