Fibonacci Numbers

GOLDEN RATIO,
RECURRENCES

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

Fibonacci function

**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:

Virahanka 600–800
Gopala before 1135
Hemacandra about 1150

The so-called Fibonacci
numbers in ancient and
medieval India,
Parmarad 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 fib(int n) {
    if ( n <= 1) return n;
    return fib(n-1) + fib(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)

Announcements

2

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.

Announcements

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

You will 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 now on Exams page of course
website. Not all answers yet.

Announcements

4

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

Must be completed by Saturday night. 1 DEC

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

Divide a line into two parts:
Call long part $a$ and short part $b$

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

Solution is the golden ratio, $\Phi$

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

Fibonacci, golden ratio, golden angle

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

\[
\frac{f(n)}{f(n-1)} \text{ is close to } \Phi.
\]

So $\Phi \cdot f(n-1)$ is close to $f(n)$

Use formula to calculate $f(n)$ from $f(n-1)$

In fact,

\[
\lim_{n \to \infty} \frac{f(n)}{f(n-1)} = \Phi
\]

Golden ratio and Fibonacci numbers: inextricably linked

Fibonacci function (year 1202)

Downloaded from wikipedia

Golden rectangle

Fibonacci tiling

Fibonacci spiral

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

The Parthenon
Drawing a golden rectangle with ruler and compass

How to draw a golden rectangle

hypotenuse: $\sqrt{(1*1 + (1/2)(1/2))} = \sqrt{5/4}$

Fibonacci and bees

<table>
<thead>
<tr>
<th>Level</th>
<th>MB</th>
<th>FB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MB</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>MB</td>
<td>FB</td>
</tr>
<tr>
<td>3</td>
<td>MB</td>
<td>FB</td>
</tr>
<tr>
<td>5</td>
<td>MB</td>
<td>FB</td>
</tr>
</tbody>
</table>

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 $j$

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!

Suppose you are a plant

You want to grow your leaves so that they all get a good amount of sunlight. 90 degrees, maybe?

Fibonacci in nature

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

$360/(\text{golden ratio}) = 222.492$

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).

Recall: golden angle

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

---

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>e1 = 0 + 89</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>e2</td>
<td>e1</td>
<td>0</td>
<td>e2</td>
<td>e1</td>
</tr>
<tr>
<td>e2 = (e1-0)/2</td>
<td>e2 = 0 + 55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e2</td>
<td>e1</td>
<td>e2</td>
<td>e1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 3 5 8 13 21 34 55 89 144

---

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.

---

Fibonacci search

David Ferguson.
Fibonacci 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.

---

LOUSY WAY TO COMPUTE: O(2^n)

```java
/** 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)?

19 18
18 17 17 16
17 16 16 15 16 15 15 14
Recursion for fib: $f(n) = f(n-1) + f(n-2)$

$T(0) = \alpha$

$T(1) = \alpha$

Just a recursive function

$T(n) = \alpha + T(n-1) + T(n-2)$  "reccurence 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 $O(2^n)$

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

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

$T(0) = \alpha$

$T(1) = \alpha$

$T(n) = \alpha + T(n-1) + T(n-2)$

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

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

$T(2) = 2\alpha \leq \alpha * 2^2$

$T(3) \leq \alpha * 2^3$

$T(4) \leq \alpha * 2^4$

WE CAN GO ON FOREVER LIKE THIS
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, \( f(0), f(1), \ldots f(n-1) \) are already cached.

```java
/** For 0 ≤ n < cache.size, fib(n) is cache[n]
 * If fibCached(k) has been called, its result is 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;
}
```

**Linear algorithm to calculate fib(n)**

```java
/** 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!**

```math
\begin{align*}
0 & \quad 1 \\
1 & \quad 1
\end{align*}
\begin{align*}
0 & \quad 1 & 0 & \quad 1 \\
1 & \quad 1 & 1 & \quad 1
\end{align*}
\begin{align*}
0 & \quad 1 & f_n & \quad 1 & f_{n+1} & \quad 1 & f_{n+2} \\
1 & \quad 1 & 1 & \quad 1 & 1 & \quad 1 & 1
\end{align*}
\begin{align*}
0 & \quad 1 & f_n & \quad f_{n+1} & \quad f_{n+k} & \quad f_{n+k+1}
\end{align*}
```

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

**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

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