School of Computer Science, University of Windsor

60-141: Introduction to Algorithms and Programming II
Term: Summer 2014 (July-August)
Instructor: Dr. Asish Mukhopadhyay

Assignment 2
Posted: 11th July, 2014
Due: 18 July, 2014, 11:59pm

Preamble: The rationale for this assignment is to help you understand better the utility of the array data structure as well as sharpen you problem-solving skills.

Grading Scheme: The problem is worth 10 points, 4 for the first part and 6 for the second. Mail your solution to parti@uwindsor.ca before the due date. The submission should have your name and student ID on it.

Credits: The idea of this assignment is original and mine.

Problem: Given a sequence of integers

$$a_1, a_2, a_3, \ldots, a_{n-1}, a_n,$$

a *contiguous* subsequence of the above sequence is:

$$a_i, a_{i+1}, \ldots, a_{j-1}, a_j,$$

where $1 \leq i \leq j \leq n$. Now on, by a subsequence we shall mean a contiguous subsequence.

The *maximum subsequence problem* is to determine a subsequence such that the sum

$$a_i + a_{i+1} + \ldots + a_{j-1} + a_j$$

is a *maximum*.

Thus for the sequence

-1, 10, -3, -4, 20, -6,

a maximum common subsequence is:

10, -3, -4, 20

A brute-force algorithm is to find the sum of a subsequence from index i tom index j for all possible index pairs $i \leq j$. The pseudo-code for this brute-force algorithm is given below:

Algorithm *MaxSeqBruteForce*

Input: A sequence of integers $a_1, \ldots, a_{n-1}, a_n$
Output: Indices *start* and *end* of a maximum subsequence and its sum *maxSum*

Step 1. Set $maxSum \leftarrow 0$
$start \leftarrow 0$
$end \leftarrow -1$
$i \leftarrow 1$

Step 2. If $i > n$, go to Step 9

Step 3. Set $sum \leftarrow 0$
$j \leftarrow i$

Step 4. If $j > n$, go to Step 8

Step 5. $sum \leftarrow sum + a_j$

Step 6. If $(sum > maxSum)$
$maxSum \leftarrow sum$
if $(start < i)$ $start \leftarrow i$
$end \leftarrow j$

Step 7. $j \leftarrow j + 1$ and go to Step 4

Step 8. $i \leftarrow i + 1$, go to Step 2

Step 9. Print $start, end, maxSum$ and STOP

Implement a C-program for this algorithm (4 points).

A cleverer algorithm that finds a maximum sum subsequence by a left-to-right scan exploits the fact that a maximum subsequence cannot have a prefix (a *prefix* is a set of consecutive terms beginning with the first) that sums to a negative value. Here is the pseudo code for such an algorithm.

Algorithm *MaxSeqImproved*

Input: A sequence of integers $a_1, \ldots, a_{n-1}, a_n$
Output: Indices *start* and *end* of a maximum subsequence and its sum *maxSum*

Step 1. Set $i \leftarrow 1$ // candidate start position
$j \leftarrow 1$ // sweeps the array
$maxSum \leftarrow 0$ // maximum sum
$sum \leftarrow 0$ // current sum
$start \leftarrow end \leftarrow -1$ // start and end positions of current maximum subsequence

Step 2. If $j > n$, go to Step 7

Step 3. Set $sum \leftarrow sum + a_j$

Step 4. If $(sum > maxSum)$
$maxSum \leftarrow sum$
$start \leftarrow i$
$end \leftarrow j$
go to Step 6

Step 5. If $(sum < 0)$
$i \leftarrow j + 1$ //reset i
$sum \leftarrow 0$

Step 6. $j \leftarrow j + 1$ and go to Step 2

Step 7. Print $start, end, maxSum$ and STOP

Implement a C-program for the above algorithm (6 points).