Asgmt#1- algorithm2

Ln#	Algorithm algortihm2	Cost	Times	Comments
1	num=0,num1=1,fibo=0	C1=3	1	3 variables assigned
2	For(int i=0;i<n;i++)	C2=1	1	Assigning of variable i
		C3=1	n+1	n successes and one failure
		C4=2	n	Incrementation and assigning of i
3	fibo=num+num1	C5=2	n	Addition and assigning repeated n times
4	num=num1	C6=1	n	Assignment n times
5	num1=fibo	C7=1	n	Assignment n times
6	Return num	C8=1	1	return

f(n)=C1*1+C2*1+C3(n+1)+C4(n)+ C5(n)+C6(n)+C7(n)+C8*1=6n+6

Algorithm2, with f(n)= 6n+6 :
Requires 6n+6 time units to solve a problem of size n
It requires time proportional to 6n
Its growth rate in linear

	Growth Rate Analysis			Asymptotic Analysis
F(n)	Time Units	Prop to	Rate	Big-Oh
F(n)=6n+6	6n+6	6n	linear	O(n)

algorithm3

Ln#	**Function Fibonacci**	Cost	Times	Comments
1	If(n=0) return 0	C1=1	1	We take the max cost, which is the matrix initialization and calling matrixpower, so we neglect cost of return
2	Initialize FM	C2=12	1	Assign 4 matrix elements, each costs 3
3	Call MatrixPower(n-1)	C3=1	1	Function called once
4	Return fm[0][0]	C4=3	1	Accessing two array elements, and returning a value

F(n) Fibonacci= C1*1+C2*1+C3*1+C4*1=1+12+1+3=17

Ln#	Procedure MatrixPower	Cost	Times	Comments
1	If(n>1)	C1=1	1	Comparison done one time before recursively called again + failure
2	Call MatrixPower(n/2)	C2=1	f(n/2)	Recursive calls
3	Update FM=FM*FM	C3=56	1	
4	if(n is odd)	C4=1	1	comparison
5	Update FM=FM*{{1,1},{1,0}}	C5=56	1	

F(n) MatrixPower=C1*1+C2*f(n/2)+C3*1+C4*1+C5*1=4+f(n/2)=$4\log_2 n$ +1

F(n) of algorithm3= f(n) Fibonacci +f(n) MatrixPower=17+$4\log_2 n$ +1=18+$4\log_2 n$

Algorithm3, with f(n)= 18+$4\log_2 n$:
Requires 18+$4\log_2 n$ time units to solve a problem of size n
It requires time proportional to $4\log_2 n$
Its growth rate in logarithmic

	Growth Rate Analysis			Asymptotic Analysis
F(n)	Time Units	Prop to	Rate	Big-Oh
f(n)= 18+$4\log_2 n$:	18+$4\log_2 n$	$4\log_2 n$	logarithmic	O($\log_2 n$)

References: CSPC 319 lecture slides

-"matrix operation costs from lecture material"

- "01 - Algorithm Design Patterns - (1) Recursion - soln-HW"