The University of Saskatchewan

Saskatoon, Canada

Department of Computer Science

CMPT 280– Intermediate Data Structures and Algoirthms

Assignment 1

Date Due: January 21, 2020, 9:00pm

Total Marks: 30

1 Submission Instructions

- Assignments must be submitted using Moodle.
- Responses to written (non-programming) questions must be submitted in a PDF file, plain text file (`.txt`), Rich Text file (`.rtf`), or MS Word's `.doc` or `.docx` files. Digital images of handwritten pages are also acceptable, provided that they are **clearly** legible, and in a common format such as JPEG or PNG.
- Programs must be written in Java.
- If you are using IntelliJ (or similar development environment), do not submit the whole project folder. Hand in only those files identified in Section 3.
- No late assignments will be accepted. See the course syllabus for the full late assignment policy for this class.

2 Files Provided

None.

3 What to Hand In

You must submit the following files:

assignment1{.doc | .docx | .rtf | .pdf | .txt} - your answers to questions 1 to 7. Digital images of handwritten pages are also acceptable, provided that they are **clearly** legible.

4 Your Tasks

The questions on this assignment are about timing analysis. There is no programming on this assignment.

Question 1 ():

For each of the following functions, give the tightest upper bound chosen from among the usual simple functions listed in Section 3.5 of the course readings. Answers should be expressed in big-O notation.

(a) (1 point) $f_1(n) = n \log_2 n + n^4 \log_{280} n + \frac{2^n}{42}$

(b) (1 point) $f_3(n) = 0.4n^4 + n^2 \log n^2 + \log_2(2^n)$

(c) (1 point) $f_2(n) = 4n^{0.7} + 29n \log_2 n + 280$

Question 2 ():

Suppose the **exact** time required for an algorithm A in both the best and worst cases is given by the function

$$T_A(n) = \frac{1}{280}n^2 + 42 \log n + 12n^3 + 280\sqrt{n}$$

(a) (2 points) For each of the following statements, indicate whether the statement is true or false.

1. Algorithm A is $O(\log n)$
2. Algorithm A is $O(n^2)$
3. Algorithm A is $O(n^3)$
4. Algorithm A is $O(2^n)$

(b) (1 point) What is the time complexity of algorithm A in big-Θ notation.

Question 3 ():

If possible, simplify the following expressions.

(a) (1 point) $O(n^2) + O(\log n) + O(n \log n)$

(b) (1 point) $O(2^n) \cdot O(n^2)$

(c) (1 point) $42O(n \log n) + 18O(n^3)$

(d) (1 point) $O(n^2 \log_2 n^2) + O(m)$

Question 4 ():

Consider the following Java code fragment:

```
// Print out all ordered pairs of numbers between 1 and n
for(i = 1; i <= n; i++) {
   for(j = 1; j <= n; j++) {
      System.out.println( i + ",␣" + j) ;
   }
}
```

(a) (3 points) Use the *statement counting approach* to determine the exact number of statements that are executed when we run this code fragment as a function of `n`. Show all of your calculations.

(b) (1 point) Express the answer you obtained in part (a) in big-Θ notation.

Question 5 ():

Consider the following pseudocode:

```
Algorithm roundRobinTournament(a)
This algorithm generates the list of matches that must be
played in a round-robin pirate-dueling tournament (a tournament where
each pirate duels each other pirate exactly once).

a is an array of strings containing names of pirates in the tournament

n = a.length
for i = 0 to n-1
   for j = i+1 to n-1
      print a[i] + "␣duels␣" + a[j] + ",␣Yarrr!"
```

(a) (6 points) Use the *statement counting approach* to determine the exact number of statements that are executed by this pseudocode as a function of n. Show all of your calculations..

(b) (1 point) Express the answer you obtained in part a) in big-Θ notation.

Question 6 (3 points):

Using the active operation approach, determine the time complexity of the pseudocode in question 5. Show all your work and express your final answer in Big-Θ notation.

Question 7 (6 points):

Consider the following pseudocode.

```
Algorithm multiSearch( data, target ):
data:  a list of arrays of integers; in each array the
       integers are sorted in ascending order; the list
       'data' has a cursor.
target: an integer

// Iterate over the arrays in the list 'data' using
// its cursor:
data.goFirst()
found = false
while( !data.after() and !found ) {
   // search for integer 'target' in A
   found = binarySearch(data.currentItem(), target)
   data.goForth()
```

Using the active operation approach to timing analysis determine the time complexity of this pseudocode in the worst case. Assume that the list of arrays contains n arrays and that each array has exactly m items in it. Show all your work and express your final answer in Big-O notation.