if $c(g(n)) \leq f(n) \rightarrow f(n)$ is Ω $g(n)$

if $c\,g(n) \geq f(n) \rightarrow F(n)$ is $O(g(n))$

if $c_1 g(n) \leq f(n) \leq c_2 g(n)$ ↓ $f(n)$ is $\Theta(g(n))$

lu zeng
1693815

COMP611 — Algorithm Design and Analysis

Lab 01

What to do

Do all the tasks, and answer all the questions. You can work individually or in pairs. Record non-code answers, as you may have to give open-class feedback on them.

Question 1

n is bigger than logn

State, for each of the empty cells in this table, whether that statement is true or false for the functions $f(n), g(n)$ in the same row.

	f(n)	g(n)	f(n) is O(g(n))	f(n) is Ω(g(n))	f(n) is Θ(g(n))
a)	$n\log(n)+100$	$n-200\log(n)$	F	T	F
b)	$n^2+n^{\frac{1}{2}}$	$\frac{1}{2}n+(\frac{n}{2})^2$	T	T	T
c)	$n\log(n^3)$	$n(\log(n))^2$	T	T	T
d)	$2(n+2)(n+1)$	$2+4+6+\cdots+2n$	F	T	F
e)	$2n+\log(n)$	$2n+22$	T	T	T

Question 2

In the lecture, you learned about four different ways to implement an algorithm to compute the nth element of the Fibonacci sequence. Complete the following tasks, and answer the following questions:

1. Implement all four of the algorithms. Each implementation should take a parameter n, and return the value of the nth Fibonacci number.
2. Write code that will execute each implementation for every n from 1 to 45, and time how long it takes, printing that time to the console. If you are working in Java, use `System.nanoTime` for this. Which one performed best? How does that compare to their asymptotic complexities?

For ~~fibb fine~~ try $f(n) = n^2$

For Try 2 ~~runs~~ performed the best.

run time:

(try 1) $f(try\ 4) > f(try\ 3) > f(try\ 2)$