Fibonacci numbers: $f_n = f_{n-1} + f_{n-2}$

Recursive algorithm has computing f_n has exponential running time. The following dynamic program computes f_n in $O(n)$ time:

Fibonacci(n, p) // Compute $f_n \% p$
$F[0] \leftarrow 0$
$F[1] \leftarrow 1$
for $i \leftarrow 2$ **to** n **do**
    $F[i] \leftarrow F[i-1] + F[i-2]$
return $F[n]$

Can we do better?

Fibonacci numbers: Matrix form

Define $V_n = \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix}$. Using the definition of f_n, we get,

$$V_n = \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_{n-1} \\ f_{n-2} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} V_{n-1}$$

Iterating $n-1$ times, we get,

$$V_n = A^{n-1} V_1 = A^{n-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \text{ where } A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

Use Power to compute A^{n-1} in $O(\log n)$ time. Compute V_n, and output $V_n[0]$.