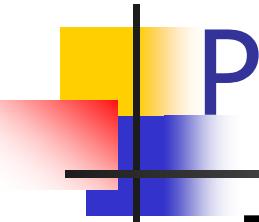


Powering Numbers & Fibonacci Revisited

Alexandre David

B2-206



Powering Numbers

- Problem: Compute a^n , where $n \in \mathbb{N}$.
- Naïve algorithm: $\Theta(n)$.
- Divide-and-conquer algorithm:

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & \text{if } n \text{ is even} \\ a^{(n-1)/2} \cdot a^{(n-1)/2} \cdot a & \text{if } n \text{ is odd} \end{cases}$$

- Recurrence:
 $T(n) = T(n/2) + \Theta(1) \Rightarrow T(n) = \Theta(\lg n)$.



Algorithm



- Iterative simple and efficient algorithm?
- Binary representation of n :

$$n = 2^0 b_0 + 2^1 b_1 + 2^2 b_2 \dots$$

- Result a^n :

$$a^n = a^{2^0 b_0} \cdot a^{2^1 b_1} \cdot a^{2^2 b_2} \dots$$

- Recurrences:

$$a^{2^n} = (a^{2^{n-1}})^2$$

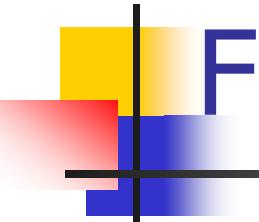
$$r_n = b_n ? a^{2^n} r_{n-1} : r_{n-1}$$

$$(r_{-1} = 1)$$



C-Implementation

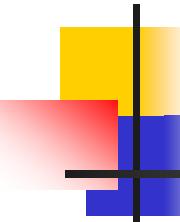
```
int power(int a, unsigned int n) {  
    int r = 1;  
    while(n != 0) {  
        if (n & 1) r *= a;  
        n >>= 1;  
        a *= a;  
    }  
    return r;  
}
```



Fibonacci Numbers

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2 \end{cases}$$

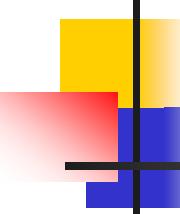
- Good algorithm?
- With good precision?



Tries

- Naïve recursive squaring: $F_n = \phi^n / \sqrt{5}$ rounded to the nearest integer.
 - $\Theta(\lg n)$ time.
 - Unreliable method because of floating-point arithmetics.
- Bottom-up computation: Compute $F_0, F_1 \dots F_n$ in order by iterative summations.
 - $\Theta(n)$ time.
- Better idea?





Recursive Squaring?

- Theorem:
$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$
- Algorithm: Recursive squaring.
Time = $\Theta(\lg n)$.

- *Proof:* Induction on n .

Base $n = 1$:

$$\begin{bmatrix} F_2 & F_1 \\ F_1 & F_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^1$$

AA1



Proof - Inductive Step

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

C-Implementation

```
int fibonacci(unsigned int n) {  
    int a[4] = {1,1,1,0};  
    int r[4] = {1,0,0,1};  
    while(n != 0) {  
        if(n & 1) mat2_mul(r, a);  
        mat2_mul(r, r);  
        n = n / 2;  
    }  
    return r[1];  
}
```

Problem:
Overflow (int) at n = 47.
Overflow (long long) at n = 93.