THE UNIVERSITY OF WESTERN ONTARIO

DEPARTMENT OF COMPUTER SCIENCE
LONDON CANADA

Analysis of Algorithms
(Computer Science 3340b)

ASSIGNMENT 1
Due date: Tuesday, Feburary 2, 2021, 11:55 PM

1. **A complete binary tree** is defined inductively as follows. A complete binary tree of height 0 consists of 1 node which is the root. A complete binary tree of height $h+1$ consists of two complete binary trees of height h whose roots are connected to a new root. Let T be a complete binary tree of height h. Prove that the number of leaves of the tree is 2^h and the size of the tree (number of nodes in T) is $2^{h+1}-1$.

2. Question 2-1 (pp. 39) a., b., and c. in the text book.

3. Question 2-4 (pp. 41) in the text book.

4. Read the text book for the definition of o and ω and answer question 3-2 (pp. 61) in the text book.

5. Question 4-2 (pp. 107) in the text book.

6. Suppose that the running time of a recursive program is represented by the following recurrence relation:

$$\begin{aligned} T(2) &\leq c \\ T(n) &\leq 2T(n/2) + cn\log_2(n) \end{aligned}$$

Determine the time complexity of the program using recurrence tree method and then prove your answer.

7. Consider the Fibonacci numbers:

$$F(n) = F(n-1) + F(n-2),\ n > 1;\ F(1) = 1;\ F(0) = 0.$$

a). Write a recursive function to compute $F(n)$ using the above definition directly. Implement your solution and print $F(i * 5)$, where $0 \leq i \leq 10$, as output.

b). Write a recursive function/procedure to compute $F(n)$ with time complexity $O(n)$ (more precisely, the time complexity should be $O(nA(n))$ when n is large, where $A(n)$ is the complexity of adding $F(n-1)$ and $F(n-2)$). Implement your solution and print $F(i*20)$, where $0 \leq i \leq 25$, as output. This program must be able to compute $F(n)$ precisely for $n \leq 500$.
Hint 1:

Let $G(n) = \begin{pmatrix} F(n) \\ F(n-1) \end{pmatrix}$: $G(n) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} G(n-1),\ n > 1;\ G(1) = \begin{pmatrix} 1 \\ 0 \end{pmatrix};\ G(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Design a recursive algorithm for $G(n)$, that means the algorithm will return both $F(n)$ and $F(n-1)$ with input parameter n.
Hint 2: Can you use a primitive type to store $F(500)$?

c) (optional for bonus credit)
Write a recursive function/procedure to compute $F(n)$ with time complexity $O(\log(n))$ (more precisely, the time complexity should be $O(\log(n)A(n))$ when n is large, where $A(n)$ is the complexity of adding or multiplying $F(i)$ and $F(j)$).

$$Hint: \quad G(n) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} G(n-1) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1} G(1), \quad n > 1$$

For programs in a), b), and c) of this question, you are **NOT** allowed to use Python. For C++ and Java, you can only use primitive types such as char, int, long and long long. You are not allowed to use large integer, such as BigInteger in Java, from the language library. You have to write your own large integer data type or object, if needed.

d) Use the Unix time facility (bash time command) to track the time needed for each algorithm. Compare the results and state your conclusion in two or three sentences.

e) Can you use your program in a) to compute $F(50)$ if int type of 4 bytes is used? Briefly explain your answer. Explain why your program in b) computes $F(500)$ precisely?

Algorithms and answers for 6 b) to 6 e) should be in **asn1_solutions.pdf**

For this question, for C++, when compiling option O2 should be considered and a makefile should be written such that the command "make clean" will remove all the "*.o" files, the command "make asn1_a" will generate an executable file "asn1_a" for 6 a) that can be run by typing "asn1_a"; the command "make asn1_b" will generate an executable file "asn1_b" for 6 b) that can be run by typing "asn1_b"; and the command "make asn1_c" will generate an executable file "asn1_c" for 6 c) that can be run by typing "asn1_c". If you are using Java, you may not need the makefile. In that case, you should have shell script files, "asn1_a", "asn1_b", and "asn1_c" such that by typing, "asn1_a", "asn1_b", and "asn1_c", your java programs will run.

You should use unix command "script" to capture the screen of the execution of your program.

Your programs have to be able to run on **compute.gaul.csd.uwo.ca** as our TAs will be marking your programs there.