Assignment 5

Date Assigned: 09/28/2018
Due: Midnight 10/05/2018 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. Consider the following numerical questions game. In this game, player 1 thinks of an integer in the range 1 to n, where n is an integer. Player 2 has to figure out this number by asking the fewest number of true/false questions.
   For example, a question may be "Is your number larger than x?"
   Assume that nobody cheats.
   (a) What is an optimal strategy if n is known? Describe your algorithm (English description 5 points), and Analyze your algorithm's running time (5 points).
   The optimal strategy if 'n' is known is to begin by asking: Is the number greater or less than n / 2. Based on the answer repeat this process until there is only one number left, which should be your answer.

   My algorithm: start is = 0, and end is = n - 1. We then enter a loop in which the middle of n (start + end / 2) is compared to x, the unknown number. after we find whether it is greater than or less than the middle of n, either the start or the end will be adjusted in order to search a certain portion of n until we are left with only one option for x.

   The running time would be O(log(n)).
   (b) What is a good strategy if n is not known? Describe your algorithm (English description 5 points), and Analyze your algorithm's running time (5 points).
   An optimal strategy may be to begin with a guess, from which we will find out if the number is higher or lower. We can then divide/multiply our guess by two in order to search further.

   My algorithm: Begin with our guess, a variable well call mid. Is x higher or lower than mid? Use the answer to determine whether mid becomes the min or the max of our next search. If it is min, multiply mid by two to find the max, and if it is max, divide mid by two to get the min. Repeat this process within a loop until we can return our answer.

   The running time would be O(2^n).
2. Let L be a list of numbers in non-decreasing order, and x be a given number. Describe an algorithm that counts the number of elements in L whose values are x (English description 5 points + Pseudocode 5 points). For example, if L = {1.3, 2.1, 2.1, 2.1, 2.1, 6.7, 7.5, 7.5, 8.6, 9.0} and x = 2.1 then the output of your algorithm should be 4. Your algorithm should run in O(log n) time (20 points).
   Important: In all of the assignments of this course, when you are asked to give an
   algorithm for a problem, you are (unless otherwise indicated) expected to
   (ii) **describe the idea behind your algorithm in English (5 points);**
   My algorithm will divide the array into two halves. The left half will be examined separate from the right half as needed, and in each half, the array will continue to be divided until we have an array that is occupied only by the value we are looking for, in which case the size is the number of occurrences.

(ii) provide pseudocode (10 points);

```
input A[], x
n = size(A), left = 0, right = n - 1
while left <= right
        mid = left + right - left / 2
        if A[mid] < x
                left = mid + 1
        end if
        else right = mid - 1
end while

start = right, left = 0, right = n - 1
while left <= right
        mid = left + right - left / 2
        if A[mid] <= x
                left = mid + 1
        end if
        else right = mid - 1
end while
end = right
return end - start
```

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

We are dividing an array in half, and then dividing these 'subarrays' in half as well in order to track a specific value. Until we have an array with only this specified value within it, we will continue to divide each of them in half. This gives us a running time of O(log(n)).

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

Note: This problem will be discussed in class. Algorithms that are O(n) or slower will be scored out of 5 points.

3. You have n coins (n may be even or odd) such that n-1 coins are of the same weight and one coin is heavier than the other coins.

You have a balance scale: you can put any number of coins on each side of the scale at one time, and it will tell you if the two sides weigh the same, or which side is lighter if they do not weigh the same.

Outline an algorithm for finding the coin with different weight.
The number of weighings using your algorithm should be O(log n).

Full credit (15 points) will be awarded for an algorithm that is O(log n). Algorithms that are O(n) or slower will be scored out of 5 points.

(i) **describe the idea behind your algorithm in English (10 points);**

My algorithm is designed to weigh the coins, and then adjust them as necessary in order to find the heavier coin. Begin by splitting the amount of coins in half, and comparing their weights. Whichever of the two halves weighs more, will then be divided in half, and the process will be repeated with the heavier sample of coins until we are left with only one coin that is heavier than another single coin.

(ii) analyze its running time (5 points).

We begin with a full sample of coins divided in half. These two halves are weighed against each other(the difference in weight is discovered), and then the heavier half is weighed again and divided again until we have our answer. The equation for the running time of this algorithm depends on how many

times we must multiply (n / 2) by 2. This is why the running time is O(log(n)).
For this problem, you do NOT need to write the pseudocode.

Section 2: Java Implementation

4. Implement problem 2 in Java (30 points).

Note:
Find a file called Problem2.java in assignment 4 folder.
Complete the method of count().
Test your method in the main method provided.
Programs that are O(n) or slower will be scored out of 10 points.

TURN-IN CHECKLIST:

1. **Answers to Section 1 (.doc/.txt), 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_5'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and push it to iLearn.**