ODL127 Algorithms and Data Structures 27 February 2006

Exercise 6: The Maximum Contiguous Subsequence Sum Problem

This is intended to get you to think in terms of algorithms. You need to think first about an algorithm to solve this problem. Once you have thought of an algorithm, then you can write a Java method to implement it. You should not expect just to be able to sit at the computer and type in code for it without first having thought it through carefully and having a plan for an algorithm that will solve the problem.

Although this is a fairly tough problem, it uses only the sort of expertise you ought to have with Java now, after your first programming course and further experience with algorithms in this course. It is an essential part of the Algorithms course that you learn by doing exercises like this, and they are set as learning exercises.

If you are still having problems with very basic definition and use of arrays, and definition and use of methods, you are falling behind and need to catch up. Make sure you do so before tackling this exercise.

Here is the problem:

Given a sequence of integers, A_1, A_2, ... , A_n, each of which may be positive or negative, find the subsequence of numbers A_i, ..., A_j which has the maximum sum of any subsequence. If all the integers are negative, return 0.

A subsequence must include all the numbers between the start and finish point. So, for example if the sequence is `{10,-20,11,-4,13,3,-5,-17,2,15,1,-7,8}` then the answer is `23` since this is the sum of the subsequence `{11,-4,13,3}` and no other subsequence has a higher sum. The answer is always at least `0`, because the empty sequence is always a possible subsequence, and its sum is `0`.

Assume the numbers are stored in a Java array `a` so `a[k-1]` stores A_k for any `k` between and including `1` and `n`. Then write a Java method which returns the maximum contiguous subsequence sum.

A program which provides a frame to test the method can be found in the directory:

`http://www.dcs.qmul.ac.uk/~mmh/DCS128/notes/code/exercises/`

in the files `MaxContSubsSum0.java` and `MaxContSubsSum1.java`.

All you have to do is fill out the body of the method `maxSubsequenceSum`. This method takes an array storing some integers as its parameter and returns the maximum contiguous subsequence sum of that array. At least, that's what it should return once you have written the code to go in it.

In `MaxContSubsSum0.java` you type in your own numbers for the sequence, but in `MaxContSubsSum1.java`, a sequence of numbers is generated at random for you.

In `MaxContSubsSum1.java`, you are asked to enter a “seed”, the length of the sequence, and the highest integer value, *max*. Integers are generated at random, both positive and negative from the range *–max* to *max*. The program takes the time before and after the method that gives the algorithm is executed. This gives a rough estimate of the timing (not an exact figure, because typically your computer will spend some of its time doing other things, e.g. maintaining the clock display, when it's obeying your code).

The “seed” is just a way of ensuring that when you run your code, a particular sequence of numbers will be generated, so if you change your code and run it again, you can make sure it’s using the same “random” numbers as it did before. Apart from that, the numbers will not appear to have any pattern.

A poor algorithm will take a long time (time for you to sit and wait for it to terminate) to find the maximum contiguous subsequence of even 1000 integers. A good algorithm will take a short time (it will appear to be instantaneous) for 1000 integers, and you won't have to wait too long for it to deal with 1000000 integers.

For an example solution, if you type 1234 for the seed, 500 for the length of the sequence, and 100 for the highest integer, the answer you should get is 1212 (which is the sum of the range from the 317th to the 477th integer inclusive).

One possible use of this algorithm is in stock market analysis. Suppose the integers represent the rise or fall of a share price over a period of time. Then the best period to own that share is the period represented by the maximum contiguous subsequence sum.

Matthew Huntbach