# Algorithms and Data Structures

Dragi Kamov

## Problem 1

a) I implemented the four methods and the code can be seen in "fibonacci_python.py"

b) I have the times written in text files together with the n and the fibonacci of that position. Only the naive recursive method is exceeding 5 seconds.

c) The bottom-up, closed form and matrix representation methods all have same fibonacci results till the 75th fibonacci and then the closed form method starts making mistakes.

d) My plot can be seen in "plot.xlsx"

## Problem 2

a) Brute-force implementation of multiplication would be a bit of multiplication between each bit from the first number with each bit of the second number, thus for every bit of the ~~seco~~ first number we would shift one place left with the next multiplication. And after that all the multiplications are being added.

example:

```
12        9
1100 · 1001
-----------
      1100
     0000
    0000
   1100
-----------
   1101100
         108
```

$$\frac{12 \cdot 9}{108}$$

So if we say that we have two numbers with n bits, while multiplicating we would have $n^2$ operations. $\Rightarrow T(n) = \Theta(n^2)$
The shifting is constant and because of that we can ignore it. For addition we also have $n^2$ operations so also $T(n) = \Theta(n^2)$

Total time: $\Theta(n^2) + \Theta(n^2) = \Theta(2n^2) \Rightarrow \boxed{T(n) = \Theta(n^2)}$

b) The divide & conquer algorithm would separate the two numbers in half multiply the halfs and then add them together. (Also we know from the problem that everytime we would be able to divide it in half)

Example in decimal:

$$23 \cdot 12 = (2 \cdot 10^1 + 3)(1 \cdot 10^1 + 2) = 2 \cdot 10^2 + 10^1(2 \cdot 2 + 3) + 3 \cdot 2 =$$
$$= 200 + 70 + 6 =$$
$$= 276$$

Example in binary:

$$1100 \cdot 1001 = (11 \cdot 2^2 + 0)(10 \cdot 2^2 + 1) = 10 \cdot 11 \cdot 2^4 + 1 \cdot 11 \cdot 2^2 + 0 \cdot 1$$
$$= 1100000 + 1100 + 0$$
$$= 1101100$$

c) So let's say that $X$ is the first number and we would divide it in $Xl$ and $X_r$ and the second number would be $Y$ and we would divide it into $Yl$ and $Y_r$.

$$X = Xl \cdot 2^{n/2} + X_r \qquad Y = Yl \cdot 2^{n/2} + Y_r$$

$$X \cdot Y = (Xl \cdot 2^{n/2} + X_r)(Yl \cdot 2^{n/2} + Y_r) =$$
$$= \underset{①}{Xl\,Yl} \cdot 2^n + 2^{n/2}(\underset{②}{Xl\,Y_r} + \underset{③}{X_r\,Yl}) + \underset{④}{X_r\,Y_r} \quad \circledast$$

So till now we still have 4 multiplications and the time complexity is $T(n) = 4T(n/2) + \theta(n)$, but we can simplify it more

$$Xl\,Y_r + X_r\,Yl = \underbrace{(Xl + X_r)(Yl + Y_r)}_{\text{And we have these from the divisions}} - Xl\,Yl - X_r\,Y_r$$

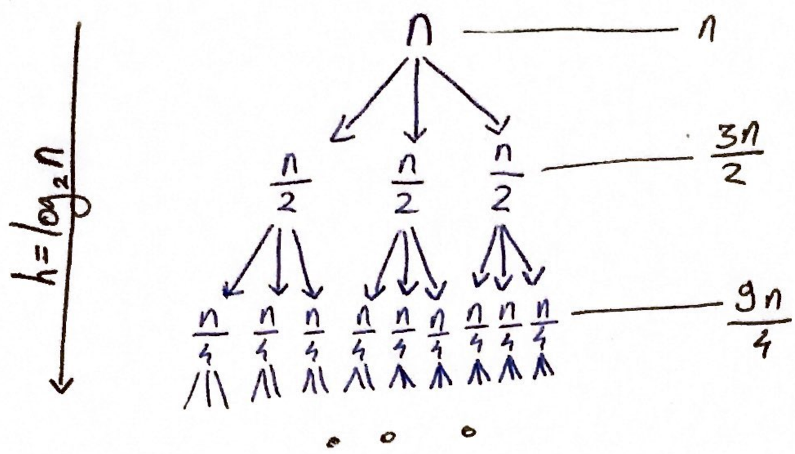
So if we substitute all these in $\circledast$ we would have 3 multiplications

$$XY = \underset{①}{Xl\,Yl} \cdot 2^n + 2^{n/2}[\underset{②}{(Xl + X_r)(Yl + Y_r)} - Xl\,Yl - X_r\,Y_r] + \underset{③}{X_r\,Y_r}$$

so now the time complexity is

$$T(n) = 3T(n/2) + \theta(n)$$

d)




$$n + \frac{3n}{2} + \frac{9n}{4} + \frac{27n}{8} + \frac{81n}{16} + \dots = \sum_{i=0}^{h} \frac{3^i n}{2^i} = \frac{1/2(3^h - 1)}{2^{h+1} - 1} n$$

$$= \frac{(3^h - 1)n}{2(2^{h+1} - 1)} = \frac{(3^h - 1)n}{2^{h+2} - 2} \approx \boxed{n^{1.58}}$$

e) $T(n) = 3T(n/2) + \theta(n)$

$f(n) = n$ $\quad\quad a = 3 \quad b = 2$

$n^{\log_b a} = n^{\log_2 3} \approx n^{1.58}$

We know that $\theta(n)$ is asymptotically smaller or equal to n.

$\varepsilon = 0.1$

$n^{1.58 - 0.1} = n^{1.48}$

$$\lim_{n \to \infty} \frac{\theta(n)}{n^{1.48}} = 0$$

$\Rightarrow$ $f(n)$ is asymptotically smaller than $g(n)$

$f(n) = O(n^{1.58 - \varepsilon})$

$\Rightarrow T(n) = \theta(n^{\log_b a}) = \theta(n^{\log_2 3})$

So this proves that the result from (d) is correct.