Homework 5

Hamza Bouhelal

March 2021

Problem 5.1:
a) Fibonacci.py
b)

n	Naive recursive	Buttom up	Closed form	Matrix representation
0	6.88769999896976 5e-05	8.414000035 372737e-06	1.9273000020 80301e-05	2.01130000050397 94e-05
1	1.31599995256692 64e-06	1.887000053 1023834e-06	2.9833999974 471226e-05	1.62800006364705 04e-06
2	2.44800003201817 16e-06	4.227999966 133211e-06	9.5320000355 05916e-06	1.77530000655679 03e-05
3	2.27900000027148 05e-06	2.405000032 0130493e-06	3.1223999940 26637e-05	1.85129999863420 38e-05
4	3.08699998186057 2e-06	1.717999907 6688546e-06	2.2360000002 663583e-06	2.73079999715264 42e-05
5	4.67699999262549 7e-06	1.914999984 36484e-06	2.0219999896 653462e-06	1.67070000998137 4e-05
6	0.00022366400003 19312	1.005699994 0388778e-05	2.1629999764 627428e-06	2.68490000507881 63e-05
8	1.92740000102276 1e-05	2.666999989 7421803e-06	2.1959999685 35034e-06	4.85350000190010 2e-05
10	6.59649999761313 7e-05	2.960000074 3812416e-06	2.1050000214 017928e-06	7.53130000248347 6e-05
13	0.00025961599999 391183	3.564000053 302152e-06	1.9599999632 19161e-06	4.09839999520045 24e-05
16	0.00091547800002 445	4.366999974 081409e-06	1.9579999843 699625e-06	8.49870000365626 8e-05
20	0.01186202700000 6492	8.585999921 706389e-06	2.0160000531 177502e-06	8.71979999601535 4e-05
25	0.09882025299998 531	1.025100004 8810965e-05	1.9569999949 45363e-06	6.89329999659094 e-05
30	1.29401628599998 73	1.203499994 0624402e-05	1.9579999843 699625e-06	8.38260000364243 8e-05

c)For n, all methods return the same Fibonacci Number except for the closed method since round is used, the larger n gets the less precise the closed method is.

Problem 5.2:

a)the asymptotic time complexity depending on the number of bits n for a brute-force implementation of the multiplication is $\Theta(n^2)$, it multiplies each bit of a with each bit of b, whenever there is a multiplication of a bit of a to a bit of b, the result is bit-shifted by the position of the bit in a , then the result is calculated by summing up all the results obtained before (and shifted).

For the addition, we see that the time complexity is $\Theta(n^2)$, We conclude that:

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

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

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

b)derive a devide and Conquer algorith by splitting the problem into two subproblems, with n a power of 2.

first we rewrite a and b in base B

$a = a1B^(m) + a2$ with a1 and a2 the left side and right side of a and m smaller than n.

$b = b1B^(m) + b2$ with b1 and b2 the left side and right side of b and m smaller than n.

for B is base 2 and m is $\frac{n}{2}$:

$a \times b = (2^{\frac{n}{2}}.a1 + a2)(2^{\frac{n}{2}}.b1 + b2)$

$= 2^n a1 \times b1 + 2^{\frac{n}{2}}(a12 + a2 \times b1) + a2 \times b2$

There is 4 multiplications, therefore: $T(n) = 4T(\frac{n}{2}) + \Theta(n)$

let's try to reduce $a1 \times b2 + a2 \times b1$

$a1 \times b2 + a2 \times b1 = (a1 + a2)(b1 + b2) - a1b1 - a2b2$

$= a1b2 + a1b1 + a2b2 + a2b1 - a1b1 - a2b2$

$= a1b2 + a2b1$

Now let's replace that in the expression obtained before:

$a \times b = (2^{\frac{n}{2}}.a1 + a2)(2^{\frac{n}{2}}.b1 + b2)$

$= 2^n a1 \times b1 + 2^{\frac{n}{2}}((a1 + a2)(b1 + b2) - a1b1 - a2b2) + a2 \times b2$

we can therefore do one multiplication less using this approach.

c)Since we can do one multiplication less using this approach, T becomes:

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

e) The Master method can be used here:

a = 3

b = 2

$n^{log_2^(3)} = n^{1.58}$

$f(n) = n$

$f(n) = O(n^{log_2^(3-e)}) = O(n^{log_2^(3-e)})O(n^{1.58-e)with e=log_2^3-1=0.58}$

Case 1 :

$T(n) = \Theta(n^{log_2^3}) = \Theta(n^{1.58})$