Student Name: Artur Gordiyenko

### 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 20.
* You should type your answers on this page, and submit it to D2L before due date.

Questions:

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?

Makes a new array of size n with values of 0.

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?

Line 3, setting of Count[i] to 0

c. [1 mark] Using common sense and your understanding of this loop, how many basic operations will be executed in Part A for an input array of with n=512 (ie: A[0..511])?

512

d. [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*.

Assume basic operation is noted as X

n-1

∑ x

i = 0

e. [1 mark] Solve your summation (from question 1d above) to closed form. *Note: Appendix A (pg 476) contains some very useful formulas to help you solve summations to closed form.*

(n-1-0+1)x = n(x)

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

a. [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 sets the order A should be in to be sorted. Ex A[0] which is 42, and should be last or 5th place.

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

Count = [0, 0, 0, 0, 0, 0]

Count = [5, 0, 0, 0, 0, 0]

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

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

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

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

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?

Line 6, it is the comparison.

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*.

d. [1 mark] Solve your summation to closed form.

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

a. [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.

Store the value of A[i] at the index of the value of Count[i]

S[0]=9

S[1]=17

S[2]=18

S[3]=23

S[4]=37

S[5]=42

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

S[Count[i]] <- A[i]

Line 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*.

Assume basic operation is noted as X

n-1

∑ x

i = 0

d. [1 mark] Solve your summation to closed form.

(n-1-0+1)x = n(x)

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*?

Line 7: The If statement.

It is executed times. Or approximately O)

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 mark] 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.

/\*Returns first key occurrence or -1 if nothing is found.

1. NumberOfOccurences (A[0..n−1], int searchKey)

2. for i ← 0 to n−1 do

3. if A[i] = searchKey

4. return i

5. return -1

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

Line 3: comparison

c. [2 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.

Assume X as basic operation

n-1

∑ x

i = 0

(n-1-0+1)x = n(x)

**Homework Question:** This question is optional (ie: not included in gade for the lab). It is here so that students who finish the earlier questions have something more challenging to work on.

**Your task:** Implement the java code for an algorithm to solve the following problem.

Problem: Building Height

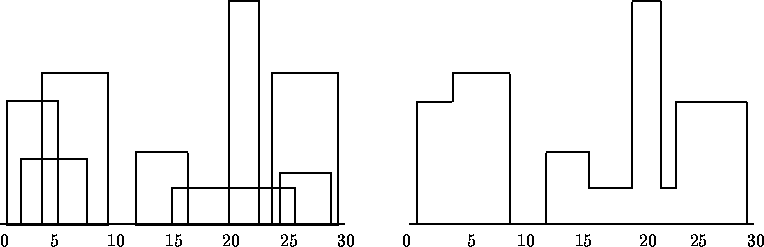
Background

With the advent of high speed graphics workstations, CAD (computer-aided design) and other areas (CAM, VLSI design) have made increasingly effective use of computers. One of the problems with drawing images is the elimination of hidden lines -- lines obscured by other parts of a drawing.

The Problem

You are to design a program to assist an architect in drawing the skyline of a city given the locations of the buildings in the city. To make the problem tractable, all buildings are rectangular in shape and they share a common bottom (the city they are built in is very flat). The city is also viewed as two-dimensional. A building is specified by an ordered triple (Li, Hi, Ri) where Li and Ri are left and right coordinates, respectively, of building i and Hi is the height of the building. In the diagram below buildings are shown on the left with triples (1,11,5), (2,6,7), (3,13,9),

(12,7,16), (14,3,25), (19,18,22), (23,13,29), (24,4,28). The skyline, shown on the right, is represented by the sequence: (1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)



**The Input**

The input is a sequence of building triples. All coordinates of buildings are positive integers less than 10,000 and there will be at least one and at most 5,000 buildings in the input file. Each building triple is on a line by itself in the input file. All integers in a triple are separated by one or more spaces. The triples will be sorted by Li the left x coordinate of the building, so the building with the smallest left x-coordinate is first in the input file.

**The Output**

The output should consist of the vector that describes the skyline as shown in the example above. In the skyline vector (v1, v2, v3,…, vn-2,vn-1,vn), the vi such that i is an even number represent a horizontal line (height). The vi such that i is an odd number represent a vertical line (x-coordinate). The skyline vector should represent the "path" taken, for example, by a bug starting at the minimum x-coordinate and traveling horizontally and vertically over all the lines that define the skyline. Thus the last entry in the skyline vector will be a 0. The coordinates must be separated by a blank space.

Sample Input

1 11 5 2 6 7 3 13 9 12 7 16 14 3 25 19 18 22 23 13 29 24 4 28

Sample Output

1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0