Jackson State University
Department of Computer Science
CSC 323 Algorithm Design and Analysis
Spring 2015
Instructor: Dr. Natarajan Meghanathan
Programming Project 2
Comparing the Performance of a Recursive vs. a Non-Recursive Algorithm to Determine the Index of the Element with the Largest Value in an Array

Due: February 19, 2015: 1 PM
Maximum Points: 100

Maximum Possible value of the elements in your arrays: 10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8

Upload your video through Google Drive/Dropbox and share with my email address; Submit the report via e-mail to **natarajan.meghanathan@jsums.edu,** See the submission section for more instructions

Important: Along with the assignment, each student should submit a cover sheet that has the following details: Student name, Semester, Assignment number, Date and time of submission of the assignment, Number of pages of the assignment excluding the cover sheet and Student signature. Assignments submitted without the cover sheet having all the above required information will not be graded.

Project Objective: In this programming project, you would implement recursive (a divide-and-conquer approach) and non-recursive (a brute-force approach) algorithms to determine the maximum element in an array and return its index value. If an array has more than one occurrence of the maximum element, then your program would return the index value corresponding to the left most occurrence of the maximum element. You will compare the running time of the recursive and non-recursive implementations for different values of the input array size.

Algorithms to Implement

Non-Recursive Algorithm

```
Algorithm MaxIndex(A, 0, n-1)

    maximumIndex = 0

    for i ← 1 to n-1 do

        if (A[maximumIndex] < A[i]) then
            maximumIndex = i
        end if

    end for

  return maximumIndex
```

Recursive Algorithm

```
Call Algorithm MaxIndex(A, 0, n − 1) where

Algorithm MaxIndex(A, l, r)
//Input: A portion of array A[0..n − 1] between indices l and r (l ≤ r)
//Output: The index of the largest element in A[l..r]
if l = r return l
else temp1 ← MaxIndex(A, l, ⌊(l + r)/2⌋)
     temp2 ← MaxIndex(A, ⌊(l + r)/2⌋ + 1, r)
     if A[temp1] ≥ A[temp2]
          return temp1
     else return temp2
```

You will be implementing both the non-recursive and recursive algorithms given above and determine running-time to return the index of the maximum element in the array; in case of a tie, the algorithms return the index of the left most occurrence of the maximum element.

You will determine the run-time for the following sizes using each of the two algorithms: 10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8

For each of the above array size, you will randomly generate that many elements in the array in a main method in your Java programs and call the recursive and non-recursive implementations as a static method. You can have one Java program with both the recursive and non-recursive algorithms implemented as separate static methods or two Java programs, one for the recursive algorithm and the other for the non-recursive algorithm.

The Java method System.nanoTime( ) returns the number of nano seconds (of long data type) passed since Jan1, 1970. Use this before calling and after returning from your recursive and non-recursive implementations to determine the run-time.

Results

You will plot an X-Y graph with the X-axis being the *logarithm* of the array input sizes (i.e., 1, 2, 3, …., 8) and the Y-axis being the running time of your algorithm in *logarithm* of the nano seconds time value time value you observe from the output of your programs. If your nano seconds time values are too larger to work with, convert them to milli seconds by dividing by 10^6.

What to submit:

(1) **A Desktop recorded video** (displaying your code, in full or in parts) with your explanation on the different sections of the code (starting from the execution in the main function) and walk through the code explaining how it will be executed starting from the input phase to the output phase. Do this for both the iterative (non-recursive) and recursive versions.
You should show the execution of the program for a sample input of the array size.
You should also display the magnitude of the actual run-time values obtained for both the iterative and recursive versions that are plotted in the results graph (with respect to the time axis) and **interpret** them.

Note that even though I am not specifying a minimum time for your video, your video explanation is expected to last at least for 8-10 minutes and should cover all of the above required explanations.
Note that the contents of the desktop/programs captured through your video should be clearly readable.

Submission of the Video: Upload the video to your Google Drive or Dropbox account and share it to natarajan.meghanathan@jsums.edu. In the notification e-mail that you send me, clearly label the subject with: Course #, Project # and short title and Semester #.

You could try using one of the **desktop recording software** (or anything of your choice):
CamStudio: http://sourceforge.net/projects/camstudio/files/legacy/
Debut: http://www.nchsoftware.com/capture/index.html

(2) A **report** (email to me) comprising of the following:

- An abstract – summarizing the results of the recursive and non-recursive algorithms
- Analysis of the theoretical worst-case run-time complexity of the recursive and non-recursive algorithms
- Your Java code for both the iterative (non-recursive) and recursive algorithms
- The result graph as described above
- Comparison the magnitude of the actual run-time values obtained for your recursive and non-recursive implementations vis-à-vis the theoretical run-time complexity for the two algorithms.

See next two pages for some start-up code

The following two Java programs generate arrays for the array size, entered as input by the user (through command-line) and call the recursive or non-recursive implementations, as appropriate. The running-time is calculated in the main method and printed. You could use these programs and implement the recursive and non-recursive algorithms and run them for each of the array sizes given.

```
import java.util.*;

// This Java program is for the recursive implementation

class maxElementRec{


   public static int MaxIndex(int[] array, int leftIndex, int rightIndex){

      // Implement the recursive algorithm here and return the maximum index

   }


   public static void main(String[] args){


   try{

      Random rand = new Random(System.currentTimeMillis());

      int arraySize = Integer.parseInt(args[0]);

      int array[] = new int[arraySize];

      for (int index = 0; index < arraySize; index++){
           array[index] = rand.nextInt();  // generates an integer from 0 to (2^32)-1
      }


      long startTime = System.nanoTime( );

      int maximumIndex = MaxIndex(array, 0, arraySize-1);

      long endTime = System.nanoTime( );

      System.out.println("maximum index is "+maximumIndex);
      System.out.println("maximum element value is "+array[maximumIndex]);
      System.out.println("running time in nano seconds is "+(endTime-startTime));
      System.out.println("running time in milli seconds is "+((double)(endTime-startTime)/1000000));

   }

  catch(Exception e){e.printStackTrace();}
  }

}
```

```
import java.util.*;

// This Java program is for the non-recursive (iterative) implementation

import java.util.*;

class maxElementNonRec{


    public static int MaxIndex(int[] array, int leftIndex, int rightIndex){

          int maximumIndex = leftIndex;

   // Complete your implementation of the non-recursive algorithm and return the maximumIndex value

     }


    public static void main(String[] args){


    try{

       Random rand = new Random(System.currentTimeMillis());

       int arraySize = Integer.parseInt(args[0]);

       int array[] = new int[arraySize];

       for (int index = 0; index < arraySize; index++){
            array[index] = rand.nextInt();  // generates an integer from 0 to (2^32)-1
       }


       long startTime = System.nanoTime( );

       int maximumIndex = MaxIndex(array, 0, arraySize-1);

       long endTime = System.nanoTime( );

       System.out.println("maximum index is "+maximumIndex);
       System.out.println("maximum element value is "+array[maximumIndex]);
       System.out.println("running time in nano seconds is "+(endTime-startTime));
       System.out.println("running time in milli seconds is "+((double)(endTime-startTime)/1000000));


     }
   catch(Exception e){e.printStackTrace();}

  }

}
```