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

FIBONACCI NUMBERS GOLDEN RATIO, RECURRENCES

Lecture 23
CS2110 – Fall 2016

Fibonacci function

2

fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2) for $n \geq 2$

0, 1, 1, 2, 3, 5, 8, 13, 21, …

In his book in 120
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

fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2) for $n \geq 2$

```
/** Return fib(n). Precondition: n ≥ 0.*/
public static int f(int n) {
   if ( n <= 1) return n;
   return f(n-1) + f(n-2);
}
```

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

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

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

4

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

whole length / long part == long part / short part

Call long part a and short part b

(a + b) / a = a / b   Solution is called Φ

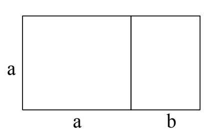
See webpage:
http://www.mathsisfun.com/numbers/golden-ratio.html

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

5

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

(a + b) / a = a / b   = Φ



Golden rectangle

See webpage:
http://www.mathsisfun.com/numbers/golden-ratio.html

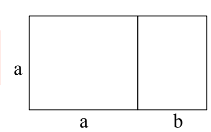
Golden ratio $\Phi = (1 + \sqrt{5})/2 = 1.61803398\cdots$

6

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

(a + b) / a = a / b   = Φ

Golden rectangle



a/b
8/5 = 1.6
13/8 = 1.625⋯
21/13= 1.615⋯
34/21 = 1.619⋯
55/34 = 1.617⋯

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 fib(n) from fib(n-1)

7

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

Since fib(n) / fib(n-1) is close to the golden ratio,

You can (golden ratio) * fib(n-1) is close to fib(n)

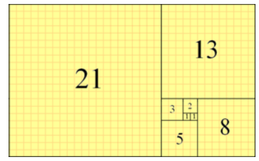
We can actually use this formula to calculate fib(n)
From fib(n-1)

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

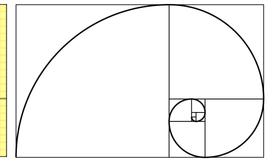
Fibonacci function (year 1202)

8

Downloaded from wikipedia



Fibonacci tiling



Fibonacci spiral

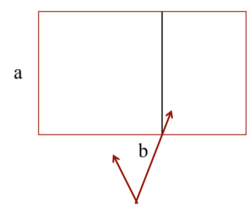
0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ⋯

The Parthenon

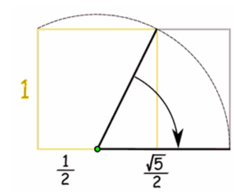
9

The golden ratio

10



golden rectangle



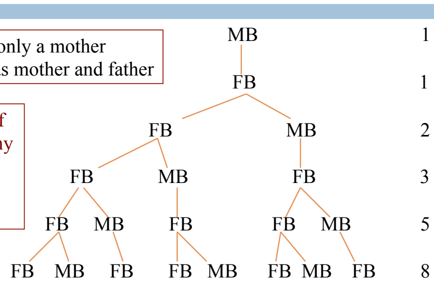
How to draw a golden rectangle

fibonacci and bees

11

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

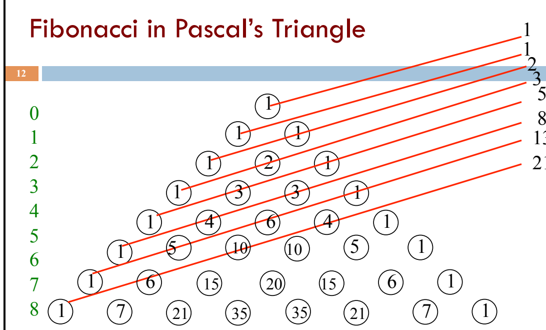
The number of ancestors at any level is a Fibonnaci number



MB: male bee, FB: female bee

Fibonacci in Pascal's Triangle

12



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

Fibonacci in nature

13

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

360/(golden ratio) = 222.492

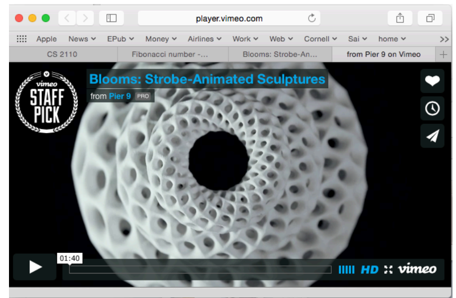
The artichoke sprouts its leafs at a constant amount of rotation: 222.5 degrees (in other words the distance between one leaf and the next is 222.5 degrees).

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

Blooms: strobe-animated sculptures

14

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



Uses of Fibonacci sequence in CS

15

Fibonacci search

Fibonacci heap data strcture

Fibonacci cubes: graphs used for interconnecting parallel and distributed systems

LOUSY WAY TO COMPUTE: O(2^n)

16

```
/** Return fib(n). Precondition: n ≥ 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)

17

T(0) = a

T(1) = a

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

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

Just a recursive function

"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 $O(2^n)$

$T(n) <= c*2^n$ for $n >= N$

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

18

T(0) = a

T(1) = a

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

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

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

$T(n) <= c*2^n$ for $n >= N$

T(2)
= <Definition>
a + T(1) + T(0)
≤ <look to the left>
$a + a * 2^1 + a * 2^0$
= <arithmetic>
a * (4)
= <arithmetic>
$a * 2^2$

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

19

T(0) = a
T(1) = a
T(n) = T(n-1) + T(n-2)

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

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

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

$T(n) <= c*2^n$ for $n >= N$

T(3)
= <Definition>
a + T(2) + T(1)
≤ <look to the left>
$a + a * 2^2 + a * 2^1$
= <arithmetic>
a * (7)
≤ <arithmetic>
$a * 2^3$

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

20

T(0) = a
T(1) = a
T(n) = T(n-1) + T(n-2)

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

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

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

$T(3) = a \le a * 2^3$

$T(n) <= c*2^n$ for $n >= N$

T(4)
= <Definition>
a + T(3) + T(2)
≤ <look to the left>
$a + a * 2^3 + a * 2^2$
= <arithmetic>
a * (13)
≤ <arithmetic>
$a * 2^4$

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

21

T(0) = a
T(1) = a
T(n) = T(n-1) + T(n-2)
$T(0) = a \le a * 2^0$
$T(1) = a \le a * 2^1$
$T(2) = a \le a * 2^2$
$T(3) = a \le a * 2^3$
$T(4) = a \le a * 2^4$

$T(n) <= c*2^n$ for $n >= N$

T(5)
= <Definition>
a + T(4) + T(3)
≤ <look to the left>
$a + a * 2^4 + a * 2^3$
= <arithmetic>
a * (25)
≤ <arithmetic>
$a * 2^5$

WE CAN GO ON FOREVER LIKE THIS

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

22

T(0) = a
T(1) = a
T(n) = T(n-1) + T(n-2)
$T(0) = a \le a * 2^0$
$T(1) = a \le a * 2^1$
$T(2) = a \le a * 2^2$
$T(3) = a \le a * 2^3$
$T(4) = a \le a * 2^4$

$T(n) <= c*2^n$ for $n >= N$

T(k)
= <Definition>
a + T(k-1) + T(k-2)
≤ <look to the left>
$a + a * 2^{k-1} + a * 2^{k-2}$
= <arithmetic>
$a * (1 + 2^{k-1} + 2^{k-2})$
≤ <arithmetic>
$a * 2^k$

Caching

23

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.

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)
$T(0) = a \le a * 2^0$
$T(1) = a \le a * 2^1$
$T(2) = a \le a * 2^2$
$T(3) = a \le a * 2^3$
$T(4) = a \le a * 2^4$
$T(5) = a \le a * 2^5$

$T(n) <= c*2^n$ for $n >= N$

Need a formal proof, somewhere.
Uses mathematical induction

"Theorem": all odd integers > 2 are prime

3, 5, 7 are primes? yes
9? experimental error
11, 13? Yes.
That's enough checking

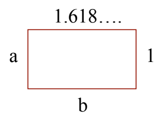
The golden ratio

25

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

$(a + b) / b = b/a$ call that value φ

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



ratio of sum of sides to longer side

=

ratio of longer side to shorter side

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

26

We won't prove it.

Requires proof by induction

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

Linear algorithm to calculate fib(n)

27

```
/** 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!

28

$f_0 = 0$

$f_1 = 1$

$f_{n+2} = f_{n+1} + f_n$

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

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

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

Logarithmic algorithm!

29

$f_0 = 0$

$f_1 = 1$

$f_{n+2} = f_{n+1} + f_n$

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

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

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!

30

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

The golden ratio again.

Prove by induction on n that

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