One-Slide Summary

• The basic recursive computation of Fibonacci can take quite a while. There are faster ways.
• We can formally measure and evaluate the cost of a computer program. We abstract away details such as processor speed and instead measure how the solving time increases as the input increases.
• $g$ is in $O(f)$ iff there exist positive constants $c$ and $n_0$ such that $g(n) \leq cf(n)$ for all $n \geq n_0$.
• If $g$ is in $O(f)$ we say that $f$ is an upper bound for $g$.

Outline

• Sneezewort and Fibonacci
• Cost of computing Fibonacci
• Cost of sorting
• Intro to Big-Oh Notation

Sneezewort

• Achillea ptarmica is real.
• It is “moste efficacious in the inflaming of the braine, and [is] therefore much used in Confusing and Befuddlement Draughts, where the wizard is desirous of producing hot-headedness and recklessness.”
  - Order of the Phoenix, p.18
• Sneezewort’s pattern of development displays the Fibonacci sequence.
Could we model Sneezewort with PS3 code?

Sneezewort Growth

First Time Unit  Second Time Unit  Offshoot

Sneezewort Numbers

1
1
2
3
5
8?
13

Pink
by Jessica Geist, Ellen Clarke

Fibo Results

>>> fibo(2)
1
>>> fibo(3)
2
>>> fibo(4)
3
>>> fibo(10)
55
>>> fibo(60)
Still working…

At least we finished.
by Dmitriy Semenov and Sara Alspaugh

Tracing Fibo

>>> (fibo 3)
| fibo(3)
| fibo(2)
| 1
| fibo(1)
| 1
| 2
| 1
| 2
| 1

To calculate fibo(5) we calculated:

- fibo(4) 1 time
- fibo(3) 2 times
- fibo(2) 3 times
- fibo(1) 2 times

= 8 calls to fibo
= fibo(6), interestingly...

5 times total

A right-wing Christmas - awwwwww......
by Andrew Baker & Emily Lam

Liberal Arts Trivia: History

• This 20th-century American inventor is credited with the phonograph, the carbon telephone transmitter, the practical electric light, and the phrase “Genius is one percent inspiration, ninety-nine percent perspiration.” He fought against Nikola Tesla's alternating current in the so-called War of the Currents.

Liberal Arts Trivia: History

• This 20th-century American inventor is credited with the phonograph, the carbon telephone transmitter, the practical electric light, and the phrase “Genius is one percent inspiration, ninety-nine percent perspiration.” He fought against Nikola Tesla's alternating current in the so-called War of the Currents.
Liberal Arts Trivia: Film Studies
• In this Oscar-nominated 2006 film, David Bowie is almost torched by Thomas Edison's goons but invents a teleportation machine for Wolverine so that he can defeat Batman in a magic trick competition because he thinks Batman killed his wife.

Liberal Arts Trivia: Physics
• Count Alessandro Antonio Anastasio Volta was a 19th-century Italian physicist. Volta studied what we now call capacitance, developing separate means to study both electrical potential $V$ and charge $Q$, and discovering that for a given object they are proportional. His experiments in “animal electricity”, in which two different metals were connected in series with frog's legs, eventually led to his most famous discovery. What was it?

```
fast_fibo
# 1 + 1 = 2
# 1 + 2 = 3
# 2 + 3 = 5
# 3 + 5 = 8
# a + b = fibo(n)

def fast_fibo(n):
    def fib_helper(a, b, left):
        if left <= 0:
            return b
        return fib_helper(b, a+b, left-1)
    return fib_helper(1, 1, n-2)
```

```
Fast-Fibo Results
>>> fast_fibo(10)
55
>>> a = time(); print fast_fibo(60); print time()-a
1548008755920
7.10487365723e-05 seconds

The original fibo would take at least 2.5 Trillion applications. A 2.5 GHz computer does 2.5 Billion simple operations per second, so 2.5 Trillion applications operations take ~1000 seconds.
```

```
# The Earth's mass is 6.0 x 10^24 kg
>>> mass_of_earth = 6 * pow(10,24)
# A typical rabbit's mass is 2.5 kilograms
>>> mass_of_rabbit = 2.5
>>> (mass_of_rabbit * fast_fibo(60)) / mass_of_earth
6.450036483e-013
>>> (mass_of_rabbit * fast_fibo(120)) / mass_of_earth
2.2326496895795693

According to Bonacci’s model, after less than 10 years, rabbits would out-weigh the Earth!
```

Broccoli Fallout by Paul DiOrio, Rachel Phillips
Evaluation Cost

Actual running times vary according to:
- How fast a processor you have
- How much memory you have
- Where data is located in memory
- How hot it is
- What else is running
- etc...

Moore’s “Law” – computing power doubles every 18 months

Measuring Cost

• How does the cost scale with the size of the input?
• If the input size increases by one, how much longer will it take?
• If the input size doubles, how much longer will it take?

Cost of Fibonacci Procedures

def fast_fibo(n):
def fibo(n):
  def fib_helper(a, b, left):
    if left <= 0:
      return b
    return fib_helper(b, a+b, left-1)
  return fib_helper(1, 1, n-2)
  if n <= 2:
    return 1
  return fibo(n-1) + fibo(n-2)
def fibo(n):
  if n <= 2:
    return 1
  return fibo(n-1) + fibo(n-2)

<table>
<thead>
<tr>
<th>Input</th>
<th>fibo</th>
<th>fast_fibo</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>mk</td>
</tr>
<tr>
<td>m+1</td>
<td></td>
<td>(m+1)k</td>
</tr>
<tr>
<td>m+2</td>
<td>at least $q^2$</td>
<td>(m+2)k</td>
</tr>
</tbody>
</table>

Cost of Fibonacci Procedures

def fast_fibo(n):
def fibo(n):
  def fib_helper(a, b, left):
    if left <= 0:
      return b
    return fib_helper(b, a+b, left-1)
  return fibo(n-1) + fibo(n-2)
def fibo(n):
  if n <= 2:
    return 1
  return fibo(n-1) + fibo(n-2)

<table>
<thead>
<tr>
<th>Input</th>
<th>fibo</th>
<th>fast_fibo</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>q</td>
<td>mk</td>
</tr>
<tr>
<td>m+1</td>
<td></td>
<td>(m+1)k</td>
</tr>
<tr>
<td>m+2</td>
<td>at least $q^2$</td>
<td>(m+2)k</td>
</tr>
</tbody>
</table>

$\Phi = \frac{1 + \sqrt{5}}{2} = "The Golden Ratio" \approx 1.618033988749895...

The Golden Ratio

Parthenon

Nautilus Shell

More Golden Ratios

CONAN O'BRIEN hair flip fits the fibonacci spiral

FIRST USA

Untitled

Nokomis McCaskill

Chris Hooe
Sorting

```python
>>> def ascending(x,y): return x - y
>>> sorted([5,2,4,1,3], ascending)
[1, 2, 3, 4, 5]
>>> def descending(x,y): return y - x
>>> sorted([5,2,4,1,3], descending)
[1, 2, 3, 4, 5]
>>> def longer(x,y): return len(x) - len(y)
>>> sorted(“allons”,”enfants”,”de”),longer)
[“de”, “allons”, “enfants”]
```

Simple Sorting

- Can we use `find_best` to implement `sort`?
  - Yes!
- Use `find_best(lst)` to find the best
- Remove it from the list
  - Adding it to the answer
- Repeat until the list is empty

```python
def sort(lst, cf): # simple sort
  if not lst: return []
  best = find_best(lst, cf)
  return [best] + sort(remove(lst, best), cf)
```

Sorting Cost

- What grows?
  - \( n \) = the number of elements in \( \text{lst} \)
- How much work are the pieces?
  - find-best:
  - remove:
Sorting Cost

- What grows?
  \[-n = \text{the number of elements in lst}\]
- How much work are the pieces?
  find-best: work scales as \(n\) (increases by one)
  remove: work scales as \(n\) (increases by one)
- How many times does sort evaluate find-best and delete? \(n\)
- Total cost: scales as \(n^2\)

Liberal Arts Trivia: Medicine

- Nicolae Paulescu was a 20\(^{th}\) century physiologist and professor of medicine. He is considered the true discoverer of hormone that causes most of the body's cells to take up glucose from the blood. His first experiments involved an aqueous pancreatic extract which, when injected into a diabetic dog, proved to have a normalizing effect on blood sugar levels. Name the hormone.

Liberal Arts Trivia: Sailing

- Name the collection of apparatus through which the force of the wind is transferred to the ship in order to propel it forward - this includes the masts, yardarms, sails, spars and cordage.
Timing Sort

def find_best(things, better):
    if len(things) == 1: return things[0]
    rest = find_best(things[1:], better)
    return things[0] if better(things[0], rest) else rest

def sort(lst, cf):
    if not lst: return []
    best = find_best(lst, cf)
    lst.remove(best)  # tricky!
    return [best] + sort(lst, cf)

def descending(x,y):
    return x > y

time_sort(k):
    a = time.time()
    sort(range(k), descending)
    return time.time() - a

print sort(range(10), descending)
print time_sort(20)
print time_sort(40)
print time_sort(80)
print time_sort(160)

Growth Notations

- \( g \in O(f) \) (“Big-Oh”)  
  \( g \) grows no faster than \( f \)  
  \( f \) is upper bound

- \( g \in \Theta(f) \) (“Theta”)  
  \( g \) grows as fast as \( f \)  
  \( f \) is tight bound

- \( g \in \Omega(f) \) (“Omega”)  
  \( g \) grows no slower than \( f \)  
  \( f \) is lower bound

Which one would we most like to know?

\( g \) is in \( O(f) \) iff there are positive constants \( c \) and \( n_0 \) such that \( g(n) \leq cf(n) \) for all \( n \geq n_0 \).

**Meaning of \( O \) (“big Oh”)**

\( g \) is in \( O(f) \) iff:

- There are positive constants \( c \) and \( n_0 \) such that \( g(n) \leq cf(n) \) for all \( n \geq n_0 \).

\( g \) is in \( O(f) \) iff there are positive constants \( c \) and \( n_0 \) such that \( g(n) \leq cf(n) \) for all \( n \geq n_0 \).
Revenge of $O$ Examples

$g$ is in $O(f)$ iff there are positive constants $c$ and $n_0$ such that $g(n) \leq cf(n)$ for all $n \geq n_0$.

Is $n+5$ in $O(n^2)$?
Is $n^2-100$ in $O(n)$?
Is $n^2$ in $O(n^3)$?

Revenge of $O$ Examples

$g$ is in $O(f)$ iff there are positive constants $c$ and $n_0$ such that $g(n) \leq cf(n)$ for all $n \geq n_0$.

Is $n+5$ in $O(n^2)$?
Yes, $c = 1$ and $n_0 = 3$ works.

Is $n^2-100$ in $O(n)$?
No, no matter what $c$ we pick, $cn^2-100 > n$ for big enough $n$.

Is $n^2$ in $O(n^3)$?
Yes, $c = 1$ and $n_0 = 1$ works.
Yes, $c = 2$ and $n_0 = 77$ works.
Yes, $c = 55$ and $n_0 = 102$ works.

Homework

- Course Book Chapter 7
  - Has a formal notation for this kind of analysis!
- Problem Set 3 due soon!