Assignment 1 (20% of total assessment marks)

Due date: 29 January 2024

Scope:

The tasks of this assignment cover the **Data Structures and Algorithms, in particular the algorithm complexity and Recursion**. The assignment covers the topics discussed in topics 1 and 2.

The assignment is divided into two parts – Part One covers the theoretical aspect of the materials discussed during classes, and Part Two covers the practicality of the concepts.

Assessment criteria:

Marks will be awarded for:

- Correct,
- Comprehensive, and
- Appropriate

application of the materials covered in this subject.

Assignment Specification:

Part One: (70 marks)

Algorithms complexity

Question 1 (10.0 marks)

Arrange the following functions into increasing order; that is, $f(n)$ should come before $g(n)$ in your list if and only if $f(n)$ is $O(g(n))$.

$6 + n^0$, $16^{\lg n}$, $\lg(n)^n$, $2^{3\lg n}$, 2^{3n}, $\log(n!)$, $\frac{1}{3}(log_3 n)^3$, $\lg\left(2^{\lg\lg n}\right)$, $3^{log_3 \lg n} + n^{2/3}$,

$(n^2+3)!$, $(1+2+3+\cdots+n)$, $2^{\lg n} \times (log_2 n)^0$.

Question 2 (10.0 marks)

Determine the big O running time of the method myMethod(int n) by counting the approximate number of operations it performs. Show all details of your answer.

```
Static void doIt (int n) {
        int i
        int j ← (2 × n)
        loop while ( j > 0 ) {
                i ← n
                loop while ( i >= 1 ) {
                        i ← i / 2
                }
                j ← j – 1
        }
}


Static int myMethod (int n) {
    sum ← 0
    for i ← 1 to n {
        sum = sum + doIt(i)
    }
    return 1
}
```

Recursion

Question 3 (20.0 marks)

Your lecturer is a funny guy; given an unsorted list of integer numbers of n elements, and to find the largest number in the list, he will first initialize a variable, let's say max, to the smallest possible number an integer can be, and randomly select an element (a number) from the unsorted list. Check if this number is greater than the variable max. If it is, he will set the variable max to the number. Next, he will discard the number from the list and create a new list. The new list now has $n - 1$ elements. He will continue with the same process on the $n - 1$ elements list until the unsorted list has no more element. When this happens, the variable max will contain the largest number in the list.

(a) Write in pseudocode a recursive implementation of the described algorithm.
(b) Analyse the asymptotic complexity of the algorithm. Give the worst-case, average-case and best-case running time in terms of Θ notation. Justify your answer.

Question 4 (20.0 marks)

For each of the following recurrence relations, determine the runtime $T(n)$ complexity. Use the Master Theorem to solve the recurrence relations if the recurrence relations can be solved using Master Theorem, otherwise use expansion (or recursive substitution) to solve the recurrence relations. You must show all your working to justify your answer. Solution without clear justification or steps scores no mark.

1. $T(n) = 4T\left(\frac{n}{2}\right) + n^2 + n,\ and\ T(1) = 1$ (4.0 marks)
2. $T(n) = 2T\left(\frac{n}{2}\right) + n\ lg^3 n,\ and\ T(1) = 1$ (4.0 marks)
3. $T(n) = 3T\left(\frac{n}{2}\right) + n\ lg\ n, and\ T(1) = 1$ (4.0 marks)
4. $T(n) = 4T\left(\frac{n}{4}\right) + n\ lg\ n,\ and\ T(1) = 1$ (4.0 marks)
5. $T(n) = T(n-1) + n^2,\ and\ T(0) = 1$ (4.0 marks)

Question 5 (10.0 marks)

Suppose you are designing a multi-player game that has n ≥ 1000 players, numbered 1 to n, interacting in an enchanted forest. The winner of this game is the first player who can meet all the other players at least once (ties are allowed). Assuming that there is a method meet (i, j), which is called each time a player i meets a player j,

i. Describe an algorithm including an appropriate data structure to keep track of the pairs of meeting players and who is the winner. **(7.0 marks)**

ii. What is the running time complexity of your algorithm? Explain your answer. **(3.0 marks)**

Part Two: (30.0 marks)

Question 6 (30.0 marks)

Given two unsorted arrays A and B. Array A consists of n numbers of integers and array B consists of m numbers of integers. The integers in array B are distinct and they are in the range $1 \cdots n$. The problem is to output the $B[i]^{th}$ smallest number in array A, for $1 \leq i \leq m$, and $m \leq n$. For example, let $A = [7, 3, 8, 21, 5, 11]$ and $B = [3, 5, 1]$, then the output is $[3, 7, 11]$ because the sorted array of the elements in array A is $[3, 5, 7, 8, 11, 21]$ and the first (1st) smallest element of array A is 3, the 3rd smallest element of array A is 7, and the 5th smallest element of array A is 11.

Write two Java, C++, or Python programs to implement the following algorithms:

a) Algorithm 1. Sort array A and Array B and produce the required output. **(10 marks)**

b) Write a report on the analysis of your implementation for Algorithm 1, using the technique we discussed in Lecture 1 (Slides 24 – 37). How much time (number of operations) does Algorithm 1 take? **(3 marks)**

c) Algorithm 2. Sort array A but do not sort array B. Produce the required output. **(10 marks)**

d) Write a report on the analysis of your implementation for Algorithm 1, using the technique we discussed in Lecture 1 (Slides 24 – 37). How much time (number of operations) does Algorithm 2 take? **(3 marks)**

e) Test run your program, screen-capture the output, and submit together with your reports for Part 8(b) and 8(d). **(4 marks)**

Standard Requirements for Question 7 (Programming question):

- Java Version – JDK 6 update 17 or higher (Using Windows), or
- C++ / C compiler – g++ 4.0 or higher (Using Windows, Linux or UBUNTU), or Python 3.x.
- Programs should be appropriately documented with comments.
- Students are to give batch / make files or compilation instruction.
- Execute your program and screen-capture the output. Include in your submission all source code and libraries plus the screen-captures.
- Students are to place all compilation and instructions on how to run the program inside a readme.txt file. Your lecturer will refer to this file when marking. Without a readme.txt or clear instructions for compilation, your lecturer will compile based on his/her computer setting; any incompatibility, will deem as failure to compile the program.
- Submission filenames are to follow the naming convention given in the submission instruction below. Do not use your own filename.

Submissions

This assignment is due by 9:00 pm (Singapore time) on Monday, 29 January 2024.

For Part One of your assignment, type your answers and save your solution in a pdf formatted file, and named it as SolutionPart1.pdf. For Part Two of your assignment, execute your program and screen capture the output. Zip your source code (in pure ASCII text format with an appropriate extension; that is .cpp for C++ codes, .java for Java codes, and .py for Python codes) together with your screen capture and named it as SolutionPart2.zip. Next zip together SolutionPart1.pdf and SolutionPart2.zip and named it as YourPUID-A1.zip, where PUID (Partner University ID) is your UOW Student Number.

Submit the files **YourPUID-A1.zip** through Moodle in the following way:

1) Access Moodle at **http://moodle.uowplatform.edu.au/**
2) To login use a Login link located in the right upper corner the Web page or in the middle of the bottom of the Web page
3) When successfully logged in, select a site CSCI203 (SP124) Algorithms and Data Structures
4) Scroll down to a section Submissions of Assignments
5) Click at Submit your Assignment 1 here link.
6) Click at a button Add Submission
7) Move a file, for example, **YourPUID-A1.zip** into an area. You can drag and drop files here to add them. You can also use a link *Add…*
8) Click at a button Save changes,
9) Click at a button Submit assignment,
10) Click at a button Continue,

A policy regarding late submissions is included in the subject outline.

Only one submission per student is accepted.

Assignment 1 is an individual assignment, and it is expected that all its tasks will be solved individually without any cooperation with the other students. Plagiarism is treated seriously. Students involved will likely receive zero. If you have any doubts, questions, etc. please consult your lecturer or tutor during lab classes or over e-mail.

End of specification