RECURSION (CONTINUED)

Lecture 9
CS2110 – Spring 2018

Prelim two weeks from today: 13 March.
1. Visit Exams page of course website, check what time your prelim is, complete assignment P1 Conflict ONLY if necessary.
2. Review session Sunday, 11 March, 1-3PM.
3. A3 is due 2 days from now, on Thursday.
4. If appropriate, please check the JavaHyperText before posting a question on the Piazza. You can get your answer instantaneously rather than have to wait for a Piazza answer.

Example: "default", "access", "modifier", "private" are well-explained in the JavaHyperText.

Why is the product of an empty bag of values 1?
Suppose bag b contains 2, 2, 5 and p is its product: 20.
Suppose we want to add 4 to the bag and keep p the product.
We do:
put 4 into the bag;
p = 4 * p;

Suppose bag b is empty and p is its product: what value?
Suppose we want to add 4 to the bag and keep p the product.
We do the same thing:
put 4 into the bag;
p = 4 * p;

For this to work, the product of the empty bag has to be 1,
since 4 = 1 * 4

// invariant: p = product of c[0..k-1]
what's the product when k == 0?

0 is the identity of + because 0 + x = x
1 is the identity of * because 1 * x = x
false is the identity of || because false || b = b
true is the identity of && because true && b = b
1 is the identity of gcd because gcd({1, x}) = x
For any such operator a, that has an identity, a of the empty bag is the identity of a.
Sum of the empty bag = 0
Product of the empty bag = 1
OR (||) of the empty bag = false.
gcd of the empty bag = 1

For any operator a, that has an identity, a of the empty bag is the identity of a.

Recap: Understanding Recursive Methods

1. Have a precise specification
2. Check that the method works in the base case(s).
3. Look at the recursive case(s). In your mind, replace each recursive call by what it does according to the spec and verify correctness.
4. (No infinite recursion) Make sure that the args of recursive calls are in some sense smaller than the pars of the method.

Problems with recursive structure

Code will be available on the course webpage.
1. exp - exponentiation, the slow way and the fast way
2. perms – list all permutations of a string
3. tile-a-kitchen – place L-shaped tiles on a kitchen floor
4. drawSierpinski – drawing the Sierpinski Triangle

http://codingbat.com/java/Recursion-1
Computing $b^n$ for $n \geq 0$

- **Power computation:**
  - $b^0 = 1$
  - If $n \neq 0$, $b^n = b \cdot b^{n-1}$
  - If $n \neq 0$ and even, $b^n = (b^2)^{n/2}$

  *Judicious use of the third property gives far better algorithm*

  Example: $3^8 = (3^2)^4$ (computed from the table)

```java
/** = b**n. Precondition: n >= 0 */
static int power(double b, int n) {
    if (n == 0) return 1;
    if (n%2 == 0) return power(b*b, n/2);
    return b * power(b, n-1);
}
```

Suppose $n = 16$

Next recursive call: $8$

Next recursive call: $4$

Next recursive call: $2$

Then $0$

$16 = 2^{4}$

Suppose $n = 2^k$

Will make $k + 2$ calls

### Table of log to the base 2

<table>
<thead>
<tr>
<th>$k$</th>
<th>$n = 2^k$</th>
<th>$\log_2(n) (= k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>2148</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>32768</td>
<td>15</td>
</tr>
</tbody>
</table>

### Difference between linear and log solutions?

```java
/** = b**n. Precondition: n >= 0 */
static int power(double b, int n) {
    if (n == 0) return 1;
    return b * power(b, n-1);
}
```

Number of recursive calls is $n$

Number of recursive calls is $\log n$

To show difference, we run linear version with bigger $n$ until out of stack space. Then run log one on that $n$. See demo.

---

### Permutations of a String

perms(abc): abc, acb, bac, bca, cab, cba

- abc acb
- bac bca
- cab cba

Recursive definition:

Each possible first letter, followed by all permutations of the remaining characters.
Tiling Elaine’s kitchen

Kitchen in Gries’s house: 8 x 8. Fridge sits on one of 1x1 squares. His wife, Elaine, wants kitchen tiled with el-shaped tiles – every square except where the refrigerator sits should be tiled.

```java
/** tile a 2^n by 2^n kitchen with 1 square filled. */
public static void tile(int n) {
    if (n == 0) return;

    // We generalize to a 2^n by 2^n kitchen
    n > 0. What can we do to get kitchens of size 2^{n-1} by 2^{n-1}
```

Sierpinski triangles

```
S triangle of depth 0

S triangle of depth 1: 3 S triangles of depth 0 drawn at the 3 vertices of the triangle
```

```
S triangle of depth 2: 3 S triangles of depth 1 drawn at the 3 vertices of the triangle
```
**Conclusion**

Recursion is a convenient and powerful way to define functions

Problems that seem insurmountable can often be solved in a “divide-and-conquer” fashion:
- Reduce a big problem to smaller problems of the same kind, solve the smaller problems
- Recombine the solutions to smaller problems to form solution for big problem

http://codingbat.com/java/Recursion-1