CSci 242: Algorithms and Data Structures

Instructor: Dr. M. E. Kim Date: 1-24-2020

Due: 11:59 PM, January 31th (Fri.), 2020. (No Extension) Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Home Assignment 1: 100 points + 20 points (optional)**

Q1. [30] **Min-Max** recursive algorithm

1. [10] Write a ***recursive*** algorithm in pseudocode, **Min-Max**, for finding *both* the *minimum* element and the *maximum* element in an array A of *n* elements. Your algorithm calls itself ***only once*** within the algorithmand should *return* a pair (*a, b*) where *a* is the minimum element and *b* is the maximum element.

**Algorithm** **Min-Max**(A, *n*)

**Input:** an Array A of *n* elements

**Output:** a pair of (*a, b*) where *a* is the *minimum* element and *b* is the *maximum* element.

1. [20] Implement your algorithm of Q1.(1) in Python, returning a pair of the maximum and the minimum elements. Run your algorithm in the inputs [30, 50, 20, 70, 10, 80, 25, 100, 60, 40].

Q2. [25] For a given algorithm below,

**Algorithm** **Loop2**(*n*):

(0) *s* ← 0

(i) **for** *i* ← 1 **to** *n* **do**

(ii) **for** *j* ← *1* **to** 2*i* **do**

(iii) *s* ← s + *j*

1. (A) [5] Count the number of primitive operations in each statement, (0) – (iii), of the algorithm and (B) [5] get the total number of primitive operations executed in the algorithm.

Assume that the variables *i* and *j* are incremented after the statement (iii) automatically, ignoring their hidden increment statements. See the Handout 2.

1. [5] Give the smallest asymptotic upper bound of the running time in 1.B)in Big-Oh notation in terms of *n.* e.g.) *O(n), O(n2),* etc.
2. [10] ***Prove*** your answer in 2) by the ***definition of big-Oh***. i.e. You have to find the positive constant *c* and *n0* that satisfies the condition of the big-Oh definition. See the examples in the slides # 22 - #26 and Handout 3.

Q3. [10] Prove that *n2 log2 n*= Ω(*n2*) by the ***definition of big-Omega***.

Q4. [10] Prove that 2*n2 - 5n - 3* = Θ(*n2*) either by the definition of big-Theta or by using the limit to the fraction of the functions.

Q5. [25] **Maxsub** algorithm in the textbook is to find the subarray whose sum is the maximum. Similarly, you can modify it to **Minsub** algorithm to find the subarray whose sum is the minimum.

1. [10] By modify the **MaxsubFastest** algorithm, write the **MinsubFastest** so that it uses only *a single loop* and, instead of computing *n*+1 different *Mt* values, it maintains just a *single variable* Mfor *Mt*s.
2. [15] Modify the **MinsubFastest** algorithm so that it returns *both* the ***value of the minimum subarray*** ***summation*** and the indices ***j*** and ***k*** that identify the minimum subarray A[*j* : *k*], i.e. a triplet of ( ***value of the minimum subarray*** ***summation, i, j*** ). The running time of your algorithm should be O(*n*).
3. [20, optional] Implement your MinsubFastest(A) algorithm in 2) in Python, printing a triplet of three values.