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

FIBONACCI NUMBERS
GOLDEN RATIO,
RECURRANCES
Separation of concerns

private void bubbleDown(int k) {
    while (size > 2*k + 1 ){
        double thisp = c[k].priority;
        double left = c[2*k+1].priority;
        double right = 0;
        if (size > 2*k +2) right = c[2*k+2].priority;
        if (right != 0  &&  thisp >  right  &&  thisp >= left) {
            swap(2*k +2, k);  k = 2*k +2;
        } else if (left < thisp) {
            swap(2*k+1, k);  k = 2*k+1;
        } else return;
    }
}

Modification of a bubble-down with an error.
For the heap shown below, this method won’t bubble (1, 5) down the right.
Separation of concerns

private void bubbleDown(int k) {
    while (size > 2*k + 1 ){
        double thisp = c[k].priority;
        double left = c[2*k+1].priority;
        double right = 0;
        if (size > 2*k +2) right = c[2*k+2].priority;
        if (right != 0  &&  thisp >  right  &&  thisp >= left) {
            swap(2*k +2, k);  k = 2*k +2;
        } else if (left < thisp) {
            swap(2*k+1, k);  k = 2*k+1;
        } else return;
    }
}

Two concerns:
1. Which is the smaller child.
2. Should a bubble-down take place.

Purpose of smallerChildOf:
separate these two concerns.

We’ll develop the two Methods in class.
Fibonacci function

$$\text{fib}(0) = 0$$
$$\text{fib}(1) = 1$$
$$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2) \quad \text{for } n \geq 2$$

0, 1, 1, 2, 3, 5, 8, 13, 21, …

In his book in 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)

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);
}

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

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 \( b \)

\[
(a + b) / a = a / b \quad \text{Solution is called } \Phi
\]

See webpage:
http://www.mathsisfun.com/numbers/golden-ratio.html
Golden ratio  $\Phi = \frac{1 + \sqrt{5}}{2} = 1.61803398\cdots$

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

$$\frac{a + b}{a} = \frac{a}{b} = \Phi$$

See webpage:
http://www.mathsisfun.com/numbers/golden-ratio.html
Golden ratio  $\Phi = (1 + \sqrt{5})/2 = 1.61803398\cdots$

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

<table>
<thead>
<tr>
<th>a/b</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8/5</td>
<td>1.6</td>
</tr>
<tr>
<td>13/8</td>
<td>1.625\cdots</td>
</tr>
<tr>
<td>21/13</td>
<td>1.615\cdots</td>
</tr>
<tr>
<td>34/21</td>
<td>1.619\cdots</td>
</tr>
<tr>
<td>55/34</td>
<td>1.617\cdots</td>
</tr>
</tbody>
</table>

For successive Fibonacci numbers $a, b$, $a/b$ is close to $\Phi$ but not quite it $\Phi$. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...
Find fib(n) from fib(n-1)

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

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

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

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

topones.weebly.com/1/post/2012/10/the-artichoke-and-fibonacci.html
Fibonacci function (year 1202)

Downloaded from wikipedia

Fibonacci tiling

Fibonacci spiral

0, 1, 1, 2, 3, 5, 8, 13, 21, 34 …
The Parthenon
The golden ratio

a

b

golden rectangle

How to draw a golden rectangle
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 $j$
Fibonacci in nature

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

\[
\frac{360}{\text{(golden ratio)}} = 222.492
\]

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

www.instructables.com/id/Blooming-Zoetrope-Sculptures/
Uses of Fibonacci sequence in CS

- Fibonacci search
- Fibonacci heap data structure
- Fibonacci cubes: graphs used for interconnecting parallel and distributed systems
LOUSY WAY TO COMPUTE: \( O(2^n) \)

/** 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)?
Recursion for fib: \( f(n) = f(n-1) + f(n-2) \)

\[
\begin{align*}
T(0) &= \alpha \\
T(1) &= \alpha \\
T(n) &= \alpha + T(n-1) + T(n-2)
\end{align*}
\]

\( T(n) \): 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 \( O(2^n) \)

\[
T(n) \leq c \cdot 2^n \text{ for } n \geq N
\]
Recursion for fib: $f(n) = f(n-1) + f(n-2)$

$T(0) = a$

$T(1) = a$

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

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

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

$T(n) \leq c \times 2^n$ for $n \geq N$

$T(2) = \langle\text{Definition}\rangle$

$= a + T(1) + T(0)$

$\leq \langle\text{look to the left}\rangle$

$a + a \times 2^1 + a \times 2^0$

$= \langle\text{arithmetic}\rangle$

$a \times (4)$

$= \langle\text{arithmetic}\rangle$

$a \times 2^2$
Recursion for fib: \( f(n) = f(n-1) + f(n-2) \)

\[
\begin{align*}
T(0) &= a \\
T(1) &= a \\
T(n) &= T(n-1) + T(n-2) \\
T(0) &= a \leq a \times 2^0 \\
T(1) &= a \leq a \times 2^1 \\
T(2) &= 2a \leq a \times 2^2 \\
T(n) &\leq c \times 2^n \text{ for } n \geq N \\
T(3) &= <\text{Definition}> \\
&= a + T(2) + T(1) \\
&\leq <\text{look to the left}> \\
&= a + a \times 2^2 + a \times 2^1 \\
&= <\text{arithmetic}> \\
&= a \times (7) \\
&\leq <\text{arithmetic}> \\
&= a \times 2^3
\end{align*}
\]
Recursion for fib: \( f(n) = f(n-1) + f(n-2) \)

\[
\begin{align*}
T(0) & = a \\
T(1) & = a \\
T(n) & = T(n-1) + T(n-2)
\end{align*}
\]

\[
\begin{align*}
T(0) & = a \leq a \times 2^0 \\
T(1) & = a \leq a \times 2^1 \\
T(2) & \leq a \times 2^2 \\
T(3) & \leq a \times 2^3
\end{align*}
\]

\[
T(n) \leq c \times 2^n \text{ for } n \geq N
\]

\[
\begin{align*}
T(4) & = \text{<Definition>}
\end{align*}
\]

\[
\begin{align*}
& a + T(3) + T(2) \\
& \leq \text{<look to the left>}
\end{align*}
\]

\[
\begin{align*}
& a + a \times 2^3 + a \times 2^2 \\
& = \text{<arithmetic>}
\end{align*}
\]

\[
\begin{align*}
& a \times (13) \\
& \leq \text{<arithmetic>}
\end{align*}
\]

\[
\begin{align*}
& a \times 2^4
\end{align*}
\]
Recursion for fib: $f(n) = f(n-1) + f(n-2)$

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

$T(0) = a \leq a \times 2^0$
$T(1) = a \leq a \times 2^1$
$T(2) \leq a \times 2^2$
$T(3) \leq a \times 2^3$
$T(4) \leq a \times 2^4$

$T(n) \leq c \times 2^n$ for $n \geq N$

$T(5) = <\text{Definition}>$
$= a + T(4) + T(3)$
$\leq <\text{look to the left}>$
$= a + a \times 2^4 + a \times 2^3$
$= <\text{arithmetic}>$
$= a \times (25)$
$\leq <\text{arithmetic}>$
$= a \times 2^5$

WE CAN GO ON FOREVER LIKE THIS
Recursion for fib:  \( f(n) = f(n-1) + f(n-2) \)

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

\[
\begin{align*}
T(0) &= a \\ 
T(1) &= a \\ 
T(n) &= T(n-1) + T(n-2) \\
T(0) &= a \leq a \cdot 2^0 \\
T(1) &= a \leq a \cdot 2^1 \\
T(2) &\leq a \cdot 2^2 \\
T(3) &\leq a \cdot 2^3 \\
T(4) &\leq a \cdot 2^4
\end{align*}
\]

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

\[
T(k) = \begin{align*}
&<\text{Definition}> \\
&= a + T(k-1) + T(k-2) \\
&\leq <\text{look to the left}> \\
&= a + a \cdot 2^{k-1} + a \cdot 2^{k-2} \\
&= <\text{arithmetic}> \\
&= a \cdot (1 + 2^{k-1} + 2^{k-2}) \\
&\leq <\text{arithmetic}> \\
&= a \cdot 2^k
\end{align*}
\]
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

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

\[(a + b) / b = b/a\]

call that value $\varphi$

\[\varphi^2 = \varphi + 1\]

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

\[
\begin{array}{c}
\text{ratio of sum of sides to longer side} \\
1.618\ldots
\end{array}
\]

\[
\begin{array}{c}
\text{ratio of longer side to shorter side} \\
= \\
\end{array}
\]

\[
\begin{array}{c}
\text{a} \\
\text{b}
\end{array}
\]
Can prove that Fibonacci recurrence is $O(\phi^n)$

We won’t prove it.

Requires proof by induction

Relies on identity $\phi^2 = \phi + 1$
/** 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)
    // 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!

\[
\begin{align*}
  f_0 &= 0 \\
  f_1 &= 1 \\
  f_{n+2} &= f_{n+1} + f_n
\end{align*}
\]

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

\[
\begin{pmatrix}
  0 & 1 \\
  1 & 1
\end{pmatrix}
^k
\begin{pmatrix}
  f_n \\
  f_{n+1}
\end{pmatrix}
= 
\begin{pmatrix}
  f_{n+k} \\
  f_{n+k+1}
\end{pmatrix}
\]
Logarithmic algorithm!

\[ f_0 = 0 \]
\[ f_1 = 1 \]
\[ f_{n+2} = f_{n+1} + f_n \]

\[
\begin{pmatrix}
0 & 1 \\
1 & 1
\end{pmatrix}
\]
\[ \begin{pmatrix}
f_n \\
f_{n+1}
\end{pmatrix}
\]
\[ = \begin{pmatrix}
f_{n+k} \\
f_{n+k+1}
\end{pmatrix}
\]

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!

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

The golden ratio again.

Prove by induction on \( n \) that

\[
fn = \frac{\phi^n - \phi'^n}{\sqrt{5}}
\]