COSC 1436 – Intro to Problem Solving II

Lab 5 Search Experiment Questions

# Name: Erman Gurses

**March 22, 2012**

1. Using the data you generated with the program, fill in the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Array size 🡪** | 1,000 Elements | 2, 500  Elements | 5,000  Elements | 10,000  Elements |
| Linear Search – Average # of comparisons | 527 | 1274 | 2570 | 5138 |
| Linear Search – Maximum # of comparisons | 1000 | 2500 | 5000 | 10000 |
| Binary Search – Maximum # of comparisons | 10 | 12 | 13 | 14 |

1. Does the Big O efficiency of the linear search match the efficiency discussed in lecture? Does it matter if the data are in order? Justify your answer.

Yes it matches because for linear search O(f(n))=n. It does not matter the data are ordered it still has O(f(n)) = n because if your data are ordered and you can check target element whether or not is bigger than in array’s elements. If it is bigger than array’s element, you can terminate while loop and the function will be f(n)= n - x. (x represents remaining elements of the array which have not been checked yet ). However according to definition of big oh we have to get rid of lower variables in function so it is still O(f(n)) = n.

1. According to our discussions, the big-O efficiency of the binary search should be O(*f*(*n*)) = O(log2*n*). Do your data support that? Justify your answer.

Absolutely, log21000=10 for 1000 elements

log22500=12 for 2500 elements

log25000=13 for 5000 elements

log21000=14 for 10000 elements

because 1024 = 210

2048 = 211

4096 = 212

8192 = 213

16348 = 214

1. Based on your data, which would you say is the most efficient search? Justify your answer.

Sure, binary search is better than linear search for this data because it is already sorted. For instance, if size of array (n) is 1000, linear search’s efficiency is n because it works as a linear function. On the other hand, binary search same size of array which is 1000 efficiency is log21000 which is approximately 10 because it works as a logarithmic function.

1. Under what situation(s) would a less-efficient search might be the search-of-choice? Does the size of the list and/or the "stability" of the data have any impact on which search might be the better choice? Justify your answer.

If data is unordered, I should choose linear search because binary search requires ordered list. So binary search has additional cost.

If data is ordered, I should choose binary search because binary search’s

Efficiency is O(log2n)

Stability affects my decision because if the list not stable I have to sort the list again and again for doing binary search so linear search is better choice in this situation If it is stable I can easily implement binary search.

Size also affects too. If the list is ordered I should choose binary search which has efficiency O(log2n)