Assignment 2

Timothy Hoang

2/10/19

CMPT435-111

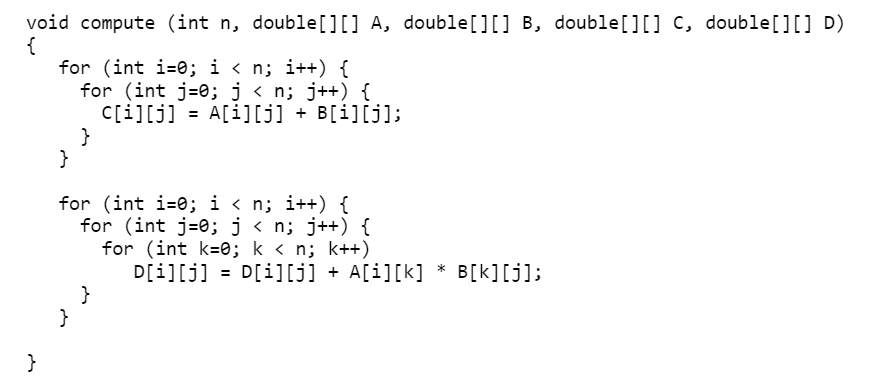
**Date Assigned: 02/04/2019**

**Due: Midnight 02/10/2019 on iLearn**

**Please read turn-in checklist at the end of this document before you start doing exercises.**

**Section 1: Pen-and-paper Exercises**

1. Analyze the following code and provide a "Big-O" estimate of its running time in terms of n. Explain your analysis.



Note: Credit will not be given only for answers - show all your work:

(5 points) steps you took to get your answer.

First line has 7 primitive operations

Second line has 7 primitive operations

Third line has 6 primitive operations

Fourth line has 7 primitive operations

Fifth line has 7 primitive operations

Sixth line has 7 primitive operations

Seventh line has 8 primitive operations

First and fourth lines run n times

Second, third, and fifth lines run n^2 times

Sixth and seventh lines run n^3 times

n\*(7+7) + n^2\*(7+6+7) + n^3\*(7+8)

(5 points) your answer.

14n + 20n^2 + 15n^3

1. Analyze the following code and provide a "Big-O" estimate of its running time in terms of n. Explain your analysis.

for (int i = 0; i < n; i \*=2)

some O(1) time statements;

end for

**Note: This question will be discussed in class.**

Note: Credit will not be given only for answers - show all your work:

(5 points) steps you took to get your answer.

i starts at 0 and is multiplied by 2 after every iteration. However 0 times anything is still 0.

(5 points) your answer.

This is an infinite loop since it is impossible for i to get past 0 through multiplication.

1. Analyze the following code and provide a "Big-O" estimate of its running time in terms of n. Explain your analysis.

k = 0//1

for (i = n / 2; i <= n; i++) //8

for (j = 2; j <= n; j = j \* 2)//7

k = k + n / 2; //5

end for

end for

**Note: This question will be discussed in class.**

Note: Credit will not be given only for answers - show all your work:

(5 points) steps you took to get your answer.

First line runs once

Second line runs n/2 times

Third and fourth lines run n/2 \* log(n) times

Multiply by primitive operations and add them up

1\*1 + n/2\*8 + n/2\*log(n)\*12 = 1 + 4n + 6n\*12log(n)

(5 points) your answer.

1 + 4n + 6n\*12log(n)

1. Analyze the following code and provide a "Big-O" estimate of its running time in terms of n. Explain your analysis.

int j = 1, i = 0; //2

while (i < n) //3

{

i = i + j; //4

j++; //3

}

**Hint: The loop variable ‘i’ is incremented by 1, 2, 3, 4, … until i becomes greater than or equal to n.**

Note: Credit will not be given only for answers - show all your work:

(5 points) steps you took to get your answer.

First line runs once

Other lines run sqrt(2n+1) times

Multiply by primitive operations then add them up

2\*1 + sqrt(2n+1) \* 10

(5 points) your answer.

2 + sqrt(2n+1) \* 10

1. Consider the following problem:

**Input:** an array A of n integers (positive, negative, or 0), elements sorted in ascending order.

**Output:** if there exists a majority element.

An element is a majority if it appears more than n/2 times. For example, if the input list is:

{0, 0, 0, 0, 0, 0, 1, 1, 2, 4, 7}

The output should be 0, as 0 appears 6 times (>n/2 = 11/2 times).

However, if the input list is:

{0, 0, 0, 1, 1, 2, 3, 10, 10}

The majority element does not exist.

Design an algorithm that solves this problem.

1. **describe the idea behind your algorithm in English (5 points);**

We check for the first number in the sequence and start the count off at 0 and increment it for every number in the array if it matches what we are checking for going through one at a time. If it does not match, we reset our count to 1 and set the new number we are checking for as the current number in the array. The max count will change if the current count exceeds the highest count we have.

**(ii) provide pseudocode (10 points);**

For (i=0; i>n; i++){

if A[i] == check{ //3

count++ //3

if (count > maxcount){ //3

maxcount = count; //2

}

}

else if A[i] != check{ //3

check = A[i]; //2

count = 1; //1

}

}

if (maxcount > A.length/2){

majority = true;

}

**(iii) analyze its running time (5 points).**

**First line has 7 primitive operations**

**Then it has to run either 11 or 9 primitive operations**

**Lastly it runs either 4 or 5 primitive operations**

**Regarding requirement (iii): Unless otherwise specified, show the steps of your analysis and present your result using big-O.**

**Big-O could be several things depending on what numbers are in the array given**

**Section 2: Java Implementation**

1. Implement problem 5 in Java.

Note:

Find a file called Problem5.java in assignment 2 folder.

Complete the method of majority().

Test your method in the main method provided following the comments.

**Important: In all of the assignments of this course, when you are asked to implement an**

**algorithm for a problem, your code will be evaluated based on:**

**5 points - Execution**

**Each file must run without error or warning on valid input described in the main method provided.**

**5 points - Within Code Documentation**

**Is the code documented for obvious understanding of the use, preconditions, and postconditions of each function?**

**20 points - Correctness**

**Is the algorithm implemented correctly? Does your method pass the test?**

**TURN-IN CHECKLIST:**

1. **Answers to Section 1 (.doc/.txt/.pdf), and to Section 2 (all your source Code (.java files)). Remember to include your name, the date, and the course number in comments near the beginning of your code/report.**
2. **Create a folder and name it 'FirstName\_LastName\_assignment\_2'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and push it to iLearn.**