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

# FIBONACCI NUMBERS GOLDEN RATIO, RECURRENCES

Lecture 26
CS2110 – Spring 2019

## Announcements

A7: NO LATE DAYS. No need to put in time and comments. We have to grade quickly. No regrade requests for A7. Grade based only on your score on a bunch of sewer systems.

Please check submission guidelines carefully. Every mistake you make in submitting A7 slows down grading of A7 and consequent delay of publishing tentative course grades.

Sewer system generated from seed -3026730162232494481 has no coins!

All regrade requests have to be in tonight.

## Announcements

Final is optional! As soon as we grade A7 and get it into the CMS, we determine tentative course grades.

Complete "assignment" Accept course grade? on the CMS by Wednesday night.

If you accept it, that IS your grade. It won't change.

Don't accept it? Take final. Can lower and well as raise grade.

More past finals are on Exams page of course website. Not all answers. We'll put last semester's on.

## Announcements

We try to make final fair.

Our experience:
For majority of students, it doesn't affect their grade.
More raise their grade than lower their grade.

| One semester: | One semester |
|---|---|
| Total taking final: 87 | 75 |
| Raised grade: 8 | 27 |
| Lowered grade: 5 | 5 |

## Announcements

Course evaluation: Completing it is part of your course assignment. Worth 1% of grade.

**Must be completed by Saturday night. 1 DEC**

**Please complete for Gries and for Clarkson**

We then get a file that says who completed the evaluation.

We do not see your evaluations until after we submit grades to to the Cornell system.

We never see names associated with evaluations.

## Announcements

Office hours:

Gries: today, Thursday, 1-3

## Fibonacci function

7

fib(0) = 0

fib(1) = 1

fib(n) = fib(n-1) + fib(n-2) for n ≥ 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)

8

fib(0) = 0

fib(1) = 1

fib(n) = fib(n-1) + fib(n-2) for n ≥ 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)

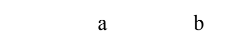
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

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

9

Divide a line into two parts:

Call long part a and short part b




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

Solution is the **golden ratio, Φ**

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

## $\Phi = (1 + \sqrt{5})/2 = 1.61803398\ldots$

10

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

fib(n) / fib(n-1) is close to Φ.
So Φ * fib(n-1) is close to fib(n)
Use formula to calculate fib(n) from fib(n-1)

In fact,

$$\lim_{n \to \infty} f(n)/fib(n-1) = \Phi$$

| a/b |
|---|
| 8/5 = 1.6 |
| 13/8 = 1.625… |
| 21/13= 1.615… |
| 34/21 = 1.619… |
| 55/34 = 1.617… |

Golden ratio and Fibonacci numbers: inextricably linked

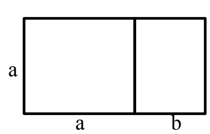
## Golden ratio $\Phi = (1 + \sqrt{5})/2 = 1.61803398\ldots$

11

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

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

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, …

## Fibonacci, golden ratio, golden angle

12

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

$$\lim_{n \to \infty} f(n)/fib(n-1) = \text{golden ratio} = 1.6180339887\ldots$$

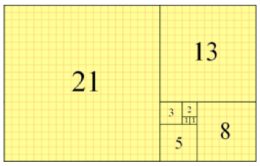
360/1.6180339887… = 222.492235…

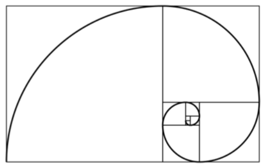
360 – 222.492235… = 137.5077 golden angle

## Fibonacci function (year 1202)

13





Golden rectangle

Fibonacci tiling

Fibonacci spiral

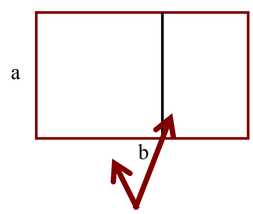
0, 1, 1, 2, 3, 5, 8, 13, 21, 34 …

## The Parthenon

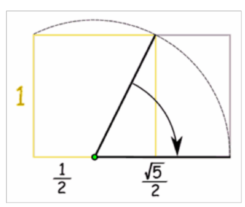
14

## Drawing a golden rectangle with ruler and compass

15




golden rectangle




How to draw a golden rectangle

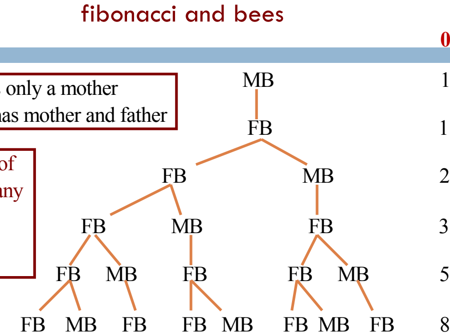
hypotenuse: $\sqrt{(1*1 + (½)(½))} = \sqrt{(5/4)}$

## fibonacci and bees

16

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

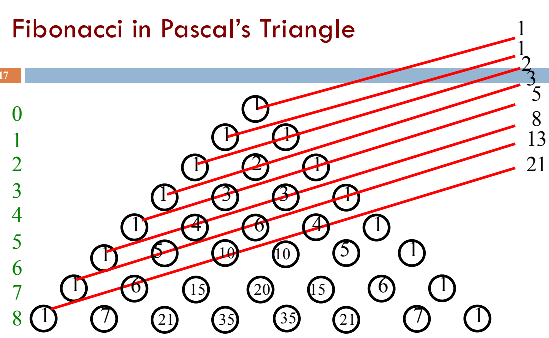
The number of ancestors at any level is a Fibonnaci number




MB: male bee, FB: female bee

## Fibonacci in Pascal's Triangle

17




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

## Suppose you are a plant

18

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

19

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

20

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).

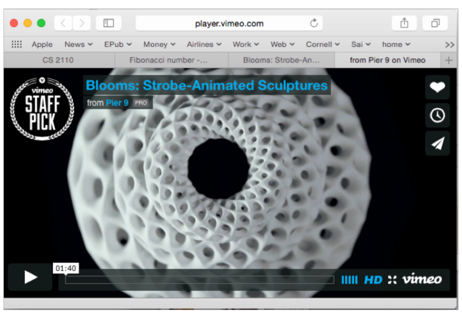
Recall: golden angle

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

## Blooms: strobe-animated sculptures

21

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




## Uses of Fibonacci sequence in CS

22

Fibonacci search

Fibonacci heap data strcture

Fibonacci cubes: graphs used for interconnecting parallel and distributed systems

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

23

binary search:
cut in half at each step

0 e1 n

e1 = (n-0)/2

0 e2 e1

e2 = (e1-0)/2

e2 e1

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

0 e1 144

e1 = 0 + 89

0 e2 e1

e2 = 0 + 55

e2 e1

2 3 5 8 13 21 34 55 89 144

## Fibonacci search history

24

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

25

David Ferguson.

Fibonaccian 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.

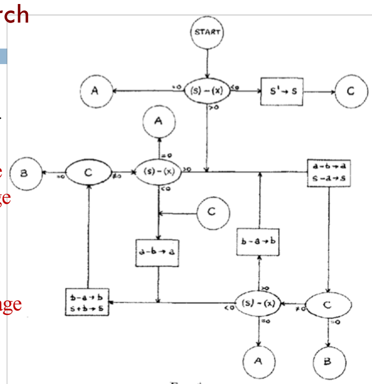


Fig. 1

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

26

```
/** 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)

27

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)

28

T(0) = a

T(1) = a

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

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

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

$\boxed{T(n) <= c*2^n \text{ for } n >= N}$

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)

29

T(0) = a

T(1) = a

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

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

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

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

$\boxed{T(n) <= c*2^n \text{ for } n >= N}$

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)

30

T(0) = a

T(1) = a

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

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

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

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

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

$\boxed{T(n) <= c*2^n \text{ for } n >= N}$

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)

31

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

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

$\boxed{T(n) \le c*2^n \text{ for } n \ge N}$

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)

32

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

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

$\boxed{T(n) \le c*2^n \text{ for } n \ge N}$

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

33

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.

## Caching

34

```
/** For 0 ≤ n < cache.size, fib(n) is cache[n]
  * If fibCached(k) has been called, its result in in cache[k] */
public static ArrayList<Integer> cache= new ArrayList<>();
```

```
/** Return fibonacci(n).   Pre: n >= 0.    Use the cache. */
public static int fibCached(int n) {
   if (n < cache.size()) return cache.get(n);
   if (n == 0) { cache.add(0); return 0; }
   if (n == 1) { cache.add(1); return 1; }

   int ans= fibCached(n-2) + fibCached(n-1);
   cache.add(ans);
   return ans;
}
```

| |
|---|
| 8 |
| 5 |
| 3 |
| 2 |
| 1 |
| 1 |
| 0 |

cache

## Linear algorithm to calculate fib(n)

35

```
/** 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!

36

$f_0 = 0$
$f_1 = 1$
$f_{n+2} = f_{n+1} + f_n$

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

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

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

## Logarithmic algorithm!

37

$f_0 = 0$
$f_1 = 1$
$f_{n+2} = f_{n+1} + f_n$

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

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

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!

38

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

The golden ratio again.

Prove by induction on n that

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