COP 3503 - Programming Assignment #2
Mathematical and Empirical Analysis
Assigned: Jan 28, 2014 (Tuesday)
Due: Feb 9, 2014 (Sunday) at 11:55 PM WebCourses time

Objective
Implement three algorithms of differing time complexities for computing the maximum contiguous subsequence sum (MCSS) of an array of numbers. Prove the Big Theta Θ time complexity of each algorithm mathematically and validate the result empirically.

Problem: Maximum Contiguous Subsequence Sum (MCSS)
Given a sequence of numbers, find the sum of a contiguous subsequence with sum no smaller than any other contiguous subsequence (there can be more than one contiguous subsequence yielding the maximum sum). For example, given the sequence (-2, 11, -4, 13, -5, -2), a maximum contiguous subsequence is (11, -4, 13) because its sum, 20, is no smaller than the sum of any other contiguous subsequence; i.e., 20 is the maximum sum of any contiguous subsequence. Therefore, the MCSS of the sequence (-2, 11, -4, 13, -5, -2) is 20. The MCSS of a sequence containing only negative numbers is 0, which is the sum of an empty subsequence, because any non-empty subsequence would yield a negative sum.

Part 1 – Implement and time the algorithms

The three algorithms of Θ(n), Θ(n^2) and Θ(n^3) time complexities, respectively, for computing MCSS have already been implemented and are available for your use; the algorithms are in the file "EmpiricalAnalysis.java" which is in the zip file on the course website located in "Modules", "Recitation 2 – MCSS and Empirical Analysis.zip." This Java file also contains code for generating a random array of length N and timing how long each algorithm takes in milliseconds (averaged over multiple runs).

In this part, you will write a program in the file **mcss.java** that accepts sequences from an input file; for each sequence, the program runs the three algorithms, prints the MCSS and prints the number of nanoseconds each algorithm takes averaged over at least 100 runs.

Input:
The input file is **mcss.txt**. The first line is the number of sequences in the file, ***M***. The following *M* lines specify the *M* sequences, one per line. Each sequence on a line is preceded by the number of integers in that sequence.

Output:
The output is to **System.out**. The output will contain *M* lines. The first line corresponds to the first inputted sequence, the second line corresponds to the second inputted sequence, etc. A line in the output begins with the MCSS of the inputted sequence corresponding to that line. The next three integers of the outputted line are the number of nanoseconds the Θ(n^3), the Θ(n^2) and the Θ(n) algorithm took, respectively, to compute the MCSS.

Sample Input:
`2`
`6 -2 11 -4 13 -5 -2`
`9 -2 1 -3 4 -1 2 1 -5 4`
	Comments on the Sample Input:
	2 sequences in the file
	The first sequence contains 6 integers and is -2 11 -4 13 -5 -2
	The second sequence contains 9 integers and is -2 1 -3 4 -1 2 1 -5 4

Sample Output:
`20 3040 1211 285`
`6 7270 2937 407`
	Comments on the Sample Output:
	The MSS, 20, of the first sequence is followed by the timings – times will vary
	The MSS, 6, of the second sequence is followed by the timings – times will vary

Part 2 – Write-up of mathematical and empirical analysis of the algorithms

Prepare a write-up in the file **mcss.pdf**[1] containing two sections, **Mathematical Analysis** and **Empirical Analysis**.

1. The first section, **Mathematical Analysis**, will contain three formal proofs that each of the three algorithms have time complexities Θ(n^3), Θ(n^2) and Θ(n), respectively. The proofs will follow the format specified by the "General Plan for Analyzing the Time Efficiency of Nonrecursive Algorithms" in section 2.3 of the book. In particular, for each of the three algorithms, first derive a summation that counts the number of basic operations of the algorithm and then determine the order of growth of the summation in terms of the Big Theta Θ notation.

2. The second section, **Empirical Analysis**, will contain three tables, one for each of the three algorithms. The first column of a table is **N**, the input size. The second column is **Time**, the number of nanoseconds taken by the algorithm with input of size N averaged over multiple runs. The third, fourth and fifth columns are **Time/N**, **Time/N^2** and **Time/N^3**, respectively, which are defined in terms of the first two columns. Use these tables to argue that the time complexities of the three algorithms are Θ(n^3), Θ(n^2) and Θ(n), respectively. The file "runtime-analysis.txt" in the zip file "Recitation 2 – MCSS and Empirical Analysis.zip" under "Modules" on the course website shows how to construct such an argument.

   Run each algorithm at least 10 times and compute the average running time for each one. The following input sizes should be used:

   Θ(N^3) Algorithm: N = 125, 250, 500, 1000, 2000.
   Θ(N^2) Algorithm: N = 25000, 50000, 100000, 200000, 400000, 800000.
   Θ(N) Algorithm: N = 100000, 200000, 400000, 800000, 1600000, 3200000, 6400000.

   You may use the code from "EmpiricalAnalysis.java" in the zip file to generate arrays of the specified sizes containing random integers. It is important not to include the generation of the input array in the timing of the algorithms.

Deliverables

You must submit **mcss.java** and **mcss.pdf** to WebCourses by 11:55 PM on Sunday, February 9, 2014. You must send your source files as an attachment using the "Add Attachments" button. Assignments that are typed into the submission box will not be accepted. Assignments that are 1 day late are deducted 25% of the points received. Assignments more than 1 day late are not accepted.

Restrictions

Your program must compile using Java 7.0 or later. It's okay to develop your program using the IDE of your choice, although Eclipse is recommended. Your program and write-up should include a header comment with the following information: your name, course number, section number, assignment title, and date. Conform exactly to the I/O specifications (timings in the output will vary). Programs that do not compile will get zero points.

Grading Details

Part 1 and Part 2 of the assignment are worth 50% of the grade each.

1 Instructions for saving as PDF in Word can be found at http://office.microsoft.com/en-us/word-help/save-as-pdf-HA010064992.aspx. If you can't convert .doc to .pdf, there are several online conversion tools, such as http://convertonlinefree.com/.