

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 $k \times k$ matrix \mathbf{A} , find the powers of \mathbf{A}^n within $O(\log n)$ 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

$$\mathbf{A}^2 = \mathbf{A}\mathbf{A} \quad (2.0.1)$$

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 computations

$$\mathbf{A}^4 = \mathbf{A}^2\mathbf{A}^2 \quad (2.0.2)$$

$$\mathbf{A}^8 = \mathbf{A}^4\mathbf{A}^4 \quad (2.0.3)$$

$$\mathbf{A}^{16} = \mathbf{A}^8\mathbf{A}^8 \quad (2.0.4)$$

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

$$n = b_k b_{k-1} b_{k-2} \dots b_0 \quad (2.0.5)$$

where

$$k = \log_2 n$$

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

Now to calculate

\mathbf{A}^n

we can use

$$\mathbf{A}^n = b_k \mathbf{A}^{2^k} + b_{k-1} \mathbf{A}^{2^{k-1}} + b_{k-2} \mathbf{A}^{2^{k-2}} \dots + b_2 \mathbf{A}^4 + b_1 \mathbf{A}^2 + b_0 \mathbf{A} \quad (2.0.6)$$

Now each of the \mathbf{A}^{2^k} can be calculated by squaring the previous $\mathbf{A}^{2^{k-1}}$. Neglecting matrix addition as it only takes n^2 time to compute when compared to n^3 time of each multiplication, we can say that since \mathbf{A}^{2^k} calculation takes $O(\log_2 n)$ time, the whole computation \mathbf{A}^n takes $O(\log_2 n)$.

3 FIBONACCI

Consider the special matrix which begins with fibonacci numbers in it

$$\mathbf{F} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad (3.0.1)$$

This matrix has the fibonacci numbers as its elements

$$\mathbf{F}^2 = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix} \quad (3.0.2)$$

$$\mathbf{F}^3 = \begin{pmatrix} 3 & 2 \\ 2 & 1 \end{pmatrix} \quad (3.0.3)$$

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

$$\mathbf{F}^{n+1} = \begin{pmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{pmatrix} \quad (3.0.4)$$