CS 580
ALGORITHM DESIGN AND ANALYSIS

Section 1:
Intro – Fibonacci numbers

Vassilis Zikas

LEONARDO FIBONACCI

FIBONACCI NUMBERS

- 0,1,1,2,3,5,8,13,21,…
- $F_n = \begin{cases} F_{n-1} + F_{n-2}, & if\ n > 1 \\ 1, & if\ n = 1 \\ 0, & if\ n = 0 \end{cases}$
- Very easy to see that the grown is exponential
  - e.g. it's easy to see that $F_n \geq 2^{0.5n}$ (Why???)
- But, what about precise values?
- How can we compute F_{200}??
- **Algorithms**!!

FIBONACCI NUMBERS: FIRST ATTEMPT

- First idea for an algorithm: recursion

```
def fib1(n):
    if n = 0:
        return 0
    if n = 1:
        return 1
    return fib1(n - 1) + fib1(n - 2)
```

- Is this algorithm correct?
  - Yes, it just implements the definition

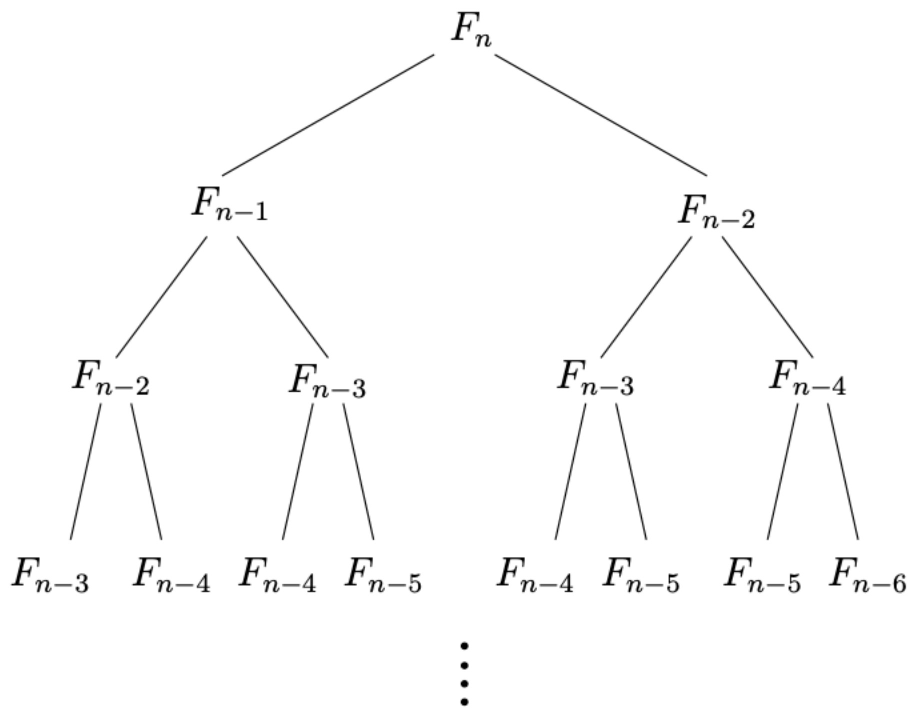
FIBONACCI NUMBERS: FIRST ATTEMPT

- How much time does it take?
- Let $T(n)$ be the number of steps.
- For $n > 2$ we have
  - $T(n) = T(n-1) + T(n-2) + 3$
  - Check the value of n and the addition
- This is bigger than F_n!!
- So $T(n) \geq F_n > 2^{0.5n}$

FIBONACCI NUMBERS: FIRST ATTEMPT

- Where did we go wrong?
- In order to compute $fib1(100)$ we call $fib1(99)$ and $fib1(98)$
- $fib1(99)$ calls $fib1(98)$ and $fib1(97)$
- Therefore, $fib1(98)$ is computed twice!

FIBONACCI NUMBERS: FIRST ATTEMPT



FIBONACCI NUMBERS: SECOND ATTEMPT

def $fib2(n)$:

if $n = 0$: return 0

create array $f[0, \ldots, n]$

$f[0] = 0, f[1] = 1$

for $i = 2, \ldots, n$:

$f[i] = f[i-1] + f[i-2]$

return $f[n]$

- Correct?
  - Yes, literally the definition of F_n

FIBONACCI NUMBERS: SECOND ATTEMPT

- Time?
- A simple loop
  - Accessing an element of an array is free
  - Plus one addition per loop
- Linear time!
- Well, perhaps not quite…

FIBONACCI NUMBERS: SECOND ATTEMPT

- When computing $f[i]$ we add $f[i-1]$ and $f[i-2]$
- These could be huge!
- F_n is exponential in n, and therefore we need linear in n bits to write it down
- Addition could not possibly take a constant number of steps!
- On the other hand, it's not terrible
    - Adding two n-bit numbers takes linear time
- Overall, $fib2(n)$ then takes $O(n^2)$ steps
    - Huge improvement!

FIBONACCI NUMBERS: THIRD ATTEMPT

- Can we do better?
- Write recursion in matrix notation

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

- $F_1 = F_1, F_2 = F_1 + F_0$

$$\begin{pmatrix} F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^2 \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

FIBONACCI NUMBERS: THIRD ATTEMPT

- General
$$\begin{pmatrix} F_{n-1} \\ F_n \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_{n-2} \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$
- Computing F_n is the same as raising a 2 by 2 matrix to the n-th power
- In order to compute A^n we roughly need $\log(n)$ matrix multiplications

FIBONACCI NUMBERS: THIRD ATTEMPT

- $A^n = A^{\frac{n}{2}} * A^{\frac{n}{2}}$
- Need one matrix multiplication to go from $A^{\frac{n}{2}}$ to A^n
- And so on
- Need $O(\log(n))$ matrix multiplications to compute A^n
- How fast is that?
- Need to compute

$$\begin{pmatrix} n-bit\ \# & n-bit\ \# \\ n-bit\ \# & n-bit\ \# \end{pmatrix} \begin{pmatrix} n-bit\ \# & n-bit\ \# \\ n-bit\ \# & n-bit\ \# \end{pmatrix}$$

FIBONACCI NUMBERS: THIRD ATTEMPT

- Naively this takes 8 multiplications and 4 additions.
- Overall, if number multiplication needs $M(n)$ operations, computing $F(n)$ would take $O(M(n)\log(n))$ operations
- Can we do better?
- Let's improve this analysis a bit…
- Claim: We can compute $F(n)$ in $O(M(n))$ operations
- Intuition: We only need to multiply n-bit numbers in the last step. In the second to last step we need to multiply $n/2$-bit numbers, and so on.
  - $M(1) + M(2) + M(4) + \cdots = \sum_{i=1}^{\log(n)} M(2^i)$
  - Notice that $M(2i) > 2M(i)$
  - Therefore running time is $O(M(n))$

FIBONACCI NUMBERS: THIRD ATTEMPT

- Naively, multiplication of n bit numbers takes $O(n^2)$ operations.
- So, no progress…
- As we'll see in a few weeks we can multiply numbers much faster than n^2!

FIBONACCI NUMBERS: FOURTH ATTEMPT

- Closed form:

$$F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

- Catch: these numbers are irrational!