Introduction

# In this lab you will implement a divide and conquer algorithm for the maximum subarray problem. Submit your answers to the questions below in a text file (e.g. Word document). Name your file in name\_surname.docx format. Submit your solution document and Java codes as a zip/rar folder in name\_surname format to Canvas.

You can use the code templates in max\_subarray.java in this lab.

## Problem Statement

Given an input array 𝐴 of 𝑛 numbers find indices 𝑖 and 𝑗 such that 1 ≤ 𝑖 < 𝑗 ≤ 𝑛 and 𝐴[𝑗] − 𝐴[𝑖]

is maximum.

## Assignment

1. (a) Implement a method called brute\_force for solving the maximum subarray problem using brute-force approach. Your method should receive an array 𝐴 of 𝑛 integers as input and it should report (i.e. print on screen) two indices 𝑖, 𝑗 such that 1 ≤ 𝑖 < 𝑗 ≤ 𝑛 and 𝐴[𝑗] − 𝐴[𝑖] is maximum. Your method should also print the maximum possible 𝐴[𝑗] − 𝐴[𝑖] difference. Note that this notation assumes that the array indices start from 1. In this brute-force approach, you

should consider all pairs of 𝑖, 𝑗. The number of such pairs is equal to your procedure should run in Θ(𝑛2) time.

i = 3 j = 4 part 1(a)= max difference = 3

(𝑛) = 2

𝑛(𝑛−1). Therefore

2

1. Choose the following array 𝑡𝑒𝑠𝑡\_𝐴 as input to the method you implemented in part (a) and verify that your method prints 𝑖 = 3, 𝑗 = 4, and maximum difference as 3.

𝑡𝑒𝑠𝑡\_𝐴 = [10, 11, 7, 10, 6]

i = 50 j = 69 part 1(a)= max difference = 99

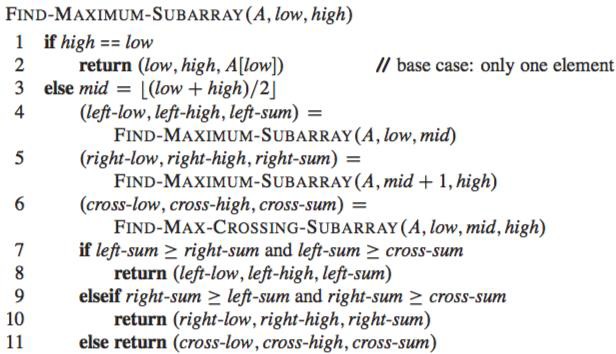
1. Choose an integer array 𝐴 of size 65536. Initialize your array by random numbers from 0 to

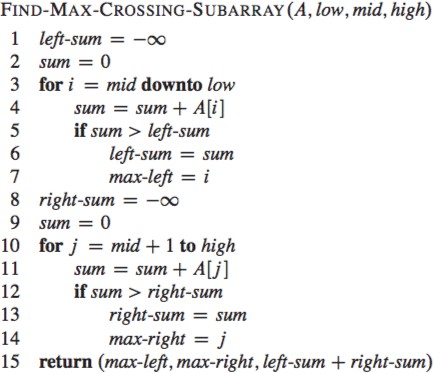
99. Call the method you implemented in part (a) and find the maximum subarray for this input array. Compute the time it takes to find the maximum subarray in nanoseconds and include this time into your report.

part 1(c)= Elapsed time : 828109500

1. (a) Implement the divide and conquer algorithm given below for finding the maximum subarray using recursion. You can use the code templates in max\_subarray.java. The

outputs of find\_max\_crossing\_subarray method can be retrived by the input array named outputs (since this method returns multiple outputs).





where A is an array of numbers corresponding to a difference array and the array indices start from 1 in these pseudo-codes.

1. Test your method using the array 𝑑𝑖𝑓𝑓\_𝑡𝑒𝑠𝑡\_𝐴 given below (which is the difference array for

𝑡𝑒𝑠𝑡\_𝐴 of part 1(b)) and verify that your method prints 3, 3, and 3 as the outputs (the first 2 are left and right indices and the last output is the maximum difference).

part 2(b)= Elapsed time : 18900

𝑑𝑖𝑓𝑓\_𝑡𝑒𝑠𝑡\_𝐴 = [1, −4, 3, −4]

1. Start with the same input array 𝐴 as in question 1 (c). Convert this array to difference array called 𝑑𝑖𝑓𝑓\_𝐴. A code template is already given in the main method of max\_subarray.java for this purpose. Find the maximum subarray of 𝐴 using the difference array called 𝑑𝑖𝑓𝑓\_𝐴 as input to the divide and conquer algorithm given above. Report the left and right indices generated by find\_maximum\_subarray method as well as the maximum difference. Then report the 𝑖, 𝑗 indices for the original input array 𝐴 at which 𝐴[𝑗] − 𝐴[𝑖] is maximum. Note that to find 𝑖 you should subtract 1 from the lower index returned by the find\_maximum\_subarray method (i.e. subtract 1 from the first output of this method). The second index returned by find\_maximum\_subarray method can be directly reported as index 𝑗. Also report the maximum difference returned by find\_maximum\_subarray method, which should be equal to the maximum 𝐴[𝑗] − 𝐴[𝑖] difference. Compute the time it takes to find the maximum subarray in nanoseconds and include this time into your report. Do you get better running time as compared to the brute-force approach?

2(c) Elapsed time : 2500

1. (Bonus) What problem size 𝑛0 gives the cross-over point at which the recursive algorithm beats the brute-force algorithm?
2. 27