Fibonacci Numbers

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

Fibonacci
(Libonacci)
1170-1240?
Statue in Pisa Italy

FIBONACCI NUMBERS
GOLDEN RATIO,
RECURRENCES

Lecture 25
CS2110 – Spring 2018

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, …

We’ll see that this is a lousy way to compute fn

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

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

\[
\frac{\text{whole length}}{\text{long part}} = \frac{\text{long part}}{\text{short part}}
\]

Call long part a and short part b

\[
\frac{a + b}{a} = \frac{a}{b}
\]

Solution is called \( \Phi \)

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

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

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

\[
\frac{a + b}{a} = \frac{a}{b} = \Phi
\]

\[
\frac{a}{b}
\]

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

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

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

\[
\frac{a + b}{a} = \frac{a}{b} = \Phi
\]

\[
\frac{a}{b}
\]

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

0, 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 \(\text{fib}(n-1)\)

**Golden ratio and Fibonacci numbers: inextricably linked**

---

**The Parthenon**

---

**Fibonacci function (year 1202)**

Downloaded from wikipedia

**Fibonacci in Pascal's Triangle**

\[ p(i|j) \text{ is the number of ways } i \text{ elements can be chosen from a set of size } j \]

---

**The golden ratio**

How to draw a golden rectangle

---

**Fibonacci in Pascal's Triangle**

\[ p(i|j) \text{ is the number of ways } i \text{ elements can be chosen from a set of size } j \]
Suppose you are a plant

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!

Fibonacci in nature

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

360/(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

Uses of Fibonacci sequence in CS

Fibonacci search

Fibonacci heap data structure

Fibonacci cubes: graphs used for interconnecting parallel and distributed systems

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

binary search: cut in half at each step

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

<table>
<thead>
<tr>
<th>0</th>
<th>e1</th>
<th>n</th>
<th>0</th>
<th>e1</th>
<th>144</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1 = (n-0)/2</td>
<td>e2</td>
<td>e1</td>
<td>e1 = 0 + 89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e2 = (e1-0)/2</td>
<td>e2</td>
<td>e1</td>
<td>e2 = 0 + 55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e2</td>
<td>e1</td>
<td>e2</td>
<td>e1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td>21</td>
</tr>
</tbody>
</table>

Blooms: strobe-animated sculptures

www.instructables.com/id/Blooming-Zoetrope-Sculptures/
Fibonacci search history


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.

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: $f(n) = f(n-1) + f(n-2)$

<table>
<thead>
<tr>
<th>$T(0)$</th>
<th>$T(1)$</th>
<th>$T(2)$</th>
<th>$T(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$a$</td>
<td>$2a$</td>
<td>$c*2^n$ for $n &gt;= N$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$T(0)$</th>
<th>$T(1)$</th>
<th>$T(2)$</th>
<th>$T(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$a$</td>
<td>$2a$</td>
<td>$c*2^n$ for $n &gt;= N$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$T(0)$</th>
<th>$T(1)$</th>
<th>$T(2)$</th>
<th>$T(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$a$</td>
<td>$2a$</td>
<td>$c*2^n$ for $n &gt;= N$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$T(0)$</th>
<th>$T(1)$</th>
<th>$T(2)$</th>
<th>$T(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$a$</td>
<td>$2a$</td>
<td>$c*2^n$ for $n &gt;= N$</td>
</tr>
</tbody>
</table>
Recursion for fib: $f(n) = f(n-1) + f(n-2)$

T(0) = 0, T(1) = 0, 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) = $a \cdot 2^2$
T(3) = $a \cdot 2^3$

T(4) = $a \cdot 2^4$

T(5) = $a \cdot (1 + 2^3 + 2^2)$

T(6) = $a \cdot 2^5$

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

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

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

\[
\frac{a + b}{b} = \frac{b}{a}
\]

Call that value $\varphi$

$\varphi^2 = \varphi + 1$ so $\varphi = \frac{1 + \sqrt{5}}{2} \approx 1.618 \ldots$

$\frac{a}{b} \approx \frac{b}{a}$ ratio of sum of sides to longer side

$\frac{b}{a}$ ratio of longer side to shorter side

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

We won't prove it.
Requires proof by induction
Relies on identity $\varphi^2 = \varphi + 1$
Linear algorithm to calculate fib(n)

```java
/** Return fib(n), for n >= 0. */
public static int fib(int n) {
    if (n <= 1) return 1;
    int p = 0, c = 1, 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!

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

```
\[
\begin{pmatrix}
0 & 1 \\
1 & 1
\end{pmatrix}
\begin{pmatrix}
f_0 \\
f_1
\end{pmatrix}
= \begin{pmatrix}
f_n \\
f_{n+1}
\end{pmatrix}
\]
```

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

Logarithmic algorithm!

```
\[
\begin{pmatrix}
0 & 1 \\
1 & 1
\end{pmatrix}
\begin{pmatrix}
0 & 1 \\
1 & 1
\end{pmatrix}
= \begin{pmatrix}
0 & 1 \\
1 & 1
\end{pmatrix}
\]
```

Another log algorithm!

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

The golden ratio again.

Prove by induction on \( n \) that

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