Fibonacci numbers

Shaik Zeeshan Ali
AI20MTECH11001

***Abstract*—This document depicts a way to setup a matrix equation to find the fibonacci sequence.**

Download all python codes from

```
https://github.com/Zeeshan−IITH/IITH−EE5609/
    new/master/codes
```

and latex-tikz codes from

```
https://github.com/Zeeshan−IITH/IITH−EE5609
```

1 Problem

Given a kxk matrix $\mathbf{A}$,find the powers of $\mathbf{A}^n$ within $O(logn)$ time.

2 Construction

For the sake of simplicity we will be calculating the powers given $n = 2^m$, where n is much larger than k.The required result will be of the form $\mathbf{A}^1, \mathbf{A}^2, \mathbf{A}^4, \mathbf{A}^8, \mathbf{A}^{16}$..

The first matrix multiplication will be

$$\mathbf{A}^2 = \mathbf{A}\mathbf{A} \tag{2.0.1}$$

Instead of using repeated multiplication by $\mathbf{A}$,we can use the previous result and square it to be closer to the result using less complutations

$$\mathbf{A}^4 = \mathbf{A}^2\mathbf{A}^2 \tag{2.0.2}$$
$$\mathbf{A}^8 = \mathbf{A}^4\mathbf{A}^4 \tag{2.0.3}$$
$$\mathbf{A}^{16} = \mathbf{A}^8\mathbf{A}^8 \tag{2.0.4}$$

So $\mathbf{A}^{2^m}$ need only m products of the resultant matrix from the previous computation.Since $m = log_2(n)$, the result can be computed in $O(log_2 n)$ time.

As a more general case any number n can be represented as a sum of powers of 2, just like binary numbers are represented with the radix 2 i.e when we represent n in binary form we get

$$n = b_k b_{k-1} b_{k-2} ... b_0 \tag{2.0.5}$$

where

$$k = \lceil \log_2 n \rceil$$

because we require that many number of bits to represent n.

Now to calculate

$$\boldsymbol{A}^n$$

we can use

$$\boldsymbol{A}^n = A^{b_k 2^k + b_{k-1}2^{k-1} + b_{k-2}2^{k-2} ... + b_2 4 + b_1 2 + b_0} \tag{2.0.6}$$
$$= A^{b_k 2^k} A^{b_{k-1}2^{k-1}} A^{b_{k-2}2^{k-2}} ....A^{b_2 4} A^{b_1 2} A^{b_0} \tag{2.0.7}$$

Now each of the $\boldsymbol{A}^{2^i}$ can be calculated by squaring the previous $\boldsymbol{A}^{2^{i-1}}.\boldsymbol{A}^{2^k}$ calculation takes k time,the whole computation $\boldsymbol{A}^n$ takes $k+k+1 = 2k+1$ time.So the order of computing A^n is $O(k) = O(\log_2 n)$.

3 Fibonacci

Consider the special matrix which begins with fibonacci numbers in it

$$\mathbf{F} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \tag{3.0.1}$$

This matrix has the fibonaci numbers as its elements

$$\mathbf{F^2} = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix} \tag{3.0.2}$$
$$\mathbf{F^3} = \begin{pmatrix} 3 & 2 \\ 2 & 1 \end{pmatrix} \tag{3.0.3}$$

Therefore the n^{th} fibonacci number F_n is the element in the first row,first column of $\mathbf{F^{n-1}}$,where $n \geq 2$.The general form will be

$$\mathbf{F^{n-1}} = \begin{pmatrix} F_n & F_{n-1} \\ F_{n-2} & F_{n-3} \end{pmatrix} \tag{3.0.4}$$