**Lab Report 1**

**CS303-2D**

**Sam Lazrak**

**Spring Semester 2018**

**Objectives:**

* Review programming in a high-level language
* Implement linear search and binary search algorithms
* Evaluate performance of linear and binary search

**In-class Assignment:**

1. Implement a method that will search a given array using the linear search algorithm.

2. Implement a method that will search a given array using a recursive binary search algorithm.

3. Write a driver program to test the methods implemented in questions 1-2. Note that you have sort the input array before using the binary search algorithm, you can use any sort method available (e.g., sort method in the Java Collection Framework).

4. Have worked reviewed by TA. My work was reviewed on Jan 10th, 2018 in lab.

Source Code:

/\*\* \*

\* @author Sam Lazrak

\* @date 21JAN2018

\* This is a program built to implement linear and binary searches.

\* Ideas from Introduction to Algorithms By Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein

\*/

import java.util.Arrays;

public class Main {

   public static void main(String[] args) {

        int[] data = {1,3,2,4,5,6,7,8,9};

       int low = 0;

       int high = data.length-1;

       Arrays.sort(data);

        System.out.println("The linear search is "+ this.linearSearch(data, 9));

       System.out.println("The binary search is "+ this.binarySearch(data, 1, low, high));

   }

   public int linearSearch(int[] array, int num){

       for (int i=0; i < array.length; i++){

           if(array[i]==num) {

               return num;

           }

       }

       return -1;

   }

   public int binarySearch(int[] array, int num, int low, int high){

       int mid = (high - low)/2;

       if(low > high){

           return -1;

       }

       if(num < array[mid]){

           return binarySearch(array, num, low, mid);

       }

       else if(num > array[mid]){

           return binarySearch(array, num, mid+1, high);

       }

       else return array[mid];

   }

}

**Output:**

*Linear search: 9*

*Binary search: 1*

**Homework Assignment:**

1. Test the program for array sizes N = 16, 32, 64, 128, 256, 512, 1024, 2048, ......, 225. Initialize the array with random numbers between the ranges 1 through N and use the same array for testing linear search and binary search. Remember to sort the array before using binary search. Use a text file with 1,000 random numbers in the range 1 through 225 as the search keys.
2. Compare the execution time for linear search and binary search. Include the time taken for sorting with the binary search time (you have to sort only once for each array size). Use a table or plot to summarize the results and document your observations and explanations in the report.

Source code:

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

import java.io.UnsupportedEncodingException;

import java.util.Arrays;

/\*\* \*

\* @author Sam Lazrak

\* @date 21JAN2018

\* This is a program built to implement linear and binary searches and test the time of each.

\* Ideas from Introduction to Algorithms By Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein

\*/

public class Main{

public static void main(String[] args) throws FileNotFoundException, UnsupportedEncodingException {

 for (int i = 1; i <= 25; i++) {

 //creates randomly sized arrays

  int size = (int) Math.pow(2, i);

  int[] array = new int[size];

  int randomNum = (int)Math.ceil(Math.random() \* size);

  fillArray(array);

  //search keys

  PrintWriter writer = new PrintWriter("data.txt", "UTF-8");

  //linear search & time

  long linear = System.nanoTime();

  linearSearch(array, randomNum);

  linear = System.nanoTime() - linear;

  long binary = System.nanoTime();

  Arrays.sort(array);

  long binarySorted = System.nanoTime();

  binarySearch(array, randomNum);

  binary = System.nanoTime() - binary;

  binarySorted = System.nanoTime() - binarySorted;

  writer.printf("%d, %d, %d, %d\n", size, linear, binary, binarySorted);

  writer.close();

  System.out.println("Linear search");

  System.out.println(size);

  System.out.println(linear + "\n");

  System.out.println("Binary search");

  System.out.println(size);

  System.out.println(binary + "\n");

  System.out.println("Binary search after sorting array");

  System.out.println(size);

  System.out.println(binarySorted + "\n");

 }

}

private static int binarySearch(int[] array, int randomNum) {

  int mid = array.length/2;

  int firstHalf = array.length - array[mid];

  if(1 > array.length) {

  System.out.println("Invalid array.");

  }

  if(randomNum < array[mid]) {

  for(int l = 0; l < firstHalf; l++) {

  if(array[l] == array[randomNum]) {

  return 1;

  }

  }

  }else if(randomNum > array[mid]){

  for(int g = firstHalf; g < array.length; g++) {

  if(g == randomNum) {

  return 1;

  }

  }

  }

return -1;

}

private static int linearSearch(int[] array, int randomNum) {

  for(int p = 0; p < array.length; p++) {

  if(array[p] == array[randomNum]) {

  return 1;

  }

  }

return -1;

}

private static void fillArray(int[] array) {

  for(int n = 0; n < array.length; n++) {

  int randomint = (int)(31 \* Math.random() + 11);

  array[n] = randomint;

  }

}

}

**Analysis of output:** Linear search seems to be faster than a plain binary search because of the sorting time. However, a binary search after the array has been sorted is always the fastest out of both. For the fastest searches presort the array at regular intervals to have it ready for a search.

**The output formatted into a table below:**

|  |  |  |  |
| --- | --- | --- | --- |
| Search Type: | Array Size: | Running Time: |  |
| Linear search | 2 | 6168 |  |
| Binary search | 2 | 145417 |  |
| Sorted Binary Search | 2 | 1781 |  |
| Linear search | 4 | 686 |  |
| Binary search | 4 | 2231 |  |
| Sorted Binary Search | 4 | 513 |  |
| Linear search | 8 | 668 |  |
| Binary search | 8 | 2111 |  |
| Sorted Binary Search | 8 | 434 |  |
| Linear search | 16 | 668 |  |
| Binary search | 16 | 3459 |  |
| Sorted Binary Search | 16 | 495 |  |
| Linear search | 32 | 956 |  |
| Binary search | 32 | 6704 |  |
| Sorted Binary Search | 32 | 829 |  |
| Linear search | 64 | 998 |  |
| Binary search | 64 | 16154 |  |
| Sorted Binary Search | 64 | 677 |  |
| Linear search | 128 | 880 |  |
| Binary search | 128 | 33526 |  |
| Sorted Binary Search | 128 | 834 |  |
| Linear search | 256 | 1171 |  |
| Binary search | 256 | 48923 |  |
| Sorted Binary Search | 256 | 877 |  |
| Linear search | 512 | 946 |  |
| Binary search | 512 | 81679 |  |
| Sorted Binary Search | 512 | 858 |  |
| Linear search | 1024 | 785 |  |
| Binary search | 1024 | 120469 |  |
| Sorted Binary Search | 1024 | 869 |  |
| Linear search | 2048 | 662 |  |
| Binary search | 2048 | 230839 |  |
| Sorted Binary Search | 2048 | 716 |  |
| Linear search | 4096 | 1597 |  |
| Binary search | 4096 | 435852 |  |
| Sorted Binary Search | 4096 | 1024 |  |
| Linear search | 8192 | 855 |  |
| Binary search | 8192 | 546899 |  |
| Sorted Binary Search | 8192 | 2037 |  |
| Linear search | 16384 | 1129 |  |
| Binary search | 16384 | 737016 |  |
| Sorted Binary Search | 16384 | 2212 |  |
| Linear search | 32768 | 864 |  |
| Binary search | 32768 | 889481 |  |
| Sorted Binary Search | 32768 | 1276 |  |
| Linear search | 65536 | 1866 |  |
| Binary search | 65536 | 2511789 |  |
| Sorted Binary Search | 65536 | 4494 |  |
| Linear search | 131072 | 1100 |  |
| Binary search | 131072 | 4149775 |  |
| Sorted Binary Search | 131072 | 4129 |  |
| Linear search | 262144 | 1772 |  |
| Binary search | 262144 | 7232558 |  |
| Sorted Binary Search | 262144 | 4335 |  |
| Linear search | 524288 | 2051 |  |
| Binary search | 524288 | 16862859 |  |
| Sorted Binary Search | 524288 | 4690 |  |
| Linear search | 1048576 | 867 |  |
| Binary search | 1048576 | 24393251 |  |
| Sorted Binary Search | 1048576 | 4425 |  |
| Linear search | 2097152 | 1484 |  |
| Binary search | 2097152 | 41754530 |  |
| Sorted Binary Search | 2097152 | 4361 |  |
| Linear search | 4194304 | 1000 |  |
| Binary search | 4194304 | 85651635 |  |
| Sorted Binary Search | 4194304 | 5491 |  |
| Linear search | 8388608 | 2426 |  |
| Binary search | 8388608 | 185750313 |  |
| Sorted Binary Search | 8388608 | 5503 |  |
| Linear search | 16777216 | 1123 |  |
| Binary search | 16777216 | 337113716 |  |
| Sorted Binary Search | 16777216 | 6430 |  |
| Linear search | 33554432 | 2561 |  |
| Binary search | 33554432 | 709361825 |  |
| Sorted Binary Search | 33554432 | 6204 |  |