

Fibonacci
(Leonardo Pisano)
1170-1240?
Statue in Pisa Italy

FIBONACCI NUMBERS
GOLDEN RATIO,
RECURRENCES



Fibonacci function

2

$$\text{fib}(0) = 0$$

$$\text{fib}(1) = 1$$

$$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2) \quad \text{for } n \geq 2$$

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

In his book in 1200
titled *Liber Abaci*

*Has nothing to do with the
famous pianist Liberaci*

But sequence described
much earlier in India:

Virahaṅka 600–800

Gopala before 1135

Hemacandra about 1150

The so-called Fibonacci
numbers in ancient and
medieval India.

Parmanad Singh, 1985
pdf on course website

Fibonacci function (year 1202)

3

$\text{fib}(0) = 0$

$\text{fib}(1) = 1$

$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$ for $n \geq 2$

*/** Return fib(n). Precondition: $n \geq 0$.*/*

```
public static int f(int n) {  
    if ( n <= 1) return n;  
    return f(n-1) + f(n-2);  
}
```

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

We'll see that this is a
lousy way to compute
 $f(n)$

Golden ratio $\Phi = (1 + \sqrt{5})/2 = 1.61803398\dots$

4

Find the golden ratio when we divide a line into two parts such that

$$\text{whole length} / \text{long part} == \text{long part} / \text{short part}$$

Call long part **a** and short part **b**



$$(a + b) / a = a / b$$

Solution is called Φ

See webpage:

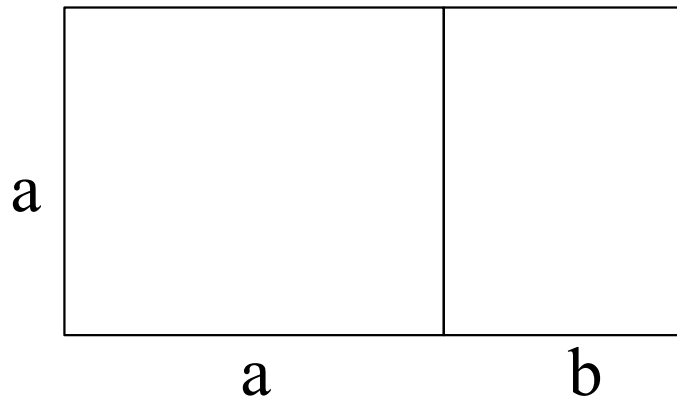
<http://www.mathsisfun.com/numbers/golden-ratio.html>

Golden ratio $\Phi = (1 + \sqrt{5})/2 = 1.61803398\dots$

5

Find the golden ratio when we divide a line into two parts a and b such that

$$(a + b) / a = a / b = \Phi$$



Golden rectangle

See webpage:

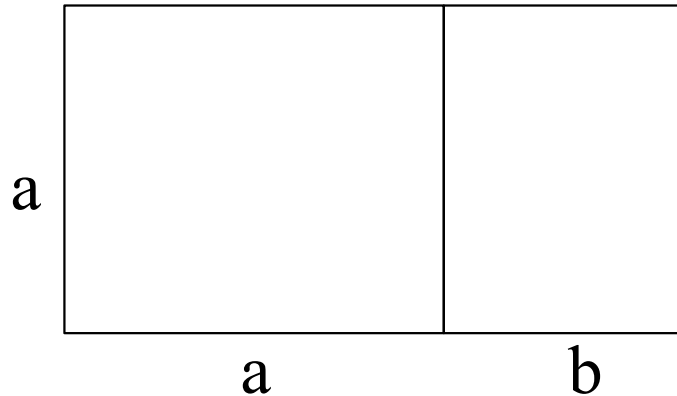
<http://www.mathsisfun.com/numbers/golden-ratio.html>

Golden ratio $\Phi = (1 + \sqrt{5})/2 = 1.61803398\dots$

6

Find the golden ratio when we divide a line into two parts a and b such that

$$(a + b) / a = a / b = \Phi$$



Golden
rectangle

a/b

$$8/5 = 1.6$$

$$13/8 = 1.625\dots$$

$$21/13 = 1.615\dots$$

$$34/21 = 1.619\dots$$

$$55/34 = 1.617\dots$$

For successive Fibonacci numbers a, b, a/b is close to Φ but not quite it Φ . 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Find $\text{fib}(n)$ from $\text{fib}(n-1)$

7

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Since $\text{fib}(n) / \text{fib}(n-1)$ is close to the golden ratio,

You can see that $(\text{golden ratio}) * \text{fib}(n-1)$ is close to $\text{fib}(n)$

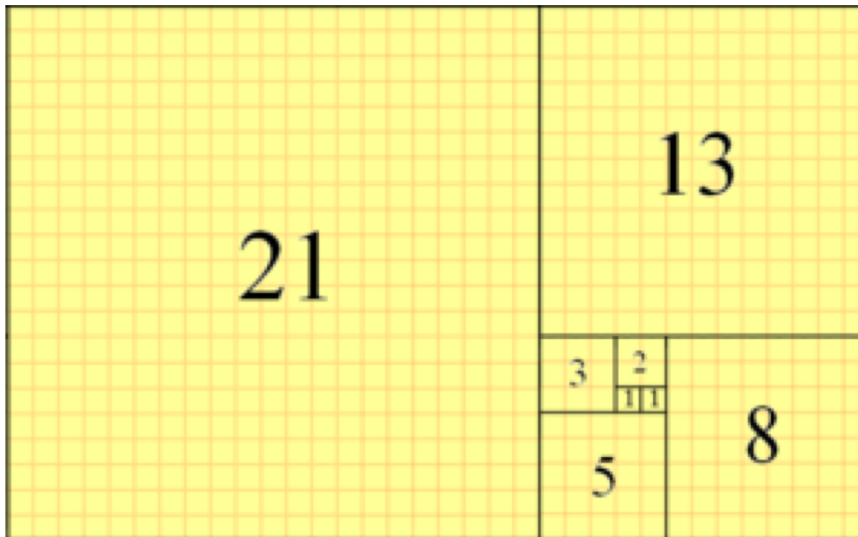
We can actually use this formula to calculate $\text{fib}(n)$
From $\text{fib}(n-1)$

Golden ratio and Fibonacci numbers: inextricably linked

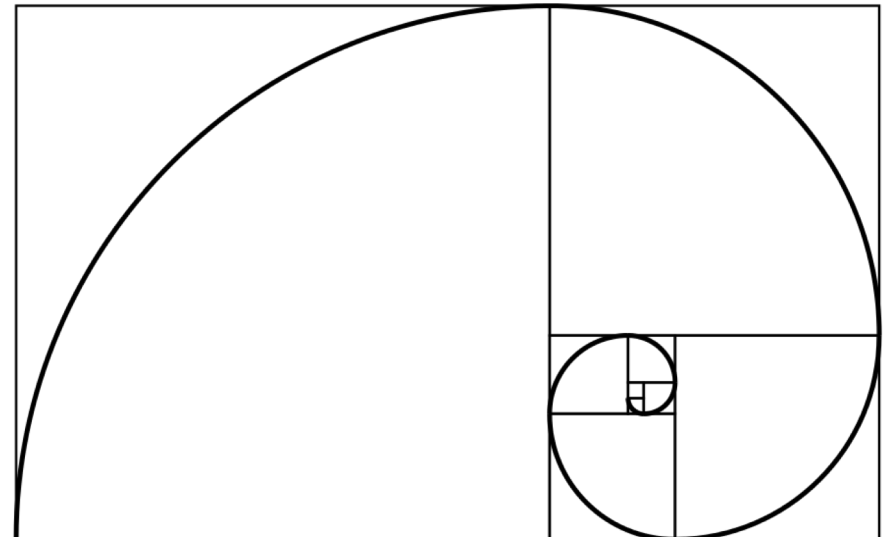
Fibonacci function (year 1202)

8

Downloaded from wikipedia



Fibonacci tiling



Fibonacci spiral

0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ...

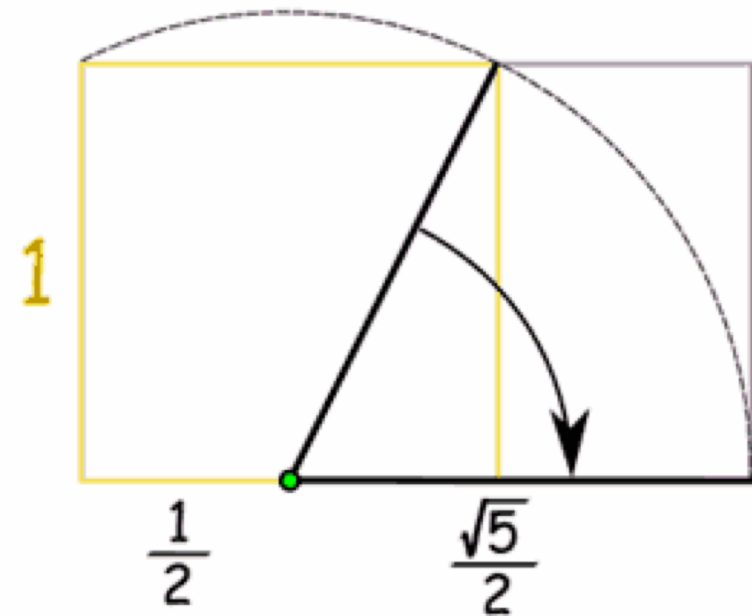
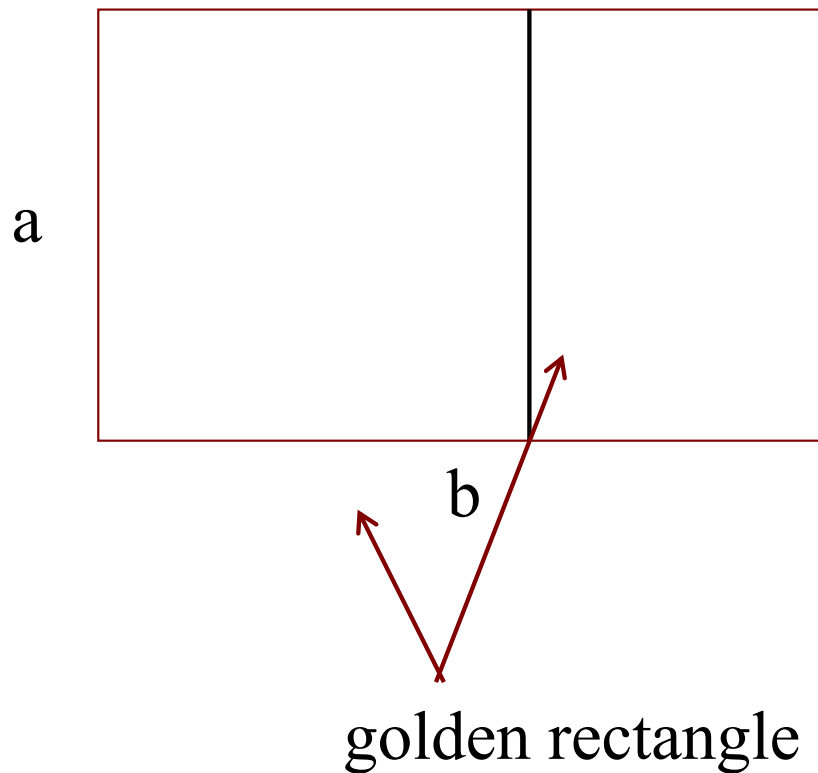
The Parthenon

9



The golden ratio

10



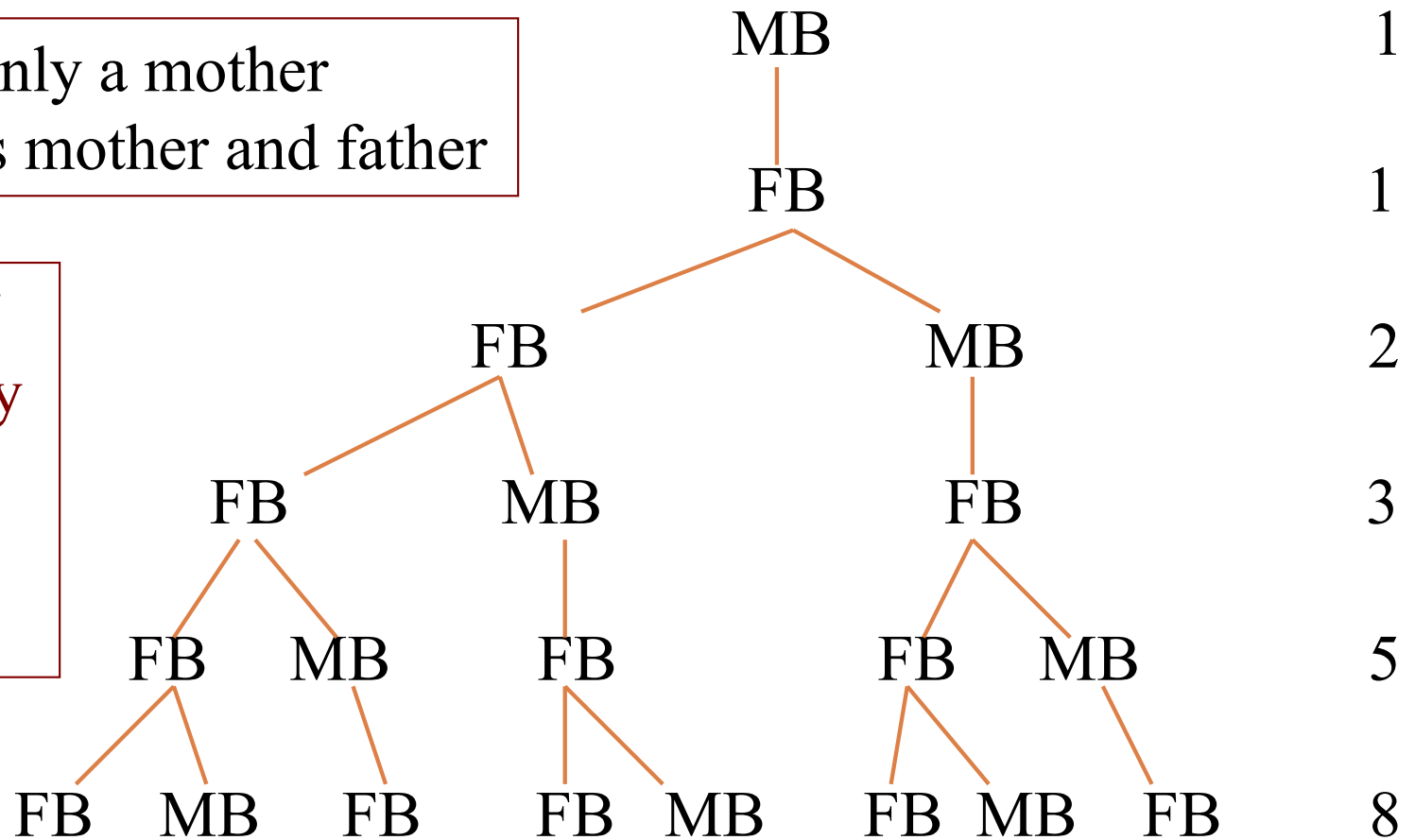
How to draw a golden rectangle

fibonacci and bees

11

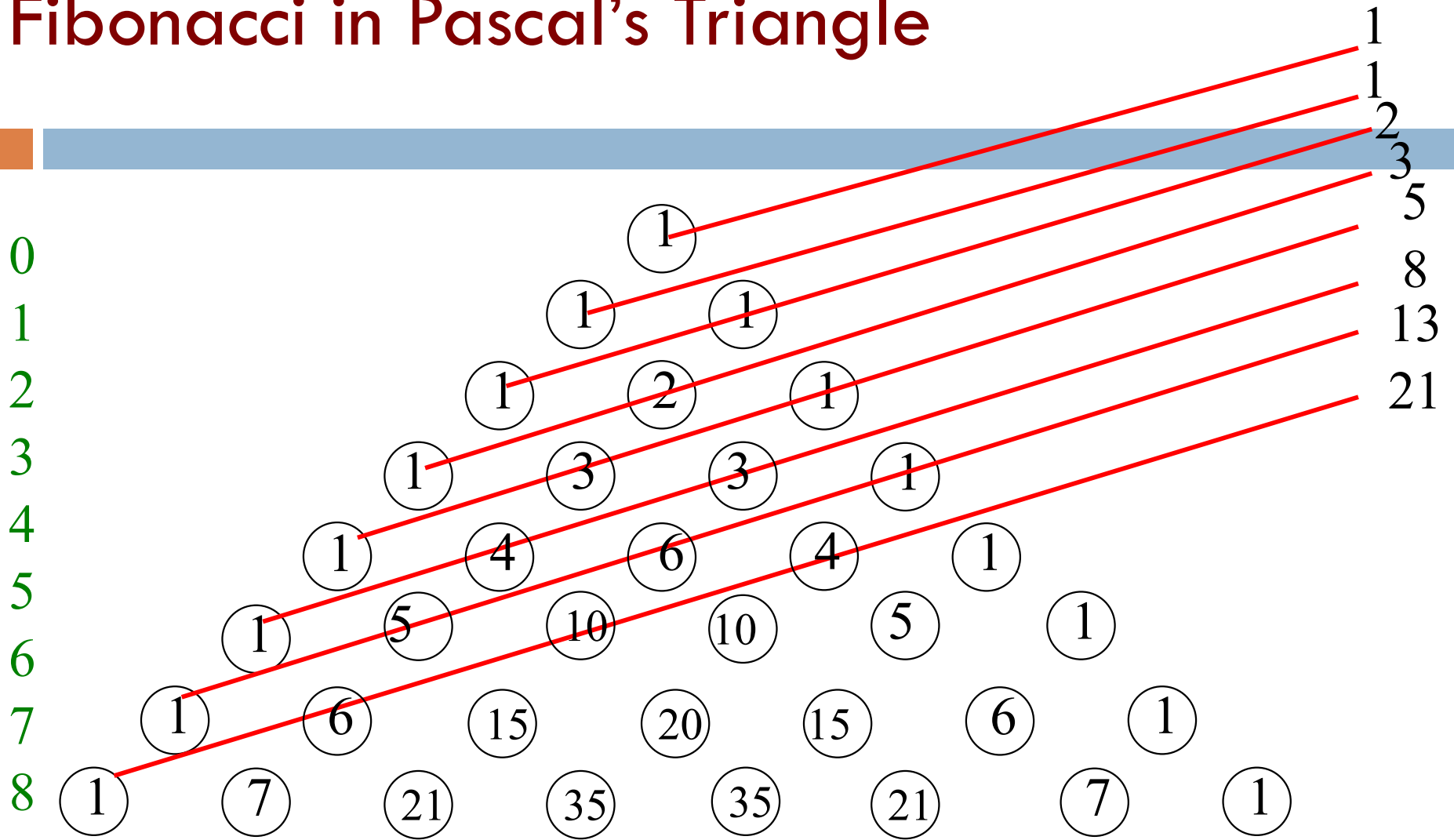
Male bee has only a mother
Female bee has mother and father

The number of
ancestors at any
level is a
Fibonacci
number



MB: male bee, FB: female bee

Fibonacci in Pascal's Triangle



$p[i][j]$ is the number of ways i elements can be chosen from a set of size j

Suppose you are a plant

13

You want to grow your leaves so that they all get a good amount of sunlight. You decide to grow them at successive angles of 180 degrees



Pretty stupid plant!

The two bottom leaves get VERY little sunlight!

Suppose you are a plant

14

You want to grow your leaves so that they all get a good amount of sunlight. 90 degrees, maybe?



**Where does the
fifth leaf go?**

Fibonacci in nature

15 The artichoke uses the Fibonacci pattern to spiral the sprouts of its flowers.

$$360/(\text{golden ratio}) = 222.492$$



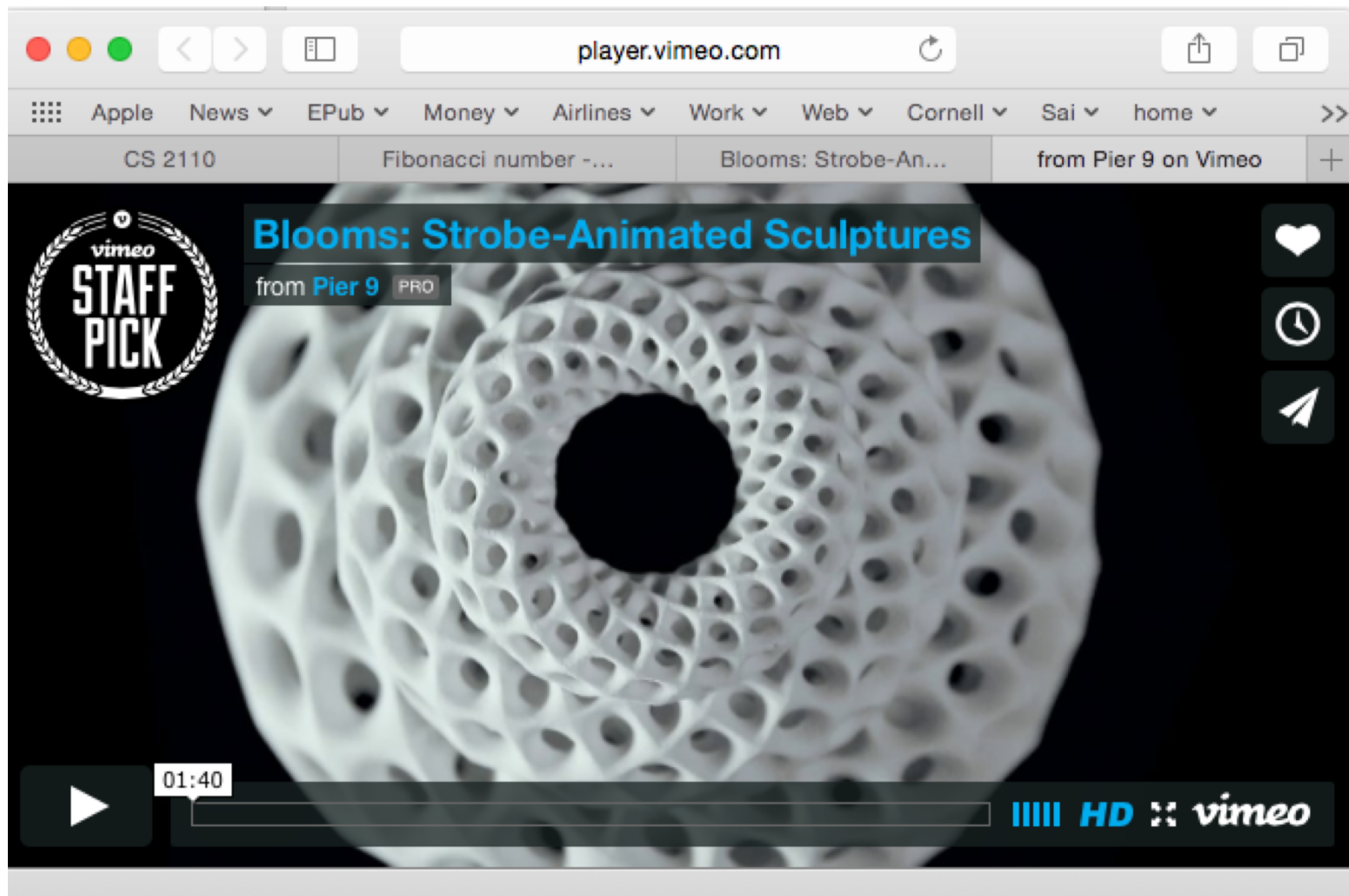
The artichoke sprouts its leaves at a constant amount of rotation: 222.5 degrees (in other words the distance between one leaf and the next is 222.5 degrees).

topones.weebly.com/1/post/2012/10/the-artichoke-and-fibonacci.html

Blooms: strobe-animated sculptures

16

www.instructables.com/id/Blooming-Zoetrope-Sculptures/



Uses of Fibonacci sequence in CS

17

Fibonacci search

Fibonacci heap data structure

Fibonacci cubes: graphs used for interconnecting parallel and distributed systems

Fibonacci search of sorted $b[0..n-1]$

18

binary search:
cut in half at each step

Fibonacci search: ($n = 144$)
cut by Fibonacci numbers

0 e1 n

0 e1 144

$$e1 = (n-0)/2$$

$$e1 = 0 + 89$$

0 e2 e1

0 e2 e1

$$e2 = (e1-0)/2$$

$$e2 = 0 + 55$$

e2 e1

e2 e1

2 3 5 8 13 21 34 55 89 144

Fibonacci search history

David Ferguson. Fibonaccian searching. Communications of the ACM, 3(12) 1960: 648

Wiki: Fibonacci search divides the array into two parts that have sizes that are consecutive Fibonacci numbers. On average, this leads to about 4% more comparisons to be executed, but only one addition and subtraction is needed to calculate the indices of the accessed array elements, while classical binary search needs bit-shift, division or multiplication.

If the data is stored on a magnetic tape where seek time depends on the current head position, a tradeoff between longer seek time and more comparisons may lead to a search algorithm that is skewed similarly to Fibonacci search.

Fibonacci search

20

David Ferguson.

Fibonacci searching.

This flowchart is how Ferguson describes the algorithm in this 1-page paper. There is some English verbiage but no code.

Only high-level language available at the time: Fortran.

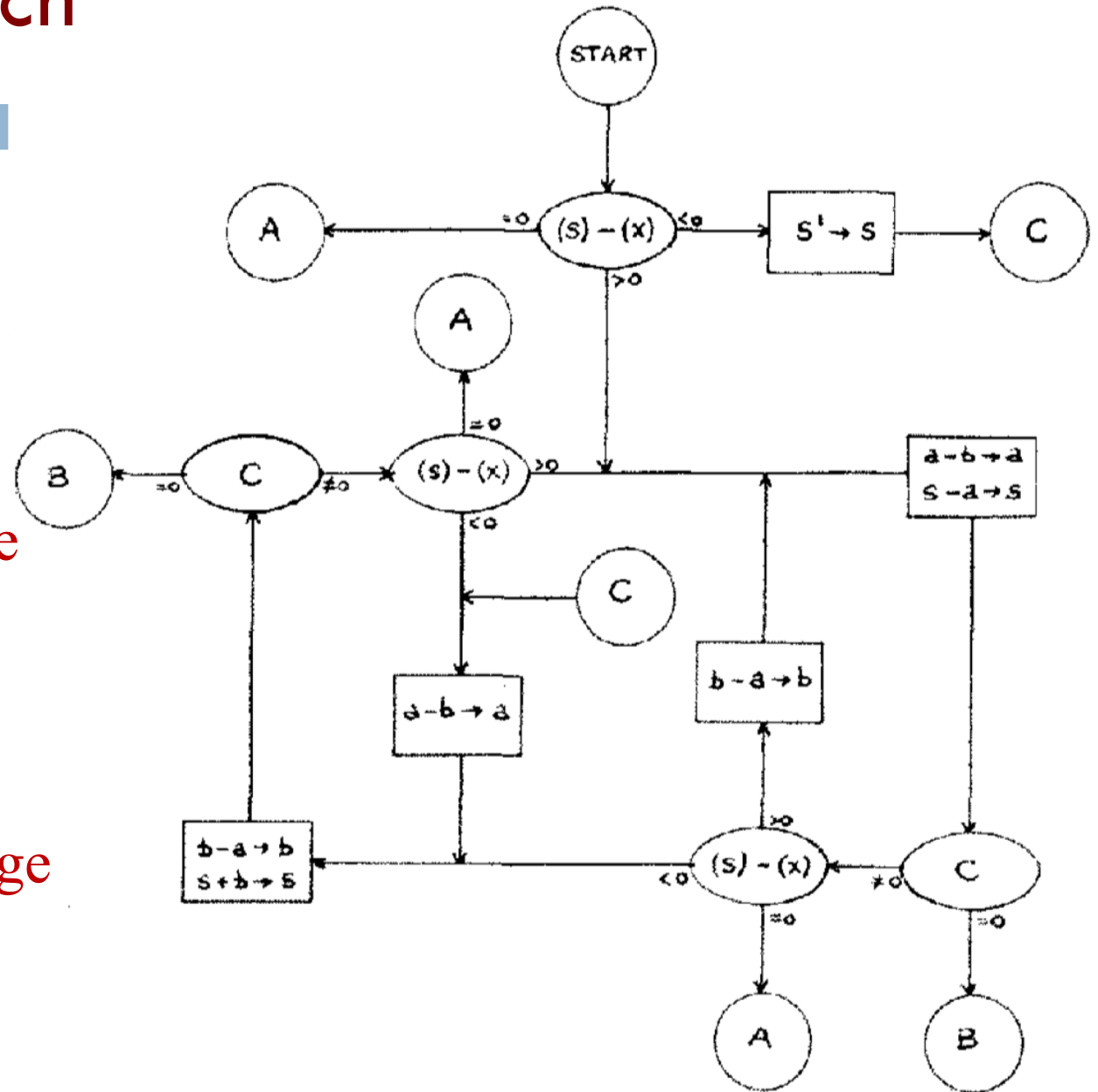


FIG. 1

LOUSY WAY TO COMPUTE: $O(2^n)$

21

```
/** Return fib(n). Precondition:  $n \geq 0$ .*/
```

```
public static int f(int n) {  
    if ( n <= 1) return n;  
    return f(n-1) + f(n-2);  
}
```

Calculates $f(15)$ 8 times!
What is complexity of $f(n)$?

20

19

18

18

17

17

16

17 16 16 15

16 15 15 14

Recursion for fib: $f(n) = f(n-1) + f(n-2)$

22

$$T(0) = a$$

$T(n)$: Time to calculate $f(n)$

$$T(1) = a$$

Just a recursive function

$$T(n) = a + T(n-1) + T(n-2)$$

“recurrence relation”

We can prove that $T(n)$ is $O(2^n)$

It's a “proof by induction”.

Proof by induction is not covered in this course.

But we can give you an idea about why $T(n)$ is $O(2^n)$

$$T(n) \leq c \cdot 2^n \quad \text{for } n \geq N$$

Recursion for fib: $f(n) = f(n-1) + f(n-2)$

23

$$T(0) = a$$

$$T(1) = a$$

$$T(n) = a + T(n-1) + T(n-2)$$

$$T(0) = a \leq a * 2^0$$

$$T(1) = a \leq a * 2^1$$

$$T(n) \leq c * 2^n \text{ for } n \geq N$$

$$\begin{aligned} & T(2) \\ = & \text{<Definition>} \\ & a + T(1) + T(0) \\ \leq & \text{<look to the left>} \\ & a + a * 2^1 + a * 2^0 \\ = & \text{<arithmetic>} \\ & a * (4) \\ = & \text{<arithmetic>} \\ & a * 2^2 \end{aligned}$$

Recursion for fib: $f(n) = f(n-1) + f(n-2)$

24

$$T(0) = a$$

$$T(1) = a$$

$$T(n) = T(n-1) + T(n-2)$$

$$T(0) = a \leq a * 2^0$$

$$T(1) = a \leq a * 2^1$$

$$T(2) = 2a \leq a * 2^2$$

$$T(n) \leq c * 2^n \text{ for } n \geq N$$

$$T(3)$$

$$= \text{<Definition>}$$

$$a + T(2) + T(1)$$

$$\leq \text{<look to the left>}$$

$$a + a * 2^2 + a * 2^1$$

$$= \text{<arithmetic>}$$

$$a * (7)$$

$$\leq \text{<arithmetic>}$$

$$a * 2^3$$

Recursion for fib: $f(n) = f(n-1) + f(n-2)$

25

$$T(0) = a$$

$$T(1) = a$$

$$T(n) = T(n-1) + T(n-2)$$

$$T(0) = a \leq a * 2^0$$

$$T(1) = a \leq a * 2^1$$

$$T(2) \leq a * 2^2$$

$$T(3) \leq a * 2^3$$

$$T(n) \leq c * 2^n \text{ for } n \geq N$$

$$T(4)$$

$$= \text{<Definition>}$$

$$a + T(3) + T(2)$$

$$\leq \text{<look to the left>}$$

$$= a + a * 2^3 + a * 2^2$$

<arithmetic>

$$a * (13)$$

$$\leq \text{<arithmetic>}$$

$$a * 2^4$$

Recursion for fib: $f(n) = f(n-1) + f(n-2)$

26

$$T(0) = a$$

$$T(1) = a$$

$$T(n) = T(n-1) + T(n-2)$$

$$T(0) = a \leq a * 2^0$$

$$T(1) = a \leq a * 2^1$$

$$T(2) \leq a * 2^2$$

$$T(3) \leq a * 2^3$$

$$T(4) \leq a * 2^4$$

$$T(n) \leq c * 2^n \text{ for } n \geq N$$

$$T(5)$$

$$= \text{<Definition>}$$

$$a + T(4) + T(3)$$

$$\leq \text{<look to the left>}$$

$$= a + a * 2^4 + a * 2^3$$

<arithmetic>

$$a * (25)$$

$$\leq \text{<arithmetic>}$$

$$a * 2^5$$

WE CAN GO ON FOREVER LIKE THIS

Recursion for fib: $f(n) = f(n-1) + f(n-2)$

27

$$T(0) = a$$

$$T(1) = a$$

$$T(n) = T(n-1) + T(n-2)$$

$$T(0) = a \leq a * 2^0$$

$$T(1) = a \leq a * 2^1$$

$$T(2) \leq a * 2^2$$

$$T(3) \leq a * 2^3$$

$$T(4) \leq a * 2^4$$

$$T(n) \leq c * 2^n \text{ for } n \geq N$$

$$T(k)$$

$$= \text{<Definition>}$$

$$a + T(k-1) + T(k-2)$$

$$\leq \text{<look to the left>}$$

$$a + a * 2^{k-1} + a * 2^{k-2}$$

$$= \text{<arithmetic>}$$

$$a * (1 + 2^{k-1} + 2^{k-2})$$

$$\leq \text{<arithmetic>}$$

$$a * 2^k$$

Caching

As values of $f(n)$ are calculated, save them in an ArrayList.
Call it a **cache**.

When asked to calculate $f(n)$ see if it is in the cache.
If yes, just return the cached value.
If no, calculate $f(n)$, add it to the cache, and return it.

Must be done in such a way that if $f(n)$ is about to be cached, $f(0)$, $f(1)$, ... $f(n-1)$ are already cached.

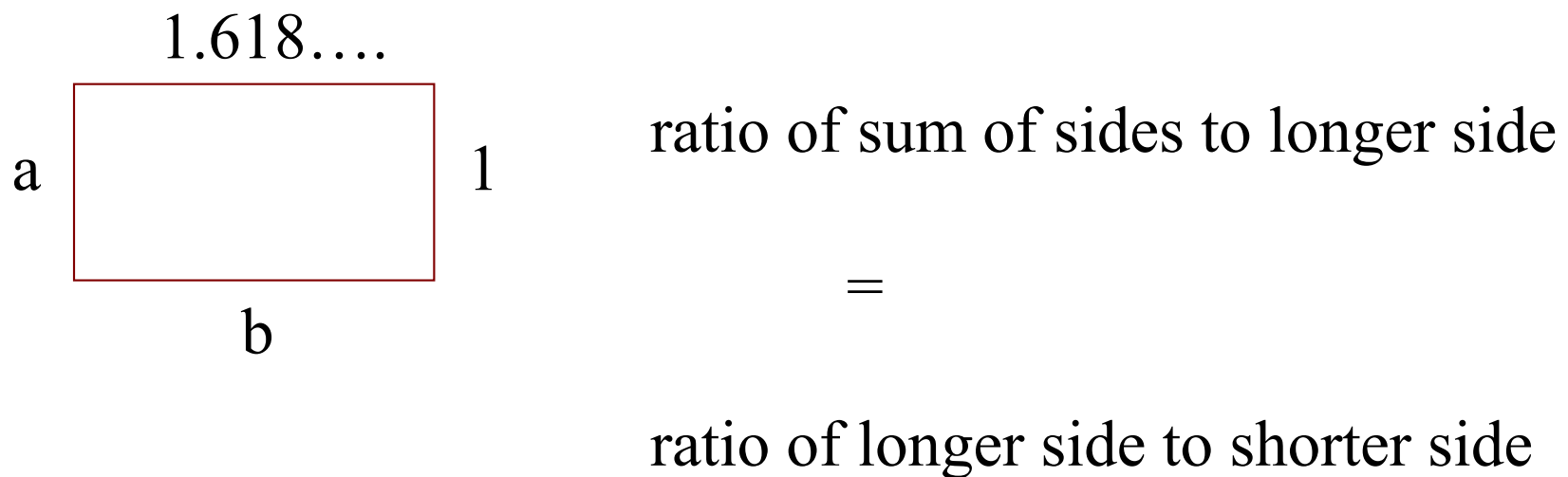
The golden ratio

29

$a > 0$ and $b > a > 0$ are in the **golden ratio** if

$$(a + b) / b = b/a \quad \text{call that value } \varphi$$

$$\varphi^2 = \varphi + 1 \quad \text{so } \varphi = (1 + \sqrt{5}) / 2 = 1.618 \dots$$



Can prove that Fibonacci recurrence is $O(\varphi^n)$

30

We won't prove it.

Requires proof by induction

Relies on identity $\varphi^2 = \varphi + 1$

Linear algorithm to calculate fib(n)

31

```
/** Return fib(n), for n >= 0. */
public static int f(int n) {
    if (n <= 1) return 1;
    int p= 0;  int c= 1;  int i= 2;
    // invariant: p = fib(i-2) and c = fib(i-1)
    while (i < n) {
        int fibi= c + p;  p= c;  c= fibi;
        i= i+1;
    }
    return c + p;
}
```

Logarithmic algorithm!

32

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_{n+2} &= f_{n+1} + f_n \end{aligned} \quad \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} f_n \\ f_{n+1} \end{pmatrix} = \begin{pmatrix} f_{n+1} \\ f_{n+2} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} f_n \\ f_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} f_{n+1} \\ f_{n+2} \end{pmatrix} = \begin{pmatrix} f_{n+2} \\ f_{n+3} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^k \begin{pmatrix} f_n \\ f_{n+1} \end{pmatrix} = \begin{pmatrix} f_{n+k} \\ f_{n+k+1} \end{pmatrix}$$

Logarithmic algorithm!

33

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_{n+2} &= f_{n+1} + f_n \end{aligned} \quad \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^k \begin{pmatrix} f_n \\ f_{n+1} \end{pmatrix} = \begin{pmatrix} f_{n+k} \\ f_{n+k+1} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^k \begin{pmatrix} f_0 \\ f_1 \end{pmatrix} = \begin{pmatrix} f_k \\ f_{k+1} \end{pmatrix}$$

You know a logarithmic algorithm for exponentiation—recursive and iterative versions

Gries and Levin
Computing a Fibonacci number in log time.
IPL 2 (October 1980), 68-69.

Another log algorithm!

34

Define $\phi = (1 + \sqrt{5}) / 2$ $\phi' = (1 - \sqrt{5}) / 2$

The golden ratio again.

Prove by induction on n that

$$f_n = (\phi^n - \phi'^n) / \sqrt{5}$$