Student Name: Kunlaya Kobunnoi Set: 3D

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 pseudo-code
* 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 do need to have your own work on your own lab at the end of the class.

**Due Date and Grading:**

* **This is a graded lab out of 10.**
* **You should type your answers on this page, and be prepared to submit it to D2L- lab1 by the end of today’s lab.**
* **Please type your answers with Blue color.**
* **Late submissions will not be marked.**
* **Don’t email the labs, it will not be marked.**

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".

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

It initializes the starting value of count to zero.

1. [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?

Line 3 Count[i] ← 0

It initializes the starting value of count to zero.

1. [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.*

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]

It fills up the number for array and initialize i and j position and compare

1. [42,17,18,23,37,9]

* i is 42
* j is 17
* compare 42 and 17

1. [5,0,0,0,0,0]

* Now i is 17 and j is 18

1. [5,1,1,1,1,0]

* Now i is 18 and j is 23

1. [5,1,2,2,2,0]
2. [5,1,2,3,3,0]
3. [5,1,2,3,4,0]

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

A key comparison on line 6 A[i] < A[j]

1. [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.

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.

Fills S array up to each sorted array

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

Line 11. S[Count[i]] ← A[i]

Assigns the value to S array.

1. [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.

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 this basic operations executed for an input array of size *n*?

A key comparison from line 6 A[i] < A[j]