从Fibonacci序列谈起

Date 2019/05/20

定义

$F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2}\ (n>1)$.

直接算F_n

迭代即可完成，耗时$O(n)$. 如何节约空间？

递归算F_n

直接按定义算运行时间为$O(\phi^n)$，指数时间！

矩阵形式

考虑 $\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^2\begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix} = \cdots = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n\begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$

令 $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} F_2 & F_1 \\ F_1 & F_0 \end{pmatrix}$ 只用求A^n即可. 幂的幂如何求？

$A^n \to A^{n/2}$ 分n为奇偶讨论即可. $O(\log n)$对数时间.

问题在于函数返回值是2×2矩阵，开销还是有点大.

奇偶形式

从A^{2m}到A^m可求 $\begin{aligned} F_{2m+1} &= F_m^2 + F_{m+1}^2 \\ F_{2m} &= (F_{m-1} + F_{m+1})F_m \end{aligned}$

Fib函数 $Fib(n) \to \begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix}$ 用$Fib(n/2)$递归

$$\begin{cases} n\text{为奇} & \begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} (2F_t + F_{t+1})F_{t+1} \\ F_t^2 + F_{t+1}^2 \end{pmatrix} \\ n\text{为偶} & \begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} F_t^2 + F_{t+1}^2 \\ (2F_t - F_{t+1})F_{t+1} \end{pmatrix} \end{cases}$$ 其中t均为$\lfloor \frac{n}{2} \rfloor$

依然保证为$O(\log n)$对数时间，但返回值只有2个数！

代码

```
std::pair<uint64_t, uint64_t> Fib(size_t n)
{
    if(n > 0)
    {
        auto PF = Fib(n/2);
        auto t0 = PF.first;
        auto t1 = PF.second;
        if(n%2 == 1)
            return {t0*t0 + t1*t1, (2*t0 + t1)*t1};
        else
            return {(2*t1 - t0)*t0, t0*t0 + t1*t1};
    }
    return {0, 1};
}
```

SUMMARY 人生之路，算法指南 @算法时空