Student Name: Jihyo Kim Set: AI

### The goals of this lab are to:

Give you the background skills that are required for you to apply the *General Plan for Analyzing Time Efficiency of Non-recursive Algorithms* (page 62 of your textbook). Ultimately this is what you need to be able to do.

Today's background skills include:

* ability to express an algorithm in pseudocode
* identification of the basic operation in an algorithm
* ability to set up summations to represent the number of times a basic operation is executed
* manipulation of summations (transform into a closed-form formula)

Note: this is an individual lab assignment in that you have to learn the material. Please feel free to discuss answers or to work through some questions with a partner if this helps you learn the material – but you must do your own work on this paper to turn in.

### Due date and marking:

* This lab is worth 15 marks.
* You should type your answers in this document, and submit this to the lab section on Learning Hub.
* Please type your answers in BLUE like this.

Consider the following algorithm from your textbook (page 23):

1. Algorithm CCS (A[0..n−1])

2. for i ¬ 0 to n−1 do

3. Count[i] ¬ 0

4. for i ¬ 0 to n-2 do

5. for j ¬ i+1 to n−1 do

6. if A[i] < A[j]

7. Count[j] ¬ Count[j]+1

8. else

9. Count[i] ¬ Count[i]+1

10. for i ¬ 0 to n−1 do

11. S[Count[i]] ¬ A[i]

12. return S

1. Consider only lines 1 to 3 of this algorithm. Call these 3 lines "Part A".

a. [1 mark] What does Part A of the algorithm do?

Assign a number, 0, to the each element in an array called Count

b. [1 mark] Assume that Part A is all there is to the algorithm. What is the basic operation in Part A, and on what line does it occur?

The basic operation in Part A is assignment and it occurs on the line number 3

c. [1 mark] Set up a summation that counts the number of times the basic operation is executed in Part A for an input array of size n and solve it. Note: Appendix A (pg 476) contains some very useful formulas to help you solve summations to closed form.

= n

n times

2. Consider only lines 4 to 9 of the algorithm from question 1. Call these 6 lines "Part B".

1. [1 mark] What does Part B of the algorithm do? Use the following array as test data, showing the contents of Count [] after each execution of the statement on line (4). A= [42, 17, 18, 23, 37, 9]

Count the number of smaller elements than the current element

i = 0 : Count = [5, 0, 0, 0, 0, 0]

i = 1 : Count = [5, 1, 0, 0, 0, 0]

i = 2 : Count = [5, 1, 2, 0, 0, 0]

i = 3 : Count = [5, 1, 2, 3, 0, 0]

i = 4 : Count = [5, 1, 2, 3, 4, 0]

i = 5 : Count = [5, 1, 2, 3, 4, 0]

b. [1 mark] What is the basic operation in Part B, and on what line does it occur?

Comparison. it occurs on line number 6

c. [1 mark] Set up a summation that counts the number of times the basic operation is executed in Part B for an input array of size n and solve your summation.

= = 25 - 10 = 15

3. Consider only lines 10 to 11 of the algorithm from question 1. Call these 2 lines "Part C".

1. [1 mark] What does Part C of the algorithm do? Use your answer from question 2a as input to Part C. Show the contents of S [] after each assignment on line 11.

Put elements in array S from array A at a specific index where its determined by values in array Count

[9, 17, 18, 23 , 37, 42]

b. [1 mark] What is the basic operation in Part C, and on what line does it occur?

Assignment on the line number 11

c. [1 mark] Set up a summation that counts the number of times the basic operation is executed in Part C for an input array of size n, and solve your summation.

= n

n times

4. [1 mark] Consider the entire CCS algorithm (including all lines in Parts A, B, and C). What is the basic operation for the entire algorithm? How many times is the basic operation executed for an input array of size *n*?

Comparison, its executed 15 times

5. Question 3 on page 51 of your textbook, which asks: *Consider a variation of sequential search that scans a list to return the number of occurrences of a given search key in the list.*

a. [3 marks] Write the pseudocode for this algorithm (count number of occurrences), using the same style of pseudocode shown in your textbook. Assume the list is implemented as an array.

GetNumOfOccurences(A[0..n-1], K)

count <- 0

for i <- 0 to n-1 do

if A[i] == K

count <- count + 1

return count

b. [1 mark] What is the basic operation in your algorithm?

Comparison

c. [1 mark] Set up a summation that counts the number of times the basic operation is executed for an array of size n, and solve your summation to closed form.

= n

n times