Assignment 9

**Please read turn-in checklist at the end of this document before you start doing exercises.**

**Problems To Hand In (Java Implementation)**:

1. Consider the following problem.

Input: A list of numbers, {A[0], A[1], A[2], . . . , A[n-1]}.

Output: The contiguous subsequence of maximum sum (a subsequence of length zero has sum zero).

A contiguous subsequence of a list A is a subsequence made up of consecutive elements of A. For instance, if A is {5, 15, −30, 10, −5, 40, 10}, then {15, −30, 10} is a contiguous subsequence but {5, 15, 40} is not. For the preceding example, the answer would be {10, −5, 40, 10}, with a sum of 55.

A dynamic programming solution is given below.

We focus on the (i+1)\_th element A[i].

Consider every possible contiguous subsequence ending with the element A[i].

You’ll notice that these sequences can be divided into two categories, the subsequence ending with (A[i-1], A[i]), and the subsequence that contains a single element A[i].

We will call the maximum sum of contiguous subsequence ending exactly at A[i] Local\_maximum(i).

Let’s say somehow I know Local\_maximum(i-1). Then we see that to calculate Local\_maximum(i), we don’t need to compute the sum of all contiguous subsequence ending at A[i] since we already know the result from contiguous subsequences ending at A[i-1]. And this leads us to the principle on which dynamic programming algorithm works.

**Local\_maximum(i) = max(A[i], A[i]+Local\_maximum(i-1))**

This way, at every index i, the problem is finding the maximum of just two numbers, A[i] and (A[i] + Local\_maximum(i-1)). Everytime we solve a subproblem Local\_maximum(i), save the result, late we can re-use the result to solve the next subproblem Local\_maximum(i+1). Note that Local\_maximum(0) would be A[0] itself.

**Thus we can create a table, or say, an array to save all the intermediate results Local\_maximum(i) and re-use the results later. The final result is the maximum of Local\_maximum(i) over all possible i.**

Implement the dynamic programming maximum contiguous subsequence Algorithm in Java.

Note:

Find a file called ContiguousSumDP.java in assignment 9 folder.

Complete the method of MaximumSumContSubsequence ().

Test your method in the main method provided following the comments.

**Full credit (30 points) will be awarded for a dynamic programming implementation of Maximum Sum of Contiguous Subsequence. Programs that are NOT dynamic programming will be scored out of 10 points.**

**Problems NOT to Hand In (Practice Problems):**

The following problems are NOT collected and NOT graded. However, some of these problems will be used in quizzes, so it is in your best interest to do practice problems.

Solutions to practical problems will be discussed AFTER quizzes (To be fair with the class, the instructor will NOT answer questions about practice problems BEFORE quizzes. If asked, the instructor will show similar problems discussed during lectures).

1. Knapsack Dynamic Programming:

<https://courses.csail.mit.edu/6.006/fall11/rec/rec21_knapsack.pdf>

1. Coin Change Dynamic Programming:

<https://people.engr.tamu.edu/andreas-klappenecker/csce411-f11/csce411-set5.pdf>

**TURN-IN CHECKLIST:**

1. **All your source code\* of “Problems To Hand In”. Remember to include your name, the date, and the course number in comments near the beginning of your code.**

**\* The source code you submit must contain all the files necessary to run your program. These files will typically be***.java***files containing programs.**

1. **Create a folder and name it 'FirstName\_LastName\_assignment\_9'. In the newly created folder copy and paste your files (.java files). Then compress the folder, and push it to Brightspace.**