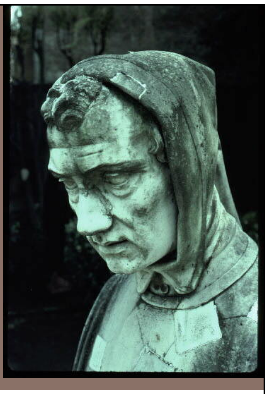
Fibonacci
(Leonardo Pisano)
1170-1240?
Statue in Pisa Italy



FIBONACCI NUMBERS AND RECURRENCES

Lecture 23
CS2110 – Fall 2015

Prelim 2 Thursday at 5:30 and 7:30

2

5:30. Statler Auditorium. Last name M..Z

7:30. Statler Auditorium. Last name A..L

If you completed assignment P2Conflict and did not hear from us, just go to your desired time

26 people 7:30 -> 5:30

07 people 5:30 -> 7:30

If you are AUTHORIZED BY CORNELL to have a quiet place or extend time and completed P2COnflict, go to Gates 405, starting from 5:15.

About implementing heaps

3

N is size of heap

```
/** Add e with priority p to the priority queue.
  * Throw an illegalArgumentException if e is already in the queue.
  * Expected time is O(log N), worst-case time is O(N). */
public void add(E e, double p) {
   if (b.contains(e)) throw new …);         O(N) operation
    b.add(size,e);   size= size+1;
                                             O(N) operation
    int ni= b.indexOf(e);  size-1;
}
```

java.util

Class ArrayList<E>

4

java.lang.Object
java.util.AbstractCollection<E>

Operations size, isEmpty, get, set, iterator, and listIterator run in constant time. Operation add runs in *amortized constant time*, that is, adding n elements requires O(n) time. All the other operations run in linear time (roughly speaking). The constant factor is low compared to that for the LinkedList implementation.

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, Serializable
```

Creating unnecessary objects

5

In bubble-up

```
b.set(k, child);
map.put(child, new Info(k, p_child));
```

BETTER:

```
b.set(k, child);
map.get(child).index= k;
```

More effective presentation

6

1. Put { on line before, with space before
2. Put space after if, else, while do, etc.
3. If the "then-part" of an if statement returns or throws, do not use else.

```
public E peek() {
    if(b.size() == 0)
    {
        throw new PC…();
    }
    else
    {
        return b.get(0);
    }
}
```

```
public E peek() {
    if (b.size() == 0) {
        throw new PC…();
    }
    return b.get(0);
}
```

Fibonacci function

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 1202 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 about1150

The so-called Fibonacci numbers in ancient and medieval India.
Parmanad Singh, 1985

Fibonacci function (year 1202)

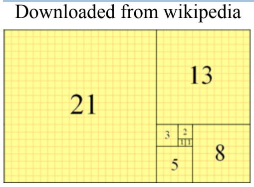
fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2) for $n \geq 2$

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

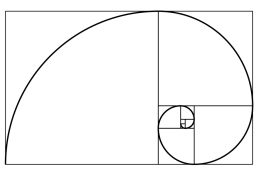
0, 1, 1, 2, 3, 5, 8, 13, 21, …

Fibonacci function (year 1202)

Downloaded from wikipedia



Fibonacci tiling



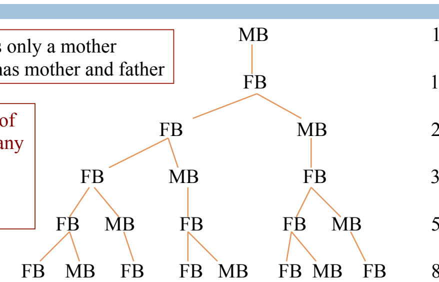
Fibonacci spiral

0, 1, 1, 2, 3, 5, 8, 13, 21, …

fibonacci and bees

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

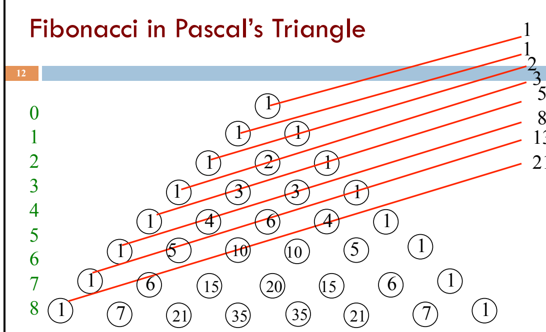
Fibonacci in nature

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

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). You can measure this rotation by dividing 360 degrees (a full spin) by the inverse of the golden ratio. We see later how the golden ratio is connected to fibonacci.

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

Fibonacci in Pascal's Triangle

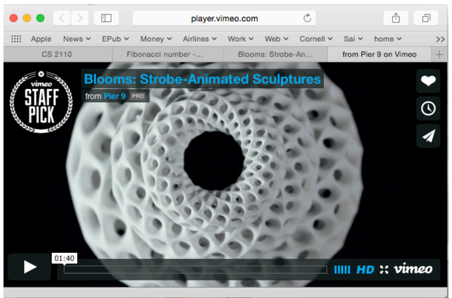


p[i][j] is the number of ways i elements can be chosen from a set of size j

Blooms: strobe-animated sculptures

13

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



Uses of Fibonacci sequence in CS

14

Fibonacci search

Fibonacci heap data strcture

Fibonacci cubes: graphs used for interconnecting parallel and distributed systems

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

15

:

f(20)

f(18) + f(19)

f(16) + f(17) + f(17) + f(18)

f(14)+ f(15) + f(15)+F(16) f(15)+f(16) + f(16)+f(17)

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

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

16

T(0) = a — "Recurrence relation" for the time

T(1) = a — It's just a recursive function

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

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)

17

T(0) = a

T(1) = a

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

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

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

$T(n) <= c*2^n$ 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)

18

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)

19

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

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

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

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

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

$T(n) <= c*2^n$ 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)

20

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

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

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

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

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

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

$T(n) <= c*2^n$ 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)

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(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>
$a * 2^k$

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)

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

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

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

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

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

$T(5) = a \le a * 2^5$

$T(n) <= c*2^n$ 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

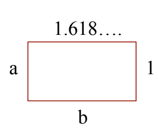
The golden ratio

23

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

(a + b) / a = a/b call that value φ

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



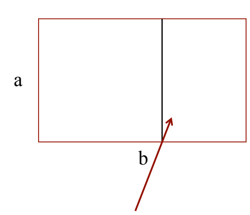
ratio of sum of sides to longer side

=

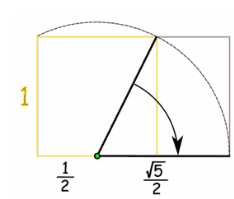
ratio of longer side to shorter side

The golden ratio

24



golden rectangle



How to draw a golden rectangle

The Parthenon

25

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

$f_0 = 0$

$f_1 = 1$

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

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

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

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

Constant-time algorithm!

29

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

The golden ratio again.

Prove by induction on n that

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

We went from $O(2^n)$ to $O(1)$