Algorithms and Data Structures
CH08-320201
Homework 3
Fibonacci Numbers and Recurrences

Problem 3.1

a. C++ source file
b.

(N)	Naïve Recursive	Bottom Up	Closed Form	Matrix Multiplication
10	0	0	0	0
20	0.001	0	0	0
30	0.011	0	0	0
40	1.194	0	0.001	0
50	-	0	0.001	0
100		0.001	0	0
500		0.001	0	0
1000		0.001		0.001
5000		0.001		0.001
10000				

c. For N greater than 40, the naïve recursive algorithm fails to return a value. For N greater than 1000, the closed form algorithm also fails to return an accurate Fibonacci sequence and for N values greater than 10,000 the remaining two algorithms also return inaccurate values for the Fibonacci sequence.
This is because the larger the number gets the algorithm cannot approximate the correct Fibonacci sequence. Because approximation loses its accuracy for the algorithms.

Algorithms & Datastructures.
HW#3
Problem 3.2

b) let $A = (a_n a_{n-1} \ldots a_1)_2$ assuming n to be a power of 2
$B = (b_n b_{n-1} \ldots b_1)_2$

A divide & conquer algorithm can be implemented by breaking A & B into 2 integers of $n/2$ bits each.

$$A = \underbrace{(a_n \ldots a_{n/2+1})_2}_{A_1} \cdot 2^{n/2} + \underbrace{(a_{n/2} \ldots a_1)_2}_{A_2}$$

$$B = \underbrace{(b_n \ldots b_{n/2+1})_2}_{B_1} \cdot 2^{n/2} + \underbrace{(b_{n/2} \ldots b_1)_2}_{B_2}$$

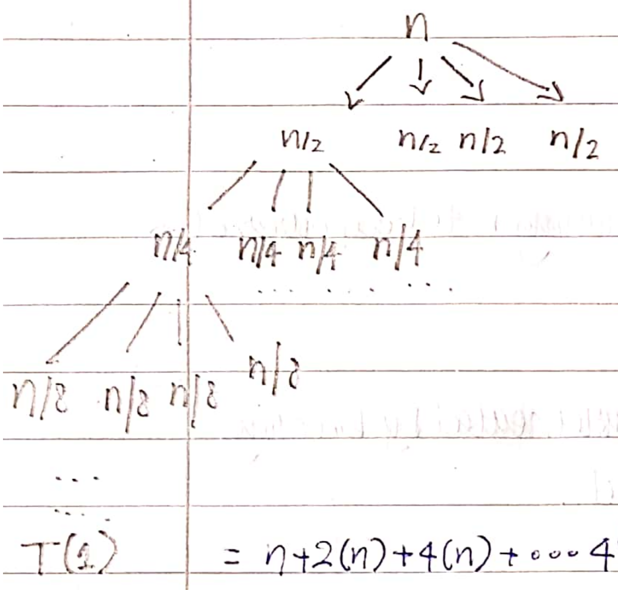
$$(AB) = A_1 \cdot B_1 \cdot 2^n + (A_1 \cdot B_2 + A_2 \cdot B_1) \cdot 2^{n/2} + A_2 \cdot B_2$$

$\Rightarrow$ reduced to 4 multiplication of $n/2$ bit integers, 3 additions of integers with $2n$ bits and 2 shifts.

c) $T(n) = 4T(n/2) + cn$

Scanned by CamScanner

d). $T(n) = 4T(n/2) + cn.$



n

$2(n)$

$4(n)$

$8(n)$

$= n + 2(n) + 4(n) + \cdots 4^{\log n}(T)(1)$

$= n \sum_{i=0}^{\log_2 n - 1} 2^i + \Theta(4^{\log n})$

$= n(2^{\log n} - 1)/(2-1) + \Theta(2^2)^{\log n}$

$= \Theta(n^2) + \Theta(n^2) = \Theta(n^2).$

e) Master Theorem.

$n^\wedge \log_b a = n^2$

$T(n) = \Theta(n^2).$

$\therefore$ proven