ADS Homework 5

Sanjay Timilsena

March 2020

1 Problem 5.1

1.1 a)

Please go through attached files fib_recursion.cpp, fib_bottomup.cpp, fib_closedform.cpp and fib_matrix.cpp.

1.2 b)

The data file for each program are attached as recurison.txt, bottomup.cpp, closedform.cpp and matrix.cpp which includes the input, output and the time taken respectively.

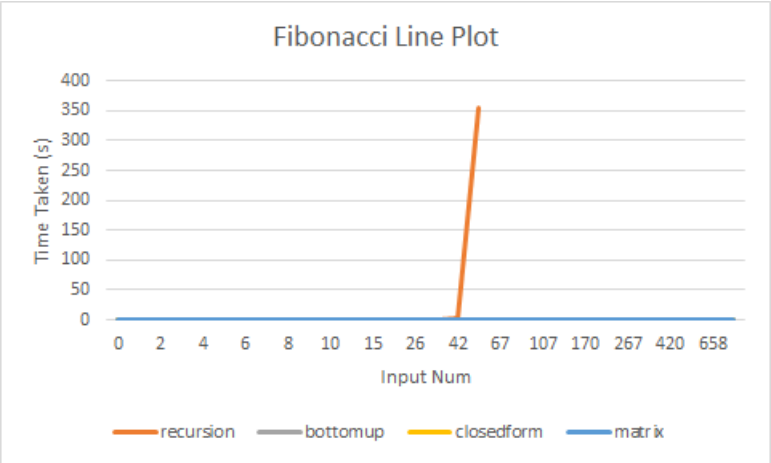
The required table is attached as Table.xlsx, an excel file.

1.3 c)

Here, for small value of input (< 50) every program gives the same output. While when the input value gets higher, there is change in output because of low range of data type in some cases like for matrix, as well as rounding off errors in cases like closedform.

1.4 d)

The graph with given input on x-axis and time taken on y-axis is as below.



2 Problem 5.2

2.1 a)

Here, addition, subtraction, and bit shifting can be done in linear time, i.e., in $\theta(n)$. For given two numbers a and b with n bits each, the number of multiplications require for each bits of one number is n as it has to be multiplied with n bits of another number. The number of bit shifting is n for each multiplication that occurs. So, for brute-force implementation of the multiplication the asymptotic time complexity can be represented as:

$$T(n) = n\theta(n) + n\theta(n)$$

$$T(n) = 2n\theta(n)$$

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

2.2 b)

For the given two numbers a and b with n bits each, we divide both into two divisions a_H, a_L and b_H, b_H where H subscript represents bits in higher order(left) and L subscript represents bits in lower order(right). The base of the numbers when converted into bits is provided 2. So, the two sub-problems can be represented as follows using the naive algorithm.

$x = a_H * 2^{n/2} + a_L$
$y = b_H * 2^{n/2} + b_L$
Now multiplying, we get,

$$x.y = a_H b_H 2^n + (a_H b_L + b_H a_L)2^{n/2} + a_L b_L$$

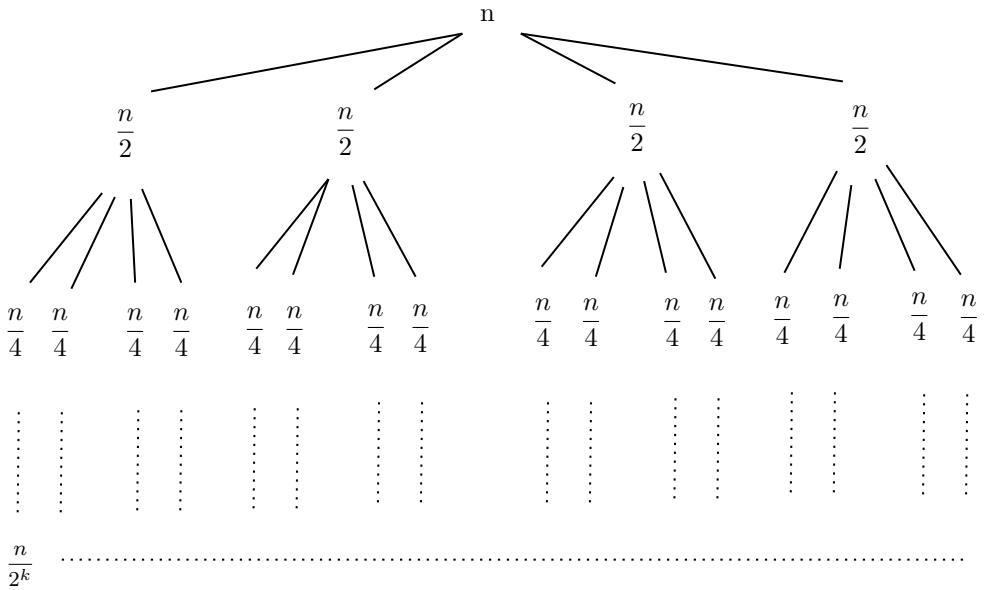
2.3 c)

This algorithm contains four multiplications for n/2 bits of numbers, 3 addition and 2 bit shifting. So, the time complexity can be represented as:

$$T(n) = 4T(n/2) + O(n)$$

2.4 d)

The recursion tree is below:



From recursion tree, we have,

For termination,

$2^k = n$

At each level the cost is changing by $2^i n$ where i ranges from 0 to k. The total cost is the summation of costs at each level.

So, the time complexity order is given by,

$$T(n) = \sum_{i=0}^{k} 2^i n$$

$$T(n) = n[1 + 2 + 2^2 + ... + 2^k]$$

$$T(n) = n\frac{1(2^k - 1)}{2 - 1}$$

$$T(n) = n * (n - 1)$$

$$T(n) = O(n^2)$$

2.5 e)

Here, $T(n) = O(n^{log_2 4}) = O(n^2)$

Using master theorem, by case 1, it gives,

$$f(n) = O(n^{log_2 4 - \epsilon}) = O(n^{2-\epsilon})$$

It is true when $\epsilon = 1$ such that $f(n)$ is polynomially smaller than $T(n)$.
Therefore,

$$T(n) = O(n^2)$$