Assignment 5

Timothy Zwart Cmpt\_435\_11\_18F

**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.

1. 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).
   1. Do a binary search in order to get O(log n) times. O log n because it keep cutting the search in half. First answer total divide by 2. If answer is correct stop. If answer isn’t correct is it higher or lower. Higher then take current answer minus 1 and divide by 2. If lower take current answer add 1 then add N then divide by 2. Repeat until finished.
      1. Answer = N/2

While (answer != #){

If (answer >#){

Answer=(Answer-1)/2

}

If(answer < #){

Answer=((Answer+1)+N)/2

}

}

1. 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).
   1. If n is unknown you must you will have to guess exponentially and repeat after every guess until a result has a lower and then begin a binary search. O(logN);
      1. Int i=0;

Answer=1;

While(answer < #){

i++;

Answer = 2^i

}

If(answer != #){

Int uppernumber = 2^i

Int lowernumber =2^i-1

Answer = ((2^i)-2^(i-1))/2) +lowernumber;

While(answer != #){

If (answer >#){

Answer=(Answer-1)+lower number number/2

}

If(answer < #){

Answer=((Answer+1)+ Uppernumber)/2

}

}

}

1. 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**

1. **describe the idea behind your algorithm in English (5 points);**
   1. Find the first time the element occurs and the last time the element occurs then minus the index of the second element from the index of the first. Begin with a binary search looking to the starting from the mid element in the array. If element being looked for is greater than the midpoint the midpoint becomes equal to one more, else the midpoint is moved one over. Repeat for the second element. Then take the first time it occurs index and minus the start index from it.
2. **provide pseudocode (10 points);** 
   1. count=0;

int midpoint;

int left;

int n=array.length;

int right = n-1;

while(right >= left){

midpoint =(left+right)/2;

if( x > A[midpoint]){

left = midpoint +1;

}else{ right =midpoint – 1;}

}

Int start =right;

Left =0;

While(right >= left){

Midpoint = (left+right)/2;

If(x>= A[midpoint]){

If(x>= A[midpoint]){

Left = midpoint +1;}else{

Right = midpoint -1;

}

}

Int end =right

Count= End-start

}

1. **analyze its running time (5 points).**
   1. O(Log n) uses only binary searches. Binary Search for first element and then binary search for second element and then add then minus first from second. Log n because binary search 1 = log n and binary search 2 = log n. Become Big 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.**

1. 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.**

1. **describe the idea behind your algorithm in English (10 points);**
   1. Divide the piles in two then if it is an amount odd amount then remove one coin. If the two piles are equal then the extra coin is the heavy one. What ever pile is heavier then divide in half again and loop.
   2. Coin Pile

While(coin not found){

Coin Pile/2

1st half= coin pile/2

2nd half = coin pile/2

If(1sthalf = 1 && 2nd half = 1){

If (1st half> 2nd){

Result = 1st half;

}else {Result = 2nd half}

}

If (1sthalf >2nd half){

Return 1sthalf as pile 1

}else{

Return 2nd half as pile 1

}

}

1. **analyze its running time (5 points).**
   1. Log n because it is only a binary search to find the coin. All are binary searches so there would be nothing else added to make above a Big O(log n)

**For this problem, you do NOT need to write the pseudocode.**

**Section 2: Java Implementation**

1. 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.**